MobiEdgeSim: A Simulator for Large-Scale Mobile MEC Server Scenarios

Tianhao Zhang[a], Owen Gallagher[b], Aqeel H. Kazmi[c] and Siobhán Clarke[d]
School of Computer Science and Statistics, Trinity College Dublin, College Green, Dublin 2, Ireland

Keywords: Multi-Access Edge Computing, Service Placement, Dynamic Scheduling, Resource Allocation.

Abstract: Multi-access edge computing (MEC) is an emerging network architecture that brings computational resources closer to users, enabling localized computation and real-time task responses. While numerous simulators have been developed to explore MEC environments, most assume static MEC servers and focus on user mobility. However, this static assumption limits the exploration of mobile MEC servers and their potential benefits in dynamic environments. In this paper, we present *MobiEdgeSim*, a simulation framework for large-scale static and mobile MEC server scenarios, where mobile MEC servers may be deployed on buses, trams, trains or other mobile vehicles. The simulator is built on top of the OMNeT++ and Simu5G frameworks, integrating SUMO for realistic road traffic simulations and Veins for seamless mobility and communication modelling. The framework supports large-scale simulations, configurable scenarios, complex network design, dynamic mobile simulations based on real-world transportation systems, and evaluation of matrices under diverse conditions. By introducing mobility-aware MEC server designs, this work enables researchers to study complex urban environments, and optimize resource efficiency in large-scale mobile networks. The performance of *MobiEdgeSim* is evaluated under varying scenarios and service placement strategies.

1 INTRODUCTION

The advent of fifth-generation (5G) networks has brought revolutionary improvements in communication technology, characterized by ultra-low latency, high bandwidth, and real-time access to radio network information. These features have unlocked new opportunities for deploying applications and services closer to end users by leveraging the Radio Access Network (RAN) edges. This paradigm shift, known as Multi-access Edge Computing (MEC), enhances computational efficiency and responsiveness by reducing the dependency on centralized data centres. However, the dynamic nature of 5G networks, with highly mobile user equipment (UE) and even mobile MEC servers, introduces significant challenges. As illustrated in Figure 1, the mobility of both users and servers increases the complexity of resource management tasks, such as service placement and allocation, requiring advanced computational strategies to ensure optimal system performance in real-time.

Service placement plays a critical role in ensuring Quality of Service (QoS) in MEC environments.

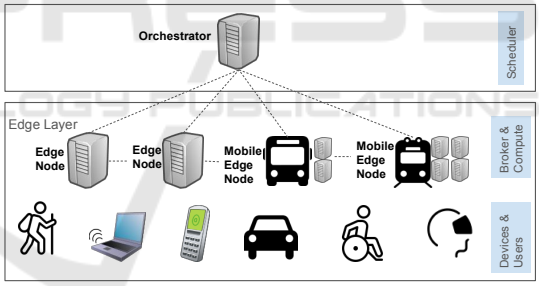

Figure 1: System overview: dynamic service placement in static and mobile MEC nodes.

Researchers and MEC network managers must thoroughly evaluate various placement strategies under realistic conditions to optimize performance, necessitating robust simulation tools. Applications and services are deployed to various edge locations based on real-time network conditions, geographical locations, service types, and resource availability. These dynamic factors make it imperative to develop efficient strategies for task distribution to optimize performance while maintaining QoS under varying conditions.

Simulators are indispensable tools for researchers navigating the complexities and dynamic nature of 5G-based services. Given the unpredictable nature of network conditions, user mobility, and uneven service

[a] https://orcid.org/0009-0004-0932-0278
[b] https://orcid.org/0009-0008-9795-0390
[c] https://orcid.org/0000-0002-8365-9892
[d] https://orcid.org/0000-0001-5721-9976

Zhang, T., Gallagher, O., Kazmi, A. H. and Clarke, S.
MobiEdgeSim: A Simulator for Large-Scale Mobile MEC Server Scenarios.
DOI: 10.5220/0013529300003970
In *Proceedings of the 15th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2025)*, pages 81-92
ISBN: 978-989-758-759-7; ISSN: 2184-2841

distribution, testing in real-world environments is often challenging. By utilizing simulators, researchers can effectively model and evaluate resource management techniques, analyse the feasibility of novel algorithms, and assess the performance of latency-sensitive applications under diverse scenarios.

Simulators play a critical role in studying MEC environments, yet existing tools face significant limitations in modelling dynamic and complex scenarios. For instance, INET (Mészáros et al., 2019), an OMNeT++-based framework, excels in network-level simulations but lacks integration with mobility models, limiting its applicability to dynamic user or server scenarios. Similarly, NS-3 (Riley and Henderson, 2010), widely used for protocol-level studies, cannot incorporate real-world road traffic conditions, which are essential for capturing the impact of congestion or smooth traffic flows on MEC performance. Veins (Sommer et al., 2019) and MAACO (Cabrera et al., 2022) provide partial solutions by integrating mobility with network communication but remain focused on user mobility, overlooking the potential of server mobility, such as deploying MEC servers on buses or trains.

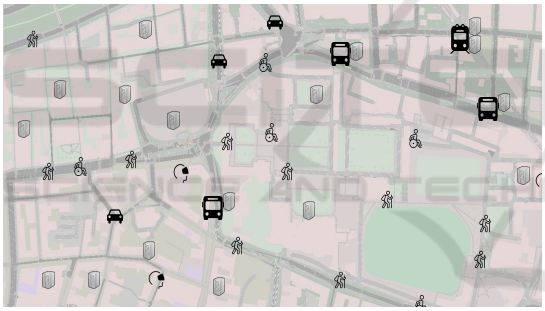

Figure 2: Mobile MEC server scenarios.

These limitations are particularly pronounced in large-scale urban networks, where dynamic interactions between mobile users, real-time traffic conditions, and computational resources play a crucial role. As illustrated in Figure 2, smart city applications such as AR support for people with disabilities, real-time navigation services, and autonomous driving require simulators that support scalability and flexibility while incorporating realistic mobility and communication dynamics. However, most existing tools cannot model these scenarios effectively, as they assume static MEC servers and lack support for advanced application designs across diverse environments.

To address these gaps, we propose a novel simulation framework called *MobiEdgeSim*, whose source code is publicly available on GitHub (Zhang et al., 2025), which overcomes these limitations by integrating static and mobile MEC servers, realistic road traffic dynamics, and large-scale simulation capabilities, and provides flexibility for evaluating diverse application scenarios. Built on top of OMNeT++, Simu5G, Veins, and SUMO, our framework supports investigations into mobility-aware service placement, resource allocation, and system performance under highly dynamic conditions. Through detailed configurations of urban and vehicular mobility, we allow researchers to explore how servers in motion can efficiently respond to shifting network demands and reduce response times for critical applications.

The rest of this paper is organized as follows. Section 2 reviews related work and highlights the gaps in existing MEC simulation frameworks. Section 3 presents our simulation framework design, elaborating how we integrate Simu5G, Veins, and SUMO under OMNeT++. Section 4 describes the modelling approach for both static and mobile nodes, including MEC server extensions and resource allocation mechanisms. Section 5 evaluates the performance of the framework under varying scenarios and service placement strategies, demonstrating the benefits of introducing mobile MEC servers. Finally, Section 6 concludes the paper with a summary of key findings and outlines future research directions.

2 RELATED WORK

We conducted a thorough analysis of both cloud and network simulators to identify effective solutions for modelling large-scale mobile Edge Computing (MEC) server scenarios.

In the category of cloud simulators, CloudSim is a widely used simulation toolkit designed for modelling and evaluating resource provisioning strategies in centralized cloud computing environments (Calheiros et al., 2011). Its modular design and extensibility make it a popular choice among researchers. CloudSim was initially developed for static cloud data centers, making it inherently unsuitable for edge computing and Internet of Things (IoT) scenarios. This limitation means it does not natively support features like mobile user equipment (UE), distributed edge nodes, or dynamic resource allocation.

Several extensions to CloudSim aim to incorporate features more relevant to edge and fog computing: **iFogSim** (Gupta et al., 2017) adds a fog layer on top of CloudSim, enabling IoT device modelling and task scheduling, but lacks native support for complex mobility of edge nodes and realistic network-layer interactions. **EdgeCloudSim** (Sonmez et al., 2018) extends CloudSim with mobility models and simplified communication protocols to evaluate edge resource

offloading. However, it focuses on static edge infrastructure and does not address mobile MEC servers. **IoTSim-Edge** (Nandan Jha et al., 2019) includes features for energy modelling, heterogeneous communication protocols, and directed acyclic graph (DAG) application structures. Despite improved support for IoT-edge interactions, it does not natively handle the movement of MEC nodes or integrate traffic conditions at scale.

YAFS (Lera et al., 2019) (Yet Another Fog Simulator) is a Python-based framework emphasizing resource management in fog and edge infrastructures. Its customizable topologies and dynamic application deployments facilitate latency, energy, and network utilization analysis. However, YAFS is not designed for protocol-level experiments or integration with realistic vehicular or road traffic simulators.

FogNetSim++ (Qayyum et al., 2018), built on top of OMNeT++, supports mobility models, resource scheduling, and handover mechanisms in fog environments. Nevertheless, it does not directly incorporate advanced road traffic simulation tools such as SUMO, limiting its applicability for urban-scale or highly mobile edge scenarios. Moreover, FogNetSim++ remains focused on user mobility rather than the mobility of MEC servers themselves.

In the category of network simulators, **NS-3** (Riley and Henderson, 2010) and **OMNeT++** (Varga and Hornig, 2010) are both prominent discrete-event network simulators, extensively used for protocol-level research in wired, wireless, and vehicular networks. They provide low-level control over networking stacks and support a range of mobility models, making them ideal for studying communication performance. However, their core design does not directly address higher-level aspects such as task scheduling, dynamic service placement, or orchestration of edge resources. Add-on frameworks like **Veins** (Sommer et al., 2019) (for vehicular networks in OMNeT++) enhance mobility and network simulation capabilities. While these extensions support user or vehicle mobility, they do not fully capture resource management or dynamic computations at the edge, particularly when MEC servers themselves are in motion. **Simu5G** is another extension of OMNeT++ designed to simulate 5G New Radio (NR) networks. It integrates with the ETSI MEC architecture to support the evaluation of MEC service placement and resource allocation strategies in standardized 5G environments.

In summary, although existing simulation tools and frameworks offer robust capabilities for either network-layer evaluations or static resource management, they generally lack integrated support for large-scale scenarios where both users and MEC servers are mobile. Such environments, especially in 5G and beyond networks, require a holistic approach that fuses realistic traffic mobility with dynamic service placement and resource orchestration. We propose *MobiEdgeSim*, a new simulation framework that builds on Simu5G, Veins, and SUMO. This framework models static and mobile MEC servers at scale, allowing for large-scale urban simulations that effectively account for mobile server nodes and realistic traffic conditions.

3 SIMULATOR FRAMEWORK DESIGN

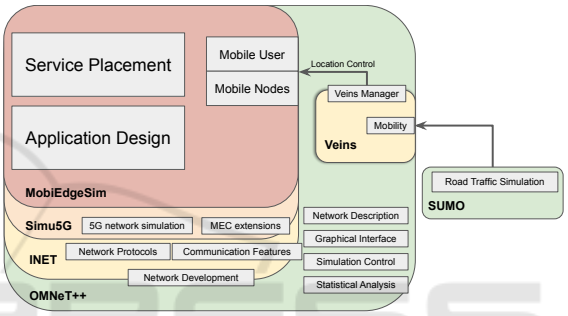

Figure 3: An overview of MobiEdgeSim architecture.

Figure 3 provides an overview of *MobiEdgeSim*, which unifies Simu5G, Veins, and SUMO within the OMNeT++ environment. The objective is to provide an integrated solution that captures both networking protocols and real-world traffic dynamics, making it well-suited for large-scale MEC scenarios that involve both static and mobile MEC servers. The framework integrates mechanisms for the rapid relocation of MEC nodes, facilitating experiments tailored to latency-sensitive applications where the goal is to effectively alleviate network congestion and reduce overall latency. In addition, it uses intricate vehicle mobility patterns to accurately simulate city-wide traffic fluctuations, demonstrating the impact of these variations on communication throughput and resource utilization. The framework also serves as an essential resource for researchers, allowing them to conduct simulations related to adaptive resource scheduling and service placement problems. It enables testing of solutions aimed at maintaining QoS requirements in the event of sudden changes in network topology.

To accomplish the objective, the environment combines a comprehensive 5G NR model (Simu5G), realistic vehicular traces (SUMO), and a unifying traffic-network simulation module (Veins). By ex-

tending this stack to support mobile MEC nodes, the framework enables in-depth investigations into mobility-aware service placement, resource management, and orchestration strategies in urban and highway settings. This holistic approach ensures that varying user demands, shifting network topologies, and realistic traffic conditions are all accounted for when evaluating MEC related scenarios at large scales.

3.1 Core Simulation Tools

Simu5G is an OMNeT++ based framework that implements a detailed 5G New Radio (NR) model, including physical layer abstractions, protocol stacks, and radio resource management (Nardini et al., 2020). It allows simulation of ultra-low latency and high-bandwidth cellular communications, which are fundamental features in modern MEC deployments. Simu5G provides a detailed 5G New Radio (NR) model, including realistic scheduling, handover procedures, and channel propagation effects. Moreover, it incorporates basic modeling capabilities for Multi-access Edge Computing (MEC), following the ETSI standards, allowing simulations of static MEC host deployment and service placement strategies (Noferi et al., 2023). This level of fidelity makes Simu5G the backbone for evaluating how diverse service placement or offloading strategies influence end-to-end performance, such as latency, throughput, and packet delivery ratio, in complex 5G scenarios.

Veins is a popular vehicular network simulation framework that runs on top of OMNeT++, focusing on mobility modelling for vehicles, V2X (Vehicle to Everything) communication protocols, and vehicular ad hoc networks (Sommer et al., 2019). By connecting to a road traffic simulator like SUMO (Simulation of Urban Mobility), Veins can inject realistic vehicle trajectories, speed profiles, and traffic conditions into OMNeT++. This synergy enables the simulation of large-scale vehicular scenarios, from small city grids to highway systems.

SUMO is a microscopic, open-source road traffic simulator that provides detailed mobility data for each individual vehicle, including acceleration, deceleration, lane changes, and interactions with traffic lights (Lopez et al., 2018). Integrating SUMO with OMNeT++ (via Veins) allows a highly accurate representation of on-road mobility patterns, congestion hot spots, and signal timing effects.

Together, **Simu5G, Veins, and SUMO** offer a robust foundation for simulating both the networking and mobility aspects essential for MEC research. However, to address the unique challenges posed by mobile MEC servers, additional extensions and modules were developed.

3.2 Framework Extensions

In order to enhance the capabilities of Simu5G, Veins, and SUMO for mobile MEC server scenarios, we developed several extensions in Simu5G. Figure 4 summarizes the key contributions of these extensions, while the following subsections discuss each in detail.

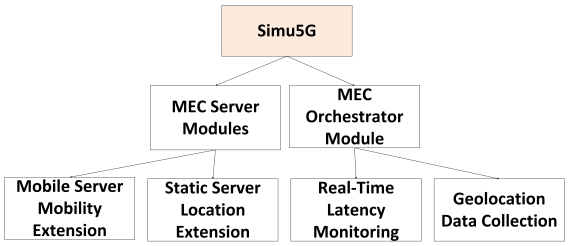

Figure 4: Key framework extensions developed to enhance the capabilities of Simu5G.

3.2.1 Mobile MEC Servers

To enable mobility for MEC servers, we introduced a *Cellular* module within the UPF (User Plane Function) submodule of the MEC server. This *Cellular* module is designed to receive wireless signals and establish essential communication links required for MEC operations, enabling mobile MEC servers to seamlessly connect with the user equipment and exchange radio network information.

Furthermore, we integrated Veins' mobility module into the MEC server component, allowing MEC nodes to move dynamically across simulated road networks. This integration utilizes Veins' mobility interfaces to emulate realistic movement patterns, such as vehicles, trams, or trains, ensuring that the simulation closely reflects real-world urban mobility scenarios.

In addition, the routing table `configurator` module was updated to an `auto-configurator` to accommodate the unique initialization requirements of mobile MEC nodes in OMNeT++. Unlike traditional configurators in INET, which are designed primarily for static nodes, the auto-configurator is capable of dynamically initializing routing tables for nodes that appear after the initial setup phase. This feature is essential for mobile nodes generated through Veins, as their initialization process differs from that of static nodes. By leveraging the auto-configurator, the routing tables are correctly configured for mobile MEC nodes, ensuring seamless integration and reliable operation within the simulation environment.

3.2.2 Static MEC Server Enhancements

For static MEC server modules, we incorporated latitude and longitude parameters and mapped them to specific locations on the SUMO-generated road network during the creation of the NED file. By assigning these geographical coordinates, each static MEC server is precisely positioned in the simulation environment corresponding to its designated location on the SUMO map. This setup ensures accurate alignment between the simulated network topology and real-world geographical layouts, enabling realistic service placement scenarios.

When researchers design their own networks, this approach provides flexibility to define static MEC server locations based on their specific SUMO road networks and architecture requirements. Researchers can strategically place static MEC servers at key points, such as near intersections, public transport hubs, or high-demand areas, to simulate various urban or rural network designs. These predefined coordinates are passed to the MEC orchestrator, which can allow investigations into making informed decisions about service placement, resource allocation, and task scheduling.

This enhancement, along with the Veins mobility module integrated into mobile MEC servers as mentioned in the last section, ensures that mobility-aware algorithms can efficiently utilize both static and mobile nodes. By leveraging the precise location data, the orchestrator is equipped with additional decision-making metrics, such as geographical proximity and spatial distribution of static MEC servers. This enhancement allows the orchestrator to make more informed and context-aware decisions, allowing researchers to test and improve the efficiency and performance of mobility-aware algorithms in the simulation.

3.2.3 Orchestrator Data Collection and Node Selection

We extended the MEC Orchestrator's capabilities by implementing more comprehensive data-gathering mechanisms, enabling more advanced node selection decisions. Instead of relying solely on simple metrics such as available CPU or RAM, the Orchestrator now collects and aggregates a broader set of parameters from both the network and application layers. Specifically:

- **Latency between User and MEC Servers.** At fixed intervals, the system measures the one-way latency between user equipment (UE) and their serving MEC nodes by transmitting timestamped UDP packets. The difference between the embedded timestamp and the arrival time is computed, ensuring the Orchestrator maintains up-to-date delay information under dynamic mobility and load conditions.
- **Geographical Distance.** Both mobile MEC servers and UEs track their latitude and longitude through Veins and get static MEC servers position through the parameter presetting. This enables the Orchestrator to accurately calculate distances for location-aware scheduling heuristics, facilitating the decisions.
- **User-Required Resources.** Detailed CPU, RAM, and disk requirements for each incoming service request prevent tasks from being assigned to under-provisioned nodes.
- **MEC Servers' Current Resources.** Real-time snapshots of available RAM, disk space, and CPU cycles across all MEC servers (both static and mobile) would allow the Orchestrator to balance load more effectively and mitigate overload scenarios.

By aggregating and analysing these richer data sources, the enhanced Orchestrator can adapt to sudden changes in network load, user distribution, or server mobility. Moreover, this expanded data collection opens the door for testing more sophisticated algorithms (e.g., multi-objective optimization or machine learning) that demand a global, fine-grained view of the network state. Consequently, the upgraded Orchestrator provides a flexible foundation for researchers exploring diverse scheduling and resource allocation strategies under realistic urban-scale scenarios.

3.3 Framework Integration

Our simulation framework leverages SUMO for road traffic simulation and OMNeT++ as the discrete event simulation platform. Veins, as a vehicular network simulation framework, bridges the gap between SUMO and OMNeT++, enabling real-time mobility updates through TraCI (Traffic Control Interface). Meanwhile, INET provides fundamental networking models, including IP-based communication and wireless protocols, which are required for Simu5G's operation within OMNeT++.

Simu5G, which models 5G New Radio (NR) communication, operates within OMNeT++ using INET's network stack and interacts with Veins' mobility models to ensure accurate representation of MEC server movement. As MEC nodes are deployed on vehicles, Simu5G dynamically updates their connectivity status, ensuring seamless handovers between 5G

base stations and allowing latency-sensitive applications to operate under mobile conditions. Our extended Simu5G framework builds upon INET and Veins_INET (a subproject of Veins) to utilize realistic mobility data from SUMO through Veins via TraCI. Although this integration leverages existing tools, our primary contribution is the introduction of mobile MEC servers to Simu5G. Specifically, by enabling MEC servers to dynamically follow the mobility patterns provided by SUMO, our extensions significantly enrich the types of scenarios researchers can simulate.

The successful integration of these components allows researchers to simulate real-world MEC deployment scenarios with a high degree of realism. Through this framework, users can evaluate various task placement strategies, resource allocation policies, and network configurations under diverse urban mobility conditions. Building on these extensions, our simulation framework seamlessly integrates Simu5G, Veins, and SUMO with newly developed modules for mobile MEC servers. This integration process involves defining urban environments within SUMO to generate realistic traffic patterns and vehicle movements, utilizing SUMO's detailed mobility data to drive Veins' vehicular models for an accurate representation of vehicle behaviours and interactions, and deploying MEC servers on selected vehicles, trams, or trains, each following mobility patterns governed by Veins' mobility module. In the next section, we provide a detailed description of how a researcher can design their simulation and integrate our simulation framework to support their required capabilities.

4 MODELLING A SIMULATION

This section outlines the workflow for modelling and executing a simulation scenario using *MobiEdgeSim*. Figure5 illustrates the key steps in setting up and running a large-scale simulation scenario in *MobiEdgeSim*, from selecting and processing the raw OpenStreetMap (OSM) data to configuring the OMNeT++ environment and collecting performance metrics. We will elaborate on each step in detail in the following sections, providing researchers with the necessary guidance to replicate or customize large-scale simulations that integrate both static and mobile MEC servers, enabling comprehensive evaluations.

4.1 Road Traffic Modelling with SUMO

Realistic road traffic modelling is essential for accurately assessing the behaviour of mobile nodes, including vehicles equipped with MEC servers. To achieve this, SUMO (Simulation of Urban Mobility) provides a high-fidelity, microscopic traffic simulation framework capable of generating individual vehicle movements based on real-world road networks. However, to create realistic urban traffic scenarios, SUMO requires detailed road network data as input.

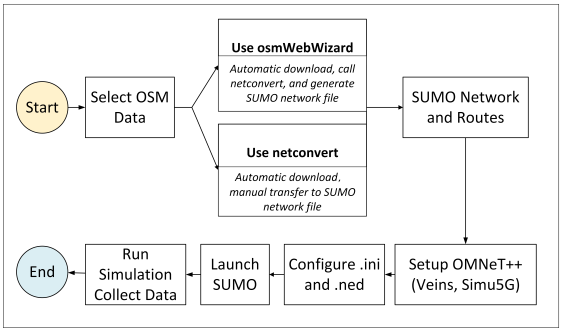

Figure 5: Workflow of simulation modelling.

OpenStreetMap (OSM) is an open-access geographic database that provides comprehensive road network information and is a valuable data source for SUMO-based simulations (OpenStreetMap contributors, 2017). However, OSM data is not directly compatible with SUMO and must first be converted into SUMO's `.net.xml` format. This conversion can be performed using `netconvert`, a SUMO tool designed to process OSM files and generate road networks that adhere to SUMO's simulation framework.

To further simplify this process, SUMO provides an automated tool, `osmWebWizard.py`, which directly retrieves OSM data, processes it with `netconvert`, and generates a complete simulation scenario. This tool allows users to select a geographic region of interest, specify vehicle types, and automatically create all necessary SUMO files, including the road network (`.net.xml`), vehicle routes (`.rou.xml`), and a road simulation configuration file (`.sumocfg`). By eliminating the need for manual data downloading and conversion, `osmWebWizard.py` significantly reduces the complexity of setting up SUMO-based simulations, making it particularly useful for large-scale traffic studies and urban mobility analysis.

For users requiring more control over network conversion parameters, `netconvert` can be used independently. This allows for fine-tuned adjustments such as filtering specific road types, modifying intersection behaviours, or inferring traffic light placements using additional parameters. Both approaches ensure the accurate integration of real-world road infrastructure into SUMO, enabling researchers to model realistic traffic conditions while maintaining flexibility in scenario design.

After generating the SUMO scenario, Veins establishes a TraCI connection that continuously retrieves mobility updates such as vehicle positions, speeds, and lane changes, and relays them to OMNeT++ in real time. This approach allows both user equipment and mobile MEC servers to move along realistic road trajectories that reflect actual traffic conditions in the network simulation. Meanwhile, **Simu5G** integrates 5G New Radio (NR) communication models into OMNeT++, ensuring that both the mobility and the details of the network layer are captured. Once the modules are properly configured, researchers can fine-tune additional parameters, includingg node placement, resource constraints, and traffic densities, before proceeding to environment configuration and performance evaluation. Section 4.2 describes how to modify these parameters using the NED (Network Description) files and the INI (Initialization) files and outlines the general procedure for executing experiments and collecting performance metrics.

4.2 Environment Configuration

The simulation environment configuration is handled primarily through two key files: the NED file and the INI file, which allow researchers to design network topologies, specify simulation parameters, and define node behaviour according to their experimental requirements.

The NED file is used to construct the network topology by specifying various simulation components. Users can define different numbers of User Equipment (UE), gNB base stations, UPF (User Plane Function), and MEC servers. Static nodes, such as gNBs, UPFs, and static MEC servers, are explicitly placed in the network topology and do not change position during the simulation. In contrast, dynamic nodes, including UE and mobile MEC hosts, are defined as vector objects and their movement is governed by SUMO and Veins. This ensures that the number, location, and mobility of these nodes are dynamically controlled according to a SUMO traffic simulation.

The INI file is used to configure simulation runtime settings, application parameters, Veins-specific configurations, and other module-specific parameters. It allows researchers to customize by adjusting the total execution time, defining mobility behaviours, setting resource constraints for MEC servers, and specifying communication parameters for different network elements. The INI file also ensures proper integration with SUMO by defining the connection through Veins' parameters, enabling seamless mobility updates. By modifying the INI file, users can fine-tune their experiments to reflect different network conditions and application requirements without altering the underlying network model.

By modifying these configuration files, users can design and execute experiments tailored to their specific research objectives. This flexibility enables simulations of varying complexity, enabling researchers to experiment with both small-scale test beds and large-scale urban deployments.

4.3 Execution and Data Collection

OMNeT++ offers a built-in statistical framework for collecting simulation results and logging runtime parameters. The simulator supports various output formats, including scalar and vector files, which can be automatically generated based on predefined collection rules specified in the INI file. These files store key performance indicators (KPIs) such as packet transmission rates, latency, and network utilization. OMNeT++ also allows researchers to implement custom result-handling mechanisms by defining specific statistical modules or writing output data directly to external files. In this project, simulation results are stored in `.csv` format, enabling easy post-processing and visualization using external tools. By leveraging OMNeT++'s flexible data collection system, researchers can efficiently analyse the impact of different network configurations and mobility scenarios.

5 PERFORMANCE EVALUATION

In this section, we demonstrate *MobiEdgeSim*'s ability to simulate large-scale edge environments with both static and mobile MEC servers. Specifically, we evaluate how well the framework supports scalability in node populations, dynamic resource constraints under mobility, and different task-placement strategies in realistic road traffic conditions. Our goal is to show that *MobiEdgeSim* effectively captures the performance trade-off of MEC deployments when faced with a variety of node types, workloads, and mobility patterns.

5.1 Experimental Setup

Network Topology and Node Deployment. Our simulation network is designed to replicate a realistic urban environment by incorporating both mobile and static nodes, enabling us to compare the impact of MEC server mobility. Specifically, we draw inspiration from previous experimental setups (Kazmi

Table 1: Evaluation configurations: Number of Static and Mobile MEC Servers.

Configuration	Static Nodes	Mobile Nodes	Users
Config 1	100	0	100
Config 2	70	30	100
Config 3	300	0	100
Config 4	210	90	100

et al., 2025), adopting similar ranges for the number of nodes, mobility patterns, and traffic conditions. User Equipments (UEs) are randomly distributed throughout the simulation area, with their mobility pattern generated using SUMO and managed through Veins. For the static modules deployment, MEC servers are uniformly distributed across the map. The MEC servers are deployed according to four distinct configurations, as summarized in Table 1:

- **Configuration 1.** Comprises 100 static MEC servers and 100 user equipments. This baseline configuration is used to assess the system's performance in the absence of server mobility.
- **Configuration 2.** Incorporates a hybrid deployment consisting of 70 static MEC servers and 30 mobile MEC servers, along with 100 user equipments. This setup introduces mobility and allows investigation of dynamic resource allocation strategies.
- **Configuration 3.** Expands the static deployment to 300 MEC servers, maintaining 100 user equipments. This configuration is designed to evaluate the scalability of the system under large-scale static infrastructure.
- **Configuration 4.** Represents a large-scale hybrid deployment featuring 210 static MEC servers and 90 mobile MEC servers. This configuration combines increased node density with significant mobility, enabling analysis of complex orchestration in dynamic environments.

Table 2: Resource allocations for Users, Mobile MEC Servers, and Static MEC Servers.

	UE	Static Server	Mobile Server
RAM	2-16MB	1-10GB	0.5-5GB
Disk	20-160MB	1-2GB	0.5-1GB
CPU	10-160 MIPS	100-500 MIPS	50-250 MIPS

Resource Allocation. The resource parameters for UEs and MEC servers are detailed in Table 2. During initialization, each node is assigned a maximum resource capacity by randomly picking a value from these ranges. Additionally, to better simulate a realistic workload environment, each node is initialized with a randomly assigned baseline utilization ranging from 10% to 90% of its total capacity. UEs are modelled as resource-constrained devices, with memory ranging from 2 MB to 16 MB, disk capacities between 20 MB and 160 MB, and CPU speeds from 10 MIPS to 160 MIPS. In contrast, MEC servers are provisioned with substantially higher resources. Specifically, static MEC servers receive memory between 1 GB and 10 GB, disk capacities from 1 GB to 2 GB, and CPU speeds ranging from 100 MIPS to 500 MIPS, while mobile MEC servers are allocated slightly reduced resources—memory from 0.5 GB to 5 GB, disk capacities from 0.5 GB to 1 GB, and CPU speeds between 50 MIPS and 250 MIPS. These ranges are designed to reflect realistic hardware constraints and allow our simulation to capture performance variability, as each node's resource capacity is randomly determined within these bounds at runtime.

Placement Strategies. We evaluate three distinct task placement strategies in this evaluation. Each strategy attempts to select a suitable MEC server for service placement. If no server meets the resource requirements, the placement request fails, and the task is deferred until the MEC Orchestrator initiates the next placement attempt.

The first strategy, **Random**, selects an available MEC server at random from the pool of candidates with sufficient resources. The second strategy, **BestFit** (Hussein et al., 2019), systematically examines resource availability across all servers and selects the one that best matches the task requirements by maximizing surplus resources. The third strategy, **ClosestFit** (Ouyang et al., 2018; Moubayed et al., 2020), calculates the geographical distance between the user and each MEC server, selecting the closest one that can accommodate the task. In the following subsections, we assess the performance of these strategies under varying conditions.

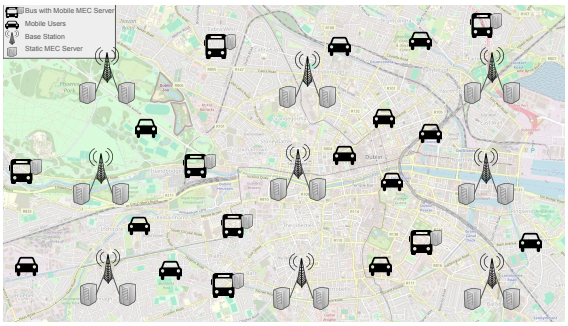

Figure 6: Conceptual overview of the simulated area in the city of Dublin.

Simulation Area and Mobility Parameters. As shown in Figure 6, our experiments take place in a 12 km × 10 km region of central Dublin, derived from an OpenStreetMap road network. We employ

the default urban driving model, which includes regular traffic lights, speed limits, and potential congestion at intersections. User devices are modelled as cars following randomly generated routes, while mobile MEC servers are mounted on buses that traverse pre-defined public transport routes. Vehicles adhere to local road speed regulations and traffic signals, but their actual speed may be influenced due to real-time traffic conditions. By reflecting genuine street layouts and multi-vehicle interactions, this setup captures realistic urban dynamics and allows us to evaluate how mobility-aware MEC scheduling strategies adapt under city-scale scenarios.

5.2 Discussion of Performance

To validate the effectiveness and practicality of our simulator, we performed experiments using the configurations described in Section 5.1. Specifically, these configurations were chosen to demonstrate how *MobiEdgeSim* handles varying numbers of static and mobile MEC servers, diverse resource allocations, and user mobility patterns. During the experiments, each user generates a service placement request once per second, resulting in more than 4000 placement attempts for each configuration. By evaluating system performance under these controlled and statistically significant conditions, our aim is to illustrate the capability of *MobiEdgeSim* to accurately model real-world MEC challenges, including latency-sensitive workloads, resource bottlenecks, and dynamic server mobility.

5.3 Resource Utilization of Static and Mobile MEC Servers

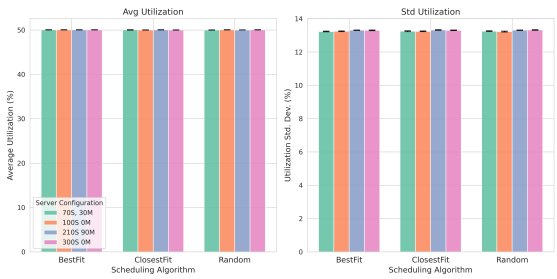

Figure 7: Average and standard deviation of MEC server resource utilization across different scheduling strategies and server configurations.

Figures 7 show the average and standard deviation of resource utilization across different scheduling strategies and server configurations. In all scenarios, utilization values appear similar with minimal differences among scheduling strategies. We attribute this consistency to the broad random baseline resource utilization (10%–90%) assigned at simulation start, which masks the impact scheduling policies would otherwise have. Moreover, our main goal in this work is to demonstrate the effectiveness of *MobiEdgeSim* in handling static and mobile MEC setups rather than optimizing utilization (as the selected placement models do not directly optimize the resource utilization). In future work, we intend to include the placement algorithms which can optimize resources utilization and adopt to more diverse workload patterns to highlight the differences in scheduling strategies and expose potential improvements in utilization.

5.4 Key QoS Metrics: Distance and Latency

To demonstrate how *MobiEdgeSim* measures essential QoS metrics in a mobile MEC environment, we focus on two representative indicators: average user-to-server distance and average latency. Specifically, the distance metric refers to the geographic distance calculated based on latitude and longitude coordinates between each user equipment and its selected MEC server. The latency metric represents the average network delay, measured by calculating the transmission time of timestamped UDP packets traveling between user equipment and the assigned MEC server.

5.4.1 Distance

Figure 8 shows the average distance between a user and its serving edge server under three scheduling algorithms and four server configurations. Our simulator records the real-time location of each user and server, allowing us to compute distances when making decision for each service placement request.

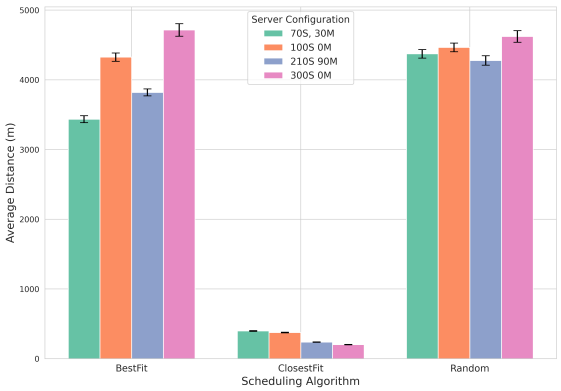

Figure 8: Average distance of connection between user equipment and edge server with 95% confidence intervals.

Under the BestFit strategy, the average user-to-server distance increases with server pool size and server mobility, ranging from approximately 3,000m in the smaller-scale 70S, 30M (green) scenario to nearly 5,000m in the 300S configuration. This trend suggests that BestFit—while prioritizing resource availability—can inadvertently select servers located farther from the requesting user, especially as the total server count grows. By contrast, the ClosestFit algorithm consistently achieves minimal distances (generally under 1,000m) because it specifically aims to choose the server with the shortest geographic separation. Interestingly, configurations with large mobile deployments 210S, 90M (blue) can yield slightly reduced distances under ClosestFit due to mobile nodes being closer to users on average, yet the improvements are modest if other factors (such as resource constraints) come into play. Finally, with the Random approach, distances tend to cluster in the mid-to-high range (approximately 3,500m to 4,500m), reflecting the lack of any proximity-based selection mechanism.

5.4.2 Latency

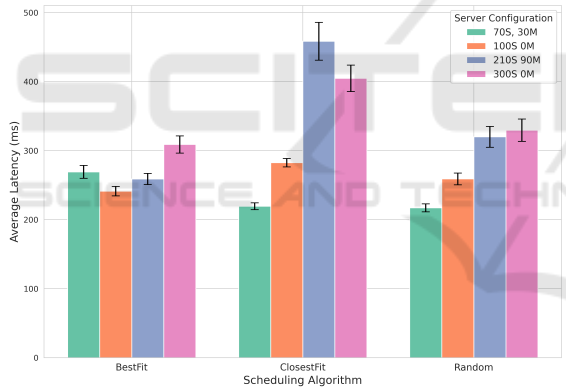

Figure 9: Average latency of connection between user equipment and edge server with 95% confidence intervals.

We also track the latency from a user to the selected MEC server. Figure 9 presents the average latency under the same four server configurations mentioned above. By correlating location-based metrics with resource utilization, *MobiEdgeSim* can reveal how scheduling strategies adapt to changing network conditions.

Results indicate that, with respect to latency, the BestFit strategy typically achieves intermediate or relatively low latency levels (around 200–320ms) across all tested configurations. While this resource-centric approach can sometimes result in larger physical distances (as observed in the BestFit distance metrics), it generally avoids overloading any single server by assigning tasks to whichever node currently appears to have the most available resources. Such transient resource availability can result in subsequent latency increases, particularly in large-scale, highly dynamic environments. In contrast, ClosestFit exhibits a broader range of latency outcomes, with the 210 static servers and 90 mobile servers configuration reaching the highest average latency (about 500ms). This trend implies that ClosestFit, despite minimizing geographic distance, risks imposing significant loads on geographically popular servers, creating queuing delays that negate the benefits of proximity. Meanwhile, the Random strategy falls somewhere in between (roughly 250–360ms), offering neither the distance-based benefits of ClosestFit nor the load-balancing advantage of BestFit. Its variability further underscores the importance of employing informed selection criteria in multi-access edge computing, especially when numerous static and mobile servers coexist.

Overall, these findings highlight a critical trade-off between physical proximity and server load balancing. While ClosestFit can effectively minimize distance, it may exacerbate congestion in heavily utilized regions. Conversely, BestFit helps maintain relatively even resource usage yet can increase propagation distance and occasionally make suboptimal assignments under transient spikes in demand. This evaluation demonstrates that *MobiEdgeSim* can accurately capture the complex interplay between geographic proximity, server mobility, and dynamic resource allocation. The framework provides researchers with a robust tool for systematically evaluating MEC deployment strategies under realistic urban conditions, facilitating deeper insights into how these factors jointly influence end-to-end service latency and performance.

6 CONCLUSIONS

In this paper, we presented *MobiEdgeSim*, an OMNeT++-based simulation framework that integrates Simu5G, Veins, and SUMO to support large-scale dynamic scenarios involving both static and mobile MEC servers. By combining detailed 5G New Radio models, vehicular mobility simulations, and realistic road traffic conditions, our framework addresses a significant gap in existing MEC simulators, many of which overlook the complexities introduced by server mobility.

Through our proof-of-concept experiments, we have demonstrated that mobile MEC servers can effectively complement static edge nodes, enhancing resource utilization and potentially reducing latency

for real-time applications. Our comparative evaluations of different placement strategies (Random, BestFit, and ClosestFit) reveal that resource-aware and distance-aware algorithms can each yield unique advantages depending on the network topology and workload distribution. Notably, even a modest number of mobile MEC servers can significantly improve system performance in congested or high-density scenarios.

Our future work will primarily involve utilizing *MobiEdgeSim* to explore more complex and diverse application scenarios. This includes supporting richer and more complex application scenarios, where a single service may consist of multiple functions that can be distributed across different edge servers to further optimize performance. By simulating a wider range of user requirements and more advanced server-selection strategies (such as multi-objective heuristics or learning-based methods), we aim to evaluate how mobile MEC servers can best support next-generation, latency-sensitive applications. By continuing to refine our platform and experimenting with broader real-world use cases and orchestration at the edge, we aim to provide deeper insights into how to harness the synergy of static and mobile edge servers in next-generation networks.

ACKNOWLEDGEMENTS

This publication has emanated from research conducted with the financial support of Taighde Éireann – Research Ireland under Grant number 13/RC/2077_P2 at CONNECT: the Research Ireland Centre for Future Networks, this work was also supported by VMware by Broadcom.

REFERENCES

Cabrera, C., Svorobej, S., Palade, A., Kazmi, A., and Clarke, S. (2022). Maaco: A dynamic service placement model for smart cities. *IEEE Transactions on Services Computing*, 16(1):424–437.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50.

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

Hussein, M. K., Mousa, M. H., and Alqarni, M. A. (2019). A placement architecture for a container as a service (caas) in a cloud environment. *Journal of Cloud Computing*, 8(1):7.

Kazmi, A. H., Staffolani, A., Zhang, T., Cabrera, C., and Clarke, S. (2025). Dynamic service placement in edge computing: A comparative evaluation of nature-inspired algorithms. *IEEE Access*, 13:2653–2670.

Lera, I., Guerrero, C., and Juiz, C. (2019). Yafs: A simulator for iot scenarios in fog computing. *IEEE Access*, 7:91745–91758.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. (2018). Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE.

Mészáros, L., Varga, A., and Kirsche, M. (2019). Inet framework. *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*, pages 55–106.

Moubayed, A., Shami, A., Heidari, P., Larabi, A., and Brunner, R. (2020). Edge-enabled v2x service placement for intelligent transportation systems. *IEEE Transactions on Mobile Computing*, pages 1–1.

Nandan Jha, D., Alwasel, K., Alshoshan, A., Huang, X., Naha, R. K., Battula, S. K., Garg, S., Puthal, D., James, P., Zomaya, A. Y., et al. (2019). Iotsim-edge: A simulation framework for modeling the behaviour of iot and edge computing environments. *arXiv e-prints*, pages arXiv–1910.

Nardini, G., Stea, G., Virdis, A., Sabella, D., et al. (2020). Simu5g: a system-level simulator for 5g networks. In *Proceedings of the 10th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pages 68–80. SciTePress.

Noferi, A., Nardini, G., Stea, G., and Virdis, A. (2023). Rapid prototyping and performance evaluation of etsi mec-based applications. *Simulation Modelling Practice and Theory*, 123:102700.

OpenStreetMap contributors (2017). Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

Ouyang, T., Zhou, Z., and Chen, X. (2018). Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. *IEEE Journal on Selected Areas in Communications*, 36(10):2333–2345.

Qayyum, T., Malik, A. W., Khattak, M. A. K., Khalid, O., and Khan, S. U. (2018). Fognetsim++: A toolkit for modeling and simulation of distributed fog environment. *IEEE Access*, 6:63570–63583.

Riley, G. F. and Henderson, T. R. (2010). The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer.

Sommer, C., Eckhoff, D., Brummer, A., Buse, D. S., Hagenauer, F., Joerer, S., and Segata, M. (2019). Veins – the open source vehicular network simulation framework. In Virdis, A. and Kirsche, M., editors, *Recent Advances in Network Simulation*. Springer.

Sonmez, C., Ozgovde, A., and Ersoy, C. (2018). Edgecloudsim: An environment for performance evaluation of edge computing systems. *Transactions on Emerging Telecommunications Technologies*, 29(11):e3493.

Varga, A. and Hornig, R. (2010). An overview of the omnet++ simulation environment. ICST.

Zhang, T., Gallagher, O., Kazmi, A. H., and Clarke, S. (2025). Mobiedgesim: A simulator for large-scale mobile mec server scenarios. https://github.com/MobiEdgeSim/MobiEdgeSim_Cellular. Accessed: 2025-03-31.