The Journal of Supercomputing (2024) 80:12564–12584
https://doi.org/10.1007/s11227-024-05926-z

Energy-focused simulation of edge computing architectures in 5G networks

Blas Gómez[1] · Estefanía Coronado[1,2] · José Villalón[1] · Antonio Garrido[1]

Accepted: 21 January 2024 / Published online: 17 February 2024

Abstract
While cloud computing is crucial in processing data from devices with low computational power, the latency introduced by the Internet backhaul limits real-time applications. By situating computing resources at the network's edge, edge computing offers low-latency services by offloading computations from high-performance computing (HPC) data centers to the edge servers, reducing wide Area network (WAN) strain. As a result, edge computing has unlocked opportunities for innovative applications that were previously unfeasible, such as connected vehicles or medical robotics. Nonetheless, deploying the infrastructure required to support edge computing services raises sustainability and energy consumption concerns. Consequently, the development of tools enabling researchers to explore innovative approaches to reducing the energy impact of edge computing is crucial. In this work, we present MintEDGE, a network simulator focused on the energy consumption of edge computing. Our simulator allows testing energy-saving approaches and task placement algorithms in realistic large-scale scenarios encompassing entire regions.

Keywords Network simulator · Multi-access edge computing · Energy efficiency · Edge computing · 5G

✉ Blas Gómez
blas.gomez@uclm.es

Estefanía Coronado
estefania.coronado@uclm.es

José Villalón
josemiguel.villalon@uclm.es

Antonio Garrido
antonio.garrido@uclm.es

1 High-Performance Networks and Architectures, Universidad de Castilla-La Mancha, Albacete, Spain

2 I2CAT Foundation, Barcelona, Spain

1 Introduction

High-performance computing (HPC) has become indispensable in our quest for smarter digital infrastructures, enabling the advanced processing of, and the making of inferences from the data gathered by a multitude of devices that lack the computational power to perform the processing themselves. However, while current sensors and smart devices typically offload this computation to their cloud back-end, this paradigm introduces a challenge in terms of latency due to the delay introduced by the Internet backhaul. This hinders the development of new applications to which timely decision-making is crucial, such as autonomous driving or medical robotics. Consequently, there is a growing demand for computing capabilities in the proximity of the users and, therefore, a shift in how data is processed and services delivered, moving from cloud computing toward edge computing. In this context, cellular networks, as the user's entry point to the network, can play a crucial role in providing these computing resources via edge servers alongside the Radio Access Network (RAN). This deployment of computing capabilities at the edge of the network has a profound impact in reducing latency and enabling a myriad of new applications, while also reducing the pressure on the capacity of the Wide Area Networks (WANs). These edge servers require HPC capabilities to handle complex tasks, such as image recognition, machine learning, or real-time sensor data processing. Moreover, the exponential growth in data generation poses a significant challenge to communication capacity. To effectively address this issue, it has become imperative to perform data pre-processing at the network edge [1].

Global warming and the ongoing energy crisis have heightened awareness of the energy footprint linked to communication and computing infrastructures. The European Union predicts that by 2025, edge systems will account for 12% of the computing infrastructures energy footprint [2]. Additionally, several studies [3, 4] emphasize the importance of designing these infrastructures with energy efficiency and sustainability in mind. Consequently, developing efficient techniques to mitigate the energy footprint of edge computing is essential. However, evaluating these techniques using hardware testbeds can be challenging due to their cost, limited scalability, and lack of hardware diversity. In this context, network simulators provide a valuable and practical tool for network research, offering cost-effectiveness, a controlled experimentation environment, scalability, and reproducibility. They expedite protocol development and foster global collaboration by providing accessible tools to model diverse network scenarios. In fact, the literature already offers different edge computing simulators such as EdgeCloudSim [5] and iFogSim2 [6]. However, these simulators prioritize network performance and quality of service (QoS) over energy consumption and scalability. On the other hand, the authors of [7] present LEAF, a fog and edge computing simulator that, in contrast to [5, 6], is focused on energy consumption. However, LEAF lacks QoS and performance metrics. Furthermore, the ambitious design goals of 6G networks entail a growing dependence on Artificial Intelligence (AI) and Machine Learning (ML). Consequently, providing researchers with tools to simulate realistic user mobility and test the accuracy of different workload predictors for proactive approaches is also crucial.

To address these challenges, this work presents a multi-tier simulator for energy-aware strategies in edge computing (MintEDGE). MintEDGE is a versatile edge computing simulation framework that prioritizes energy consumption while also including QoS and performance metrics, extending our previous work [8]. MintEDGE permits the configuration of infrastructure features, orchestration, and user mobility, empowering researchers to test innovative energy optimization strategies and workload predictors in large-scale scenarios, such as entire cities or regions with numerous users. MintEDGE aims to provide a tool for researchers to evaluate strategies for energy-efficient resource allocations for current infrastructures, such as service placement, VM scaling, and the selective deactivation of edge servers. In particular, the contributions of this work are as follows:

(i) We identify the desirable features of a modular, RAN agnostic edge computing simulator.
(ii) We outline MintEDGE's architecture, which is designed for 5G and edge computing but can be easily extended to other access networks or architectures such as Wi-Fi or Multi-access Edge Computing (MEC) [9].
(iii) We describe MintEDGE's key functionalities, such as cellular user configuration (e.g., mobility, request arrival rate), the definition of edge services requirements, the introduction of workload predictors, and real map testing scenarios thanks to integration with the state-of-the-art urban mobility simulator developed by Alvarez et al., SUMO [10].
(iv) We employ MintEDGE to comprehensively evaluate the real infrastructure of an Mobile Network Operator (MNO) in The Netherlands in scenarios of various sizes, demonstrating its capacity to cope with large-scale realistic scenarios.
(v) MintEDGE is released under a permissive MIT license, allowing free use, modification, and distribution.[1]

The rest of the paper is organized as follows. Section 2 provides an overview of the related work and Sect. 3 describes MintEDGE's requirements and architecture, providing insights into the network and energy models. Then, Sect. 4 reports on the performance evaluation, and Sect. 5 contains the conclusions and future work.

2 Related work

Network simulators provide an efficient way to conduct controlled experiments, allowing researchers to enhance their understanding of networking principles and develop innovative solutions while minimizing costs and real-world deployments risks. Traditional network simulators, such as NS-3 [11] or OMNeT++ [12], extensively model all networking layers, with particular emphasis on the physical layer, including error rates and interference models. However, they lack representation of

[1] MintEDGE is available at https://github.com/blasf1/MintEDGE.

the computational resources of the network in architectures such as edge and fog computing. Furthermore, these simulators lack dynamism in the placement of tasks and applications and the assignment of resources, and their high level of detail results in slow simulation speeds for larger-scale scenarios, which is essential when testing orchestration approaches. For this reason, researchers and developers have introduced new simulators for edge and fog computing architectures, and in this section, we provide an overview of some of them.

EdgeCloudSim [5] is based on CloudSim [13], a simulation framework that enables the modeling and simulation of cloud computing infrastructures. EdgeCloudSim adds functionality to CloudSim to support edge computing scenarios in an easy-to-use and efficient manner, as well as providing a modular architecture that makes it possible to focus on both the network and the computational resources. However, EdgeCloudSim lacks energy consumption models and customizable mobility models in its current version. Another simulator based on CloudSim is iFogSim [14], which focuses on edge and fog computing with a significant emphasis on IoT applications. IFogSim has evolved into iFogSim2 [6], incorporating new scenarios, use cases and customizable mobility. While iFogSim and iFogSim2 inherit basic energy consumption metrics from CloudSim, their simplistic model, which is based on two power levels (idle and busy), lacks the ability to assess individual computing nodes or model energy consumption in the transport network. In [15], the authors introduce PFogSim, a Fog Computing simulator based on EdgeCloudSim and iFogSim. PFogSim extends the capabilities of its predecessors by allowing for multilayered architectures but does not provide any energy metrics. Conversely, Yet Another Fog Simulator (YAFS) [16], based on complex network theory, concentrates on the relationships between infrastructure nodes and applications in IoT scenarios. However, YAFS does not include energy or user mobility models.

In [17], the authors introduce PureEdgeSim, which uses analytical and numerical modelling research on the edge-to-cloud continuum. It primarily focuses on large-scale IoT and offers an extensive energy model for battery-constrained devices. However, it lacks models for the energy consumption of edge servers and the transport network. Moreover, PureEdgeSim does not support customizable mobility, and details on the mobility model, which seems to follow random routes, are not provided. The authors of [18] present FogTorchPi, a tool focused on the placement of applications in the computing resources of Fog Computing nodes, and in particular on the study of the feasibility of different deployments. However, while it provides extensive QoS attributes and a cost model, it does not provide any energy modelling. In [7] the authors introduce LEAF, another Fog simulator whose primary focus is on large deployment energy consumption. They use a linear energy model, which accounts for energy consumption per basic operation carried out on the CPU, showing its accuracy and scalability. However, they focus on energy consumption, and they do not provide essential QoS metrics such as latency. Additionally, LEAF lacks the implementation of a user mobility model.

The above overview, summarized in Table 1, highlights the fact that current state-of-the-art simulators often lack customizable and accurate mobility models, accurate energy modeling, or both. Moreover, some of them struggle to handle large-scale scenarios, and most focus on IoT and Fog Computing, thus ignoring edge computing

Table 1 Summary of simulators' characteristics

Simulator	Energy modeling	Scalability	Customizable mobility
EdgeCloudSim [5]	No	Bad	No
iFogSim2 [6]	End devices only	Bad	Yes
pFogSim [15]	No	Fair	No
YAFS [16]	No	Fair	No
PureEdgeSim [17]	End devices only	Good	No
LEAF [7]	Yes	Good	No
MintEDGE	Yes	Good	Yes. Realistic traces. SUMO integration

infrastructure's energy consumption and placement strategies, not supporting architectures such as ETSI's [9]. Our proposal, MintEDGE, is a modular lightweight simulation framework that addresses these shortcomings by incorporating accurate mobility models, including the use of mobility traces such as [19] or [20] thanks to the integration of SUMO, and by providing accurate and lightweight energy consumption models thanks to the integration of LEAF's energy model. Consequently, MintEDGE is the first simulator that includes, simultaneously, infrastructure's energy, performance, and resource utilization metrics for edge computing architectures where edge servers are hosted in the RAN such as [9], empowering researchers to design and assess new placement and energy-saving strategies.

3 MintEDGE: multi-tier simulator for energy-aware strategies in edge computing

MintEDGE has been designed to cover a wide variety of practical use cases. In general, any use case, regardless of its size, where there is an edge computing infrastructure whose energy consumption can be optimized using different strategies. This usually happens when the workload changes during the day, with the infrastructure being escalated to cope with the peak hours. MintEDGE allows testing the effects of dynamic resource allocation strategies, especially those designed to reduce the infrastructure's energy consumption. Furthermore, MintEDGE facilitates the assessment of how these strategies impact the Quality of Service (QoS) of various configurable services. Examples of this include smart cities offering vehicular safety services where MNOs want to save energy during off-peak hours when fewer resources are needed to attend to all the demand and private networks in large factories with multiple BSs or Access Points (APs) that offer critical safety services. Our design choices are detailed in the following subsection.

3.1 Design choices

MintEDGE is designed to be inherently modular and versatile, empowering researchers to customize it to test different placement strategies and workload

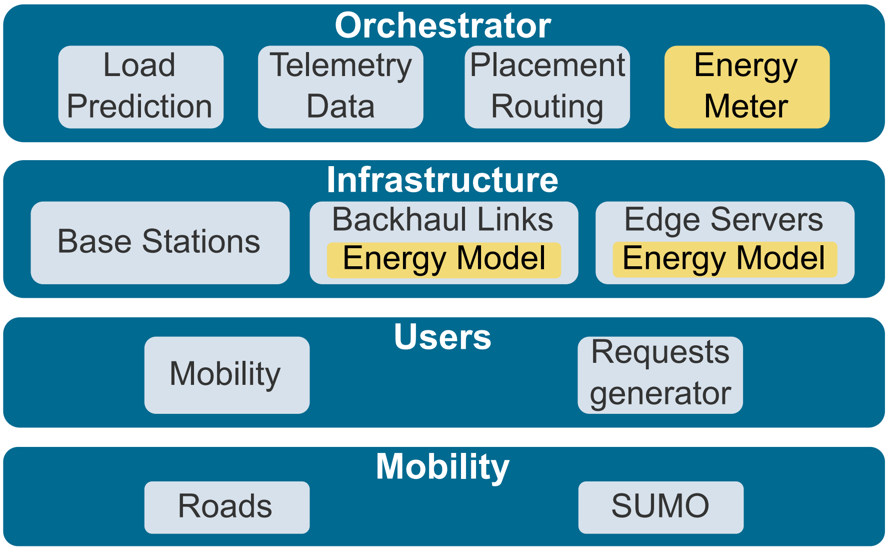


Fig. 1 MintEDGE's architecture

prediction experiments. Furthermore, to support large-scale realistic scenarios, MintEDGE is characterized by a light computational footprint, achieved through high levels of abstraction. MintEDGE is programmed in Python using SimPy and SUMO libraries as its core. While Python can be less memory efficient than other languages, such as C++, it can still be very optimized using slots and the correct external libraries. MintEDGE uses Python due to its ease of use and maintenance, especially due to its large ecosystem of libraries, especially when it comes to AI and ML, which can be a fundamental part of the strategies researchers can evaluate using MintEDGE. Moreover, Python facilitates working with big amounts of generated KPIs using data frames. In essence, MintEDGE has been designed with the following specific set of characteristics in mind:

- *Modularity in the Orchestrator's main tasks* The orchestrator layer plays a pivotal role in providing a global network perspective, as it manages tasks such as service placement and request routing, resource allocation, telemetry data acquisition, and workload forecasting (if needed). Its design focuses on making these modules interchangeable (as depicted in Fig. 1), with particular attention given to placement strategies and the workload prediction modules. This emphasis aligns with the objectives of MintEDGE, which are tailored to streamline the development of energy-efficient placement algorithms, typically involving proactive strategies that incorporate workload predictors.
- *Accountability for energy consumption* Many of the state-of-the-art simulators discussed in Sect. 2 primarily concentrate on IoT devices, which are often powered by batteries, and their energy consumption. In contrast, MintEDGE shifts its focus to the energy consumption associated with placement strate-

gies. As a result, it accounts for the energy consumed by both colocated computing resources within the network (represented by the edge servers) and the transport network, responsible for transmitting requests from wireless access points to the serving edge server. To model these aspects, MintEDGE builds upon the linear power model initially introduced in LEAF [7] and extends it to account for the booting energy of the edge servers. Further details of the energy model are provided below.

- *RAN agnosticism* By default, MintEDGE simulates a 5G infrastructure with a colocated edge computing deployment in which 5G Base Stations (BSs) also serve as hosts for the edge servers. Nonetheless, MintEDGE is designed to be RAN-agnostic. In practical terms, this means that alternative RAN technologies, such as Wi-Fi or different architectural frameworks, such as MEC, can be seamlessly incorporated into MintEDGE. While in a real implementation, moving from 5G to Wi-Fi would require big infrastructure changes, the complete abstraction of the RAN of MintEDGE makes it possible to do this with a few changes to the orchestrator. In its current version, the orchestrator also handles the government of the RAN. In Wi-Fi, it would act both as an orchestrator and SDN controller. This entity would be in charge of allocating resources on the edge infrastructure, the government of the transport network, and the Wi-Fi RAN. This versatile design not only enhances MintEDGE's adaptability but also contributes to its efficiency by abstracting the complexities of RAN operations, allowing MintEDGE to accommodate larger-scale scenarios.
- *Configuration flexibility* MintEDGE offers complete configurability, enabling users to adjust various hardware parameters, including backhaul link capacity, edge server capacity, and service prerequisites such as delay budget or minimum capacity. This flexibility ensures that MintEDGE can be finely tuned to accommodate various hardware configurations and network scenarios. Moreover, MintEDGE allows fully configurable workloads, enabling the definition of user counts, their distribution over time or the request arrival rate for each service.
- *Simulation of large-scale realistic scenarios* MNOs frequently provide their services over vast geographical areas, covering entire countries with millions of users. To ensure the scalability of their networks and mitigate the risk of single points of failure, MNOs often deploy separate orchestrators (typically hosted at the Service Management and Orchestration (SMO) layer) to oversee different regions. Even within this framework, each Orchestrator can manage areas with tens of thousands of users. Consequently, MintEDGE is designed with a focus on efficiency, minimizing real-time computational demands without sacrificing memory usage.
- *Realistic mobility modeling* Network users typically exhibit non-random behaviors, often following discernible patterns, such as commuting between home and university, adhering to daily routines, and periods of rest. Therefore, the mobility model must be customizable to represent these patterns accurately. MintEDGE allows researchers and practitioners to use real mobility traces, which can be sourced from real-world data collection efforts or generated using various methodologies such as those described in [19] and [20]. Furthermore, MintEDGE

integrates with SUMO [10], enhancing the realism of users' mobility even when using random routes, as SUMO incorporates its own micromobility models.[2]

3.2 MintEDGE's architecture

MintEDGE is designed to simulate a 5G RAN and edge computing, but its adaptable architecture makes it easily configurable for various RANs and architectures. In this section, we look at the specific components that make up the foundation of MintEDGE. In essence, MintEDGE's architectural framework, as shown in Fig. 1, comprises four fundamental blocks: Orchestrator, Infrastructure, Users, and Mobility.

3.2.1 Orchestrator

The orchestrator layer plays a central role in MintEDGE, providing a global view of the simulated system. Users leverage this view to develop innovative service placement and resource allocation strategies hosted and executed by the orchestrator. The orchestrator also handles, if required by the user, the control and execution of machine learning models for workload prediction, which is essential for evaluating proactive energy-saving approaches. The orchestrator can configure the infrastructure on the basis of these strategies' output, aggregate telemetry data from the underlying infrastructure layers, and offer the option of storing this data for performance evaluation or feeding it to other components, including placement strategies and workload predictors. Additionally, the orchestrator includes the energy meter block, which is responsible for collecting data from the infrastructure's energy models.

Note that the orchestrator hosts a Load Prediction module. By default, MintEDGE provides an ideal predictor. Thanks to its flexible implementation in Python, researchers can inherit from this class to introduce their own predictors using any library or framework from the rich ecosystem available in Python, such as Scikit-learn, TensorFlow, and Keras. This module can be disabled for those strategies that do not require the use of prediction.

3.2.2 Infrastructure

The infrastructure layer is responsible for modelling the behavior of the infrastructure, encompassing both computing resources and network components. This effectively manages requests originating from the users layer, simulating network and computing delays, managing user connections and request rejection according to hardware capacity and resource allocation, as well as the selected policy (FIFO by default). MintEDGE operates on the assumption that edge servers are colocated with the BSs. This design choice facilitates the exploration of diverse server-BS deployments and server placement algorithms. Furthermore, this concept can be extended to encompass other radio access points as needed. The modular architecture of

[2] Micromobility refers to each user and their dynamics individually. This includes the interaction of the user with the environment, e.g., cars stuck in traffic or waiting at traffic lights.

MintEDGE allows the seamless integration of energy models into any component of the infrastructure. By default, edge servers and backhaul links come with their energy models, but incorporating BSs or orchestrator models would be seamless if required by the user. Researchers can easily import real infrastructure data with BS location and their backhaul connections or generate random ones. The infrastructure hosts user-defined services. MintEDGE users need to define the arrival rate, the workload of a request, the input and output data volumes, and the maximum admissible delay for each defined service. The orchestrator defines the amount of resources that the infrastructure allocates to each one of these services according to the allocation strategy being evaluated.

By default, the infrastructure provides the orchestrator with telemetry information related to the energy consumption (idle, workload-caused and backhaul), server utilization, average delay per service, maximum delay per service, number of requests over the maximum service delay, workload in requests per second (total and for each BS) and rejected requests per service and BS. These KPIs can be stored for evaluation or used by the orchestrator to optimize resource allocation either for performance or energy efficiency.

3.2.3 Users

Within the users layer, individual users initiate requests for one or more services and send them to the connected BS. The request arrival rate depends on the requested service and follows an easily configurable distribution, with the default setting being Poisson. MintEDGE's users can also configure the request arrival rate for each simulated service providing them with the ability to represent a big diversity of workloads and applications. Within this layer, various user types governed by distinct mobility models are available: vehicles (cars), pedestrians, and stationary users. These mobility models, detailed in the mobility layer description below, regulate how users navigate and interact with the simulated environment, granting each user type access to different applications or services.

3.2.4 Mobility

When evaluating strategies or approaches reliant on workload prediction, the inclusion of realistic mobility models is of paramount importance. In this regard, MintEDGE seamlessly integrates with SUMO [10], an open-source traffic simulator designed to create and simulate detailed and realistic traffic scenarios. SUMO models the movement of vehicles, pedestrians, and other forms of transportation within a specified road network. The integration of SUMO within the MintEDGE framework ensures user-friendliness, eliminating the need for users to understand the inner workings of SUMO. Through this integration, MintEDGE gains the ability to model the dynamics of each individual user comprehensively. This includes the interaction of the user with the environment, e.g., cars stuck in traffic or waiting for pedestrians to cross the road. To model the mobility of the simulation scenario, MintEDGE's users need to create a road network. MintEDGE provides two ways of doing so:

(i) *Real-world geographical coordinates:* Researchers can specify the geographical coordinates by defining the real-world area to be evaluated. MintEDGE builds the SUMO road network automatically by using the OpenStreetMap API [21].

(ii) *SUMO network file:* Alternatively, MintEDGE users can provide a pre-existing SUMO network file, thus allowing more flexibility in the evaluation, whether representing a real-world scenario or a synthetic one.

We need to differentiate two types of mobility: macromobility and micromobility. Macromobility refers to the dynamics of traffic flows, i.e., how many users are there and where they are at each time of the day. The user of MintEDGE must provide this information as a crucial part of their evaluation scenario. MintEDGE's users can provide macromobility information in two ways:

(i) *SUMO trace file:* MintEDGE provides full control over both user count and user mobility. The framework accommodates the utilization of SUMO trace files, which record the routes followed by each user. There are two ways to obtain these traces: (1) Generate them synthetically from traffic counting data from different points of the road network such as [22]. Different works have developed methods to transform this data into SUMO traces such as [19] and [20]. (2) Using real GPS data from mobile devices. SUMO provides tools to transform this data into a trace file. However, this last method tends to be more expensive, and the data are often not publicly available due to privacy concerns.

(ii) *User traffic distribution:* For those use cases where the accuracy of mobility is less critical or where a trace file is not available, MintEDGE presents an alternative option, enabling the utilization of random routes. In this mode, researchers only need to specify a constant user count or a user count distribution expressing the user count for each hour of the day. This feature streamlines the configuration for evaluations in which meticulous mobility modeling is not a primary concern.

Once macromobility dynamics have been provided by the user, MintEDGE uses SUMO to control micromobility, which refers to the movement of every single user in the scenario, assuming that their behavior depends on the users surrounding them, e.g., in a single-lane street, a car cannot go faster than the car immediately in front of it. SUMO's micromobility model is detailed in [23].

3.3 Network model

By default, MintEDGE considers a 5G RAN with a finite set of BSs, $\mathcal{B} = \{BS_1, \ldots, BS_N\}$. The BSs are interconnected with each other and with the core network through a given set of links $\mathcal{L}$. Each link ℓ is identified by its source and destination BSs and associated link capacity denoted by $\alpha_{i,j}$, which is configurable by the user. The orchestrator has access to the connectivity graph and the routing matrix Γ

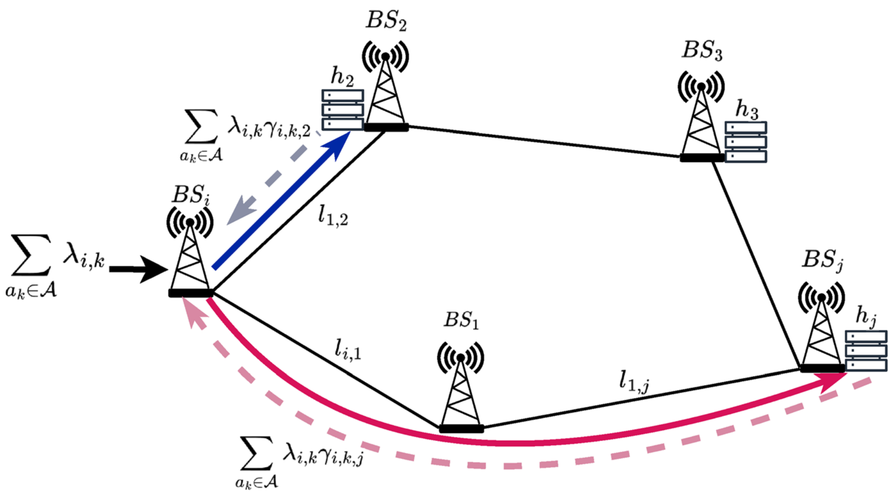

BS_2
BS_3
h_2
h_3
$\sum_{a_k \in \mathcal{A}} \lambda_{i,k}\gamma_{i,k,2}$
BS_i
$l_{1,2}$
$\sum_{a_k \in \mathcal{A}} \lambda_{i,k}$
BS_j
h_j
BS_1
$l_{i,1}$
$l_{1,j}$
$\sum_{a_k \in \mathcal{A}} \lambda_{i,k}\gamma_{i,k,j}$

Fig. 2 Computing requests received by each BS are forwarded to the edge servers that will execute the computing task as determined by the orchestrator. In the above example, incoming requests received at BS_i are forwarded to either BS_2 or to BS_j's edge servers

that determines the routing between two BSs. MintEDGE assumes that the link capacity from BS_i to BS_j is equal to the capacity in the reverse direction. Hence, each link is represented as a three tuple: $\ell_{i,j} = \langle BS_i, BS_j, \alpha_{i,j} \rangle$. An example of a small network following this model is shown in Fig. 2. This cellular network has a colocated set of edge servers denoted by $\mathcal{H}$. MintEDGE's network model assumes that each edge server is associated with a BS, and MintEDGE's users can decide what BSs have an edge server. The presence of an edge server at BS_i is indicated by a binary variable e_i. The maximum capacity in operations per second of an edge server $h_m \in \mathcal{H}$ is denoted by $C_m^{\max}$, which is also configurable. Each edge server can accommodate a finite set $\mathcal{A}$ of computation services, e.g., vehicular safety or medical robotics. The orchestrator is in charge of allocating a specific capacity of each server to each service. Any user $u_i \in \mathcal{U}$ can access the services deployed on any edge server through their serving BS and following the backhaul routes from this BS to the BS that hosts the target edge server. Each computing request $a_k \in \mathcal{A}$ is characterized by a four tuple: $\langle o_k, V_k^{\text{in}}, V_k^{\text{out}}, T_k^{\max} \rangle$, where o_k is the workload generated by the request, V_k^{in} is the size of the input data, V_k^{out} is the size of the outcome of the computation, and $T_k^{\max}$ is the delay budget for the task. The four components of the tuple are configurable by the user. Requests to BS_i for a particular service a_k arrive at a service request rate $\lambda_{i,k}$, which is also configured by MintEDGE's users. The resulting workload due to a_k per unit time is given by $o_k\lambda_{i,k}$.

The edge computing infrastructure is managed by the orchestrator layer, which also handles the government of the RAN and the transport network. At any given time, the orchestrator needs to assign computing requests to edge servers and assign network and computing resources to each service on the servers and the backhaul network, according to the placement and routing strategy introduced by MintEDGE's users, which are the object of the evaluation. This can be done by configuring two matrices:

- *Placement matrix:* Each element $\gamma_{i,k,j} \in [0,1]$ represents the fraction of requests for service a_k received by BS_i to be computed by an edge server hosted at BS_j. When no requests are being sent to BS_j, then $\gamma_{i,k,j} = 0$.
- *Allocation matrix:* Each element $\beta_{k,j} \in [0,1]$ represents the fraction of computing capacity allocated for a service a_k on edge server h_j.

A greedy strategy that places each request on the closest server with free capacity is implemented by default. Moreover, we allow for the possibility of shutting down edge servers, so MintEDGE user-defined strategies can also configure an on/off status variable, denoted by $\eta_j \in \{0,1\}$, where $\eta_i = 0$ means that h_j is off.

Another important part of the network model, especially in an edge computing environment, is the delay experienced, which consists of: (i) the time to upload a request and the input data to the serving BS (T^u); (ii) the time to route the request toward the serving edge server (T^r); (iii) the time to compute the response (T^c); (iv) the time to route the output back from the serving edge server to the serving BS (T^o); and (v) the time to download the output from the serving BS (T^d).

By default, MintEDGE provides a model that assumes that capacity constraints of servers and links are respected, and therefore it does not consider queuing delays. However, the design of MintEDGE makes it possible to seamlessly replace this model with more intricate ones. When the serving BS also hosts the serving edge server, T^r is 0, i.e., the request is not routed through the backhaul links. Let us now focus on how MintEDGE models each one of the components:

- T^u: With the data rate of the radio link at BS_i, denoted by R_i, which can be obtained with Shannon–Hartley's theorem, $T^u_{i,k}$ is given by:

$$T^u_{i,k} = \frac{V^{\text{in}}_k}{R_i} \qquad \text{[seconds]}. \tag{1}$$

- T^r: The backhaul routing time depends on the link capacity $\alpha_{o,p}$ and the size of the request's input V^{in}_k. The backhaul latency can be calculated as the sum of the delays of all the links along the path[3]:

$$T^r_{i,k,j} = \sum_{\ell_{i,j} \in \mathcal{L}} \frac{V^{\text{in}}_k p^{o,p}_{i,j}}{\alpha_{o,p}}, \tag{2}$$

where $p^{o,p}_{i,j}$ indicates whether $\ell_{o,p}$ is on the shortest path from BS_i to BS_j.
- T^c: The computing time of a request depends on the fraction $\beta_{k,j}$ of computing resources allocated to service a_k on edge server h_j and the total capacity $C^{\max}_j$ of h_j:

[3] Requests can experience queuing delays if the capacity limits of the links or the edge servers are exceeded. We assume node processing and propagation delays to be negligible.

$$T^c_{i,k} = \frac{o_k}{\beta_{k,j} C_j^{\max}} \text{ where } \gamma_{i,k,j} > 0. \tag{3}$$

The calculation of T^d and T^o is performed in the same manner as that of T^u and T^r, respectively, but using the output data volume V^{out} instead of V^{in}.

3.4 Energy model

MintEDGE uses LEAF's linear energy model and extends it to include booting energy consumption for a more realistic evaluation of strategies involving edge server shutdown and startup. According to this energy model and the network model explained above, the energy consumption can be divided into: (i) energy used by the edge servers to execute the computing requests; (ii) energy used by the links to route requests from the serving BS to the corresponding edge server; and (iii) energy consumed in the boot process when an edge server is shut down and then turned on.

3.4.1 Edge servers' energy

As mentioned above, an edge server $h_m \in \mathcal{H}$ can perform $C_m^{\max}$ operations per second. When not performing user operations, it has a specific baseline energy consumption, denoted by E_m^{idle}. Each operation executed on h_m adds a certain energy, denoted by E_m, that needs to be added on top of E_m^{idle}. By taking the request arrival rate $\lambda_{i,k}$ at each BS_i, the workload of a request o_k and the fraction of assigned requests $\gamma_{i,k,m}$ for service a_k, the number of operations per second executed on an edge server h_m due to a_k is given by:

$$O_{k,j} = \sum_{BS_i \in \mathcal{B}} \gamma_{i,k,j} o_k \lambda_{i,k} \quad \forall a_k \in \mathcal{A}. \tag{4}$$

If we consider all the services, the total number of operations on h_j can be calculated as:

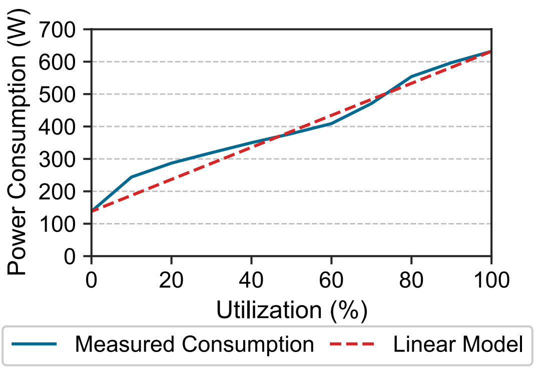
$$O_j = \sum_{a_k \in \mathcal{A}} O_{k,j}. \tag{5}$$

With this, the power consumed by the set of edge servers that are active, i.e., $\eta_m = 1$, is given by:

$$\sum_{h_m \in \mathcal{H}} \eta_m \left(E_m^{\text{idle}} + O_m E_m \right). \tag{6}$$

This energy model strikes a good trade-off between simplicity and accuracy. As illustrated by real-world benchmarks such as [24], a linear increase of energy consumption according to CPU occupation is a robust approximation to the actual consumption, as shown in Fig. 3. While it is true that a better approximation could be obtained by adjusting the energy-per-operation values depending on different load

Fig. 3 Measured energy consumption of an EPYC 8534P CPU [24] and its consumption according to the linear model



levels, the linear model strikes a good balance in terms of simplicity and accuracy, as shown in [7]. The linear model's complexity is $\mathcal{O}(n)$, where n is the number of servers in $\mathcal{H}$. Each server calculates its energy consumption at every time slot and sends the value to the orchestrator, enabling the latter to compute the total consumption of the servers by adding all the received values.

3.4.2 Routing of the computing requests

Depending on the defined placement strategy, computing requests may not be attended to on the server hosted at the serving BS, e.g., the serving BS does not have an edge server, or the edge server's capacity is full. Such situations result in additional energy consumption, as indicated in [7]. The energy consumption can be calculated using a hardware-specific parameter $\sigma_{o,p}$ that denotes the energy consumption incurred when transmitting one bit through a particular link $\ell_{o,p} \in \mathcal{L}$ (typically in J/bit). This parameter can be easily configured in MintEDGE. By considering both the computing input data volume (V_k^{in}) and the corresponding output data volume (V_k^{out}), as well as the number of requests traversing each link, we can calculate the volume of the total data traversing a link as follows:

$$V_{o,p} = \sum_{a_k \in \mathcal{A}} \sum_{BS_i \in \mathcal{B}} \sum_{BS_j \in \mathcal{B}} \left(V_k^{\text{in}} + V_k^{\text{out}}\right) \lambda_{i,k} \gamma_{i,k,j} p_{i,j}^{o,p} \quad \forall \ell_{o,p} \in \mathcal{L}, \tag{7}$$

where $p_{i,j}^{o,p}$ indicates whether $\ell_{o,p}$ is on the shortest path from BS_i to BS_j. Assuming that requests are always routed along the shortest path, this indicates whether the requests originating at BS_i and routed to BS_j (indicated by $\lambda_{i,k}\gamma_{i,k,j}$) have to traverse $\ell_{o,p}$. Consequently, with the data volume traversing a link $\ell_{o,p}$ in bits and its $\sigma_{o,p}$, the total energy consumed by the set of links $\mathcal{L}$ is given by:

$$\sum_{\ell_{o,p} \in \mathcal{L}} \sigma_{o,p} V_{o,p}. \tag{8}$$

Thanks to its modular design, other energy models, such as nonlinear dependency between data volume and transmission energy, can seamlessly be incorporated into

Table 2 Simulation parameters

	Elburg	Maastricht	Utrecht
Number of BSs	27	41	76
Number of cars	375	2500	5700
Number of pedestrians	100	500	1400
Number of stationary users	20	100	280
Max. capacity ($C^{\max}$, ops/s)	11260532		
Idle consumption (E^{idle})	222 W		
Max. consumption (E^{max})	696 W		
Link capacity (α)	10 Gbps		
Link energy consumption (σ)	5.9 nJ/bit		

MintEDGE. At each time slot, each link calculates its energy consumption and sends the value to the orchestrator. The orchestrator can then calculate the total energy consumption of the backhaul by adding the consumption of each link. Thus, the complexity of the backhaul energy model is $\mathcal{O}(m)$ where m is the number of links in $\mathcal{L}$.

3.4.3 Boot process

Turning on edge servers also consumes energy, even if no user operation can be performed during the boot process. Thus, MintEDGE accounts for this energy to facilitate the evaluation of strategies that involve turning off edge servers. An inactive edge server $h_m \in \mathcal{H}$ takes a configurable setup time T^s_m before it is fully operational and ready to serve requests. During this setup time, the power consumption remains constant and is represented as P^s_m [25]. Therefore, the total power consumed during the edge server boot process is given by $E^{boot}_m = T^s_m P^s_m$. The boot energy is calculated by each server on the corresponding time slot and sent to the orchestrator along with the computing energy. Therefore, it does not add further complexity to the model.

4 Evaluation

In this section, we evaluate the performance of MintEDGE using three distinct scenarios, which will vary in terms of geographical area and the number of users involved. By doing so, we can provide a comprehensive view of MintEDGE's performance and scalability and its ability to simulate various numbers of users and BSs.

4.1 Scenarios and parameters

To show how MintEDGE scales, we chose three different scenarios based on the real data [26] for the infrastructure of a single MNO in 3 Dutch cities with different population sizes, namely Elburg, Maastricht and Utrecht. The number of BSs and users in each scenario is shown in Table 2. The number of users is calculated by

Table 3 Requirements of the services evaluated

Service type	Max. delay (ms)	Arrival rate	Input data	Output data	ops/s
Video analytics	30	6	1500 kB	20 B	30000
Augmented reality	15	0.5	1500 kB	25 kB	50000
Vehicular safety	5	10	1600 B	100 B	7000

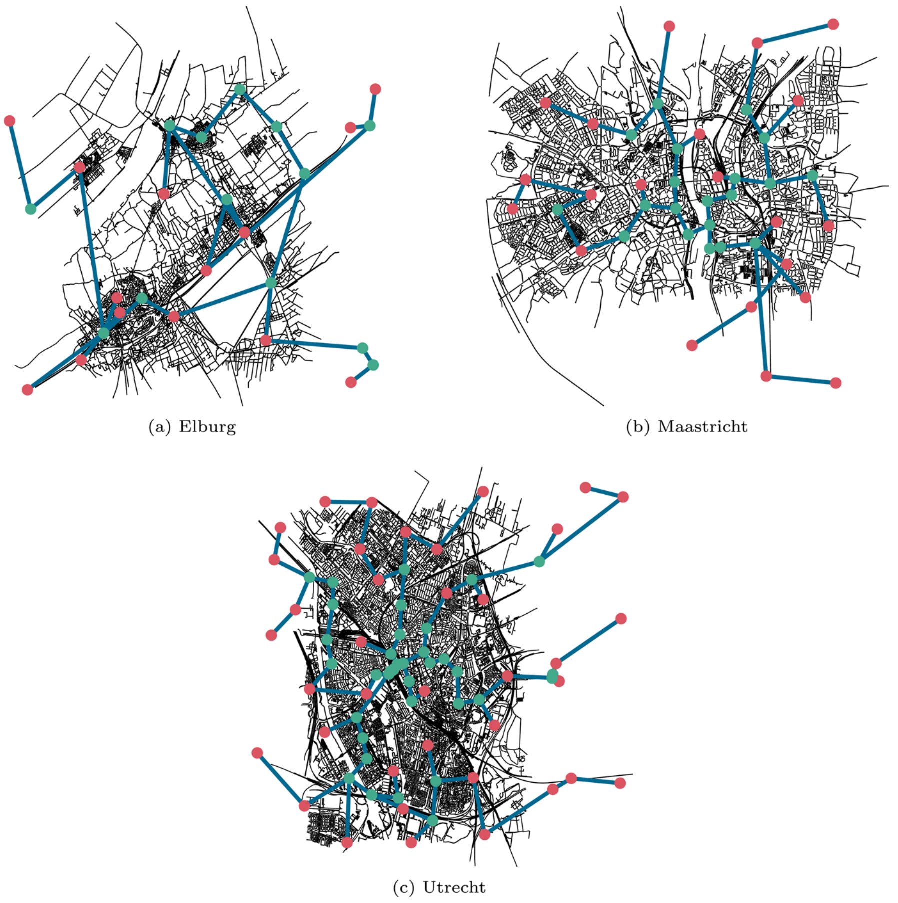


Fig. 4 Map of the three selected scenarios with the position of the BSs. In green are the BSs that host an edge server, and in red are those that do not

taking into account a 1% market penetration share for the augmented reality service and a 10% market penetration share for the vehicular safety service. In the case of the video analytics service, we consider 15 users (CCTV cameras) per km^2 in Maastricht and proportionally scale the number of users to match the population of the other two scenarios. Figure 4 displays the three scenarios, in which half of the BSs

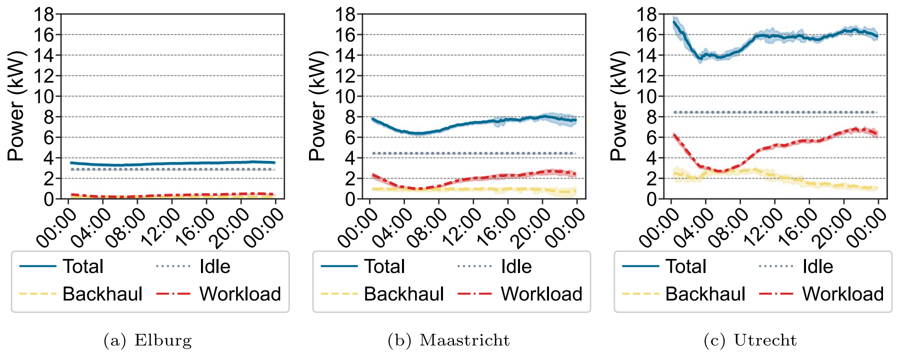


Fig. 5 Energy consumption for each component in the three evaluated scenarios

are shown with green icons to indicate the presence of edge servers. All the edge servers have identical hardware, having a capacity of 11,260,532 operations per second, a baseline energy consumption of 222 W, and a maximum power limit of 696 W [24]. From the maximum and idle energy consumption, the energy required per operation is given by $E_m = (E_m^{max} - E_m^{\text{idle}})/C_m^{\text{max}}$. Therefore, each operation requires 42.1 μJ. We assume that the BSs are connected through the X2 interface using 10 Gbps fiber optics links, whose layout is partially obtained from [26]. However, as this information is insufficient to connect all the BSs, some links have been added. We simulate three different latency-constrained computing services, each defined by its unique characteristics listed, as in Table 3. Throughout this evaluation, we use MintEDGE's random route functionality, adapting the number of users according to the scenarios outlined in Table 2. There are three kinds of users: cars, pedestrians, and stationary users. The first two follow a user count distribution as described in [27], with user activity ranging from 2% of active users at 06:00 to 16% at 21:00. The count of stationary users remains constant throughout the simulation. For each scenario, we conduct two sets of simulations, the first one generating functional results and execution time data, while the second set involves a memory profiler to measure the memory usage. We separate the two sets because the memory profiler introduces significant overhead that can affect the execution time. All the simulations were carried out on a 9-year-old PC equipped with an i5-4590T CPU running at 2 GHz and 16 GB of DDR3 RAM.

4.2 Results discussion

MintEDGE provides detailed insights into the components contributing to total energy consumption, namely idle energy consumption (baseline energy consumption of the servers), energy consumption during server workload, and backhaul energy consumption. We simulate a 24-hour period for each of the three cities. Figure 5 depicts the three energy consumption components and the total consumption resulting from adding them together. In particular, Fig. 5a shows the energy consumption for Elburg, in which 83.6% corresponds to the idle energy

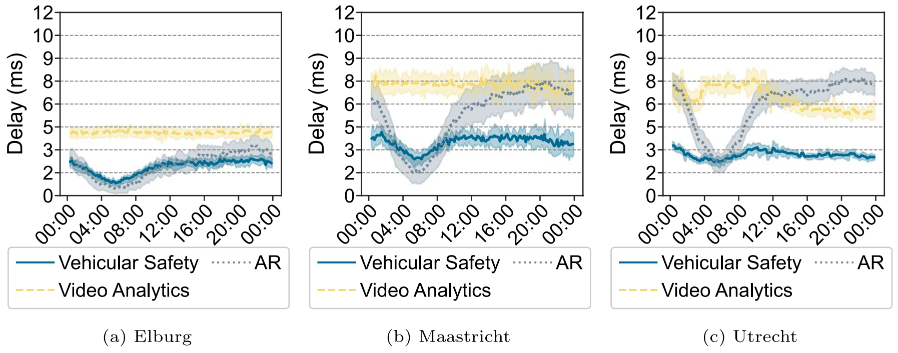


Fig. 6 Delay experienced for each service type in the three evaluated scenarios

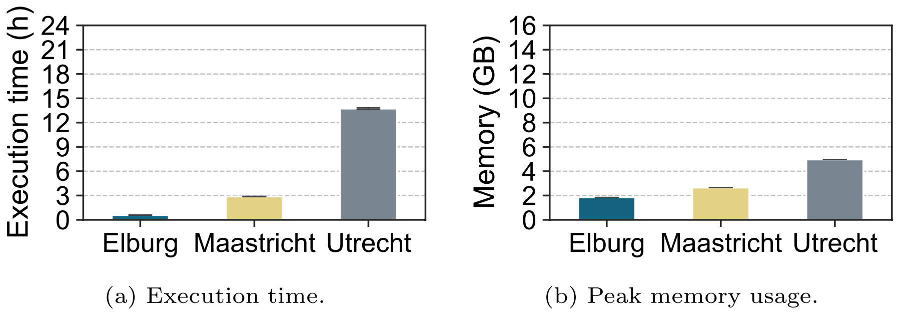


Fig. 7 MintEDGE performance in terms of memory usage and execution time

consumption, 5.8% to the backhaul consumption, and the remaining 10.6% corresponds to the workload energy. The idle energy consumption is the most significant contributor to the total consumption due to the low number of users in this scenario. As the number of users increases, we observe a corresponding growth in workload energy consumption. This can be seen in Fig. 5b, where the workload energy consumption constitutes 25.9% of the total, with the idle and backhaul consumptions accounting for 60.8% and 13.3%, respectively. In the largest scenario, as depicted in Fig. 5c, this effect is more pronounced, with workload energy comprising 32.4% of the total consumption, while idle and backhaul consumptions are 55% and 12.6%, respectively. As the scenario size increases, encompassing more users and base stations, energy consumption grows accordingly, doubling from Fig. 5a to Fig. 5b and once again from Fig. 5b to c. In all three scenarios, idle energy consumption constitutes the most significant portion of the infrastructure's total energy footprint, highlighting substantial optimization potential.

The average delay for each one of the services evaluated is shown in Fig. 6. In all three scenarios, an increase in the average delay is experienced during peak hours (around 21:00) for vehicular safety and augmented reality services due to higher network load. On the contrary, the video analytics service has a constant delay as the number of stationary users (the ones using this service) remains stable. The infrastructure in Elburg could handle a higher load, as demonstrated by the low delay shown in Fig. 6a and the low portion of workload energy consumption (Fig. 5a).

The average execution time for the three evaluated scenarios is shown in Fig. 7a. The 24-hour simulation for Elburg took 35 min, for Maastricht, it took 2 h and 53 min, and for Utrecht 13 h and 44 min. MintEDGE shows good scalability, with an acceleration against the real clock of 40 times in the smallest scenario (Elburg) and 1.7 times in the largest scenario (Utrecht). These promising results are achieved on a regular PC with an old CPU, showing the good performance and scalability of MintEDGE. Considering the three evaluated scenarios, the average acceleration against the real clock is 17 times. Another important factor with regards to scalability is the use of RAM. The PC we used has 16GB of RAM, but none of the scenarios comes close to using the whole capacity, with Elburg taking up 1.8 GB of RAM, Maastricht taking up 2.7 GB, and Utrecht taking up 5 GB, as shown in Fig. 7b.

5 Conclusion

In this work, we have presented MintEDGE, a network simulator dedicated to studying the energy consumption dynamics of edge computing. MintEDGE empowers researchers and practitioners to explore innovative strategies, such as task placement algorithms, with the aim of reducing the energy footprint of the Edge infrastructure. MintEDGE's abstraction level allows it to simulate large-scale scenarios, and its design will enable researchers to fully customize mobility and test proactive approaches with workload predictors. The evaluation we performed demonstrates that MintEDGE can achieve a significant speed-up against the real clock, even when running on older user-oriented PCs. In the future, we plan to extend MintEDGE by introducing a higher level of parallelism in user processes to further boost its performance. We also plan to introduce easier configuration to architectural aspects of the network, such as supporting more than one server per BS, having servers not colocated at BSs, and different service arrival rate distributions. In addition, we plan to expand its capabilities to generate synthetic mobility traces using only traffic counting data, and we aim to incorporate delay models to broaden MintEDGE's applicability and utility.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work is part of the R&D project PID2021-123627OB-C52 funded by the Spanish Ministry of Science and Innovation, Agencia Estatal de Investigación/10.13039/501100011033 and the European Regional Development Fund: "a way of making Europe." This work is also part of the R&D project PID2022-142332OA-I00 funded by the Spanish Ministry of Science and Innovation, Agencia Estatal de Investigación/10.13039/501100011033 and the European Regional Development Fund: "a way of making Europe." This work is also funded by the European Social Fund and Universidad de Castilla-La Mancha under grant 2019-PREDUCLM-10921, the Government of Castilla-La Mancha under project SBPLY/21/180501/000195 and Universidad de Castilla-La Mancha under project 2023-GRIN-34056. This work is also supported by the European Union "NextGenerationEU/PRTR," the Spanish Ministry of Science and Innovation, and Agencia Estatal de Investigación/10.13039/501100011033 under project IJC2020-043058-I.

Data Availability The code used in this work is available at https://github.com/blasf1/MintEDGE. All data generated or analyzed during this study are included in this article.

Declarations

Ethical approval Not applicable.

References

1. The decadal plan for semiconductors a pivotal roadmap outlining research priorities. Technical Report, Semiconductor Research Corporation (2021). https://www.src.org/about/decadal-plan/ Accessed 31 Aug 2023
2. Energy-efficient cloud computing technologies and policies for an eco-friendly cloud market (2020). https://digital-strategy.ec.europa.eu/en/library/energy-efficient-cloud-computing-technologies-and-policies-eco-friendly-cloud-market Accessed 31 Aug 2023
3. Perin G, Berno M, Erseghe T, Rossi M (2022) Towards sustainable edge computing through renewable energy resources and online, distributed and predictive scheduling. IEEE Trans Netw Serv Manag 19(1):306–321. https://doi.org/10.1109/TNSM.2021.3112796
4. Jacob R, Vanbever L (2022) The internet of tomorrow must sleep more and grow old. In: Workshop on Sustainable Computer Systems Design and Implementation (HotCarbon 2022), La Jolla, CA, USA
5. Sonmez C, Ozgovde A, Ersoy C (2017) EdgeCloudSim: an environment for performance evaluation of edge computing systems. In: Proceedings of International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, Spain. https://doi.org/10.1109/FMEC.2017.7946405
6. Mahmud R, Pallewatta S, Goudarzi M, Buyya R (2022) iFogSim2: an extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. J Syst Softw 190:111351. https://doi.org/10.1016/j.jss.2022.111351
7. Wiesner P, Thamsen L (2021) LEAF: simulating large energy-aware fog computing environments. In: Proceedings of IEEE International Conference on Fog and Edge Computing (ICFEC), Melbourne, Australia. https://doi.org/10.1109/ICFEC51620.2021.00012
8. Gómez B, Bayhan S, Coronado E, Villalón J, Garrido A (2023) MintEDGE: Multi-Tier SImulator for ENergy-Aware STrategies in Edge Computing. In: Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom), Madrid, Spain. https://doi.org/10.1145/3570361.3615727
9. ETSI: MEC 003-V3.1.1-Multi-access Edge Computing (MEC) (2022) Framework and Reference Architecture 1: 1–29
10. Alvarez Lopez P, Behrisch M, Bieker-Walz L, Erdmann J, Flötteröd Y-P, Hilbrich R, Lücken L, Rummel J, Wagner P, Wießner E (2018) Microscopic Traffic Simulation using SUMO. In: Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC), Maui, Hawaii, USA. https://doi.org/10.1109/ITSC.2018.8569938
11. Riley GF, Henderson TR (2010) The ns-3 Network Simulator, pp 15–34. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-12331-3_2
12. Varga A, Hornig R (2008) An overview of the OMNeT++ simulation environment. In: Proceedings of International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems, Marseille, France, p 60. https://doi.org/10.1145/1416222.1416290

13. Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50. https://doi.org/10.1002/spe.995
14. Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R (2017) iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Softw Pract Exp 47(9):1275–1296. https://doi.org/10.1002/spe.2509
15. Wang Q (2019) PFogSim: A Simulator for Evaluating Dynamic and Layered Fog Computing Environments. PhD thesis, Auburn University
16. Lera I, Guerrero C, Juiz C (2019) YAFS: a simulator for IoT scenarios in fog computing. IEEE Access 7:91745–91758. https://doi.org/10.1109/ACCESS.2019.2927895
17. Mechalikh C, Taktak H, Moussa F (2021) PureEdgeSim: a simulation framework for performance evaluation of cloud, edge and mist computing environments. Comput Sci Inf Syst 18(1):43–66. https://doi.org/10.2298/CSIS200301042M
18. Brogi A, Forti S, Ibrahim A (2019) 9. Predictive analysis to support fog application deployment, pp 191–221. Wiley, New York, US. https://doi.org/10.1002/9781119525080.ch9
19. Uppoor S, Trullols-Cruces O, Fiore M, Barcelo-Ordinas JM (2014) Generation and analysis of a large-scale urban vehicular mobility dataset. IEEE Trans Mob Comput 13(5):1061–1075. https://doi.org/10.1109/TMC.2013.27
20. Pigné Y, Danoy G, Bouvry P (2011) A vehicular mobility model based on real traffic counting data. In: Proceedings of Communication Technologies for Vehicles, Oberpfaffenhofen, Germany, pp 131–142. https://doi.org/10.1007/978-3-642-19786-4_12
21. API - OpenStreetMap Wiki. https://wiki.openstreetmap.org/wiki/API Accessed 25 Aug 2023
22. Government of Luxembourg: Portail des Travaux publics. Comptage du trafic. https://travaux.public.lu/fr/infos-trafic/comptage.html Accessed 25 Dec 2023
23. Lopez PA, Behrisch M, Bieker-Walz L, Erdmann J, Flötteröd Y-P, Hilbrich R, Lücken L, Rummel J, Wagner P, Wiessner E (2018) Microscopic Traffic Simulation using SUMO. In: Proceedings of IEEE International Conference on Intelligent Transportation Systems (ITSC), Maui, Hawaii, USA, pp 2575–2582 . https://doi.org/10.1109/ITSC.2018.8569938
24. Standard Performance Evaluation Corporation: SPECpower results. https://www.spec.org/power_ssj2008/results/ Accessed 25 Aug 2023
25. Gandhi A (2013) Dynamic server provisioning for data center power management. PhD thesis, Carnegie Mellon University
26. Antennekaart. https://antennekaart.nl Accessed 25 Aug 2023
27. METIS-II Mobile and Wireless Communications Enablers for Twenty–Twenty Information Society II (2020). https://metis-ii.5g-ppp.eu/ Accessed 25 Aug 2023