Modelling and Simulating the Computing Continuum with ENIGMA

Elias Del-Pozo-Puñal
Departamento de Arquitectura y Tecnología de Sistemas Informáticos
Universidad Politécnica de Madrid
Madrid, Spain
e.delpozo@upm.es

Felix Garcia-Carballeira, Jesus Carretero
Computer Science and Engineering Department
Universidad Carlos III de Madrid
Leganés, Spain
{fgcarbal,jcarrete}@inf.uc3m.es

***Abstract*—The explosive growth of the Internet of Things (IoT) paradigm has driven the emergence of highly complex, multi-layered network architectures encompassing Cloud, Edge, and Fog Computing. This Computing Continuum reduces latency and distributes computational load efficiently across the network. However, before physically deploying these costly infrastructures, researchers and engineers require robust simulation platforms to model dynamic environments and analyze key behavioral metrics (e.g., power consumption, CPU usage, and network bandwidth). This paper focuses on addressing the critical challenge of scalable simulation within the Computing Continuum. In this paper, we introduce ENIGMA, a highly scalable simulation platform, and we specifically evaluate environments that integrate mobile devices. The results of the evaluation show that ENIGMA is 10x faster than YAFS and scales far beyond IFogSim2.**

***Index Terms*—Computing Continuum, Internet of Things, Simulation.**

I. Introduction

The explosive growth of the Internet of Things (IoT) paradigm has driven the emergence of highly complex, multi-layered network architectures encompassing Cloud, Edge, and Fog Computing. This Computing Continuum reduces latency and distributes computational load efficiently across the network. As the deployment of 5G/6G networks and massive IoT ecosystems accelerates, the Computing Continuum is rapidly becoming the standard architecture for modern digital services.

However, before physically deploying these costly infrastructures, researchers and engineers require robust simulation platforms to model dynamic environments and analyze key behavioral metrics (e.g., power consumption, CPU usage, and network bandwidth).

The scope of this paper focuses on addressing the critical challenge of scalable simulation within the Computing Continuum, specifically highlighting environments that integrate mobile devices. In this paper, we introduce **ENIGMA**, a highly scalable simulation platform. ENIGMA provides immense practical value by directly addressing the critical challenges researchers face today in designing massive IoT ecosystems: achieving massive scalability for IoT deployments, natively modelling mobile devices, and offering a graphical mapping API for location-based analysis without compromising simulation performance.

This work has been funded by State Research Agency under the project ADAMAS (Ref. PID2025-167966NB-I00) .

ENIGMA [1] allows testing new distributed devices and algorithms before physical testbeds, which are prohibitively expensive, time-consuming, and difficult to reproduce.

Main contributions of this paper are:

- A discussion of the architectural challenges of distributing computation across the network, the role of mobile IoT devices, and why simulation is a mandatory step before the physical deployment of new routing, offloading, or resource-allocation algorithms.
- The ENIGMA Simulator architecture. We detail ENIGMA's underlying architecture and analyze how it efficiently simulates massive infrastructures while extracting metrics such as application execution time, energy consumption, and CPU/network usage.
- We will present ENIGMA's native compatibility with mobile devices and show the results of the simulation to visualize node mobility on graphical maps. Different use cases will be presented.

The rest of this paper is organized as follows. Section II reviews related work. Section III presents challenges in the Computing Continuum. Section IV presents the design of the ENIGMA simulator. Section V describes the case study and experimental results. Finally, Section VI concludes the paper.

II. Background and Related Work

The most effective method for evaluating the performance of any service management policy is to deploy it in a real-world environment. However, deploying real-world Edge and Fog Computing environments at such a scale is a complex task. They incorporate many IoT devices and compute nodes, in both staggered and flat order, with large amounts of batch or streaming data and distributed applications. However, the lack of global Edge and Fog service providers that provide tools for infrastructure management, such as commercial Cloud platforms like Microsoft Azure and Amazon AWS, leaves researchers forced to request Edge and Fog Computing environments on their own to perform costly policy assessments [11].

The complexity of the Computing Continuum has motivated the development of simulators for evaluating resource man-

979-8-3195-0167-7/26/$31.00 ©2026 IEEE

agement strategies before deployment [2]. These simulators provide users with valuable insight into the potential costs associated with deploying an environment or running specific jobs, including power consumption, bandwidth usage, and data center congestion. CloudSim [3] and its edge and fog extensions, such as iFogSim [4] and EdgeCloudSim [5], are widely used to optimize task placement, measure network congestion, and estimate energy consumption. Other frameworks focus on SLA and QoS constraints: RECAP [6] evaluates trade-offs between energy consumption and operational costs, FogTorch [7] analyzes deployment policies under QoS requirements, and YAFS (Yet Another Fog Simulator) [8] supports complex IoT topologies, including node failures and mobility-aware placement strategies.

Other simulators target workflows and stream processing workloads. WRENCH [9] provides abstractions for computation, storage, and network modelling in distributed workflows. IoTSim-Stream [10] extends CloudSim for IoT stream processing in multi-cloud environments and uses an XML-based graph model to represent application patterns. Finally, PureEdgeSim [12], provides a simulation framework for performance evaluation of cloud, edge, and mist computing environments.

Therefore, simulation platforms become essential tools for understanding and evaluating the behavior of such distributed architectures before real deployment. These tools abstract hardware resources, such as CPU, GPU, memory, and storage, as well as network topologies, allowing researchers to prototype algorithms and applications across diverse environments.

As shown in the Section, existing simulators vary significantly in terms of supported features, scalability, extensibility, and representational fidelity. In addition, they allow resource consumption to be reduced by implementing the desired topology in the event of a failure, thus maximizing profits [13]. However, they often fail to adequately model the protocol-level interactions and data lifecycle management across the entire Continuum. Moreover, few simulators allow the flexible integration of new communication technologies or hybrid orchestration strategies.

III. Understanding the Computing Continuum

The progressive digitization of the physical environment has been marked by the proliferation of connected devices, distributed systems, and services deployed at different levels of abstraction and proximity to data sources.

Edge Computing [15] is a paradigm that brings computing and information storage closer to the smart sensors, end users, mobile devices, or IoT devices themselves. Thus, one of the objectives achieved by this paradigm is to minimize the physical distance, response time, and bandwidth used during the transmission of information between both ends.

Fog Computing focuses more on the decentralization of Cloud Computing, located between data-producing devices and the Cloud itself. This paradigm aims to reduce data processing in the Cloud and move it closer to the Edge of the network, as close to the devices as possible, thereby offering faster services and resources to end users. In general terms, Fog Computing is an extension of Cloud Computing, but with the particularity of being physically closer to the devices. This intermediate layer provides computational resources distributed throughout the network and can be found in elements such as gateways, routers, or local servers, commonly referred to as fog nodes. These enable data generated near them to be processed without requiring direct access to the Cloud [16], thus providing scalability, reduced latency, and reduced data congestion.

As can be seen, both Edge and Fog Computing are very similar paradigms in terms of functionality. Both deal with the computation of tasks or data received from sensors and end devices within their components, aiming to reduce the computational load on servers and the network. Both paradigms can help organizations reduce network load and increase the response rate to customers.

Cloud Computing has traditionally represented the backbone of large-scale data processing and storage. As the final tier in the Computing Continuum, the Cloud offers virtually unlimited computational resources, scalability, and centralized management. It is widely adopted for handling intensive workloads, such as large-scale analytics, machine learning training, and long-term data archiving. However, while the Cloud remains a critical component, its limitations become apparent when addressing latency-sensitive or geographically distributed IoT applications [17]. In this context, Cloud Computing operates in complementarity with Edge and Fog layers, offloading non-time-critical tasks and supporting the overall architecture with orchestration, long-term storage, and global data fusion capabilities. It also facilitates ubiquitous, on-demand access to a shared pool of configurable resources, which can be quickly released with minimal management effort on the part of the service provider.

This technological evolution has led to a staggered configuration of computational layers–IoT, Edge, Fog, and Cloud–which together form what is known as the Computing Continuum.

The increasing interdependence of IoT, Edge, Fog, and Cloud Computing has given rise to the Computing Continuum paradigm, which is a conceptual and operational model that promotes the seamless integration of all computing layers, from the physical Edge to centralized Cloud infrastructures. Rather than viewing each tier in isolation, the Continuum treats them as cooperating entities within a distributed, hierarchical ecosystem, as discussed in [18]. It enables data processing closer to the data sources, reducing both latency and bandwidth consumption. Moreover, it facilitates the creation of decentralized systems where multiple environments collaborate to process and share data seamlessly. This results in a continuous data flow across edge, fog, and cloud layers [14].

Figure 1 illustrates a hierarchical Computing Continuum architecture that includes the Edge Computing layer directly interacting with IoT devices, such as sensors and actuators, responsible for generating and collecting data at the network's periphery. The figure shows the integration of Cloud Computing, Fog Computing, and Edge Computing, showing data

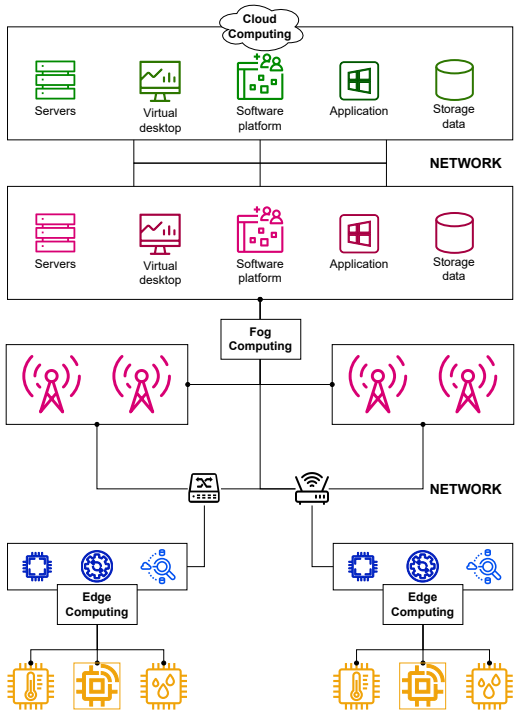


Fig. 1: Diagram illustrating the integration of the Computing Continuum layers.

flow from physical sensors to centralized cloud infrastructure. As shown in the figure, the Cloud layer sits at the top of the hierarchical architecture, offering centralized resources and services that interact with the underlying Fog and Edge layers.

However, the field of Computing Continuum research has been conducted in an ad-hoc manner, lacking a solid theoretical and engineering basis for resource and performance management. This has led developers to frequently resort to ad-hoc reasoning approaches to address performance and resource utilization problems. Still, many challenges are present in the Computing Continuum field, such as infrastructure and resource management across the Continuum, resource allocation and scheduling, orchestration and task placement, or data placement and storage management. In this context, simulation platforms, such as ENIGMA, become essential tools for understanding and evaluating the behavior of such distributed architectures before real deployment.

IV. ENIGMA Design and Architecture

ENIGMA[1] [21], [22] reproduces the behavior of applications distributed along the computing continuum [19], [20] before any real deployment takes place, reporting CPU usage, energy consumption, network bandwidth and execution time. Two requirements drove its design: *scalability*, understood as the ability to add components and simulate large infrastructures without degrading the simulator itself, and *mobility*, since a large fraction of IoT devices can be geolocated and their position conditions where tasks are placed.

The tool is built on top of SimGrid [24], [25][2], which acts as a platform for transmitting data packets among components represented as tasks. SimGrid was chosen because it offers two levels of interaction - one for users who look for simplicity and do not want to know the internals of the simulator, another for users who prefer to customize the infrastructure by modifying or adding elements - and because it provides validated network models with high performance [26]. ENIGMA is a *centralized* simulator: the whole simulation runs on a single machine, as SimGrid executes the entire simulation logic locally.

A. Layered Architecture

Components are structured in three layers interconnected through the network, as shown in Figure 2. The Cloud layer is the top tier, designed for high-processing tasks and centralized storage; its nodes process the jobs sent from below and may store information for later access. The Fog layer acts as an intermediary that stores and processes data closer to the source, reducing latency and alleviating the burden on the Cloud. The IoT-Edge layer contains the devices that generate tasks and periodically offload them.

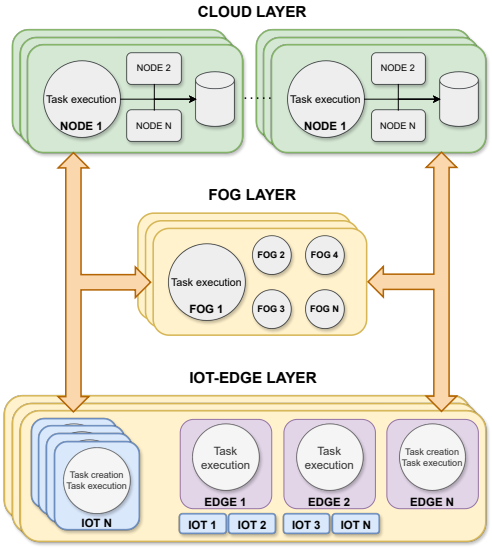


Fig. 2: ENIGMA's layered architecture.

The flexibility of the last layer is its key characteristic. In its simplest form, it contains only IoT devices that send tasks directly to the Fog or the Cloud. Alternatively, those devices may offload to nearby Edge components that fully or partially process the incoming tasks. ENIGMA also supports configurations with Edge components and no traditional IoT devices, where these elements generate or receive tasks and execute them locally, forwarding them upwards only when their complexity demands it. This makes the layer modular and

[1]https://github.com/ENIGMA-Sim/ENIGMA

[2]https://simgrid.org/

adaptable to deployments ranging from centralized to highly distributed scenarios.

Every configurable element exposes a few key parameters: the number of elements per group, the processor speed in FLOPS, the number of cores, and the latency and bandwidth of the links. Optionally, ENIGMA computes the watts consumed by each component [27]. Storage is not modeled in the current version, as the focus is placed on computation and communication; recent SimGrid releases do support storage metrics, which opens the door to including disk throughput or capacity limits in future versions [28].

B. Describing a Platform

The user fills in a CSV file that a Python script transforms into the XML platform file required by SimGrid, containing the processing capacity, the connectivity characteristics, and the remaining per-node parameters. This is flexible, but estimating some values requires prior knowledge of the modeled hardware. The theoretical peak performance of a CPU can be approximated as:

$$FLOPS = Cores \times ClockFreq \times \frac{FLOPs}{Cycle} \qquad (1)$$

where $FLOPs/Cycle$ depends on the SIMD width and FPU capabilities of the architecture. In practice users take these figures from documentation or from benchmarks of comparable devices: published measurements of single-board computers [29], [30] report about 0.4 GFLOPS for a Raspberry Pi 1 and up to 9.7 GFLOPS for a Raspberry Pi 4, which is a reasonable reference when configuring Edge nodes.

Listing 1 shows an excerpt of the resulting file for a Cloud cluster, a Fog cluster, and a cluster of Edge devices. Each cluster describes one layer through its prefix, its numerical range, its speed and core count, and its internal bandwidth and latency, and is reached through a router. Once the elements exist, the inter-layer links and the routes that traverse them are declared [31]; in the example, both hops provide 1 Gbps with 10 ms of latency.

Listing 1: Excerpt of the XML platform file generated by ENIGMA.

```
<platform version="4.1">
 <AS id="AS0" routing="Full">
  <cluster id="cloud0" prefix="cloud-0-" radical="0-9"
    speed="20Gf,10Gf,5Gf" core="12" bw="2Gbps" lat="1ms"
    bb_bw="2Gbps" bb_lat="1ms" router_id="Cloud0"/>
  <cluster id="fog0" prefix="fog-0-" radical="0-4"
    speed="16Gf,8Gf,4Gf" core="8" bw="1Gbps" lat="2ms"
    bb_bw="1Gbps" bb_lat="2ms" router_id="Fog0"/>
  <cluster id="iot0" prefix="iot-0-" radical="0-49"
    speed="9400Mf,3000Mf,2000Mf" core="4" bw="1Gbps"
    lat="2ms" bb_bw="1Gbps" bb_lat="2ms" router_id="IoT0"
        />
  <link id="linkiot0fog0" latency="10ms" bandwidth="1Gbps
       "/>
  <link id="linkfog0cloud0" latency="10ms" bandwidth="1
       Gbps"/>
  <ASroute src="iot0" dst="fog0" gw_src="IoT0" gw_dst="
       Fog0">
    <link_ctn id="linkiot0fog0"/>
  </ASroute>
  ...
 </AS>
</platform>
```

SimGrid supports three communication models built on an abstraction of TCP behaviour [32]: a raw model that ignores congestion and bandwidth sharing, fast but unrealistic; CM02 [33], which adds constant overheads derived from empirical measurements but still ignores contention; and LV08 [26], [34], the default, which considers latency and bandwidth sharing among concurrent flows using validated analytical formulas. ENIGMA adopts LV08, as it offers a realistic yet accessible representation of the network and keeps results interpretable without knowledge of low-level network dynamics. Switching models, together with the recent SimGrid plugins for link failures and battery-aware devices, is left for future versions.

C. Execution and Application Models

The algorithms of Figure 3 are the base on which ENIGMA builds its simulations and cover the most general cases; the code base is fully modifiable, so users may add operations for more complex scenarios without any restriction from the simulator.

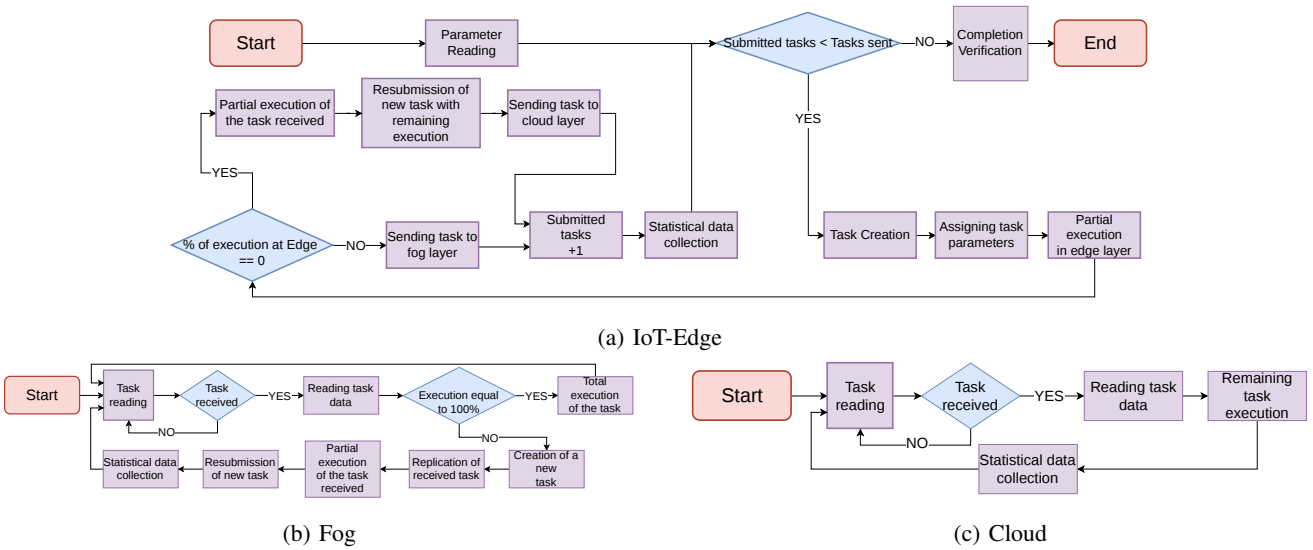
At the IoT-Edge layer (Figure 3a), while the number of submitted tasks is below the target, new tasks are created and the fraction that can be executed locally is evaluated: if it is zero the whole task is sent upwards, otherwise the partial execution is performed and the remaining work becomes a new task forwarded to the Fog or the Cloud. Statistics are collected at every iteration. The Fog layer (Figure 3b) runs an endless loop that, on receiving a task, either executes it entirely or performs a partial execution and creates a new task with the replicated data and the pending work, forwarding it to another Fog node or to the Cloud. The Cloud layer (Figure 3c) reads the data of each incoming task, executes the remaining work, and collects the statistics.

These flows can be simplified or extended: devices may merely collect and forward data, Fog nodes may redirect tasks to nodes closer to their origin or store data locally, and the Cloud may forward work to another geographic region for cost or data-locality reasons. Deciding how each element behaves at every instant is one of the main advantages of ENIGMA.

Two families of applications are supported, and the choice determines how the simulation ends. *Data flow* applications manage the volumes of data continuously generated by sensors: each block carries a processing requirement in FLOPS that every node compares against its own capacity, and once processed the data may be stored, replicated or split following a divide-and-conquer strategy that spreads the load over several processing points. The information never returns to its origin, so execution ends at the last node. *Service processing-oriented* applications instead send a response back to the emitting device once the work is complete, closing the communication cycle.

V. Experimental Results

This section analyses its scalability against YAFS [35] and iFogSim2 [36], and evaluates the mobility model on a real use case. All the experiments were executed on a machine with



(a) IoT-Edge

(b) Fog

(c) Cloud

Fig. 3: Execution algorithm of each layer.

an Intel Xeon Silver 4114 at 2.2 GHz with 20 cores, 125 GiB of DDR4 RAM, Gigabit Ethernet, a 7 TiB HDD and Ubuntu 23.04.

A. Scalability Analysis

The first experiment increases the number of participating devices from 100 to 1,000 sensors that collect information, send it to an intermediate Fog layer and finally store it in a Cloud node. Among the tools reviewed in Section II, only YAFS and iFogSim2 could be compared fairly: the rest either cannot handle infrastructures with many devices or consume considerably more resources. For iFogSim2 we used its JSON input files; for YAFS a custom Python script was needed to define all components, connections and services; and ENIGMA supported the equivalent simulation through the SimGrid simulation tool.

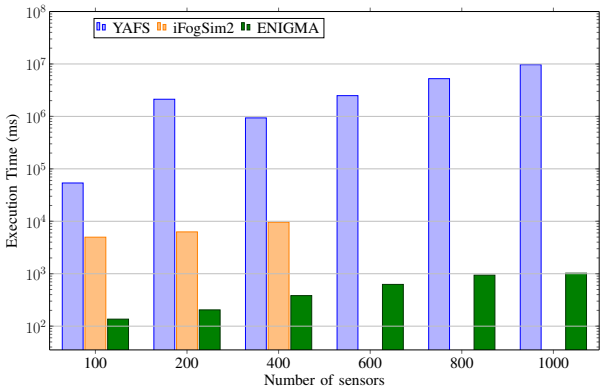


Fig. 4: Evaluation of the execution time of YAFS, iFogSim2, and ENIGMA as the number of sensors in the simulation increases. The graph shows the scalability behavior of each simulator in milliseconds (ms), using **logarithmic scale**.

Figure 4 shows the execution time on a logarithmic scale. YAFS needs approximately 53 seconds for 100 sensors and around 160 minutes for 1,000, an exponential growth that reflects a limited ability to handle large infrastructures. ENIGMA increases far more gently, from 136 ms with 100 devices to 1,034 ms with 1,000, while iFogSim2 ranges between 5 and 10 seconds.

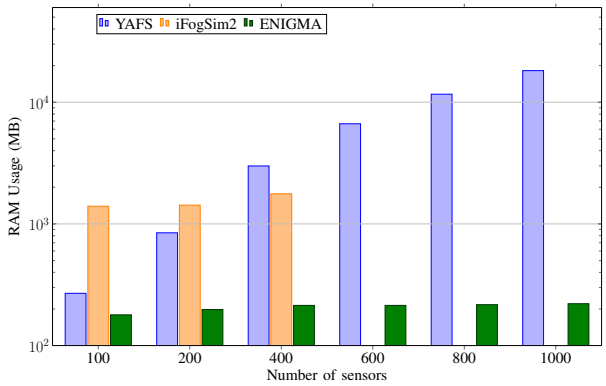


Fig. 5: Evaluation of RAM consumption between YAFS, iFogSim2, and ENIGMA as the number of simulated sensors increases. The Y-axis is in **logarithmic scale**.

Memory usage, in Figure 5, follows the same pattern. ENIGMA remains remarkably stable, from approximately 180 MB with 100 devices to 221 MB with 1,000. iFogSim2 starts at 1,398 MB and grows to 1,768 MB at 400 sensors; beyond that point it can still initialise the infrastructure but fails to complete the execution, which is why its results are excluded from both figures for larger scenarios. YAFS starts from a moderate 260 MB but reaches a peak of about 18 GB in the largest case, revealing severe scalability issues.

This stability stems from the architecture of SimGrid, whose event-driven model processes discrete events instead

of instantiating heavy-weight objects per entity or message: components are represented as lightweight abstractions - processes and hosts - and internal structures are reused efficiently. ENIGMA therefore not only matches iFogSim2 in efficiency but surpasses it in scalability, making it suitable for large-scale IoT simulations without compromising execution time or resources.

B. Evaluation of the Mobility Model

The second experiment evaluates the simulator when sensors update their location while sending data, using buses of the city of Madrid as a case study. An API was developed to incorporate the mobility and location of IoT devices within the simulator, allowing evaluation of how proximity and movement affect the performance of the system.

The first test deploys 500 devices - about 25% of the operational fleet of the EMT of Madrid, which exceeds 2,000 buses - moving for 10,000 seconds and reporting to a Fog layer of five substations placed at the north, south, east, west and center of the city, which act as intermediate processing nodes. Before each transmission, a device computes its distance to the five substations and selects the nearest one, modelling the proximity-driven behavior of Edge devices. A previous Python script generated the initial coordinates, constrained to the boundaries of the city. The substations collect the updated position every 600 seconds, an interval long enough for the sensors to cover a significant portion of the map, so ENIGMA produces 16 snapshots over the whole simulation. Feeding them to Folium yields the interactive map of Figure 6, where each point is the location of a bus at a given timestamp and a time slider animates the movement, confirming that the generated coordinates and mobility patterns were correctly integrated.

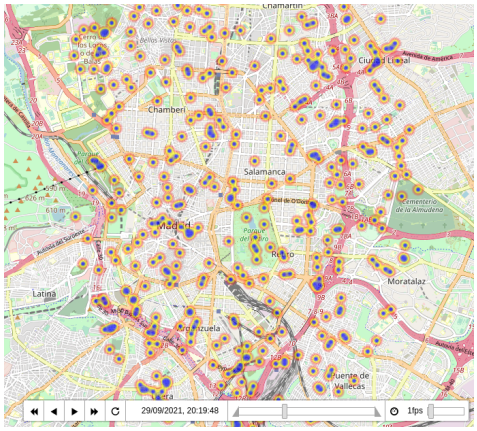


Fig. 6: Positions and mobility of 500 IoT-enabled buses interacting with fixed substations throughout Madrid.

The second test reproduces nine real EMT routes converging on three common destinations, a number that covers different areas of the city while keeping the results manageable. Each bus carries sensors that record the ambient temperature every second, and the passenger count together with the GPS position every 10 seconds [23]. Data are sent to the nearest Fog node at each transmission; the base stations, whose positions are fixed before the simulation starts, organize and partially process the information before relaying it to the central server, which consolidates the GPS data every 30 seconds to avoid overloading the map while keeping the routes accurate enough.

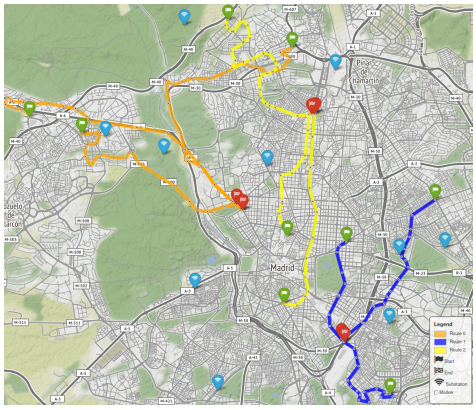


Fig. 7: Simulated bus routes, colored by destination. Markers indicate the origin (green), the destination (red) and the intermediate stations (blue), and expose the statistics recorded at each point.

Results are stored in a CSV file that a user-provided script turns into the map of Figure 7, where three colors identify the destinations and the markers distinguish origins, destinations, and intermediate stations. Selecting any intermediate point displays the data associated with it - position, temperature and number of passengers - so the analysis of the information collected along the route is considerably simplified. Overall, the mobility API integrates location awareness into IoT devices deployed over Cloud, Edge or Fog infrastructures, and its combination with Folium turns the output of the simulation into an intuitive tool for exploring mobility patterns.

VI. Conclusion

In this paper, we have presented ENIGMA, a simulation tool that enables the modelling and analysis of Computing Computing environments, evaluating cost, performance, and resource usage in IoT-based scenarios.

To evaluate its performance, it was compared with state-of-the-art simulators in the field of IoT and data stream-based applications. A scalability study was performed, analyzing how they respond as the number of IoT devices increases. The results of the evaluation show that ENIGMA is 10x faster than YAFS and scales far beyond IfogSim2. A real-world deployment was also tested using buses of the city of Madrid as a case study. The results show that ENIGMA is valid for simulating scenarios with mobility, those where sensors update their location while sending data.

After this work, some directions that will be addressed in future work are adding support for new communication protocols, providing multicloud support in parallel, and incorporating latency modelling in ENIGMA to reflect realistic connectivity delays in mobile deployments.

REFERENCES

[1] E. Del-Pozo-Puñal, F. García-Carballeira, and D. Camarmas-Alonso, "A scalable simulator for cloud, fog and edge computing platforms with mobility support", Future Generation Computer Systems, vol. 144, pp. 117–130, 2023.

[2] M. Bendechache, S. Svorobej, P. Takako Endo, and T. Lynn, "Simulating resource management across the cloud-to-thing continuum: A survey and future directions," Future Internet, vol. 12, no. 6, p. 95, 2020.

[3] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, "Cloudsim: a toolkit for modelling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and experience, vol. 41, no. 1, pp. 23–50, 2011.

[4] H. Gupta, A. Vahid Dastjerdi, S. Ghosh, and R. Buyya, "ifogsim: A toolkit for modelling and simulation of resource management techniques in the internet of things, edge and fog computing environments," Software: Practice and Experience, vol. 47, no. 9, pp. 1275–1296, 2017.

[5] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," Transactions on Emerging Telecommunications Technologies, vol. 29, no. 11, p. e3493, 2018.

[6] J. Byrne, S. Svorobej, A. Gourinovitch, D. M. Elango, P. Liston, P. J. Byrne, and T. Lynn, "Recap simulator: Simulation of cloud/edge/fog computing scenarios," in 2017 Winter Simulation Conference (WSC). IEEE, 2017, pp. 4568–4569.

[7] 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), 2017, pp. 105–114.

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

[9] H. Casanova, S. Pandey, J. Oeth, R. Tanaka, F. Suter, and R. F. Da Silva, "Wrench: A framework for simulating workflow management systems," in 2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS). IEEE, 2018, pp. 74–85.

[10] M. Barika, S. Garg, A. Chan, R. N. Calheiros, and R. Ranjan, "Iotsim-stream: Modelling stream graph application in cloud simulation," Future Generation Computer Systems, vol. 99, 2019, pp. 86–105.

[11] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Edge affinity-based management of applications in fog computing environments," Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing, 2019, pp. 61-70.

[12] C. Mechalikh, T. Hajer, and M. Faouzi, "PureEdgeSim: A simulation framework for performance evaluation of cloud, edge and mist computing environments," Comput. Sci. Inf. Syst. Vol 18 No. 1, 2021, pp. 43-66.

[13] M. Gill and D. Singh, "A comprehensive study of simulation frameworks and research directions in fog computing," Computer Science Review 40, 2021, p. 100391.

[14] A. Maia, A. Boutouchent, Y. Kardjadja, M. Gherari, E. G. Soyak, M. Saqib, K. Boussekar, I. Cilbir, S. Habibi, S. O. Ali, et al., "A survey on integrated computing, caching, and communication in the cloud-to-edge continuum", Computer Communications, 219 (2024), pp. 128-152.

[15] M. Satyanarayanan, "Edge computing," Computer, 50(10):36–38, 2017. ISSN 0018-9162.

[16] W. Javed et al. "A review on fog computing for the internet of things," 2021 International Conference on Innovative Computing (ICIC). IEEE, 2021, pp. 1-7.

[17] A.K. Sandhu, "Big data with cloud computing: Discussions and challenges." Big Data Mining and Analytics 5.1 (2021) pp. 32-40.

[18] S. Risco et al. "Rescheduling serverless workloads across the cloud-to-edge continuum." Future Generation Computer Systems 153 (2024): 457-466.

[19] S. Dustdar, V. C. Pujol, and P. K. Donta. On distributed computing continuum systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):4092–4105, 2023.

[20] A. Al-Dulaimy et al.. The computing continuum: From IoT to the cloud. *Internet of Things*, 27:101272, 2024.

[21] E. Del-Pozo-Puñal and F. García-Carballeira. A generic simulator for Edge Computing platforms. In *International Conference on High Performance Computing and Simulation (HPCS)*, 2020.

[22] E. Del-Pozo-Puñal, F. García-Carballeira, and D. Camarmas-Alonso. A scalable simulator for cloud, fog and edge computing platforms with mobility support. *Future Generation Computer Systems*, 144:117–130, 2023.

[23] C. Martinez-Rendon, D. Camarmas-Alonso, J. Carretero, and J. L. Gonzalez-Compean. On the continuous contract verification using blockchain and real-time data. *Cluster Computing*, pages 1–23, 2021.

[24] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917, 2014.

[25] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter. Lowering entry barriers to developing custom simulators of distributed applications and platforms with SimGrid. *Parallel Computing*, 123:103125, 2025.

[26] P. Velho and A. Legrand. Accuracy study and improvement of network simulation in the SimGrid framework. In *2nd International Conference on Simulation Tools and Techniques (SIMUTools)*, Rome, Italy, 2009.

[27] M. Muniswamaiah, T. Agerwala, and C. C. Tappert. Energy consumption for IoT streaming applications. In *2020 IEEE International Conferences on Cyber Security and Cloud Computing (CSCloud) / Edge Computing and Scalable Cloud (EdgeCom)*, pages 174–177, 2020.

[28] E. Del-Pozo-Puñal, F. García-Carballeira, D. Camarmas-Alonso, and A. Calderón-Mateos. Hierarchical and distributed data storage for computing continuum. *Future Generation Computer Systems*, 174:107931, 2026.

[29] M. F. Cloutier, C. Paradis, and V. M. Weaver. A Raspberry Pi cluster instrumented for fine-grained power measurement. *Electronics*, 5(4):61, 2016.

[30] E. Gamess and S. Hernandez. Performance evaluation of different Raspberry Pi models for a broad spectrum of interests. 2022.

[31] SimGrid Project. Advanced routing — SimGrid documentation. https://simgrid.org/doc/latest/Platform_routing.html. Accessed: 2026-08-25.

[32] SimGrid Project. The SimGrid models — SimGrid documentation. https://simgrid.org/doc/latest/Models.html#the-tcp-models. Accessed: 2026-08-25.

[33] H. Casanova, A. Legrand, and M. Quinson. SimGrid: A generic framework for large-scale distributed experiments. In *10th International Conference on Computer modelling and Simulation (UKSim 2008)*, pages 126–131, 2008.

[34] M. Geier, L. Nussbaum, and M. Quinson. On the convergence of experimental methodologies for distributed systems: Where do we stand? In *4th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, 2013.

[35] I. Lera, C. Guerrero, and C. Juiz. YAFS: A simulator for IoT scenarios in fog computing. *IEEE Access*, 7:91745–91758, 2019.

[36] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya. iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Journal of Systems and Software*, 190:111351, 2022.