*Chapter 14*

# Simulators and Emulators for Edge Computing

*Atakan Aral[1], Vincenzo De Maio[1]*

In this chapter, we perform a study of the existing tools for the evaluation of Fog/Edge infrastructures. First, we analyze the state-of-the-art in the simulation of Fog/Edge infrastructures and determine the main challenges in simulation and modeling such infrastructures. Then, we use a scientific methodology to identify the most important simulation and emulation tools, identifying their main characteristics, and define a classification. Each tool is then described in detail, and compared with the others. Finally, we conclude the chapter with a discussion about future research directions in the area.

## 14.1 Introduction

Modeling and simulation of Cloud computing infrastructures have attracted significant interest in the distributed computing research community. The reason for this has to be found not only in the commercial interest and widespread diffusion of such infrastructures but also in the fact that testing and validation of resource management strategies for Cloud computing is very challenging for researchers. This is due to three main factors: (1) the use of commercial infrastructures, such as Amazon, Azure or Google Cloud, or private experimental testbeds, affects the reproducibility of experiments; (2) some measurements might not be available on commercial infrastructures, due to privacy issues, and (3) deployment and management of a real-world Cloud infrastructure to perform experiments is either costly or time demanding. The use of simulation and emulation tools for Cloud infrastructures allows testing of provisioning and resource management strategies before deploying them on a real Cloud infrastructure, minimizing risks for researchers and Cloud providers. For these reasons, the development of accurate simulation and emulation tools is of paramount importance for the advancement of research in this field.

Considering the recent affirmation of Fog/Edge computing, modeling and simulation of Fog/Edge computational resources become even more complex. This is because in comparison with Cloud computing, the application areas of Fog/Edge

[1]Institute of Information Systems Engineering, Vienna University of Technology
Favoritenstrasse 9-11, 1040 Vienna, Austria
{first name}@ec.tuwien.ac.at

computing vary by the extent of (1) geographical distribution, (2) scale, (3) heterogeneity, (4) processing type (batch or near real-time), (5) mobility, and (6) the interplay between the Edge, the Fog, and the Cloud [1]. These challenges are described in the following sections.

*Geographical distribution*
While Cloud data centers are distributed around the globe, and often very far from the source of the data, Fog/Edge nodes are deployed in close proximity to the data sources. This geographical proximity to the data sources has effects on latency and the time required for data transfer and processing, which has to be considered in the modeling of Fog/Edge nodes.

*Scale*
Due to the higher number of devices that can be present in a Fog/Edge infrastructures (e.g. Cloud data centers, Fog micro data centers, Edge devices, Mobile devices, and IoT devices), and the consequently larger scale of systems to be modeled, there is the need for providing models and simulators/emulators that can deal with the greater scale of such systems. Another issue in this sense is related to the evaluation: while there is a plethora of real-world datasets for the execution of Cloud applications, at the moment we can notice a lack of real-world data and difficulty of measurements for systems of such scale.

*Heterogeneity*
While Cloud infrastructures rely on centralized data centers and relatively homogeneous commodity hardware, Fog/Edge infrastructures move computation closer to the source of data, relying on heterogeneous computing facilities (e.g. mobile devices, IoT devices, and micro data centers). Moreover, some of these computing facilities are not designed to perform the processing required by typical Fog/Edge applications, demanding for different modeling approaches, more advanced than the one used for typical computing facilities.

*Processing type*
Typical Fog/Edge applications range from batch processing applications to near real-time applications. Each of these applications has different objectives and characteristics, requiring different modeling strategies. For this reason, the simulation and the modeling of Fog/Edge application should provide ways to model a wide range of applications.

*Mobility*
Modeling the mobility of mobile computing devices becomes of paramount importance in the context of Fog/Edge applications. In fact, many works such as [2, 3], investigate the benefits of offloading mobile application to Fog/Edge infrastructure. In addition, mobile devices can also be seen as Edge devices, due to their computational capabilities [4]. In order to accurately model Fog/Edge infrastructures, then, it is necessary to provide accurate modeling of different mobility patterns, ranging

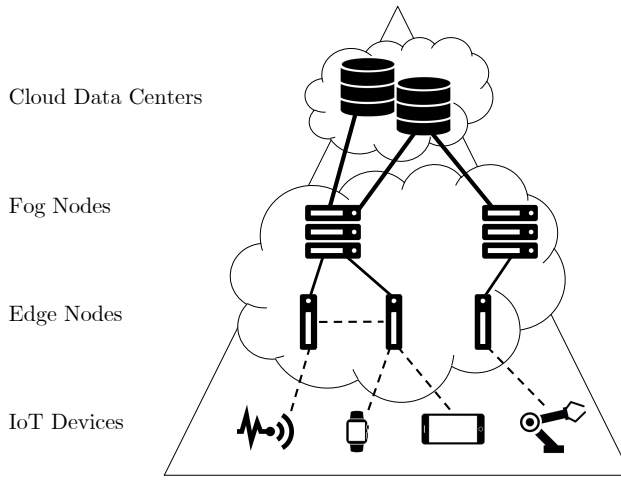


*Figure 14.1 Fog and Edge Computing Architecture.*

from pedestrian to vehicular mobility, and the influence of mobility over computations and network communications.

*The interplay between Edge, Fog, and Cloud*

According to [5], Fog/Edge computing is not supposed to replace Cloud computing, but mostly to interoperate with it: for example, large-scale batch computation could be executed on the Cloud, while near real-time applications could perform low latency data processing on the Fog/Edge layers (as demonstrated in Figure 14.1). To this end, there is the need to model the different types of processing at different layers of the infrastructure, as well as orchestration and network connections available at different layers.

Such challenges cannot be addressed by typical Cloud simulators, and require specifically tailored solutions for simulating such infrastructures. To the best of our knowledge, there exists only a single literature survey on simulating fog and edge computing [1]. Our work extends this study in the following ways:

- We provide a detailed classification of the simulators, identifying six key criteria regarding the modeling capability. Among these, only mobility is considered in previous work [1];
- We widen the scope of the reviewed simulators/emulators by considering newer tools. The total number of tools increased from seven to eleven. Particularly, FogExplorer, YAFS, RECAP, and Sleipnir simulators, as well as Cloud4IoT emulator, are missing in previous work. On the other hand, we exclude IOTSim [6] because it models IoT applications that are hosted on centralized Cloud and not in Fog/Edge infrastructures;
- We explicitly describe the research methodology of our systematic literature review and classification;

- We redraw the high-level architecture diagrams of the simulators and emulators (where available) in a uniform format for quick comparison.

The rest of the chapter is structured as follows. In section 14.2, we introduce our proposed classification and the methodology behind it. Then we proceed to describe existing solutions for deterministic simulators (section 14.3), stochastic simulators (section 14.4), and emulators (section 14.5). Finally, we compare the simulators, discuss future research directions, and draw conclusions in section 14.6.

## 14.2 Classification of Simulators

### *14.2.1 Research Methodology*

A systematic literature review follows precisely defined steps and reduces bias. It also increases the reproducibility of the results. In this section, we describe the methodology used for discovering simulators and emulators, as well as for classification. We utilized Google Scholar digital library to discover candidate works for our study. The following query is executed to match only the titles of the works.

```
("edge" OR "fog")("simulator" OR "simulation"
OR "emulator" OR "emulation" OR "evaluation")
```

Additionally, we limited the publication year from 2017 onward. After several trials, no relevant work is discovered before that year. The query is first executed in December 2018 to obtain the initial set of work. It is repeated in July 2019 to account for works that are more recent. Indeed, YAFS is included in our study only after the second query. This procedure returned 406 results in total, which are then subjected to a selection procedure based on the following exclusion criteria inspired by [7].

- Works that are not published in a peer-reviewed venue.
- Works that do not specifically address Fog/Edge computing infrastructures.
- Works that do not propose a general-purpose simulator or emulator.

Consequently, we obtained 11 relevant works to include in our study. Then we proceeded to extract main contributions and highlighted features from each work and the descriptions in the corresponding source code repository. This procedure yielded 29 features. After combining redundant features and excluding generic features that are supported by all simulators, six classification features remained. Excluded generic features include high-level network model (e.g. topology definition) and resource management.

### *14.2.2 Features*

In this section, we describe the main features of the simulation and emulation tools considered in our classification. Our classification is presented in Tables 14.1. Emulators are excluded from this classification because they refer to a specific platform and are used mostly to test the deployment of applications on this specific environment. In each column, '**Y**' indicates a supported feature, 'N' a missing feature, and

'N*' a feature that is currently not supported but indicated as a future work either in the paper or source code repository. These tables also contain general information about each simulator including underlying simulation platform, license for the source code availability, the programming language used, and recent activity of the project. We assume that the projects with a git commit within the last six months are active (**Y**) as of July 2019.

*Behavior*
We propose the deterministic or stochastic behavior of the simulator as the main distinctive feature. As described in the following two sections, this feature may completely change the way the simulator is utilized. The top six simulators in Table 14.1 are deterministic, whereas bottom two (separated by a bar) are stochastic.

*Energy Model*
This feature indicates whether the energy consumption of the Fog/Edge nodes or user devices is simulated or emulated.

*Cost Model*
A cost model accounts for the monetary aspects of the computation based on different pricing schemes.

*Low-Level Network Model*
All simulators support high-level network topology design. This feature is about whether packet-level granularity is available.

*Mobility Model*
Mobility indicates the possibility of the user and/or compute node movement during the simulation runtime.

*Failure Model*
This feature indicates whether it is supported to simulate the failure and unavailability of certain nodes or links.

## 14.3 Deterministic Simulators

Simulators introduced in this section do not contain random variables; hence, they produce always the same output given a particular input. Deterministic models are known to be easier to build and implement than stochastic ones. We review following six deterministic simulators: iFogSim, EdgeCloudSim, FogNetSim++, FogExplorer, YAFS, and RECAP.

### *14.3.1 iFogSim*

To the best of our knowledge, iFogSim[2] is the first simulator specifically addressing Fog/Edge computing infrastructures. At the time of writing, it is also the most

[2]https://github.com/Cloudslab/iFogSim

*Table 14.1 Comparison of Fog/Edge computing simulators (N* indicates a planned but not yet implemented feature).*

| | General Information | | | | Classification | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Edge/Fog Simulator** | **Underlying Platform** | **Source Availability** | **Prog. Language** | **Currently Active** | **Energy Model** | **Cost Model** | **Low-Level Network** | **Mobility Model** | **Failure Model** |
| iFogSim | CloudSim | Apache v2.0 | Java | N | **Y** | **Y** | N | N* | N* |
| EdgeCloudSim | CloudSim | GPL v3.0 | Java | **Y** | N* | N | N | **Y** | N* |
| FogNetSim++ | OMNeT++ | GPL v3.0 | C++ | N | **Y** | **Y** | **Y** | **Y** | N |
| FogExplorer | N/A | MIT License | JavaScript | N | N | **Y** | N | N | N |
| YAFS | SimPy | MIT License | Python | **Y** | N* | N | N | **Y** | **Y** |
| RECAP | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A |
| FogTorchPi | N/A | MIT License | Java | N | **Y** | **Y** | N | N | **Y** |
| Sleipnir | FogTorchPi | MIT License | Java | **Y** | **Y** | **Y** | N | **Y** | **Y** |

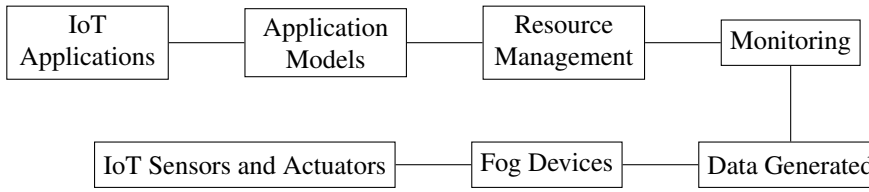


*Figure 14.2 High-level architecture of iFogSim.*

cited work that is discussed here and arguably the most widely used one. iFogSim is developed for IoT and Fog environments and considers the impact of resource management techniques in latency, network congestion, energy consumption, and operational costs [8].

The simulator supports the Sense-Process-Actuate model, that is, data are produced by the sensors, processed by the Fog nodes, and consumed by the actuators. Alternatively, the stream-processing model is also allowed. In this model, streaming data from the sensors are processed online at the Fog nodes and then forwarded to cloud data centers for long-term analytics. The fog computing environment is modeled as a layered architecture shown in Figure 14.2, which includes the following elements from the bottom to the top.

1. **IoT sensors/actuators** are the sources and the sinks of the data, respectively. They both interact with the environment or an external physical system.
2. **Fog devices** are the main computation sources. They host application modules and can be located anywhere between the cloud and the network edge.
3. **Data streams** are either IoT data emitted from the sensors or log data generated by the Fog devices.
4. **Infrastructure monitoring** collects log data from Fog devices, sensors, and actuators. It keeps track of resource use, power consumption, and availability.
5. **Resource management** is where placement and scheduling decisions are made. They are responsible for confirming QoS constraints. Complex resource management mechanisms such as migration can be implemented by the user.
6. **Application model** represents the IoT application as a directed graph based on the distributed data flow model (DDF).

iFogSim is implemented as an extension to the widely-used event-based cloud computing simulator, CloudSim[3] [9]. Two application placement strategies, namely cloud-only and edge-ward placement, are already implemented as reference. Additionally, a graphical user interface is available to describe the physical network topology intuitively. Instructions to build the simulation of a Fog/Edge environment are provided in [10].

An extension to iFogSim that supports data placement strategies is also available[4]. This extension also supports the formulation of the data placement problem

[3]http://www.cloudbus.org/cloudsim/
[4]https://github.com/medislam/iFogSimWithDataPlacement

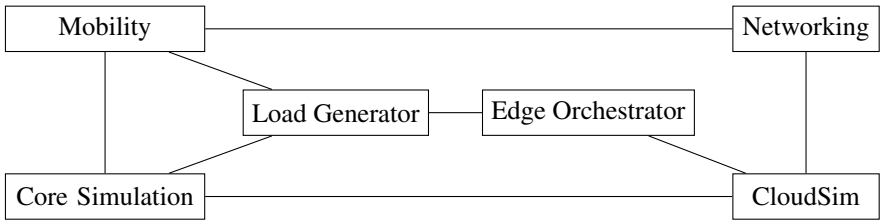


*Figure 14.3 High-level architecture of EdgeCloudSim.*

as a Mixed Integer Linear Program (MILP) and includes a parallel solver in order to decrease the simulation time [11].

### *14.3.2 EdgeCloudSim*

EdgeCloudSim[5] is another extension to CloudSim with edge computing support. Based on our literature review, it is the only deterministic simulator that targets the edge computing systems specifically, instead of the whole Fog computing continuum. The focal points of the simulator in addition to the general features of CloudSim include; edge-specific applications, mobility, and wide-area or wireless network. The five main modules of EdgeCloudSim are shown in Figure 14.3 and described as follows [12].

1. **Core simulation** supports the XML-based configuration of simulation scenarios and detailed logging to facilitate the analysis of results.
2. **Edge orchestrator** manages the available resources, particularly taking offloading, placement, and replication decisions. In EdgeCloudSim, there is a single centralized orchestrator.
3. **Networking** module deals mainly with wireless networks and wide-area networks (WAN), which are missing in CloudSim. Both WLAN and cellular networks are supported for the communication between the edge servers and the users. WAN design follows the single server queue model.
4. **Mobility** of the users is simulated in EdgeCloudSim via a nomadic mobility model. It is possible to extend this model for other mobility strategies such as vehicular mobility.
5. **Load generator** provides the capability of defining different application types and generate application tasks according to a Poisson arrival process, by default.

EdgeCloudSim is designed flexibly through the factory pattern that facilitates extensions. Indeed, in a recent study [13], it is extended with a point-of-interest based mobility model and a node availability model and that is based on the failure–repair process.

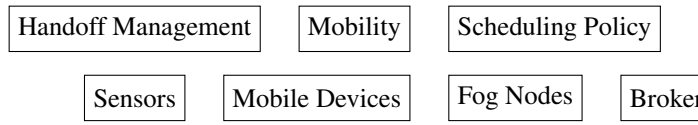


*Figure 14.4 High-level architecture of FogNetSim++.*

### *14.3.3 FogNetSim++*

FogNetSim++[6] [14] is a simulator that focuses on the network aspects of the Fog/Edge computing infrastructure rather than data generation and consumption. This results in the consideration of packet drop/error rate, network congestion, and channel collision. It allows the creation of a network system with static and dynamic nodes and supports various Fog/Edge protocols for communication, including MQTT, CoAP, and AMQP. Another focus of the simulator is the realistic mobility of the users. It supports various mobility models as well as handover mechanisms. The simulation kernel library is provided by the widely used discrete event library, OMNeT++[7]. FogNetSim++ modules are developed as an extension to the INET framework[8], which is an OMNeT++ model suite for wired, wireless and mobile networks. Design architecture of the simulator is given in Figure 14.4. Three main modules are described as follows.

1. **Broker** is the centralized resource manager, which is responsible for task scheduling, execution, and handover. Handovers are carried out due to either user mobility or load balancing. The broker also manages the communication between Fog nodes, Cloud data center, and end-users.
2. **Fog node** is the provider of the computation. It is a static node located on a network gateway. They communicate with users and sensors through wireless access points.
3. **User and sensor** are the sources of the data and requests. Different from sensors, which only generate data, users can also receive data. Random waypoints as well as Mass, Gauss Markov, Chaing, Circle, Linear, and vehicle mobility models are available.

The simulator also includes a graphical user interface as well as ready-to-use pricing models and an energy model. Pricing models include pay-as-you-go, subscription, pay-for-resources, and a hybrid one; whereas the energy model considers the consumption of both the device and Fog node. The scheduling policy is left to the programmer to implement.

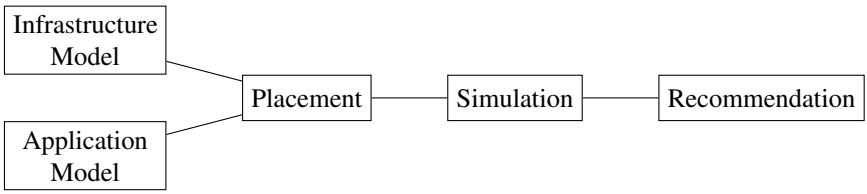


*Figure 14.5 High-level architecture of FogExplorer.*

### *14.3.4 FogExplorer*

FogExlorer[9] is an interactive simulator intended for the design phase of Fog/Edge infrastructures, and particularly for the definition of the application architecture, the runtime infrastructure, and the mapping between these two [15]. It is based on iterative modeling and simulation shown in Figure 14.5 with the following steps.

1. High-level modeling of the application (modules and inter-module data streams) and the infrastructure (machines and interconnections).
2. Manual placement of the application modules on Fog machines.
3. Simulation and calculation of QoS and cost.
4. Recommendations for optimizing the placement by highlighting under-provisioned resources and missing connections.

Since FogExplorer targets the application design phase, both application and infrastructure models are quite abstract [16]. Application modules can be of three types: sources, which produce the data; services, which process them; and sinks, which consume them. Sources are defined by an output rate and services by processing time as well as the ratio of output size. Outputs of the sources and services are either duplicated or equally distributed to all subsequent services or sinks. Required bandwidth between the modules is derived from these data. All three types of modules also have predefined memory requirements. Infrastructure nodes are defined by a processing performance indicator, available memory, and unit memory price. Connections, on the other hand, have available latency and bandwidth, as well as bandwidth cost properties.

After the user defines the environment and suggests a module placement, the simulator calculates four metrics: processing cost, processing time, transmission cost, and transmission time. Additionally, it highlights under-provisioned machines and connections as well as the cases that a data stream cannot flow due to missing connections between the machines. Based on this feedback, the user can update the models or placement and a new simulation is triggered. FogExplorer is implemented as a front-end only web-based tool with JavaScript.

[5]https://github.com/CagataySonmez/EdgeCloudSim
[6]https://github.com/rtqayyum/fognetsimpp
[7]https://omnetpp.org/
[8]https://inet.omnetpp.org/
[9]https://github.com/OpenFogStack/FogExplorer

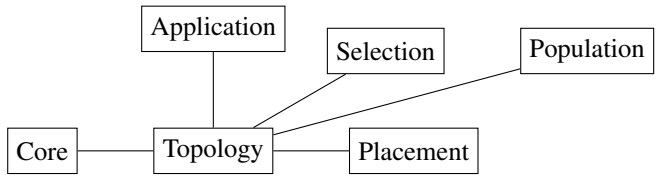


*Figure 14.6 High-level architecture of YAFS.*

### *14.3.5 YAFS*

Yet Another Fog Simulator[10] focuses on network topology aspects of Fog/Edge computing. It is developed on top of the SimPy discrete-event simulator framework[11]. Similar to EdgeCloudSim, YAFS supports text-based (JSON) configuration and ready to analyze CSV results. Authors list the network, workload sources, customized placement, custom processes, post-simulation data analysis, and scenario definition as the highlights of the YAFS [17]. The following are the main modules of the simulator that provide these features.

1. **Topology and entity modeling** implements the complex network theory (e.g. scale-free networks) and represents the Fog devices and the inter-communication links as a graph. It is possible to import CAIDA and BRITE topology models in different file formats.
2. **Application model** follows the distributed data flow (DDF) model similar to iFogSim. Here, an application is defined as a directed acyclic graph where the nodes represent the tasks and the links their interoperability.
3. **Dynamic policies** deal with resource selection, task placement, and resource allocation. It is also possible to define custom processes that model user mobility and resource unavailability (due to failures).
4. **Results** include two types of events: task execution and network transmissions, which are logged to CSV files. It is then possible to visualize these events as graph animations.

The high-level architecture of YAFS is presented in Figure 14.6. Among all simulators, it seems to be most actively developed and maintained one with recent improvements. For instance, the latest version allows importing GPX traces for the user and device mobility.

### *14.3.6 RECAP Simulator*

The RECAP simulator [18] is currently under development within the scope of the Horizon 2020 project *Reliable Capacity Provisioning and Enhanced Remediation for Distributed Cloud Applications* (RECAP)[12]. The project promises autonomous and optimal strategies for reliable and predictable capacity provisioning, application placement, and infrastructure management [19]. The scale of the considered

[10]https://github.com/acsicuib/YAFS
[11]https://simpy.readthedocs.io/
[12]https://recap-project.eu/

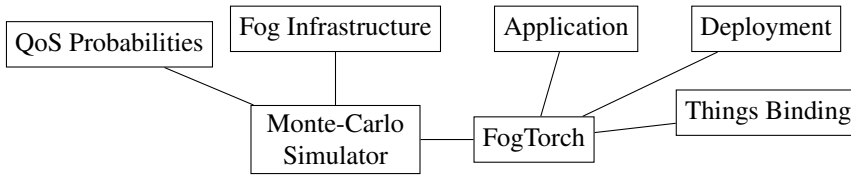


*Figure 14.7 High-level architecture of FogTorchPI.*

systems hinders full-scale deployment for experimentation. Correspondingly, the RECAP simulator will cooperate with the RECAP optimizer component to simulate distributed cloud applications as well as the underlying infrastructure for a reproducible and controllable evaluation. The simulator will consist of: (1) Experiment Manager, which gathers all models and experimental system information; (2) Simulation Manager, which controls the simulation; (3) Event Coordinator, which serializes simulation results and manages optimization and failure events; and (4) Event Generator, which injects events such as new tasks or failures to the simulation. At the time of writing, further details regarding the implementation and features of the simulators are not published, to the best of our knowledge. This refrains us from providing a fair comparison to other simulators. Hence, the RECAP simulator is excluded from our proposed classification.

## 14.4 Stochastic Simulators

In this section, we describe simulations based on stochastic models of Fog/Edge infrastructures. The main advantage of using stochastic simulators is that in contexts with high mobility (e.g. VANETs, mobile computing) or with highly unreliable resources (e.g. IoT), the availability of resources can be easily modeled as a stochastic process, as done in works like [20, 21]. In the following, we describe two state-of-the-art simulators: FogTorchPi and Sleipnir.

### *14.4.1 FogTorchPI*

FogTorchPi[13] [5] is an open-source Java simulator of Fog/Edge infrastructures with the high-level architecture depicted in Figure 14.7. FogTorchPi input consists of:

- A description of a Fog/Edge infrastructure *i*. In this description, it is needed to specify the (1) IoT devices (a.k.a. "Thing", in the simulator terminology; (2) Fog/Edge and Cloud data centers available for application deployment. For each one of these data centers it is needed to specify the hardware capabilities (number of cores, amount of RAM and storage); (3) Network infrastructure, along with the probability distributions of the QoS (latency, bandwidth) available on the communication links (Cloud-to-Fog, Fog-to-Fog and Fog-to-Things); (4) Cost for purchasing Cloud/Fog virtual instances.

[13]https://github.com/di-unipi-socc/FogTorchPI

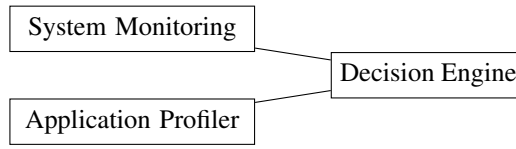


*Figure 14.8 High-level architecture of Sleipnir.*

- An application description $a$. Each application is composed of different components. For each component, it is needed to specify (1) hardware (CPU, RAM, storage), software (OS, libraries, frameworks) and IoT device binding; (2) the QoS (e.g. latency and bandwidth) needed to adequately support component-component and component-Thing interactions after application's deployment.
- An IoT mapping $\tau$, describing the requirements of each connection between each application component and IoT device (Thing);
- A deployment policy $\delta$ that describes on which nodes components can be deployed to respect security or business-related constraints.

Starting from the input $(i, a, \tau, \delta)$, FogTorchPi determines admissible deployments for $a$ over infrastructure $i$, given the QoS requirements of $a$, the IoT mapping $\tau$ and the deployment policy $\delta$. QoS requirements of $a$ are determined by the user, while QoS available on the infrastructure is determined by sampling the probability distribution for latency and bandwidth specified in the simulator setup. FogTorchPi follows a Monte-Carlo approach to identify the admissible deployments. First, it performs a sampling of the QoS available on each communication link, according to the probability distribution specified. At the end of the sampling phase, the simulator performs a random admissible deployment of each application component, such that respects (1) the QoS requirement of application $a$, (2) the IoT mapping $\tau$ and (3) the deployment policy $\delta$. These two phases are repeated for a given number of iterations. At the end of the iterations, the resulting deployments are collected in a histogram, in order to calculate the frequency of each admissible deployment. For each deployment in the histogram, FogTorchPi also calculates the QoS measures and its cost, according to the cost model specified in the infrastructure description $i$.

### *14.4.2 Sleipnir*

SLEIPNIR[14] (Spark-enabled mobiLe Edge offloadIng Platform moNte-carlo sImulatoR) is an extended version of FogTorchPi, described in [20]. The main improvement of SLEIPNIR with respect to its predecessor is the fact that it runs on the Apache Spark platform [22] that allows it to easily scale according to the underlying computational resources. In addition to its predecessor, it also provides simulation support for mobile devices and for mobility traces coming from SUMO. At the time we write, simulator architecture is as given in Figure 14.8 and its input consists of:

- A description of an Edge infrastructure $\mathscr{I}$. This description includes the number and specification of Cloud, Edge, and mobile nodes respectively described the

[14]https://github.com/vindem/sleipnir

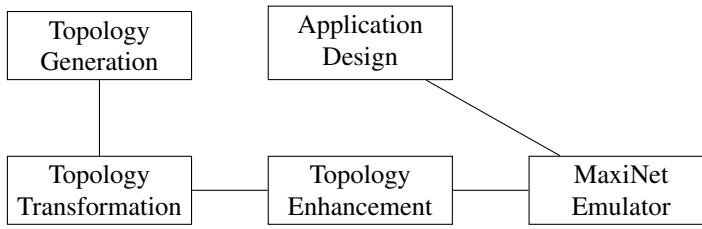


*Figure 14.9 High-level architecture of EmuFog.*

sets $\mathscr{C}$, $\mathscr{E}$ and $\mathscr{D}$, and a directed graph $\mathscr{N}$ modeling the network connections between the nodes, as in FogTorchPi;
- A map $\mathscr{M}$ of a geographical area where to place the Edge nodes, according to a placement algorithm. At the moment, the simulator offers only the possibility of deploying Edge over three areas of the city of Vienna;
- An optional mobility trace, coming from SUMO, modeling the mobility of mobile devices over the selected map;
- A workflow $\mathscr{W}$, composed of different mobile application description $\mathscr{A}$. Each application is described as a Directed Acyclic Graph (DAG). At this moment, the simulator offers five different types of applications: NAVIGATOR, modeling a navigation app; CHESS, modeling a chess game on a smartphone; FACEBOOK, modeling the posting of a picture on facebook; FACERECOGNIZER, modeling a photo-processing app, and ANTIVIRUS, modeling a virus scan. In the workflow, it is also possible to specify the frequency at which each application occurs.

Starting from $\mathscr{I}, \mathscr{M}, \mathscr{W}$, the simulator identifies admissible deployments of tasks in $\mathscr{W}$ over infrastructure $\mathscr{I}$. The user specifies the QoS requirement for the workflow, based on which it tries to determine an admissible deployment of tasks on the infrastructure. QoS available on the infrastructure is determined by sampling the probability distribution for latency and bandwidth specified in the simulator setup. The sampling of infrastructure and workflow is repeated for a given number of iterations. At the end of the iterations, the resulting deployments are collected in a histogram, in order to calculate the frequency of each admissible deployment.

Applications and offloading policies are defined in [20], while in [21] it has also been used for comparing different Edge provisioning methods.

## 14.5 Emulators

### *14.5.1 EmuFog*

EmuFog[15] provides a test environment for Fog computing, built on top of MaxiNet [23] (an extension of Mininet [24] that allows the emulation of datacenter network spanning over multiple network nodes). The main design objectives of EmuFog

[15]https://github.com/emufog/emufog


*Figure 14.10 High-level architecture of Cloud4IoT.*

are (1) scalability, allowing the emulation of large-scale Fog/Edge computing scenarios; (2) emulation of real application/workloads, allowing the developer to package real-world application and run them in the emulated environment; (3) extensibility, to allow each framework's component to be replaced by custom-built components, suiting the scenario to be emulated. The emulation workflow consists of four main steps depicted in Figure 14.9.

1. **Topology generation**: in this phase, the developer generates a network topology using tools like BRITE [25]. EmuFog also allows to load network topology from a topology database, which ensures reproducibility of experiments on real-world network topology;
2. **Topology transformation**: in this phase, the network topology generated in the topology generation phase is converted in the EmuFog network model, which is seen as an undirected graph where each network device is an autonomous system;
3. **Topology enhancement**: in this phase, the network topology is enhanced by adding the Fog/Edge nodes. To this end, first, the edge of the network topology is determined, and then Fog/Edge nodes are placed in the network according to a placement policy. Placement policy is specified in a Fog/Edge configuration file.
4. **Deployment and execution**: in this phase, the enhanced topology is deployed in the emulation environment. Fog/Edge nodes are placed in the emulated network, while the applications are deployed on them in form of Docker containers.

In [26], the emulator is described in detail, including also performance evaluation.

At the time we write, EmuFog does not provide emulation of mobility, therefore does not allow the evaluation of mobile devices (e.g. drones, vehicles and mobile phones). Also, no emulation of hierarchical Fog/Edge environments is provided.

### *14.5.2 Cloud4IoT*

Cloud4IoT [27] is a lightweight PaaS platform specialized for Edge/Cloud applications, designed for the native support of IoT. Its logical model (Figure 14.10) is composed of three layers: (1) the *Central Cloud platform*, which hosts the central controller functionality and offers additional scalability when needed. This layer works as an orchestrator and a scheduler for the workload running on the platform, and it is implemented on a private OpenStack-based Cloud; (2) the *Edge Cloud Modules*, which are designed to let the data-intensive applications run close to the source of data (in this case, the IoT devices, also defined as IoT gateways). Such modules are small-sized servers with limited storage and computational power. Such servers can be used to offload the workload from the edges to the cloud and vice

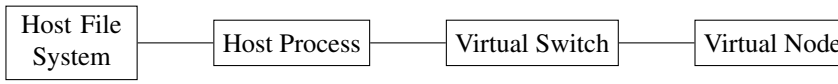


*Figure 14.11 High-level architecture of Fogbed.*

versa. Moreover, they also improve resilience, providing additional resources to the IoT gateways; (3) the *IoT gateways*, representing the hardware interface with objects and the acquisition of data from IoT sensor objects. Cloud4IoT mainly supports two types of applications: IoT support applications and Data logic/processing.

IoT-specific applications support the deployment and the maintenance of new objects/sensors in the field. Such applications offer the following services: (1) *discovery* of new objects attached to an IoT gateway; (2) *retrieval* of the firmware version suitable with OS and model of the needed object; (3) *dispatch* of the data collected on the edge modules connected to the IoT gateway; (4) *installation* of new applications to support and manage newly acquired and/or updated objects.

The latter type of applications instead is deployed, scheduled and orchestrated from the central cloud onto the edge modules, according to the current platform condition. The deployment is performed according to the users' latency requirements: applications with high latency requirements can be deployed on the cloud, while an application with short latency requirements can be deployed on the Edge modules. In the current version of Cloud4IoT, orchestration is performed by employing a simple threshold-based mechanism.

### *14.5.3 Fogbed*

Fogbed is described in [28] with the simplified architecture in Figure 14.11. It is described as a framework and integration toolset for rapid prototyping of fog components in a virtualized environment. The simulator is based on Mininet [24] and Docker. It extends the Mininet emulator by allowing the use of Docker containers as virtual nodes. Fogbed emulation works through the deployment of pre-configured container images, such as:

1. *Cloud container image*: containers of this type emulate virtual resources for IoT applications. They can also act as a virtual cloud instance or act as proxies for cloud services located in remote data centers.
2. *Fog container image*: containers of this type emulate Fog nodes that perform processing and storage of data coming from the IoT gateways. It also performs analytics and filtering of raw data.
3. *Edge container image*: containers of this type emulate IoT gateways or smart IoT devices. It is also possible to simulate sensors and send data to a virtual Fog instance.

The emulation workflow goes as follows: first, the developer provides the images that he/she wants to instantiate to perform the emulation. Second, the developer defines a network topology for application testing. In contrast with EmuFog (see Section 14.5.1), Fogbed employs its own language for the definition of topology.

Afterward, Fogbed executes the emulator with the described topology. Once the emulator starts its execution, the developer starts the management system that connects to the emulated environment using the instance API. The management system then deploys the application on the platform and starts the required services in the virtual nodes. Afterward, the application can run in the platform, and network flow statistics are collected and stored for future analysis. In [28], the authors describe a use case of Fogbed for the development of a health monitoring application using Cloud/Fog infrastructures and data coming from simulated wearable sensors. At the current stage, the emulator does not offer emulation of features like mobile offloading, fault tolerance and reliable management and security. This might be improved by developing new functionalities that allow injecting failures and simulating security attacks. In addition, a deeper study of the emulation scalability would be required.

## 14.6 Discussion

In this chapter, we discussed the main issues about modeling and simulation of Fog/Edge infrastructures. Based on these issues, we identified the main characteristics of the most used solutions in the literature for the simulation and emulation of Fog/Edge infrastructures and designed a classification for these tools. Finally, we described the most used solutions in the literature. According to our analysis, we notice that there is no perfect solution for simulation of Fog/Edge infrastructure since each of the existing tools is specifically tailored for the given characteristics. Moreover, at the time we write, some research challenges are left open. We describe them in the rest of this section.

*Reliability*
Fog/Edge computing hardware is prone to failures due to geographical dispersion, limited resources, and the absence of advanced support systems as in cloud data centers [29]. The reliability of the resources has a direct impact on other QoS parameters such as latency and energy efficiency. However, this aspect of edge computing is mostly overlooked in existing simulators. It is left as future work in two most widely used simulators, namely iFogSim and EdgeCloudSim. Other works such as YAFS and FogTorchPi support dynamic availability of compute nodes and network links due to failures. More detailed failure models (e.g. correlated failures or Byzantine faults) are needed for the realistic simulation of the Fog/Edge environment.

*Network simulation*
The advent of distributed services rapidly removes the borders between computing and networking systems and demands their joint consideration. Reflecting this circumstance, all Fog/Edge simulators and emulators support network modeling to a certain extent. However, network models are limited to high-level topology design in almost all cases. Only FogNetSim++ support low-level details such as packet routing, switching, etc. This is because it is developed as an extension to an existing network simulator, OMNeT++.

*Validation*
At the time we write, most of the simulation and emulation frameworks lack an extensive and rigorous evaluation of results. This is because of the lack of real-world implementations of large-scale Fog/Edge infrastructures. Also, the only commercial providers delivering Edge services are Amazon, with Lambda@Edge[16] and Azure IoT Edge[17]. However, commercial platforms do not allow to measure all the parameters that are needed by researchers, which poses several issues in collecting the data that are necessary for validation of the simulations. In papers like [30], validation is performed on data coming from a smart building use case. However, such datasets are used only as an example of the workload of typical Fog/Edge infrastructures. In many works, validation is performed using Cloud or IoT datasets, which are not suitable to simulate near real-time applications typical of Fog/Edge infrastructures. In the future, there is the need for developing datasets representing real-world Fog/Edge infrastructures or deploying more real-world infrastructures that allow to perform measurements or to deploy applications for validation of simulations.

*Missing general-purpose solution*
As our classification in Table 14.1 demonstrates, there currently exists no general-purpose solution for all simulation and emulation needs. iFogSim is the most widely used simulation tool despite the absence of low-level network, mobility, and failure modeling support. In problems, where the network plays the most important role, one may prefer FogNetSim++. Alternatively, if reliability is the main focus, YAFS, FogTorchPi, or Sleipnir might be chosen. However, these solutions are not yet as mature or stable as iFogSim.

## 14.7 Conclusion

In this chapter, we propose a classification of all existing simulators and emulators for Fog/Edge computing. First, we describe the main challenges in simulation and modeling for Fog/Edge infrastructures. Then, we describe the research methodology that we use to identify the main characteristics of Fog/Edge simulators and emulators and classify existing tools for simulation and emulation of Fog/Edge infrastructures. Afterward, we describe Fog/Edge simulators and emulators, identifying their pros and cons. Finally, we describe the possible future research directions and identify issues that should be solved to deliver an accurate simulation of Fog/Edge infrastructures. In the future, we expect increased interest in the simulation and modeling of such infrastructures. Such interest will be even more encouraged by the increasing diffusion of Fog/Edge infrastructures that will help to produce dataset that can be used to perform validation and give more insights on the shape of real-world Fog/Edge infrastructures.

[16]https://aws.amazon.com/en/lambda/edge/
[17]https://azure.microsoft.com/en-us/services/iot-edge/

## Acknowledgements

This work has been funded through the Rucon project (Runtime Control in Multi Clouds), FWF Y 904 START-Programm 2015.

## References

[1] Sergej Svorobej, Patricia Takako Endo, Malika Bendechache, Christos Filelis-Papadopoulos, Konstantinos M Giannoutakis, George A Gravvanis, Dimitrios Tzovaras, James Byrne, and Theo Lynn. Simulating fog and edge computing scenarios: An overview and research challenges. *Future Internet*, 11(3):55, 2019.

[2] Q. Zhu, B. Si, F. Yang, and Y. Ma. Task offloading decision in fog computing system. *China Communications*, 14(11):59–68, 2017.

[3] Long Chen, Jigang Wu, Xin Long, and Zikai Zhang. ENGINE: cost effective offloading in mobile edge computing with fog-cloud cooperation. *CoRR*, abs/1711.01683, 2017.

[4] A. Reiter, B. Prnster, and T. Zefferer. Hybrid mobile edge computing: Unleashing the full potential of edge computing in mobile device use cases. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 935–944, 2017.

[5] A. Brogi, S. Forti, and A. Ibrahim. How to best deploy your fog applications, probably. In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, pages 105–114, May 2017.

[6] Xuezhi Zeng, Saurabh Kumar Garg, Peter Strazdins, Prem Prakash Jayaraman, Dimitrios Georgakopoulos, and Rajiv Ranjan. Iotsim: A simulator for analysing iot applications. *Journal of Systems Architecture*, 72:93–107, 2017.

[7] Pooyan Jamshidi, Aakash Ahmad, and Claus Pahl. Cloud migration research: a systematic review. *IEEE Transactions on Cloud Computing*, 1(2):142–157, 2013.

[8] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.

[9] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50, 2011.

[10] Redowan Mahmud and Rajkumar Buyya. Modelling and simulation of fog and edge computing environments using ifogsim toolkit. *Fog and Edge Computing: Principles and Paradigms*, pages 1–35, 2019.

[11] Mohammed Islam Naas, Jalil Boukhobza, Philippe Raipin Parvedy, and Laurent Lemarchand. An extension to ifogsim to enable the design of data placement strategies. In *IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, pages 1–8. IEEE, 2018.

[12] Cagatay Sonmez, Atay Ozgovde, and Cem Ersoy. Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11):e3493, 2018.

[13] Atakan Aral, Ivona Brandic, Rafael Brundo Uriarte, Rocco De Nicola, and Vincenzo Scoca. Addressing application latency requirements through edge scheduling. *Journal of Grid Computing*, to appear, 2019.

[14] Tariq Qayyum, Asad Waqar Malik, Muazzam A Khan Khattak, Osman Khalid, and Samee U Khan. Fognetsim++: A toolkit for modeling and simulation of distributed fog environment. *IEEE Access*, 6:63570–63583, 2018.

[15] Jonathan Hasenburg, Sebastian Werner, and David Bermbach. Fogexplorer. In *Proceedings of the 19th International Middleware Conference (Posters)*, pages 1–2. ACM, 2018.

[16] Jonathan Hasenburg, Sebastian Werner, and David Bermbach. Supporting the evaluation of fog-based iot applications during the design phase. In *Proceedings of the 5th Workshop on Middleware and Applications for the Internet of Things*, pages 1–6. ACM, 2018.

[17] Isaac Lera, Carlos Guerrero, and Carlos Juiz. Yafs: A simulator for iot scenarios in fog computing. *IEEE Access*, 7:91745–91758, 2019.

[18] James Byrne, Sergej Svorobej, Anna Gourinovitch, Divyaa Manimaran Elango, Paul Liston, Peter J Byrne, and Theo Lynn. Recap simulator: Simulation of cloud/edge/fog computing scenarios. In *2017 Winter Simulation Conference (WSC)*, pages 4568–4569. IEEE, 2017.

[19] Per-Olov Östberg, James Byrne, Paolo Casari, Philip Eardley, Antonio Fernandez Anta, Johan Forsman, John Kennedy, Thang Le Duc, Manuel Noya Marino, Radhika Loomba, et al. Reliable capacity provisioning for distributed cloud/edge/fog computing applications. In *2017 European conference on networks and communications (EuCNC)*, pages 1–6. IEEE, 2017.

[20] Vincenzo De Maio and Ivona Brandic. First hop mobile offloading of dag computations. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 83–92, 2018.

[21] Vincenzo De Maio and Ivona Brandic. Multi-objective mobile edge provisioning in small cell clouds. In *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, ICPE '19, pages 127–138. ACM, 2019.

[22] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache spark: A unified engine for big data processing. *Commun. ACM*, 59(11):56–65, October 2016.

[23] P. Wette, M. Drxler, A. Schwabe, F. Wallaschek, M. H. Zahraee, and H. Karl. Maxinet: Distributed emulation of software-defined networks. In *2014 IFIP Networking Conference*, pages 1–9, 2014.

[24] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th*

*ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA, 2010. ACM.

[25] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: an approach to universal topology generation. In *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 346–353, Aug 2001.

[26] Ruben Mayer, Leon Graser, Harshit Gupta, Enrique Saurez, and Umakishore Ramachandran. Emufog: Extensible and scalable emulation of large-scale fog computing infrastructures. *CoRR*, abs/1709.07563, 2017.

[27] D. Pizzolli, G. Cossu, D. Santoro, L. Capra, C. Dupont, D. Charalampos, F. De Pellegrini, F. Antonelli, and S. Cretti. Cloud4iot: A heterogeneous, distributed and autonomic cloud platform for the iot. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 476–479, 2016.

[28] A. Coutinho, F. Greve, C. Prazeres, and J. Cardoso. Fogbed: A rapid-prototyping emulation environment for fog computing. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7, May 2018.

[29] Atakan Aral and Ivona Brandic. Dependency mining for service resilience at the edge. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 228–242. IEEE, 2018.

[30] I. Lujic, V. De Maio, and I. Brandic. Adaptive recovery of incomplete datasets for edge analytics. In *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, pages 1–10, 2018.