

Fog Computing
2026/27 - Autumn 2026 - UiO

Fog Simulators

Prof. Paulo Ferreira

paulofe@ifi.uio.no

UiO/IFI/PT

© Paulo Ferreira

A comparative analysis of simulators for the Cloud to Fog continuum. David Perez Abreu, Karima Velasquez, Marilia Curado, Edmundo Monteiro. Journal of Simulation Modelling Practice and Theory, Volume 101, 2020, Pages 1-27, ISSN 1569-190X, Elsevier. https://doi.org/10.1016/j.simpat.2019.102029.

Introduction (1/4)

Fog computing introduction and is characteristics



- **Fog computing** was envisioned to:
  - **minimize the request-response time of applications**
  - besides **providing computing resources and connectivity** to centralized services (i.e., Cloud) to the devices located at the Edge of the communication infrastructure
- To fulfill its purpose, the Fog paradigm has to **guarantee the following characteristics**:
  - location awareness and geographical distribution, low-latency, mobility support, heterogeneity data support, real-time interaction, federation and scalability of services/applications
- This inherently creates a **more complex scenario** to deploy application modules and services and to manage the resources in general
- **Novel mechanisms and procedures**, specifically designed for the Cloud to Fog continuum are needed to exploit the characteristics of the Edge of the communication infrastructure, such as low-latency

© Paulo Ferreira

Introduction (2/4)

Need to simulate Fog computing



- To evaluate the **correctness and performance** of these novel mechanisms and procedures, an **evaluation must be carried out**:
  - **testing in real-world working infrastructures or scenarios is not practical**, since one failure during the experiment could impact not only other tests, but also running services and applications
  - thus, **isolated environments are needed** before launching novel solutions to the real-world
  - creating a testbed for this kind of scenarios could carry **high costs** since there are many different devices and communication links involved, so this is not always a viable option
  - **simulation represents a solution** to evaluate the proposed mechanisms and procedures during an early stage in their development
  - however, **using simulation brings a new challenge in itself: selecting the right simulator**

© Paulo Ferreira

Introduction (3/4)

Several simulators for Cloud and Fog computing



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

© Paulo Ferreira

Introduction (4/4)

Analysis of simulators for Cloud and Fog computing



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

© Paulo Ferreira

Previous Work (1/3)

Comparison of simulators for Cloud and Fog computing



- **Simulators**:
  - CloudSim, CloudAnalyst, GreenCloud, MDCSim, iCanCloud, NetworkCloudSim, EMUSIM, GroudSim, DCSim, MR-CloudSim, SmartSim, and SimIC, Omnet++
  - the main limitation the mentioned above simulators is the **lack of mobility support**
  - some of them can currently be considered **deprecated since they have had no recent activity** (i.e., over five years), or
  - show **difficulties for downloading** the tool (e.g., broken links, missing files)
- **Comparison**:
  - comparison was limited to a conceptual analysis of simulators **only focused on Cloud**, and
  - several tools under analysis could be considered as deprecated, such as iCanCloud and MDCSim
- **CloudSim** is one of the most widely known with lots of extensions:
  - NetworkCloudSim, FederatedCloudSim, TeachCloud, FTCloudSim, WorkflowSim, ElasticSim, CloudAnalyst, CloudSimSDN

© Paulo Ferreira

Previous Work (2/3)

There are several simulators for Cloud and Fog computing



- **There are many others**:
  - but from all the reviewed work, it is noticeable that many of the analyzed simulation tools are deprecated or show some inconvenient accessing the source code or executable file
- Moreover, such works are **mainly focused on the evaluation of Cloud simulation tools**
- We address a set of Cloud/Fog continuum simulation tools or Cloud simulation tools where:
  - the **concept of Fog can be included in the experiments**, combining a conceptual and practical evaluation
- Sometimes we will use the **terms Fog and Edge interchangeably**

© Paulo Ferreira

Previous Work (3/3)



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

© Paulo Ferreira

Cloud/Fog Simulators

Model of simulators for Cloud and Fog computing and which simulators



- To **model the Cloud/Fog continuum**, up until the lower levels that represent the IoT devices:
  - the **simulator must include some characteristics** that would lead to more realistic results in such a complex environment
  - for example, **it must be possible to model features such as location awareness and low-latency, mobility support, as well as interoperability and scalability mechanisms**
  - a **proper Cloud/Fog simulator** should include support for these features
- The **simulators** addressed were **chosen based on**:
  - **fulfillment of the Fog features previously mentioned**, as well as
  - regarding **their acceptance** (measured by the citation number), and
  - the **support provided by the community** (measured by availability of tutorials, documentation, examples, and/or discussion groups)
  - the **acceptance and community support help** in the selection by allowing to discard tools that have not been adopted by the community
- The **simulators** are the following:
  - iFogSim, CloudSimSDN, YAFS, EmuFog, FogTorchπ, EdgeCloudSim

© Paulo Ferreira

iFogSim (1/4)

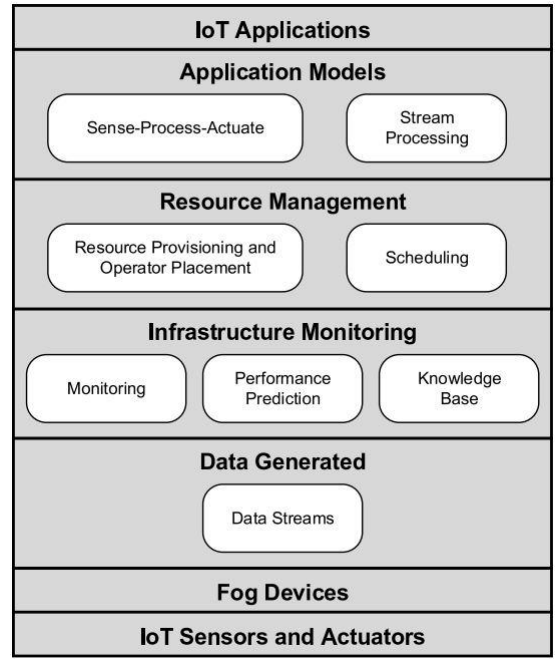


- **iFogSim** is a toolkit that allows:
  - **modeling IoT applications in Cloud/Fog environments**
  - is **based on Java**, is **open-source** and is an **extension of the very well-known CloudSim**
  - the evaluation of the impact of resource management techniques on:
    - network congestion, energy consumption, latency, and cost
- iFogSim physical **topologies can be defined using a GUI** (Graphical User Interface)
- **Saved topologies** are stored:
  - in JSON format, or
  - programmatically using Java
- An **application is built by a set of modules** (processing elements) that communicate among each other
- There is **no mobility support in iFogSim**:
  - however, there is another branch based on iFogSim focused on mobility support and Virtual Machine (VM) migrations, called MyiFogSim

© Paulo Ferreira

iFogSim (2/4)

- iFogSim architecture is based on **layers** that are responsible for specific tasks:
  - **at the bottom, there is the layer that comprises the IoT devices** (e.g., sensors and actuators) that interact with the real-world, representing the source or sink of data
- Such **devices** can be:
  - customized to simulate the effects of sending data (**sensors**) or
  - performing actions (**actuators**),
  - in order to model more sophisticated devices, such as smart cameras, environmental sensors, and mobile vehicles



© Paulo Ferreira

iFogSim (3/4)

- iFogSim also defines **Fog devices as any network element capable of hosting application modules**
- Such Fog devices are:
  - organized in a **hierarchical tree topology**,
  - where the **communication is only possible between a parent-child pair**
- The architecture offers **three main services**:
  1. **monitoring components**, to keep track of resource usage, power consumption, and availability of devices (i.e., sensors, actuators, and Fog devices);
  2. **resource management**, to guarantee that Quality of Service (QoS) requirements are met, placement and scheduler components identify the best candidates for hosting an application module and allocate the resources for said module; and
  3. **power monitoring**, to monitor the energy consumption during the simulation
- **Power consumption** receives a special treatment given that in Fog and IoT-based environments:
  - this becomes a **critical resource** since the devices are usually battery-constrained

© Paulo Ferreira

iFogSim (4/4)



- The **cost and energy models are inherited from CloudSim and adapted for iFogSim**
- iFogSim **supports VMs resource elasticity**
- iFogSim is **commonly used to**:
  - **assess allocation strategies** to measure their impact on **energy consumption and on latency**, and
  - also for **evaluating architectures** for Fog environments
- **By only offering tree-based topology support, it is not possible to model other types of topologies**

© Paulo Ferreira

CloudSimSDN (1/4)



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

© Paulo Ferreira

CloudSimSDN (2/4)



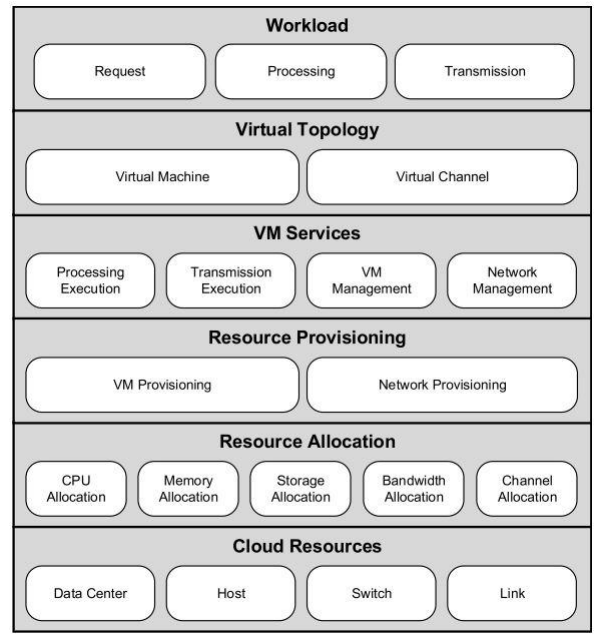
- Input **topology can be provided via a JSON file**, as program codes (written in Java), or via a GUI
- Despite CloudSimSDN **claims that it is possible to provide/build a topology using a GUI**:
  - the packages that support this functionality are **not available** in the main repository of the simulator
- During simulations:
  - the resources (i.e., CPU power, memory, and storage size) defined by the user **characterize the physical devices where VMs are placed**
  - the entities that can be modeled include **datacenters, physical hosts, links, VMs, VM schedulers, and workload schedulers**
  - a physical node can be an **SDN host or an SDN switch** (i.e., Core, Aggregation, Edge)
- There is **no mobility support** included in CloudSimSDN:
  - nevertheless, it is **possible to use migration mechanisms provided by the CloudSim simulation core**

© Paulo Ferreira

CloudSimSDN (3/4)



- The architecture rests over the **CloudSim** core
- It has some **layers** to manage:
  - **cloud resources** (i.e., datacenter, host switch, and link definitions), **resource allocation**, and
  - **provisioning, the VM services, virtual topology, and workloads**
- On top of all this, there are **layers** to:
  - support **user code and scenario description**



© Paulo Ferreira

CloudSimSDN (4/4)



- As with iFogSim, **the cost and energy models are adapted from CloudSim**:
  - it **supports VMs elasticity and migration**, besides the possibility to use or define federation models
- CloudSimSDN is **infrastructure-oriented**:
  - it is commonly used in **research works regarding datacenters and their performance** mainly when the scenarios involve Virtual Networks (VN) and SDN techniques, such as mechanisms for Virtual Network Embedding (VNE)
- Having **SDN support** makes it a good solution for infrastructure-based studies:
  - but it is **not as fairly suited for works based on applications**
- In CloudSimSDN, **applications are modeled via a set of tasks** (i.e., compute processing and network transmission) that will be processed in a VM:
  - CloudSimSDN assumes that **VMs use their network bandwidth entirely during their lifetime**, thus just one long packet transmission workload for each VM is given in the workload file
  - the aforementioned **assumption impacts the granularity level of the application**, restricting the communication process between the modules

© Paulo Ferreira

YAFS (1/3)

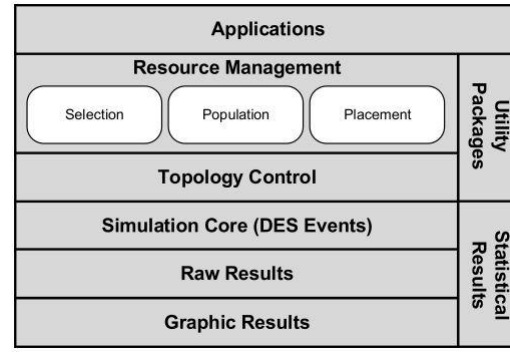


- **Yet Another Fog Simulator** (YAFS) is a simulator for Cloud/Fog:
  - main focus is the **performance evaluation of placement, scheduling, and routing strategies**
- Some of the metrics reported by YAFS include **network utilization, network delay, response time, and waiting time**:
  - **users can calculate other QoS metrics** since raw resulting data is reported in a Comma-Separated Values (CVS)-based log
- YAFS **uses Python as programming language**, is **open-source**, and can be downloaded for free under the **MIT license**
- **Topologies can be defined using a JSON-format file**:
  - enabling the use of other tools (i.e., BRITE, CAIDA) to generate the **desired scenario**
- **Applications are modeled as a set of modules that run services**, following the concept defined by iFogSim:
  - thus, a methodology is used to represent the applications where **nodes are modules and edges define the data exchanged among them**

© Paulo Ferreira

YAFS (2/3)

- With YAFS **it is possible to define a mobility pattern of the application model** along the communication infrastructure
- The architecture is defined by **six classes**
- When the **simulation is running**:
  - the **selection, placement, and population classes will create events dynamically** (e.g., remove or add a node or link in the communication infrastructure)
  - this **enables a more realistic simulation** by allowing the appearance of changes in the topology that could represent failures during the execution,
  - thus, **it is possible to evaluate more complex policies** able to handle these issues



- Additionally, **YAFS supports a set of graphics libraries** that allow the visualization of the simulations performed

© Paulo Ferreira

YAFS (3/3)

- It is possible to **define nodes' attributes**:
  - it will **enable the characterization of cost and energy models**
- There is also **support for scalability** by means of:
  - clustering definition of Cloud and Fog nodes
- YAFS uses a **simulator engine based on discrete events**:
  - since YAFS is a recent tool, there is **less available research using it for experiments**
  - however, **it has been used in works focused on IoT applications deployed in the Fog** to appraise the efficacy of placement policies
- **Graph animations** are also provided by generating plots of the network on each event and generating a video with the combination of said plots:
  - this **eases** the analysis of the results
  - furthermore, by providing the results in a raw format, it is possible to **customize the metrics** for a particular need
- Being released in 2019:
  - this simulator **has not been as used in research works as others more mature**, such as iFogSim

© Paulo Ferreira

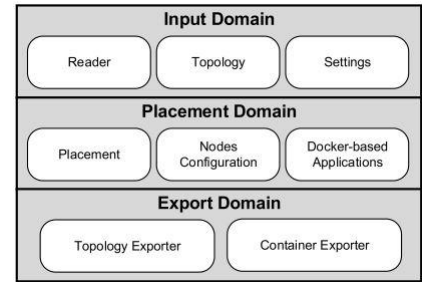
EmuFog (1/3)

- EmuFog is an emulation framework for Fog environments that **allows the simulation of Docker-based applications**
- EmuFog **builds up from MaxiNet, which in turn is based on Mininet**
- Respecting **reported metrics**, based on the fact that **EmuFog uses MaxiNet**:
  - it is **possible to keep track of events in each local node** as a log regarding CPU and memory consumption
  - however, there is **not a generic interface to deal with more global metrics**, such as response time
- The programming language used is **Java**, it is available under the **MIT license**, and it is **open-source** and **free to download**
- Regarding the topology, it is **possible to use** (JSON-format file):
  - **conventional topology generators** such as BRITE,
  - or also **importing real-world datasets** like from CAIDA
- The **topology can also be customized** to fulfill users' needs:
  - **users can define the placement of Fog computing nodes on top of the specified topology** according to different placement policies and also by specifying the Fog nodes' capabilities and expected workload
- **Mobility is not supported**, neither for clients nor for Fog nodes

© Paulo Ferreira

EmuFog (2/3)

- The EmuFog architecture is **divided into three domains**:
  1. the **input** domain, with the logical description of the experimental setting, including topologies and initialization values;
  2. the **placement** domain which is involved with the execution of the placement mechanisms; and
  3. the **export** domain, focused on generating the output files needed to run the experiments
- The **workflow begins with**:
  - the **topology generation** (either external or customized) to a topology enhancement by applying a placement strategy taking into consideration the Fog nodes' characteristics and the Edge occupancy
  - this, **together with the Dockerbased application model, is fed to the MaxiNet engine for emulation**
- It is **possible to use and tune cost models**, but **there is no energy model available**



- Regarding **federation and elasticity**, Emufog supports adaptive functions from the topology infrastructure:
  - however, it **does not define any mechanism to deal with computational nodes elasticity**

© Paulo Ferreira

EmuFog (3/3)



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

© Paulo Ferreira

FogTorchπ (1/3)

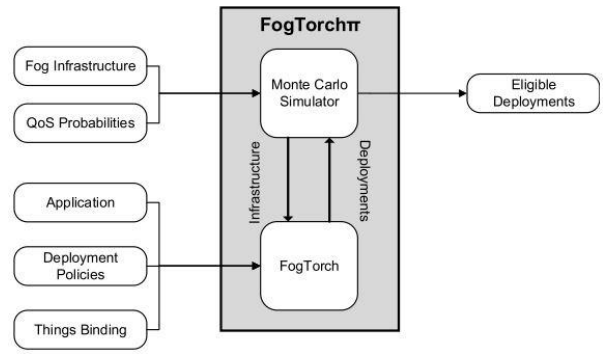


- **FogTorchπ** is a tool based on a model which aims to support QoS-aware deployment of IoT applications in Fog environments:
  - **implements variations in communication links**, which are used as inputs
- **Results are aggregated by computing two metrics** for each obtained deployment:
  - **RQoS assurance** (percentage of runs that generated that deployment), and
  - **Fog resource consumption** (aggregated average percentage of consumed RAM and storage)
- The tool uses **Java as programming language**, is published under the **MIT license**, and is **free to download**
- The **Fog infrastructure consists of IoT devices, Fog nodes and Cloud datacenters, and the communication links** that connect them
- The model is abstract enough to allow the definition of **arbitrary topologies**
- **Applications are defined as a triplet** composed by the **software modules**, the interactions between them, and **users' requests**
- **FogTorchπ does not support mobility in its simulations**

© Paulo Ferreira

FogTorchπ (2/3)

- FogTorchπ uses **latency and bandwidth**:
  - **differentiating between upload and download per link** as QoS attributes
- For each run, a particular **Fog infrastructure is provided as input**:
  - choosing a QoS profile for each communication link
- Finally, the **output is a set of eligible deployments**



© Paulo Ferreira

FogTorchπ (3/3)

- Applications are viewed as a set of independently deployable components that work together and are **bound to QoS constraints**
- It is possible to **define costs and energy models** and use them during the simulation process
- Concerning **scalability**:
  - **FogTorchπ assumes that Cloud and Fog nodes have unbounded hardware capabilities** which denote that real **scalability and elasticity support is not contemplated in the simulator**
- This tool has **previously been used in works** related to offloading in Cloud/Fog environments
- For simple deployment evaluation, FogTorchπ could be a good option since it is focused on this task:
  - nevertheless, **FogTorchπ is referred to by its authors as a prototype tool**, thus it might not have enough maturity

© Paulo Ferreira

EdgeCloudSim (1/4)



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

© Paulo Ferreira

EdgeCloudSim (2/4)

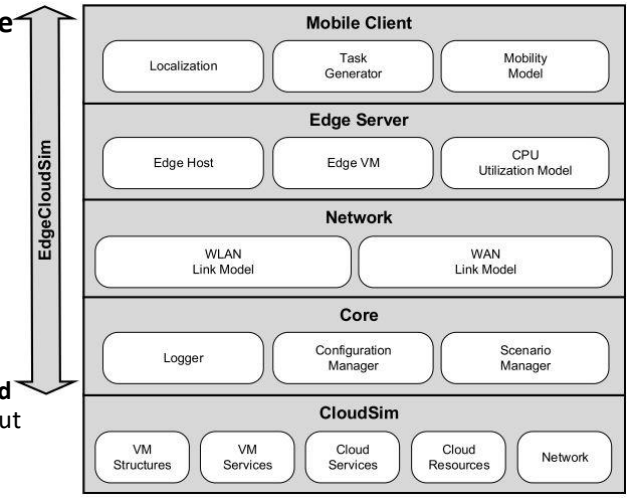


- Regarding the **topology**:
  - EdgeCloudSim allows to work with **arbitrary topologies**, modeling Wireless LAN (WLAN), LAN, and WAN
  - it is possible to **define network link delays, which can vary during the simulation**
- The **application data model is handled by a specific module called Task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

© Paulo Ferreira

EdgeCloudSim (3/4)

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



© Paulo Ferreira

EdgeCloudSim (4/4)

- There is a possibility to use a cost model, but **no energy model is available**
- There is also a **lack of migration support**, and it is **only possible to group nodes that belong to the same tier**:
  - thus rendering impossible to group nodes from the entire Cloud/Fog spectrum
- **EdgeCloudSim** has been used to evaluate:
  - **orchestration** in the Edge,
  - **VM allocation policies and proposed Edge architectures**, as well
  - as **fault-tolerance** mechanisms
- EdgeCloudSim provides the **possibility to define mobility, network link, and edge server models to measure different aspects of Cloud/Fog environments**:
  - making it **suitable to evaluate the behavior of IoT and Edge applications/services**
  - **however, the simulator uses a single server queue model to calculate the network delay**, restricting the behaviour of network access technologies to a simple model which impacts its accuracy

© Paulo Ferreira

Summary of Simulators Characteristics (1/13)

- This table summarizes the non-technical features of the simulators

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

- **Latest release** refers to the year in which the latest release of the tool was published
- **First release** corresponds to the year of the launch of the first version of the tool (as listed on its GitHub repository), in order to have an idea on its maturity
- **Date of publication** indicates the year on which the paper that introduces the tool was published
- **Community support** measures the backing of the creators of the tool and the community in terms of availability of examples, tutorials/documentation, and/or existence of community groups:
  - the Community support is measured in High (all three factors are present), Moderate (two factors present), and Low (only one-factor present)

© Paulo Ferreira

Summary of Simulators Characteristics (2/13)

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

- **Citations** accounts for the number of hits obtained in Google Scholar when searching the name of each tool
- **Citations** and **Latest Release** are accounted as to July 2019
- It is noticeable that:
  - **CloudSimSDN, YAFS, EmuFog, and EdgeCloudSim are the most recently updated projects**, while
  - **FogTorchπ and iFogSim are not so far behind** (two years since the latest update)
- The **most mature tools** (according to the Release Date and Date of Publication) are:
  - **iFogSim, CloudSimSDN, and FogTorchπ**, with **YAFS being the most recent**
- This is also reflected in the Citation category, where **EmuFog and YAFS have significantly fewer citations**:
  - this could be highly influenced by the lack of maturity of the tools and their recent creation

© Paulo Ferreira

Summary of Simulators Characteristics (3/13)



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

- Regarding the **Community support, there is no official discussion group for any of the simulators**:
  - however, **there are a couple of groups related to CloudSim**,
  - which is the **base simulator from where iFogSim, CloudSimSDN, and EdgeCloudSim build up**, and
  - **some threads in these groups** are dedicated to these simulators
- **Almost all the simulators include them in their source code, except for EmuFog**

© Paulo Ferreira

Summary of Simulators Characteristics (4/13)



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

- About the tutorials or documentation:
  - **iFogSim and FogTorchπ do not offer any documentation**
  - for **iFogSim just a couple of examples are provided with the source code**
  - **CloudSimSDN, YAFS, EmuFog, and EdgeCloudSim include in their repositories a guide for installation and examples on how to use them**
  - furthermore, **YAFS also has a more detailed document covering the basic concepts related to the simulator**

© Paulo Ferreira

Summary of Simulators Characteristics (5/13)

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

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

© Paulo Ferreira

Summary of Simulators Characteristics (6/13)

- This table summarizes the technical features of the simulators

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

- The row **Language** means the programming language used to code the experiments
- The **Topologies** category refers to the types of network topologies that are supported by the tool
- The presence of the feature **Fault Injection** indicates if the tool is able to simulate random failures in the topology or if dynamic topologies are supported
- **Application Model** refers to the representation of an application in the context of the tool
- **Mobility** indicates whether the simulator has support for this feature

© Paulo Ferreira

Summary of Simulators Characteristics (7/13)

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

- The category **Cost** and **Energy Model** indicates if there is already a model (for cost and/or energy) implemented or the possibility for it to be added by the user via built-in features
- The row named **Federation and Scalability** shows if there is support to define federations and scale upwards or downwards the resources used by the VMs or building clusters (i.e., grouping or ungrouping computational nodes) in the Cloud/Fog environment
- Since **all evaluated tools include support for Fog environments (except EmuFog, which uses emulation),** this feature are not included in the table

© Paulo Ferreira

Summary of Simulators Characteristics (8/13)

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

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

© Paulo Ferreira

Summary of Simulators Characteristics (9/13)

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

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

© Paulo Ferreira

Summary of Simulators Characteristics (10/13)

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

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

© Paulo Ferreira

Summary of Simulators Characteristics (11/13)

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

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

© Paulo Ferreira

Summary of Simulators Characteristics (12/13)

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

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

© Paulo Ferreira

Summary of Simulators Characteristics (13/13)

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

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

© Paulo Ferreira

Summary of Simulators Metrics (1/2)

- In the following table, the presence of a checkmark indicates that the simulator reports that metric, while a dash says otherwise

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

© Paulo Ferreira

Summary of Simulators Metrics (2/2)

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

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

© Paulo Ferreira

Discussion (1/2)

- In the case of **Fog computing**:
  - there are several simulators, but **each one offers different features that adapt to specific evaluations**
- **EmuFog is the only emulation tool**, being the other five simulation tools
- **Most of the tools are Java-based**, being YAFS the only one based on Python
- **iFogSim turned out to be the only one that restricts the topology supported to a tree**, while the rest accept an arbitrary topology
- The **modular application model is the most adopted one** (iFogSim, YAFS, FogTorchπ, and EdgeCloudSim)
- According to the **citation count**:
  - **iFogSim is the most used Cloud/Fog simulation tool**,
  - while **EmuFog is the least used following the same criteria**
- **3 simulators were chosen among the 6 presented**:
  - **iFogSim, CloudSimDNS, and YAFS**
  - a **practical assessment was carried out for such simulators**

© Paulo Ferreira

Discussion (2/2)



- YAFS:
  - was the **most time-consuming simulator**
  - was the simulator that **consumed the lesser amount of resources**
  - **more fitted for application performance evaluation**
  - provides **beneficial tools to evaluate the impact of population and placement of application modules in a communication infrastructure**
- iFogSim:
  - the simulator that **consumed more resources**
  - **more fitted for application performance evaluation**
  - offers **better support regarding the simulation of algorithms and mechanisms that require measurement of resource consumption inside Virtual Machines**
- CloudSimSDN:
  - is **more oriented for assessing of infrastructure**, making it suitable to validate Virtual Network Embedding mechanisms, as well as, to perform experiments related to SDN environments

© Paulo Ferreira

Conclusion



- Simulation has **proven to be an important tool** to evaluate novel solutions in networking and communication environments
- In the case of Fog computing, **there are several simulators**, but each one offers different features that adapt to specific evaluations
- **EmuFog is the only emulation tool**, being the other five simulation tools
- **Most of the tools are Java-based**, being YAFS the only one based on Python
- **iFogSim turned out to be the only one that restricts the topology supported to a tree**, while the rest accept an arbitrary topology
- The **modular application model is the most adopted one** (iFogSim, YAFS, FogTorchπ, and EdgeCloudSim)
- According to the **citation count, iFogSim is the most used Cloud/Fog simulation tool, while EmuFog is the least used following the same criteria**

© Paulo Ferreira