Kurdistan Journal of Applied Research (KJAR)
Print-ISSN: 2411-7684 | Electronic-ISSN: 2411-7706

Original Article

A Trace-Driven and Dataset-Calibrated Simulation Testbed for Priority-Aware Resource Allocation in Mobile IoT Networks

Mohammad Esmail Ahmad [a, c], **Miran Taha Abdullah** [b, d]

[a] Technical College of Informatics, Sulaimani Polytechnic University, Sulaymaniyah, Iraq.
[b] Computer Science Department, College of Science, University of Sulaimani, Sulaymaniyah, Iraq.
[c] Statistics and Informatics Department, College of Administration and Economics, University of Sulaimani, Sulaymaniyah, Iraq.
[d] Information Technology Department, Tishk International University, Kurdistan Region, Sulaymaniyah, Iraq.

Submitted: 30 June 2026
Revised: 05 November 2026
Accepted: 18 November 2026

*** Corresponding Author**: mohammad.ahmad@uinsul.edu.iq

Keywords: Mobile IoT, Resource allocation, Simulation testbed, Trace-driven calibration, UAV networks, Priority-aware workload, Quality of Experience, Dataset validation.

How to cite this paper: M. E. Ahmed, M. T. Abdullah, "A Trace-Driven and Dataset-Calibrated Simulation Testbed for Priority-Aware Resource Allocation in Mobile IoT Networks", KJAR, vol. 11, no. 2, pp. 237-263, December 2026, doi: 10.24017/science.2026.2.13

Copyright: © 2026 by the authors. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY-NC-ND 4.0)

Abstract: Mobile Internet of Things (IoT) networks support Unmanned Aerial Vehicle (UAV)-assisted services, vehicular systems, wearable healthcare, maritime monitoring, and industrial automation under limited battery, bandwidth, and computing resources. Efficient resource allocation maintains service quality and reliability as communication conditions, traffic demand, and resources change. However, existing network, edge/fog, and federated-learning simulation tools usually focus on different abstraction levels and do not directly provide one repeatable evaluation pipeline for priority-aware mobile IoT resource allocation under joint battery, bandwidth, central processing unit (CPU), communication, and traffic-priority constraints. This paper proposes a configurable, trace-driven, dataset-calibrated testbed for priority-aware mobile IoT resource allocation. The testbed provides a shared state-action-metric interface for trace calibration, priority-level decisions, resource-use logging, profile extension, and benchmark reporting. Each node observes the remaining battery, bandwidth, CPU, backlog, channel quality, round-trip time, and packet-loss probability, while each action allocates transmission effort according to the traffic-priority level. The testbed supports configurable mobile IoT profiles and uses a UAV/drone scenario as the validated dataset-calibrated case. Communication, flight/energy, and image-workload records are mapped to network, battery, and workload-cost variables, and validation checks whether the simulator preserves key communication-trace statistics. Across 18,000 simulator-used communication observations, the mean and standard-deviation differences ranged from 0.475% to 1.833%, with Kolmogorov–Smirnov distances of 0.0041 for round-trip time and 0.0023 for packet-loss probability; lightweight one-step profiling executed 100,000 mixed nodes with 1,633.512 MiB additional memory and 5,134.722 node-step updates per second. This study suggests that the proposed testbed provides a repeatable evaluation environment for comparing priority-aware resource-allocation methods across mobile IoT scenarios while separating dataset-calibrated validation from configurable scenario extension.

1. Introduction

Mobile Internet of Things (IoT) networks are increasingly used in systems where sensing, communication, computation, and service delivery must operate close to moving or resource-limited devices. These systems appear in Unmanned Aerial Vehicle (UAV)-enabled edge computing, vehicular edge computing, healthcare and wearable IoT, maritime IoT, and edge/fog-supported industrial services [1-

6]. In these environments, mobile nodes may collect data, process local observations, and transmit information to a gateway, access point, edge server, base station, or cloud-connected service. However, such nodes usually operate with limited battery energy, wireless bandwidth, and onboard computing capacity, making resource allocation a central requirement for reliable service [1, 2, 4, 7].

The allocation problem becomes more difficult when transmitted data are classified according to service importance. UAV-enabled edge systems must jointly consider computation, communication, storage, mobility, and energy limitations [3]. Vehicular edge systems must also handle changing connectivity, service latency, computation demand, and limited vehicle-edge resources [4]. Similar resource and responsiveness constraints appear in healthcare, wearable, and maritime IoT systems [5, 6]. These examples show that mobile IoT traffic may include both routine data and high-importance information. Therefore, an evaluation environment should represent not only the amount of transmitted data, but also traffic priority and the resource cost of delivery.

Mobile IoT traffic is often classified according to service importance, especially in priority-aware UAV, edge, and IoT service-evaluation settings [8, 9]. For example, some data may be critical and require immediate delivery, while other data may be important or normal and can tolerate more delay. Because of this classification, evaluating only total throughput or total delivery may not describe service performance accurately. An allocation method that delivers many normal data items but misses critical information may appear efficient from a raw delivery perspective, even though it performs poorly for the intended service. For this reason, this paper uses Quality of Experience (QoE) as a priority-aware service metric. In this context, QoE gives higher importance to critical traffic than to routine traffic, so it reflects the usefulness of the delivered service rather than only the amount of delivered data. In wireless adaptive delivery, QoE can be affected by delay, packet loss, wireless channel behavior, and receiver-side conditions [10, 11]. Therefore, QoE should be considered alongside priority-class delivery, packet loss, latency, throughput, and resource usage. This joint reporting is needed because QoE alone may hide resource overspending; an allocation method may improve service quality only by consuming excessive battery energy, wireless bandwidth, or onboard computing capacity.

A reproducible simulation testbed is needed because physical experiments with many mobile IoT nodes are costly, difficult to repeat, and hard to scale under controlled conditions [12]. In this paper, the term simulation testbed refers to a software-based evaluation environment that models mobile IoT nodes, resource constraints, priority-aware traffic, communication variation, and performance metrics under repeatable execution rules. Existing IoT and edge/fog simulation studies and simulators [13] provide established support for repeatable evaluation. For example, iFogSim [14] supports IoT, edge, and fog resource-management simulation, iFogSim2 [15] extends this direction with mobility-aware service migration and microservice orchestration, YAFS [16] supports customizable fog/IoT simulation, and PureEdgeSim [17] supports cloud, edge, and mist computing evaluation. These tools provide broad simulation capabilities, whereas the present work focuses on a narrower evaluation need: priority-aware mobile IoT resource allocation under joint battery, bandwidth, central processing unit (CPU), communication, and traffic-priority constraints.

A second challenge is that fully synthetic simulation may be less directly connected to empirical behavior. Trace-driven and dataset-calibrated simulation can reduce this gap by transforming external records into simulator-facing variables. In this paper, communication traces represent Round-Trip Time (RTT), packet-loss probability, and channel-quality behavior; flight and energy records support battery-drain behavior in the validated UAV case; and UAV image workload data estimate workload-dependent resource costs for critical, important, and normal traffic classes. In this way, the testbed retains the repeatability of simulation while making the validated UAV case more interpretable than a purely random environment.

The research problem addressed in this paper is formulated as follows: how can priority-aware resource-allocation methods in mobile IoT networks be evaluated through a reproducible testbed that jointly represents traffic priority, communication variation, resource constraints, and service-quality outcomes? To address this problem, the paper develops a mobile IoT simulation testbed that uses the

UAV/drone case as the detailed dataset-calibrated scenario and extends the same interface to configurable non-UAV profiles. The resulting environment allows centralized learning, federated learning, heuristic, and non-learning methods to be tested under comparable conditions.

The aim of this paper is to develop and validate a reusable evaluation environment for priority-aware mobile IoT resource-allocation studies. The testbed is designed to support controlled evaluation under constrained battery, bandwidth, and CPU resources while preserving a common state, action, logging, and metric interface across different method families.

The main contributions of this paper are summarized as follows:

- A reusable mobile IoT simulation testbed is developed for priority-aware resource-allocation studies. The validated UAV case integrates communication traces, flight and energy records, and UAV image workload data into one simulation environment.
- A common per-node state-action-resource interface is defined for mobile IoT resource allocation. Each node is represented through remaining battery, remaining bandwidth, remaining CPU, normalized backlog, channel quality, normalized RTT, and packet-loss probability, while the action allocates the transmission effort across critical, important, and normal traffic classes.
- A configurable mobile IoT node-profile layer is introduced to extend the simulator beyond the validated UAV case. The layer keeps the same evaluation interface while allowing the resource capacities, mobility assumptions, traffic models, priority weights, and resource-cost scaling factors to vary by node type. The UAV profile is dataset-calibrated, while the other profiles provide configurable scenario extensions.
- A reproducible benchmarking and validation protocol is defined for evaluating different resource-allocation method families. The protocol supports centralized learning, federated learning, heuristic, and non-learning methods through the same environment interface, decision horizon, QoE definition, resource metrics, and logging structure.
- The testbed is validated through dataset-calibration checks, priority-aware execution behavior, QoE-resource reporting, benchmark-protocol evaluation, mixed-node profile validation, and large-scale simulation smoke testing. The large-scale test is used to examine simulator execution scalability in lightweight mode.

The remainder of this paper is organized as follows. Section 2 reviews the related literature and positions the proposed testbed against resource-allocation studies, learning-based IoT management, and existing simulation tools. Section 3 presents the materials and methods, including the mobile IoT simulation model, dataset calibration, trace-driven execution, benchmarking protocol, metrics, validation, and reproducibility checks. Section 4 presents the experimental results. Section 5 discusses the findings, implications, limitations, and makes a comparison with related work. Section 6 concludes the paper and outlines future research directions.

2. Related Works

This section reviews the selected studies that are relevant to the proposed mobile IoT simulation testbed. The reviewed literature is organized into five areas of focus: mobile and edge-oriented resource allocation, UAV-assisted and aerial IoT systems, learning-based and federated resource management, QoE-oriented evaluation, and simulation tools for IoT, edge, fog, and cloud environments. Some of the reviewed studies focus directly on mobile IoT settings, such as UAV-assisted or mobility-aware edge systems, while others address broader IoT and edge environments whose resource, communication, computation, and simulation issues are relevant to mobile IoT evaluation.

Several IoT and edge studies address the resource-management issues that also appear in mobile IoT systems, including communication, computation, scheduling, energy, and service constraints. Garrido-Hidalgo *et al*. [18] proposed a multi-agent approach for online resource allocation in large-scale Long Range Wide Area Network (LoRaWAN), where scheduling and spreading-factor allocation were used to improve scalability and reliability under dense IoT deployment conditions. Al-Masri *et al.* [19] proposed an energy-efficient cooperative resource-allocation and task-scheduling method for IoT environments, addressing the trade-off between energy use, execution time, and resource utilization.

Costa *et al.* [20] developed an AI-based planning and resource-allocation method for a hybrid IoT network using empirical sensor data, showing how real deployment information can guide network planning and resource decisions. Peng *et al.* [21] proposed a slicing- and auction-based resource-allocation strategy for private edge cloud systems, where allocation is formulated as a hierarchical optimization problem involving network operators, virtual operators, and terminals. Zhang *et al.* [22] studied resource allocation in multi-access edge computing using service function chaining and virtualized network-function deployment, combining static optimization and dynamic deep reinforcement learning for resource placement. These studies show that IoT resource allocation is usually multi-dimensional, involving communication, computation, scheduling, cost, energy, and service constraints. Their main emphasis is on specific IoT architectures or allocation problems. A reusable priority-aware mobile IoT testbed with common action semantics across learning, heuristic, and non-learning methods is less directly addressed in the literature.

UAV-assisted and aerial IoT studies are relevant to the validated case used in this paper. Li *et al.* [23] proposed a UAV-assisted QoS-guarantee scheme for reliable mobile edge computing, showing that UAV-Mobile Edge Computing (MEC) environments require mechanisms that connect application reliability with resource and service management. Zheng *et al.* [8] studied priority-aware path planning and user scheduling for UAV-mounted MEC networks using deep reinforcement learning, where task priority and task size influence path planning, offloading, and scheduling decisions. Li *et al.* [24] proposed a deep reinforcement learning method for task-aware multi-UAV MEC networks, jointly considering UAV trajectory design, access decisions, and resource allocation under energy and resource constraints. Zhang *et al.* [25] also studied a dual-layer aerial IoT system by jointly optimizing service caching, computation offloading, resource allocation, and three-dimensional UAV deployment. These studies show that UAV-enabled IoT and MEC systems require combined reasoning over mobility, communication, task demand, and resource limits. Their main purpose is to propose or evaluate specific optimization or learning algorithms. In contrast, the present work emphasizes the evaluation environment, where priority-aware workload behavior, trace-driven communication, dataset-calibrated image costs, and multiple method families are evaluated through the same interface.

Learning-based and federated resource-management studies also motivate the proposed evaluation design. Liang *et al.* [26] proposed a deep Q-network based resource-allocation method for industrial IoT networks, aiming to improve bandwidth utilization and energy efficiency through reinforcement learning. Loconte *et al.* [27] extended the cloud-to-edge continuum toward IoT devices through a serverless federated learning architecture, showing that IoT devices can participate in distributed training and inference across cloud, edge, and device layers. Sagar *et al.* [28] proposed a hierarchical adaptive federated reinforcement learning approach for resource allocation and task scheduling in hierarchical IoT networks, combining resource management with distributed learning and scheduling decisions. These works support the need for evaluation environments that can test centralized and federated methods under comparable operating conditions. Their focus remains either algorithmic or architectural. The present work complements this direction by providing a benchmarking environment where centralized learning, federated learning, heuristic baselines, and non-learning baselines share the same priority-aware state, action, resource-accounting, and metric pipeline.

QoE-oriented studies also motivate the proposed evaluation design because priority-aware mobile IoT evaluation should not depend on raw throughput only. Taha *et al.* [10] developed an automated model for assessing the QoE of adaptive video streaming over wireless networks, showing that wireless channel behavior, packet loss, delay, and receiver-side conditions can affect perceived service quality. Taha *et al.* [11] also proposed a QoE-adaptive management system for high-definition video streaming over wireless networks, where delay, delay variation, and packet loss are considered in service-quality management. Herrero [9] later proposed multifactor QoE scores in IoT topologies, showing that QoE can be examined in IoT systems using multiple service and network factors. These studies support the use of QoE as a service-oriented evaluation measure. The present work applies this idea to a different setting: priority-aware image traffic in mobile IoT networks, where each transmission action consumes battery, bandwidth, and CPU resources. The proposed testbed connects QoE to priority-weighted delivery, packet loss, latency, throughput, and normalized resource usage.

Simulation tools are also central to this work because the contribution is a simulation testbed. At the network-simulation level, ns-3 [29] and OMNeT++ [30] are widely used discrete-event simulation environments for communication-network and protocol-level studies. At the IoT and edge/fog level, iFogSim is one of the most widely used tools for modeling IoT, edge, and fog environments and evaluating resource-management techniques using latency, network congestion, energy consumption, and cost [14]. iFogSim2 extends this direction by supporting mobility-aware service migration, dynamic distributed clustering, and microservice orchestration in edge and fog environments [15]. YAFS provides a Python-based simulator for IoT scenarios in fog computing, allowing customizable topology, placement, routing, and simulation behavior [16]. PureEdgeSim supports the performance evaluation of cloud, edge, and mist computing environments using metrics such as delay, energy consumption, resource utilization, and task success [17]. More recently, Cantero *et al.* [31] introduced SimulateIoT-Services, a simulation framework for assessing and optimizing IoT service and resource allocation. These tools provide established support for reproducible network, IoT, and edge/fog evaluation. The proposed testbed is complementary to these simulators: it narrows the modeling scope in order to make priority-aware transmission actions, battery-bandwidth-CPU accounting, trace-driven UAV calibration, and method-family benchmarking explicit within one reusable evaluation workflow.

To clarify the role of the proposed testbed, table 1 compares representative tool categories at the capability level. The comparison is not intended to rank simulators; rather, it distinguishes packet/protocol-level simulation, edge/fog service simulation, federated-learning support, and the priority-aware resource-allocation functions required in this study. In this context, the proposed testbed is positioned as a focused evaluation environment for repeatable priority-aware mobile IoT resource allocation through a common state-action-metric interface under controlled simulation assumptions.

Table 1: Capability-level positioning of the proposed testbed against representative tool categories.

Capability	ns-3 [29] / OMNeT++ [30]	iFogSim [14] / iFogSim2 [15] / YAFS [16] / PureEdgeSim [17]	Federated-learning studies [27, 28]	Proposed testbed
Packet/protocol-level network simulation	Main purpose	Limited or abstracted	Not the main purpose	Not a target function
Edge/fog service placement and orchestration	Limited or requires extension	Main purpose	Not the main purpose	Abstracted
Federated-learning training support	Requires custom implementation	Requires custom implementation	Central focus in selected studies	Supported for benchmarked methods
Priority-class transmission actions	Requires custom implementation	Requires custom implementation	Not the main purpose	Directly supported
Battery-bandwidth-CPU resource accounting	Requires custom implementation	Partly supported, depending on model	Not the main purpose	Directly supported
UAV trace and image-workload calibration	Requires custom implementation	Requires custom implementation	Not the main purpose	Supported in the validated UAV case
Common benchmark interface for multiple method families	Requires custom implementation	Requires custom implementation	Usually method-specific	Directly supported
Mixed mobile IoT profile layer	Requires custom implementation	Partly supported or requires extension	Not the main purpose	Directly supported

IoT: Internet of Things, UAV: Unmanned Aerial Vehicle; CPU: Central Processing Unit; YAFS: Yet Another Fog Simulator.

Table 2 summarizes the positioning of the proposed work relative to representative related studies. The comparison focuses on the dimensions that matter most for this paper: environment type, mobility scope, data grounding, priority-aware workload support, resource dimensions, method families, and the relation of each study to the proposed testbed.

Table 2: Positioning of the proposed testbed relative to representative related work.

Study	Main purpose	Evidence / environment	Mobility or node scope	Data grounding	Method scope	Relation to the proposed testbed
[8]	Priority-aware UAV-MEC path planning and user scheduling	DRL-based UAV-MEC evaluation	UAV-mounted MEC	Simulation-based	DRL for path planning and scheduling	Includes priority-aware UAV-MEC scheduling; the proposed testbed extends priority handling toward dataset-calibrated image workload, multi-resource logging, and reusable benchmarking.
[9]	Multifactor QoE scoring in IoT topologies	IoT QoE evaluation	IoT topology context	Evaluation-based	QoE scoring	Supports multifactor QoE evaluation in IoT; the proposed testbed connects QoE to priority-aware action control and multi-resource accounting.
[10]	Automated QoE assessment for adaptive wireless video streaming	Wireless service-quality evaluation	Wireless service delivery	Wireless channel, delay, packet-loss, and receiver-side conditions	QoE assessment and modeling	Supports service-quality evaluation beyond throughput; the proposed testbed extends priority-aware reporting to mobile IoT resource allocation.
[11]	QoE-adaptive management for high-definition wireless video streaming	Wireless service-quality management	Wireless service delivery	Delay, delay variation, and packet loss	QoE-adaptive management	Addresses QoE-aware wireless service management; the proposed testbed additionally evaluates traffic priority and battery, bandwidth, and CPU usage.
[14]	IoT, edge, and fog resource-management simulation	General simulator	IoT/ edge/ fog	Configurable simulation	Resource-management methods	Provides fog/edge resource-management simulator; priority-aware mobile IoT transmission actions are outside its main design scope.
[15]	Mobility, clustering, and microservice management	Extended simulator	Mobility-aware edge/ fog	Configurable simulation	Service migration, clustering, orchestration	Provides mobility-aware edge/fog simulation; the proposed testbed focuses more narrowly on priority image traffic and per-node three-class transmission allocation.
[16]	Fog/IoT simulation with customizable topology	Python-based simulator	Fog/ IoT topology	Configurable simulation	Placement and routing methods	Provides fog/IoT topology simulation; the proposed testbed adds dataset-calibrated priority-aware mobile IoT resource-allocation evaluation.
[17]	Cloud, edge, and mist simulation	Simulator	Edge/ mist/ cloud	Configurable simulation	Orchestration and offloading methods	Supports edge/mist evaluation; the proposed testbed complements it with priority-aware mobile IoT QoE-resource reporting.
[18]	Online resource allocation in large-scale LoRaWAN networks	Emulation and hardware-supported validation	Large-scale IoT end nodes	Uses emulation and hardware validation	Multi-agent allocation and scheduling	Large-scale IoT allocation study; mobile IoT profiles, priority image traffic, and battery-bandwidth-CPU action accounting are outside its main scope.
[19]	Energy-efficient cooperative resource allocation and task scheduling	IoT resource-allocation evaluation	IoT task execution environment	Evaluation-based, not trace-calibrated mobile workload	Cooperative scheduling and allocation	Addresses task scheduling and energy-efficient allocation; priority-aware mobile IoT transmission control is outside its main emphasis.
[20]	AI-based planning and resource allocation in hybrid IoT networks	Empirical IoT planning using sensor data	Hybrid IoT deployment	Uses real sensor measurements	AI-supported planning and allocation	Uses data-grounded IoT planning; the proposed testbed extends this direction toward a reusable priority-aware action interface for multiple method families.
[21]	Slicing and auction-based allocation in private edge cloud systems	Simulation-based edge-cloud allocation	Private edge cloud terminals	Simulation, not trace-driven mobile IoT	Optimization and auction mechanisms	Addresses edge-cloud slicing and terminal satisfaction; mobile IoT priority traffic and QoE-resource reporting are less central to its scope.

Table 2: Continue.

[22]	Service-function-chain resource allocation in MEC	Simulated MEC network	IoT terminals and MEC servers	Simulation-based	MILP and DRL	Addresses VNF/SFC placement and MEC resource allocation; mobile IoT profile abstraction and priority traffic classes are outside its main design focus.
[23]	UAV-assisted QoS guarantee for reliable MEC	UAV-MEC service evaluation	UAV-assisted MEC	Simulation/ evaluation setting	QoS-guarantee mechanism	Supports UAV-assisted MEC reliability evaluation; the proposed testbed gives more emphasis to common interface-based benchmarking and priority-aware traffic classes.
[24]	Task-aware multi-UAV trajectory design and resource allocation	DRL-based multi-UAV MEC evaluation	Multi-UAV MEC	Simulation-based	DRL trajectory and resource allocation	Studies multi-UAV trajectory and resource allocation; the proposed testbed focuses more directly on trace-driven priority image transmission and common interface-based evaluation.
[25]	Caching, offloading, resource allocation, and UAV deployment	Dual-layer aerial IoT evaluation	Aerial IoT/ UAV deployment	Simulation-based	Joint optimization	Addresses aerial IoT optimization; common benchmarking across centralized, federated, heuristic, and non-learning method families is outside its main focus.
[26]	Deep Q-network-based resource allocation in industrial IoT	Simulation-based industrial IoT evaluation	Robotic devices in a smart-warehouse scenario	Simulated communication and resource conditions	Deep reinforcement learning	Addresses reinforcement learning-based bandwidth and energy allocation; the proposed testbed provides priority-aware image traffic, dataset calibration, and a common benchmark interface across method families.
[27]	Serverless federated learning across cloud-edge-IoT	Prototype/ case study	IoT, edge, and cloud continuum	Dataset-based case study	Federated learning	Addresses cloud-edge-IoT federated learning; the proposed testbed applies compatible evaluation ideas to priority-aware resource allocation and QoE-resource reporting.
[28]	Federated reinforcement learning for resource allocation and scheduling	Hierarchical IoT evaluation	Hierarchical IoT	Evaluation-specific	Federated reinforcement learning	Addresses federated reinforcement learning for hierarchical IoT resource management; the proposed testbed provides a reusable priority-aware mobile IoT environment for multiple method families.
[31]	IoT service and resource-allocation simulation	Simulation framework	IoT services	Simulation-based	Service/ resource optimization	The proposed testbed adds mobile IoT profiles, trace-driven UAV calibration, and method-family benchmarking.
Proposed testbed	Priority-aware resource-allocation evaluation in mobile IoT	Trace-driven and dataset-calibrated simulation testbed	UAV validated case; vehicle, wearable, ship, and mobile robot profiles	Communication traces, flight/ energy records, UAV image workload	Centralized learning, federated learning, heuristic, and non-learning methods	Provides a focused testbed that combines trace-driven behavior, mobile IoT profile abstraction, priority-aware action interface, QoE-resource metrics, method-family compatibility, and scalability smoke testing.

LoRaWAN: Long Range Wide Area Network, MEC: Mobile Edge Computing, MILP: Mixed Integer Linear Programming, DRL: Deep Reinforcement Learning, VNF: Virtualized Network Function, SFC: Service Function Chaining, QoS: Quality of Service, DQN: deep Q-network.

Taken together, the reviewed studies show that IoT and edge resource allocation commonly involves communication, computation, energy, scheduling, and service-quality constraints. UAV-assisted, federated-learning, QoE-oriented, and simulation-tool studies each address important parts of this problem. However, the reviewed work does not directly provide a single reusable priority-aware mobile IoT testbed that combines trace-driven UAV calibration, mobile IoT profile abstraction, battery-bandwidth-CPU accounting, QoE-resource reporting, and benchmark execution across the centralized, federated, heuristic, and non-learning methods. The proposed testbed should be interpreted as a focused evaluation environment rather than as a replacement for packet-level network simulators, general edge/fog simulators, or federated-learning frameworks.

3. Materials and Methods

The proposed testbed is constructed through five connected methodological stages. First, communication, flight/energy, and image-workload data are prepared for the dataset-calibrated UAV/drone case, while the profile layer provides configurable parameters for other mobile IoT scenarios. Second, a common seven-component node observation, three-component priority-aware action, and battery-bandwidth-CPU resource model are defined. Third, the trace-driven simulator executes actions by checking workload and resource feasibility, applying communication outcomes, and recording service and resource measures. Fourth, centralized, federated, heuristic, and non-learning methods are evaluated through the same environment and reporting interfaces. Finally, validation examines communication-trace preservation, priority-aware execution behavior, profile extension, and lightweight simulator scalability. Figure 1 presents the overall workflow connecting these stages.

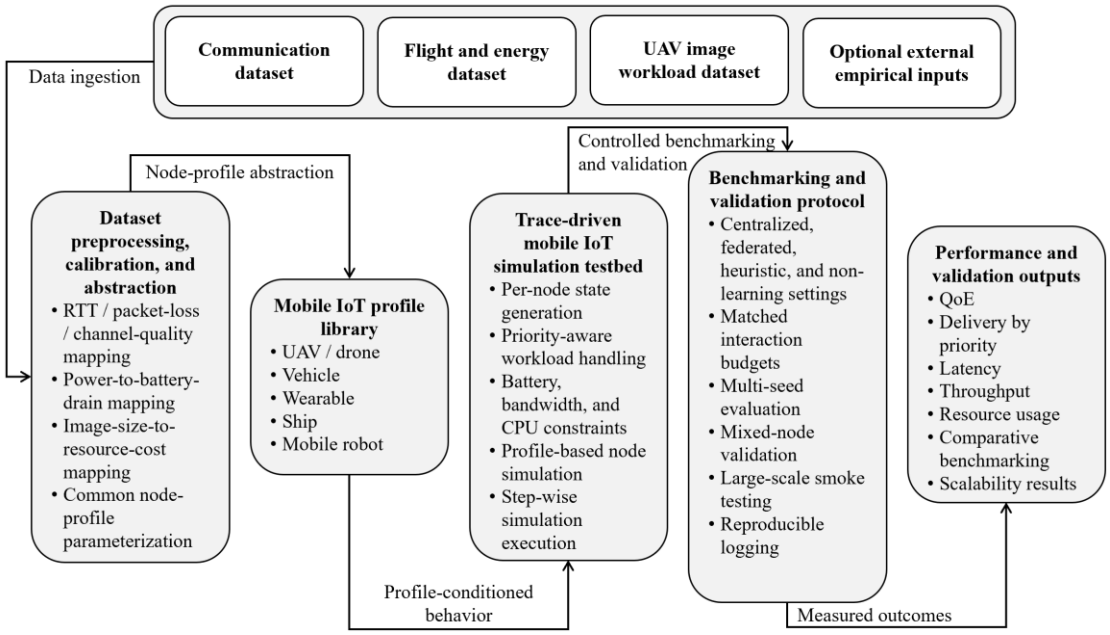

Figure 1: Overall workflow of the mobile IoT simulation testbed.

3.1. Simulation Testbed Overview and Node Representation

The proposed testbed models an uplink-oriented mobile IoT resource-allocation scenario in which mobile nodes generate priority-aware data and transmit it toward a gateway, access point, base station, edge server, or cloud-connected service. This setting is consistent with mobile IoT and edge-computing environments, where communication, computation, mobility, and energy constraints are tightly connected [1, 3, 4].

Figure 2 illustrates how the dataset-calibrated UAV case and the mobile IoT profile layer use one common per-node representation. This representation allows different method families to interact with the simulator through the same observation, action, resource, and reporting structure.

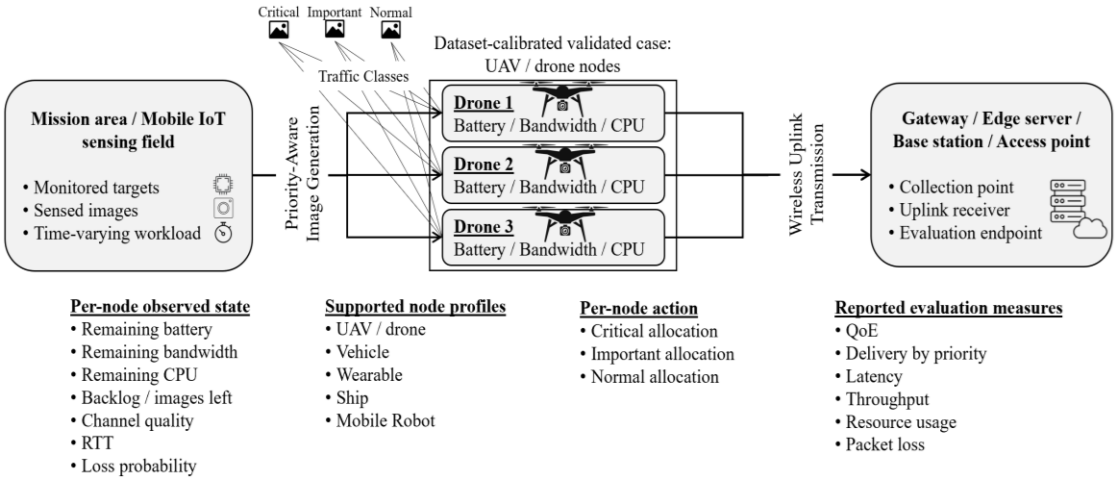

Figure 2: Dataset-calibrated UAV case and common mobile IoT node interface.

In this work, mobile IoT network is used to refer to a controlled abstraction in which multiple mobile IoT nodes share a wireless communication environment and make priority-aware transmission decisions under changing link and resource conditions. Network behavior is represented at the resource-and-link-metric level through bandwidth availability, RTT, packet-loss probability, and channel-quality variation derived from communication traces. The testbed does not model the full communication protocol stack; instead, it focuses on the resource-allocation interface required to evaluate transmission decisions under constrained battery, bandwidth, and CPU resources.

Each node is represented as an independent simulation entity with its own resource state, workload backlog, communication condition, and transmission decision. The simulator supports homogeneous populations, where all nodes use the same profile, and heterogeneous populations, where different node profiles coexist in the same episode. This design allows heterogeneity to be represented through the profile parameters while preserving a unified simulator structure.

3.2. State-Action Interface and Priority-Aware Resource Model

Each node is represented by a seven-component local state vector. For node i at decision step t, the local state is defined as:

$$s_i(t) = [b_i(t), w_i(t), c_i(t), q_i(t), h_i(t), r_i(t), \ell_i(t)] \quad (1)$$

where $b_i(t)$ is the normalized remaining battery, $w_i(t)$ is the normalized remaining bandwidth, $c_i(t)$ is the normalized remaining CPU capacity, $q_i(t)$ is the normalized backlog, $h_i(t)$ is the channel-quality indicator, $r_i(t)$ is the normalized RTT, and $\ell_i(t)$ is the packet-loss probability. All components are normalized to support consistent policy input across the node profiles and resource configurations. A normalized state representation is useful for learning-based resource-management settings because it allows heterogeneous resource conditions to be processed through a common input scale [32].

For a population of N nodes, the global observation is formed by concatenating the local states:

$$S(t) = [s_1(t), s_2(t), \dots, s_N(t)]. \quad (2)$$

The global observation dimension is $7N$. This representation preserves the per-node state while allowing centralized, federated, heuristic, and non-learning methods to operate through the same environment interface.

The action interface has priority awareness. Each node receives three action values corresponding to the transmission effort assigned to critical (cr), important (im), and normal (no) traffic:

$$a_i(t) = \left[a_i^{cr}(t), a_i^{im}(t), a_i^{no}(t)\right]. \quad (3)$$

For N nodes, the joint action is:

$$A(t) = [a_1(t), a_2(t), \dots, a_N(t)], \tag{4}$$

and the total action dimension is $3N$. In the continuous-control setting, for each priority class $p \in \{cr, im, no\}$, the action component $a_i^p(t)$ defined in equation (3) is converted into an executable integer transmission count $n_i^p(t)$ according to:

$$n_i^p(t) = \text{clip}_{[0,5]}\left(\text{rint}\left(2.5\left[\text{clip}_{[-1,1]}\left(a_i^p(t)\right) + 1\right]\right)\right) \tag{5}$$

where $\left[\text{clip}_{[-1,1]}(\,.\,)\right]$ bounds the continuous action to the supported action interval, $\text{rint}(\,.\,)$ denotes the nearest-integer rounding operation used by the implementation, and $\text{clip}_{[0,5]}(\,.\,)$ limits the resulting transmission count to the integer range from 0 to 5. Each continuous action component is converted into an executable priority-class transmission request while preserving the common action interface across continuous-control methods.

The workload is represented through three traffic classes: critical, important, and normal. Critical traffic represents urgent or high-value information that should have a higher priority when resources become scarce. Important traffic represents relevant but less urgent information, while normal traffic represents routine data that can tolerate reduced service under constrained conditions.

The simulator uses these priority classes throughout transmission, delivery accounting, loss accounting, backlog update, and resource-cost calculation. When a decision method selects a transmission action, the simulator interprets the action as the number of requested transmissions for each priority class. Successful transmissions reduce the remaining backlog, while failed transmissions caused by packet loss or insufficient resources are recorded as losses. This makes the workload behavior visible through the delivery, loss, backlog, latency, throughput, and resource-use outputs rather than only through a single reward value. This reporting choice is consistent with QoE-oriented evaluation, where service usefulness is affected by delivery quality, delay, packet loss, and other service-level factors [9-11].

Each node has three constrained resources: battery, bandwidth, and CPU. These resources are initialized at the beginning of an episode and decrease as transmissions are attempted and executed. A transmission can proceed only when the node has sufficient remaining resources. If the requested action exceeds the available resources, the simulator limits or rejects the affected transmission attempts according to the resource-feasibility check.

This resource model is intentionally simple but interpretable. Battery represents energy pressure, bandwidth represents communication capacity, and CPU represents onboard processing pressure. Together, these resources allow the testbed to evaluate whether a method improves service delivery efficiently or whether it consumes more limited resources.

3.3. Mobile IoT Profile Layer

The mobile IoT profile layer defines the node-type parameters used by the simulator, including resource capacities, mobility assumption, traffic model, priority weights, and resource-cost scaling factors. In the current implementation, the profile layer includes one validated UAV/drone profile and four configurable non-UAV scenario profiles: vehicle, wearable, ship, and mobile robot.

During environment construction, each selected profile is converted into per-node simulation parameters. These parameters determine the initial resource capacities, the workload assumptions, the priority weighting, and the scaling factors used in resource-cost calculation. After a profile is assigned to a node, the simulator applies the same state vector, action representation, logging structure, and metric-reporting procedure for all node types.

The UAV/drone profile is the dataset-calibrated profile because it is linked to the available communication, flight/energy, and image-workload datasets. The vehicle, wearable, ship, and mobile robot profiles are configurable scenario defaults for broader mobile IoT testing and future empirical calibration. This design allows homogeneous and mixed-node populations to be evaluated without changing

the benchmark protocol, while keeping the validated empirical interpretation centered on the UAV/drone case.

3.4. Dataset Inputs and Calibration Procedure

The calibration stage converts empirical inputs into simulator-facing variables for the UAV/drone case. Its purpose is to connect the simulation environment to measurable communication, energy, and image-workload behavior while preserving controlled and repeatable execution. The calibration stage uses three empirical datasets: the Drone Communication Dataset [33] (https://www.kaggle.com/dsv/10702965) provides the communication trace containing 45,289 records; the package-delivery quadcopter flight/energy dataset [34, 35] (https://doi.org/10.1184/R1/12683453.v1) provides flight and energy measurements from 209 flights and is openly available under the Creative Commons Attribution 4.0 license; and the UAV-Based Agricultural Image Dataset [36] (https://www.kaggle.com/datasets/ziya07/uav-based-agricultural-image-dataset) provides agricultural image files for workload-size calibration. Both Kaggle datasets are listed online, with access and reuse subject to their respective platform and licensing conditions.

In this study, the trace-driven simulation uses recorded communication measurements to represent time-varying link behavior, while dataset calibration derived simulator-facing parameters from empirical communication, flight/energy, and image-workload records. The profile-configuration layer additionally provided adjustable parameters for other mobile IoT scenarios. Figure 3 summarizes the relationship between these inputs, calibration procedures, and simulator components.

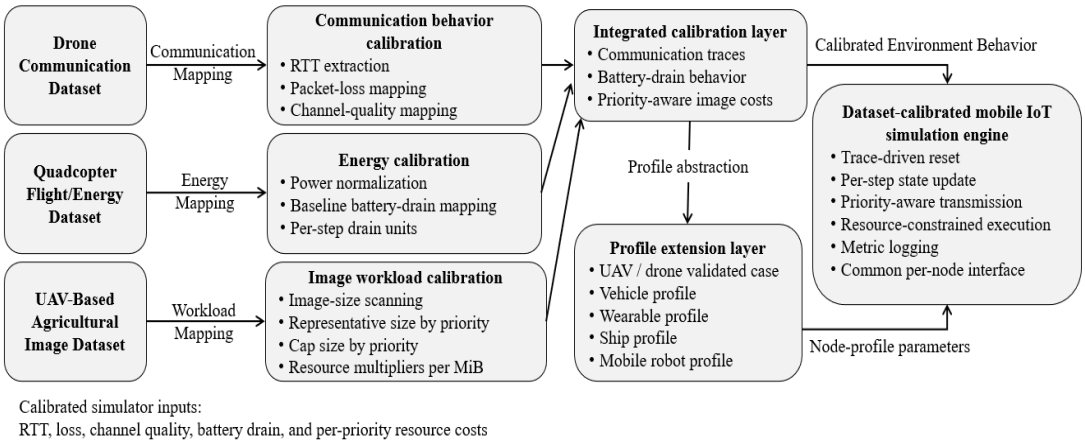

Figure 3: Dataset-to-simulator calibration pipeline.

Table 3 summarizes the methodological role of each dataset and the simulator-facing variables derived from the calibration process. It also includes the profile-configuration layer used for extending the testbed to other mobile IoT scenarios.

Table 3: Calibration inputs, profile configuration, and simulator-facing variables.

Input source	Main role in the testbed	Simulator-facing variables	Used directly for
Drone Communication Dataset	Represents time-varying wireless behavior	RTT, packet-loss probability, channel-quality proxy	Trace-driven communication update
Package-delivery quadcopter flight/energy dataset	Supports UAV baseline energy behavior	Baseline battery-drain tendency	UAV dataset-calibrated case
UAV-Based Agricultural Image Dataset	Represents image-workload variation	Priority-conditioned image-size statistics and resource-cost multipliers	Workload-dependent transmission cost
Profile configuration layer	Provides configurable mobile IoT node parameters	Node capacity, traffic assumptions, priority weights, and resource-cost scales	Profile-based scenario extension

RTT: Round-Trip Time, UAV: Unmanned Aerial Vehicle.

Communication calibration maps raw trace values into step-level link behavior. RTT is used to represent delay-related behavior, while packet-loss rate is converted into a bounded packet-loss probability. When communication-quality indicators are available, they are transformed into a channel-

quality proxy. During an episode, the simulator reads the trace values sequentially. If the episode length exceeds the available trace window, the trace index wraps around so that the environment remains executable under a fixed seed while preserving the temporal variations in delay, loss, and link quality.

The flight and energy records support baseline battery-drain behavior. When valid flight-related fields are available, the simulator derives a power-related sequence from the records and normalizes it into a per-step battery-drain tendency. This baseline drain is separated from the immediate battery cost caused by transmission actions because UAV energy pressure may result from flight operation even when the communication action is small. If the flight data are unavailable or unusable, a fallback baseline drain keeps the environment executable.

The UAV image dataset is used as a workload source rather than as a temporal trace. The simulator scans the image directory, identifies valid image files, and computes their file sizes. These statistics are used to estimate representative image sizes for critical, important, and normal traffic. Representative values describe typical image sizes, while capped values reduce the influence of unusually large files. The calibrated size values are then converted into battery, bandwidth, and CPU cost multipliers, so attempted transmissions consume resources according to workload size rather than fixed manually assigned constants only.

3.5. Trace-Driven Simulation Step Procedure

After calibration, each episode is executed as a repeated decision-step loop. At the start of an episode, the simulator initializes the node resources, workload backlog, communication state, trace position, and metric counters. During each step, the environment constructs the local state of each node, forms the global observation, receives the joint action from the evaluated method, decodes the action into per-node priority-class transmission counts, updates communication variables from the trace, checks resource feasibility, applies resource costs, and records the resulting delivery, loss, backlog, latency, throughput, QoE, and resource metrics. Algorithm 1 summarizes the simulator transition used in each decision step.

Algorithm 1: Trace-driven simulator step loop.

Input: node profiles, calibrated communication trace, calibrated workload costs, current resource states, current backlog, and evaluated resource-allocation method.

Output: next global observation, reward, delivery/loss records, latency/throughput records, and resource-usage metrics.

Initialize node resources, workload backlog, communication state, trace position, and metric counters.
For each decision step (t):
    Build the local state of each node.
    Concatenate all local states into the global observation.
    Provide the local observations and, where required, the global observation to the evaluated method.
    Receive the joint action from the evaluated method.
    Decode the joint action into per-node transmission counts for critical, important, and normal traffic; for continuous-control methods, apply the mapping in equation (5).
    Update communication variables from trace-driven values when dataset-calibrated mode is enabled.
    Check whether the requested transmissions are feasible under remaining battery, bandwidth, and CPU resources.
Apply battery, bandwidth, and CPU costs for feasible transmissions.
Update delivery, loss, backlog, latency, throughput, QoE, and resource metrics, and calculate the scalar reward according to equation (9).
Return the next observations to the evaluated method.
End the episode when the maximum step count is reached or the workload is completed.

This algorithm keeps the policy-facing interface unchanged across the evaluated method families while allowing calibrated communication and workload behavior to affect the internal environment transition.

3.6. Lightweight Large-Scale Execution Mode

The simulator supports a lightweight execution mode for scalability checks. In this mode, the communication trace is loaded once and shared across the node objects, while repeated per-node image-cost scanning is disabled. This reduces repeated data loading during large mixed-node experiments.

The purpose of this mode is to examine lightweight simulator execution scalability as the number of mobile IoT nodes increases. It evaluates environment creation, reset, observation and action handling, one-step execution time, memory growth, and node-step throughput under the simplified execution mode. It does not evaluate learning convergence, detailed network/protocol scalability, or physical deployment scalability, and it should not be interpreted as a replacement for the full dataset-calibrated benchmark mode.

3.7. Evaluation Modes and Benchmark Scope

The evaluation protocol was designed to check whether the proposed testbed can support controlled method comparison, consistent metric reporting, dataset-calibrated validation, profile-based extension, and lightweight scalability testing. The focus is on the simulator and reporting pipeline rather than on proposing a new allocation algorithm. Figure 4 summarizes the benchmarking, validation, and reproducibility workflow.

Two evaluation modes are used. The first is the dataset-calibrated UAV mode, which is used for the main validation and benchmark demonstrations because it has the strongest connection to the empirical data. The second is the profile-extension mode, which uses the mobile IoT profile layer to create homogeneous or mixed-node populations. This mode checks whether the simulator remains executable when node type, resource capacity, traffic assumption, priority weighting, and resource-cost scaling change.

The benchmark scope is defined by three requirements. First, all evaluated methods interact with the same environment interface. Second, all methods are reported using the same service and resource metrics. Third, the results are produced under common episode settings, seed control, and logging structure. These requirements keep the comparison focused on method behavior rather than differences in simulator access or reporting.

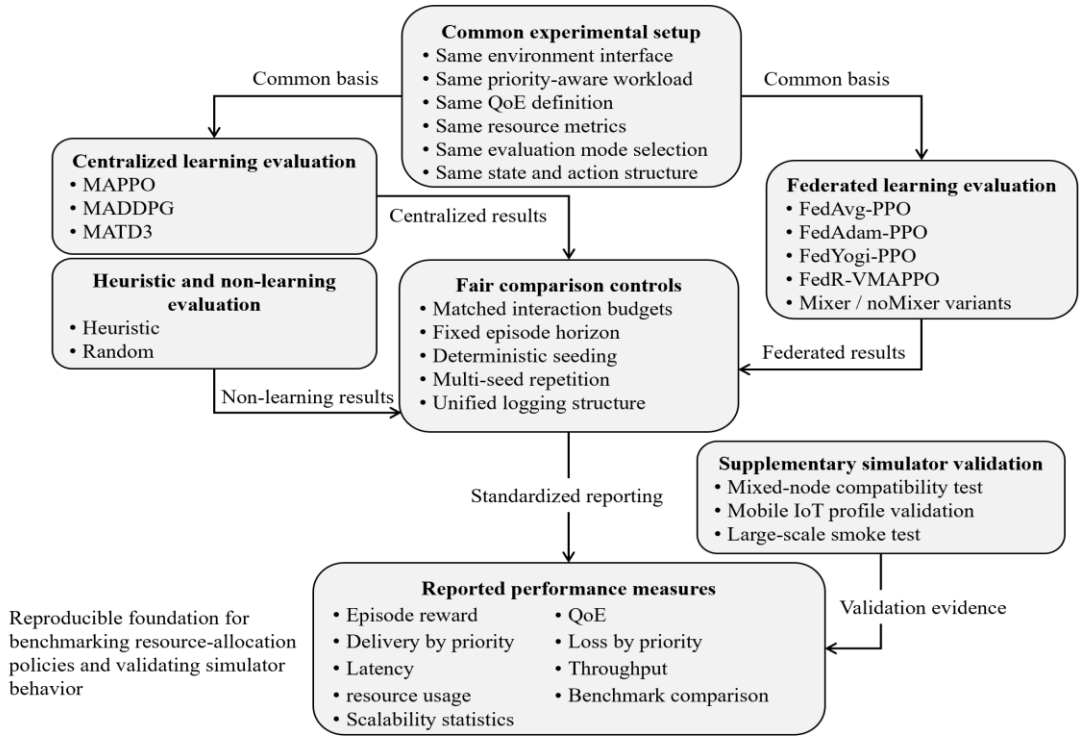

Figure 4: Benchmarking, validation, and reproducibility protocol.

3.8. Method Families and Benchmark Configuration

The selected method families are included to test whether the proposed testbed can support different decision-making styles under the same state, action, and metric interface. The centralized learning methods, including Multi-Agent Proximal Policy Optimization (MAPPO) [37], Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [38], and Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MATD3) [39], represent multi-agent reinforcement learning methods that use centralized training with decentralized execution or centralized critic information for cooperative multi-agent control. The federated variants are Proximal Policy Optimization (PPO), including Federated Averaging with PPO (FedAvg-PPO) [40], Federated Adam with PPO (FedAdam-PPO) [41], Federated Yogi with PPO (FedYogi-PPO) [41], and Federated Residual Value-Mixed Multi-Agent PPO variant (FedR-VMAPPO), and they have been included to evaluate the compatibility with distributed training and aggregation-based learning, both of which are increasingly relevant for IoT systems where data, workload, and resource conditions may be distributed across nodes or clients [27, 28]. The heuristic method provides an interpretable rule-based comparison point, while the Random method provides a non-learning reference baseline for checking whether learned or rule-based decisions produce meaningful behavior beyond uncontrolled action selection [42, 43]. Table 4 summarizes the role of each method family in the benchmark protocol.

Table 4: Method families used in the benchmarking protocol.

Method family	Examples used in the benchmark	Role in the protocol
Centralized learning	MAPPO, MADDPG, MATD3	Tests centralized or centrally trained multi-agent learning under the common simulator interface
Federated learning	FedAvg-PPO, FedAdam-PPO, FedYogi-PPO, FedR-VMAPPO variants	Tests distributed learning and aggregation-based training under matched environment access
Heuristic	Resource-aware heuristic method	Provides an interpretable rule-based comparison point.
Non-learning	Random	Provides a simple non-learning reference baseline

MAPPO: Multi-Agent Proximal Policy Optimization, MADDPG: Multi-Agent Deep Deterministic Policy Gradient, MATD3: Multi-Agent Twin Delayed Deep Deterministic Policy Gradient, PPO: Proximal Policy Optimization, FedAvg: Federated Averaging, FedAdam: Federated Adam, FedYogi: Federated Yogi, FedR-VMAPPO: Federated Residual Value-Mixed Multi-Agent PPO.

The benchmark experiments were repeated with five independent seeds (0–4). For each seed, the centralized learning methods were allocated 2,784 training episodes. Federated methods used 116 communication rounds, three clients, and eight local training episodes per client per round, giving the same total of 2,784 local training episodes per seed. The dataset-calibrated mode was enabled, and the maximum episode horizon was 120 steps, with earlier termination permitted when the workload was completed. The same environment interface, resource model, action mapping, and reporting metrics were used across all benchmark families.

Table 5 summarizes the main benchmark and validation configuration. The configuration fixes the validated empirical case, supported profile extension, per-node state and action dimensions, priority classes, constrained resources, benchmark mode, profile-validation mode, large-scale mode, and reported outputs. This configuration preserves the same simulator semantics across the method families and separates the dataset-calibrated UAV benchmark from the broader profile-extension and scalability checks.

Table 5: Benchmark, training and validation configuration.

Configuration item	Value or description
Validated empirical case	UAV/drone dataset-calibrated scenario
Supported profile extension	UAV/drone, vehicle, wearable, ship, and mobile robot profiles
Per-node state dimension	7
Global observation dimension	$7 \times N$, where N is the number of nodes
Per-node action dimension	3
Global action dimension	$3 \times N$, where N is the number of nodes
Priority classes	Critical, important, and normal
Constrained resources	Battery, bandwidth, and CPU
Table 5 cont	
Main benchmark mode	Dataset-calibrated UAV mode
Profile validation mode	Homogeneous and mixed mobile IoT node populations
Large-scale mode	Lightweight simulator execution mode
Simulator-recorded outputs	QoE, delivery, loss, latency, throughput, resource usage, and efficiency
Independent benchmark runs	Five seeds: 0, 1, 2, 3, and 4
Centralized learning budget	2,784 training episodes per seed
Federated learning budget	116 rounds, 3 clients, and 8 local episodes per client per round; 2,784 local training episodes per seed
Maximum episode horizon	120 steps; an episode can terminate earlier when the workload is completed
Dataset mode	Dataset-calibrated traces enabled
MAPPO	Actor LR 3×10^{-4}; critic LR 10^{-3}; $\gamma = 0.99$; PPO clip $= 0.2$; 4 PPO epochs; batch size $= 64$; replay/buffer size $= 10{,}000$; hidden size $= 128$
MADDPG	Actor LR 10^{-4}; critic LR 10^{-3}; $\gamma = 0.99$; $\tau = 0.01$; exploration SD $= 0.2$; batch size $= 128$; buffer size $= 100{,}000$; hidden size $= 256$
MATD3	Actor LR 10^{-4}; critic LR 10^{-3}; $\gamma = 0.99$; $\tau = 0.01$; policy noise $= 0.2$; noise clip $= 0.5$; policy delay $= 2$; batch size $= 128$; buffer size $= 100{,}000$; hidden size $= 256$
Federated PPO local learner	Actor LR 3×10^{-4}; critic LR 10^{-3}; $\gamma = 0.99$; GAE $\lambda = 0.95$; PPO clip $= 0.2$; 10 PPO epochs; batch size $= 128$; buffer size $= 10{,}000$; hidden size $= 256$; mixer hidden size $= 256$
FedAvg-PPO	Equal model-parameter averaging; Mixer and noMixer variants use the same local PPO settings
FedAdam-PPO	FedAdam server update; server LR $= 0.05$; $\beta_1 = 0.9$; $\beta_2 = 0.99$; $\epsilon = 10^{-8}$
FedYogi-PPO	FedYogi server update; server LR $= 0.02$; $\beta_1 = 0.9$; $\beta_2 = 0.99$; $\epsilon = 10^{-8}$
FedR-VMAPPO	Same local PPO settings; resource budgets (battery, bandwidth, CPU) $= (0.30, 0.55, 0.64)$; dual $LR = 0.07$; gate $k = 12$; minimum gate $= 0.2584$; linear residual schedule $0.0978 \rightarrow 0.02435$; server Adam LR $= 0.1$, $\beta_1 = 0.9$; $\beta_2 = 0.999$; $\epsilon = 10^{-8}$
Mixer/noMixer variants	Mixer uses CAR-Mixer value coordination with hidden size 256; noMixer uses the centralized critic without the mixer
Heuristic	Resource-aware deterministic rule; no trainable learning hyperparameters
Random	Random action sampling through the same action interface; no trainable learning hyperparameters

CPU: Central Processing Unit, IoT: Internet of Things, QoE: Quality of Experience, UAV: Unmanned Aerial Vehicle, GAE: Generalized Advantage Estimation.

3.9. Metrics and Reporting

The simulator records priority-level delivery and loss, latency, throughput, backlog, and battery, bandwidth, and CPU consumption during execution. These outputs allow service behavior and resource consumption to be examined separately. The cross-method numerical benchmark, however, summarizes priority-aware service quality through the weighted QoE measure and reports battery, bandwidth, CPU, total resource usage, and efficiency. The detailed service metrics remain simulator-recorded diagnostics rather than separate numerical aggregates in the cross-method summary.

Priority-weighted QoE is computed from the delivery rates of the three traffic classes:

$$QoE = \alpha_{cr} D_{cr} + \alpha_{im} D_{im} + \alpha_{no} D_{no}, \tag{6}$$

where D_{cr}, D_{im}, and D_{no} are the delivery rates for critical, important, and normal traffic, respectively, and α_{cr}, α_{im}, and α_{no} are their priority weights. For the reported experiments $\alpha_{cr} = 0.50$, $\alpha_{im} =$

0.30, and $\alpha_{no} = 0.20$. These normalized weights sum to one and preserve the intended service-priority ordering critical > important > normal. The weights are fixed across the compared methods and can be reconfigured for other application scenarios.

Resource usage is reported separately for battery, bandwidth, and CPU. A compact total resource-use score is also calculated:

$$R_{total} = \omega_b R_b + \omega_w R_w + \omega_c R_c, \tag{7}$$

where R_b, R_w, and R_c represent normalized battery, bandwidth, and CPU usage, respectively, and ω_b, ω_w, and ω_c are the corresponding resource-reporting weights. For the reported experiments, $\omega_b = 0.34$, $\omega_w = 0.33$, and $\omega_c = 0.33$. The coefficients sum to one and provide an approximately equal contribution from the three constrained resources, with 0.34 serving as the rounding remainder required for the coefficients to sum to one. The same coefficients are applied to all compared methods.

Bandwidth accounting includes both application-level transmission traffic and, for federated methods, model-weight exchange traffic. Let R_w^{app} denote the normalized bandwidth consumed by workload transmission, M_s the one-way model payload in mebibytes (MiB), G the number of federated rounds, C the number of clients, E the number of local episodes per client per round, N_d the number of drones per client, and B_{cap} the per-drone bandwidth capacity per episode. One client upload and one server broadcast per client give a recurrent model-exchange volume of $M_{FL} = 2GCM_s$. Because the testbed does not model a separate federated control channel, this volume is expressed as an equivalent normalized bandwidth overhead, $R_w^{FL} = M_{FL}/(GCEN_dB_{cap}) = 2M_s/(EN_dB_{cap})$, and the bandwidth term used for federated reporting is $R_w = \mathrm{clip}(R_w^{app} + R_w^{FL}, 0, 1)$. Under the reported configuration, $G = 116$, $C = 3$, $E = 8$, $N_d = 3$, and $B_{cap} = 50$ MiB, while the exchanged model payload is 1.058666 MiB for no-Mixer variants and 1.840916 MiB for Mixer variants. The resulting normalized overheads are 0.001764 and 0.003068, respectively. One-time initialization synchronization, where used, is included in the total training communication accounting but not in the final-window bandwidth metric because it does not occur during the final reporting rounds. No additional battery cost is inferred for model exchange because the testbed does not define an empirical radio-energy-per-byte coefficient for federated control traffic.

To summarize the service-resource trade-off, the protocol reports an efficiency score:

$$Eff = QoE - \lambda_R R_{total}, \tag{8}$$

where R_{total} is the aggregate normalized resource-use score and λ_R is the fixed resource-penalty coefficient. For the reported experiments, $\lambda_R = 0.60$. This coefficient is applied uniformly to all compared methods and is used only for the compact reporting scalarization; it does not replace the separate QoE, delivery, loss, latency, throughput, or resource-use measures.

The fixed coefficients in equations (6)–(8) are reporting design parameters rather than learned policy parameters. The QoE weights (0.50, 0.30, 0.20) encode the intended ordering critical > important > normal while retaining a non-zero contribution from all three traffic classes. The resource weights (0.34, 0.33, 0.33) approximately equal one-third weighting across battery, bandwidth, and CPU; the value 0.34 is the rounding remainder required for the three coefficients to sum to one rather than an additional preference for battery. The efficiency coefficient $\lambda_R = 0.60$ provides a fixed trade-off between normalized QoE and normalized aggregate resource use. Sensitivity to these reporting choices was examined post hoc using the recorded benchmark outputs. For the QoE weights, the alternative normalized settings (0.60, 0.25, 0.15) and (0.40, 0.35, 0.25) preserve the same priority ordering. Because each class-delivery rate lies in [0,1], either of these perturbations can change the scalar QoE by at most 0.10. Resource-efficiency sensitivity was additionally evaluated using battery-emphasized (0.44, 0.28, 0.28), bandwidth-emphasized (0.29, 0.43, 0.28), and CPU-emphasized (0.29, 0.28, 0.43) resource weights, together with $\lambda_R = 0.40$ and $\lambda_R = 0.80$.

For the learning-based methods, the simulator additionally returns a scalar cooperative reward for policy training. This training reward is distinct from the QoE, total-resource, and efficiency measures in equations (6)–(8), which are used for evaluation and reporting. Let $Q_i(t)$ denote the step-

level priority-weighted delivery quality of node i, $V_i^S(t)$ the priority-weighted value of successful transmissions, $V_i^D(t)$ the priority-weighted value of dropped transmissions, $U_i(t)$ the current weighted normalized resource usage, $\Delta U_i(t)$ the weighted incremental resource consumption during the current step, $r_i(t)$ the normalized RTT, and $I_i(t)$ an indicator equal to 1 when the node requests no transmission and 0 otherwise. For a population of N nodes, the cooperative reward returned by the environment is:

$$\mathcal{R}(t) = \sum_{i=1}^{N} \text{clip}_{[-r_{\max}, r_{\max}]} \left(\frac{\beta_Q Q_i(t) + \beta_s V_i^S(t) - \beta_d V_i^D(t) - \beta_u U_i(t) - \beta_{\Delta u} \Delta U_i(t) - \beta_r r_i(t) - \beta_0 I_i(t)}{\kappa} \right), \quad (9)$$

where $\beta_Q = 10, \beta_s = 1, \beta_d = 1, \beta_u = 8, \beta_{\Delta u} = 5, \beta_r = 2, \beta_0 = 0.5, \kappa = 50$, and $r_{\max} = 5$.

For priority class $p \in \{cr, im, no\}$, the step-level delivery component used in $Q_i(t)$ is the number of successfully transmitted items divided by $\max(1, n_i^p(t))$, where $n_i^p(t)$ is the requested transmission count defined in equation (5). The same priority ordering used by the QoE calculation is applied to these delivery components. The successful- and dropped-transmission values use weights 6, 3, and 1 for critical, important, and normal traffic, respectively. The resource terms are calculated as $U_i(t) = 0.02R_{b,i}(t) + 0.01R_{w,i}(t) + 0.01R_{c,i}(t)$ and $\Delta U_i(t) = 0.02\Delta R_{b,i}(t) + 0.01\Delta R_{w,i}(t) + 0.01\Delta R_{c,i}(t)$, where the incremental terms represent the fractions of the corresponding resource capacities consumed during the current decision step. The divisor of 50 and the clipping interval [-5,5] bound the reward magnitude used during learning.

Benchmark variability is evaluated across the five independent seeds using the sample standard deviation reported with each metric. Because the same seed set is used for all benchmark methods, the between-method differences are additionally examined using the Friedman repeated-measures test. Kendall's W is reported as the corresponding effect-size measure, and Holm correction is applied across the six metric-level omnibus tests to control for multiple testing. The statistical analysis is used to determine whether the common benchmark pipeline exposes the systematic differences among method behaviors under matched experimental conditions. Exhaustive pairwise significance ranking is not used because the purpose of the benchmark is testbed evaluation rather than algorithm-superiority testing.

3.10. Validation and Reproducibility Checks

The validation and reproducibility protocol evaluates whether the proposed testbed preserves the intended behavior across calibration, execution, benchmarking, profile extension, and large-scale profiling. First, the dataset-calibration process verifies how communication traces, flight/energy records, and UAV image workload data are transformed into simulator-facing variables. The simulator-used RTT and packet-loss values are compared with the imported communication trace to examine trace-statistics preservation. The communication validation uses 50 deterministic seeds, three drones, and 120 decision steps per seed, producing 18,000 simulator-used observations for comparison with the 45,289-record source trace. Preservation is assessed using the mean, standard deviation, 10th percentile (P10), median (P50), 90th percentile (P90), and the descriptive maximum distance between the source and simulator-used empirical cumulative distribution functions (Kolmogorov–Smirnov (KS) distance). During execution, the environment records battery, bandwidth, CPU, delivery, loss, latency, throughput, and backlog values to check whether both service-level and resource-level behavior are captured. The same protocol is also used to verify that centralized learning, federated learning, heuristic, and non-learning methods can be executed and reported through one environment pipeline. Finally, profile-extension checks examine homogeneous and mixed mobile IoT populations, while lightweight large-scale profiling reports the runtime, process-memory growth, and node-step execution throughputs.

The reproducibility controls used in the reported experiments include fixed random seeds, explicit dataset sources and paths, fixed episode and round settings, consistent state and action dimensions, common metric definitions, algorithm hyperparameters, matched centralized and federated training budgets, defined reporting windows, and the comma-separated values based logging of service and

resource outputs. The external communication, flight/energy, and image-workload datasets are identified through their original sources [33-36], while the benchmark configuration and experimental settings are summarized in tables 3–5.

4. Results

This section presents the experimental and validation results of the proposed mobile IoT simulation testbed. The results are organized according to dataset calibration, execution behavior, benchmark reporting, profile extension, and lightweight large-scale smoke testing.

4.1. Dataset-Calibration Results

The first validation step examined whether the empirical inputs were reflected in the simulator outputs. Figure 5 reports the trace-driven communication behavior derived from the imported drone communication dataset. The figure shows the time-varying RTT and packet-loss behavior used by the simulator during execution.

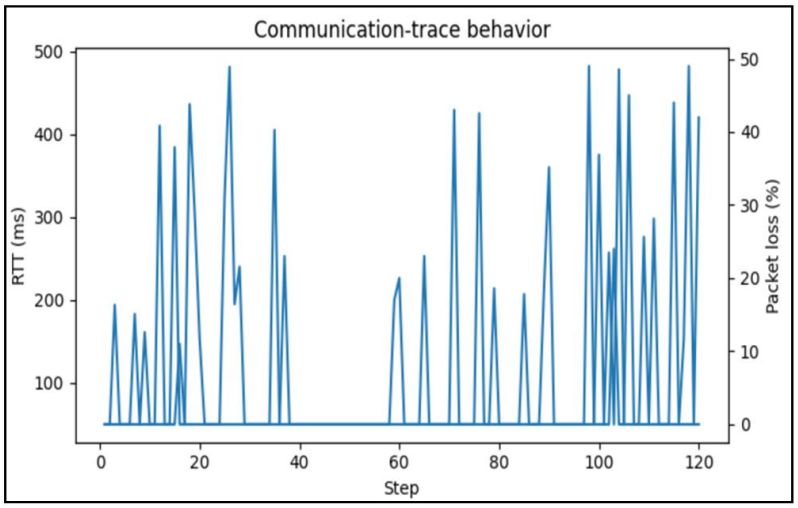

Figure 5: Trace-driven communication behavior.

Figure 6 reports the workload side of the calibration using the UAV image workload dataset. The calibration extracts image-size statistics and converts them into representative and capped workload-size values. These values are then used to derive the resource-cost multipliers for critical, important, and normal traffic.

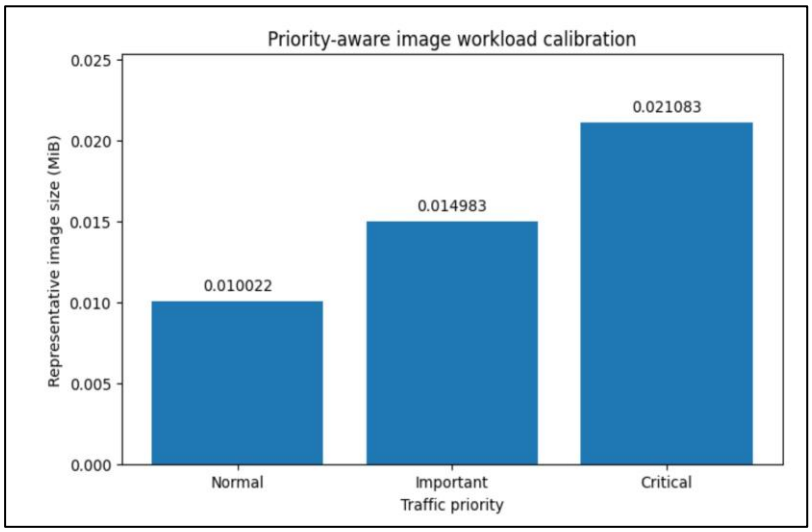

Figure 6: Image-workload calibration.

Table 6 compares the full source communication trace with the 18,000 observations sampled through simulator execution across the 50-seed validation. For RTT, the mean and standard-deviation differences were 0.475% and 0.809%, respectively; P10 and the median were unchanged, while P90 differed by 2.667%. For packet-loss probability, the mean and standard-deviation differences were 1.521% and 1.833%, respectively; P10 and the median remained zero, while P90 changed from 0 to 0.020. The empirical-distribution KS distances were 0.004071 for RTT and 0.002283 for packet-loss probability.

Table 6: Communication trace-statistics preservation between the source trace and simulator-used values.

Metric	Statistic	Full source trace	Simulator-used sample	Absolute difference (%)
RTT (ms)	Mean	100.021	100.497	0.475
RTT (ms)	Std	112.561	113.471	0.809
RTT (ms)	P10	50	50	0
RTT (ms)	Median (P50)	50	50	0
RTT (ms)	P90	300	308	2.667
Packet-loss probability	Mean	0.027792	0.028214	1.521
Packet-loss probability	Std	0.099757	0.101586	1.833
Packet-loss probability	P10	0	0	—
Packet-loss probability	Median (P50)	0	0	—
Packet-loss probability	P90	0	0.02	—

RTT: Round-Trip Time, ms: milliseconds, Std: Standard Deviation, P10: 10th percentile, P50: 50th percentile (median), P90: 90th percentile.

4.2. Execution and Resource-Behavior Results

The second validation step examined whether the simulator records resource behavior during priority-aware execution. Figure 7 reports the evolution of battery, bandwidth, and CPU usage during a representative run. The three resource dimensions are shown separately because the simulator records each constrained resource independently during transmission decisions.

The execution output also records queue-based delivery behavior. Pending workload decreases after successful transmissions, while unsuccessful or resource-blocked attempts are recorded as losses or dropped attempts. During each step, the simulator logs delivery, loss, backlog, latency, throughput, and resource-use values. These outputs show that the environment records both service-level and resource-level values during execution.

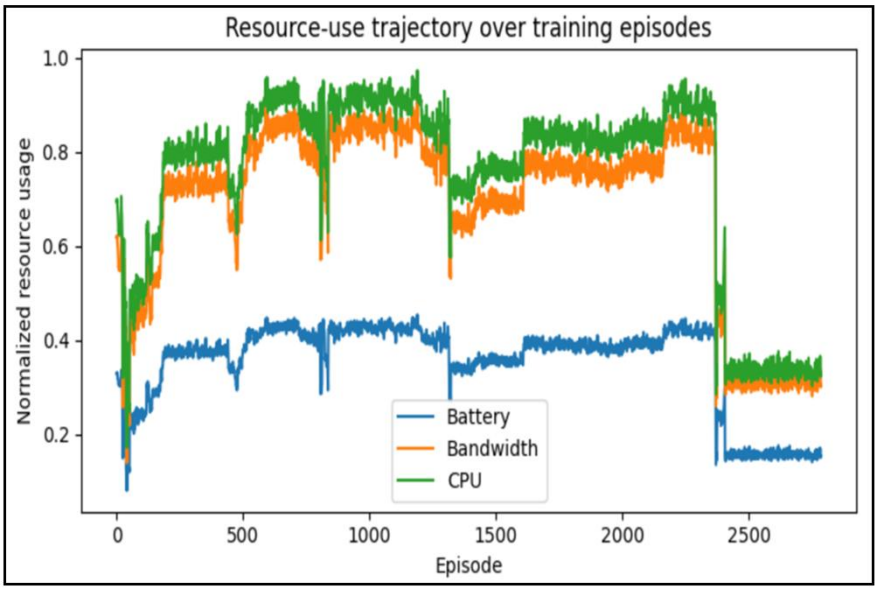

Figure 7: Multi-resource execution behavior.

4.3. *Benchmark Reporting Results Under the Dataset-Calibrated UAV Case*

The benchmark results examined whether centralized learning, federated learning, heuristic, and non-learning methods could be evaluated through the same simulator pipeline. Figure 8 illustrates the QoE-resource comparison, while table 7 summarizes the corresponding six cross-method measures: priority-weighted QoE, normalized battery usage, normalized bandwidth usage, normalized CPU usage, total resource usage, and efficiency. The simulator also records delivery, loss, latency, and throughput during execution; however, these detailed diagnostics are not presented as separate five-seed aggregates in the cross-method comparison.

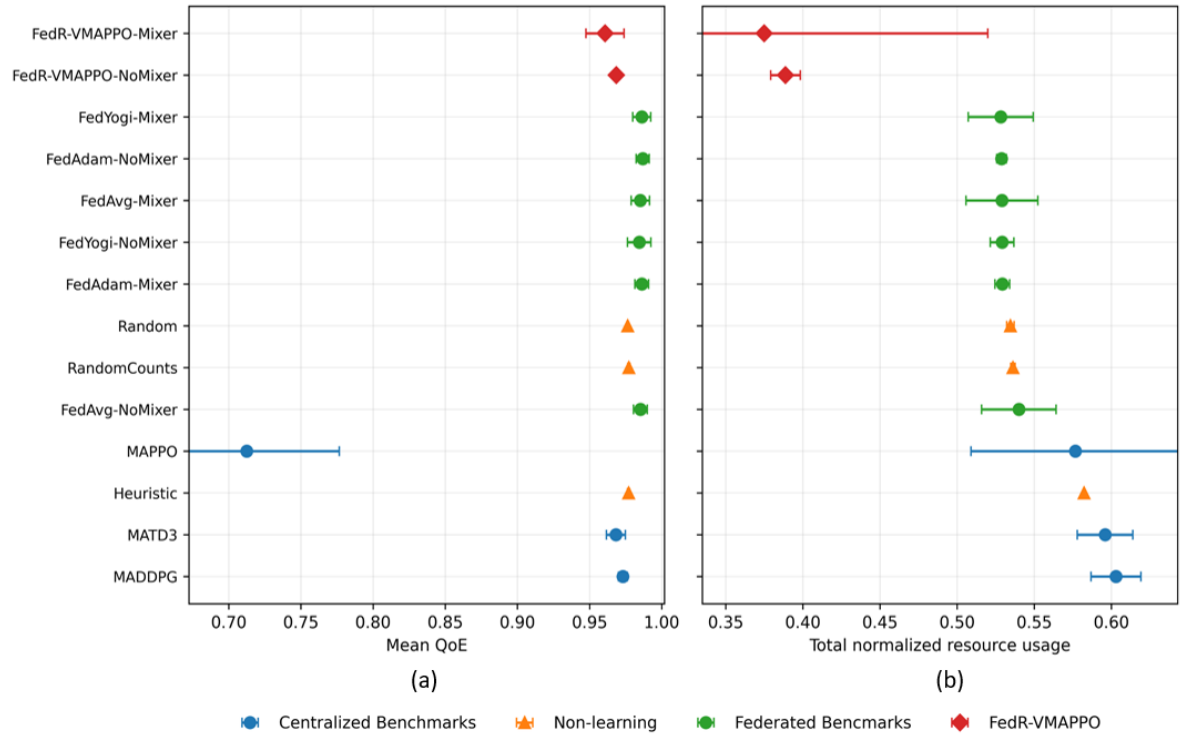

Figure 8: QoE-resource benchmark demonstration under the dataset-calibrated UAV case: (a) priority-weighted QoE; (b) total normalized resource usage.

Table 7 reports QoE, normalized battery usage, normalized bandwidth usage, normalized CPU usage, total resource usage, and efficiency for the evaluated method families. Each centralized learning run was summarized over its final 50 training episodes, while the heuristic and Random methods were summarized over their final 50 execution episodes. For federated methods, client-level metrics were first averaged within each communication round, followed by a summary over the final 10 rounds. The reported values represent the mean ± sample standard deviation of the resulting seed-level summaries across five independent seeds (0-4). No separate held-out train/test split or post-training frozen-policy evaluation was used; therefore, these results characterize late-stage benchmark behavior under the common simulator configuration rather than out-of-sample policy generalization.

Table 7: QoE-resource benchmark summary under the dataset-calibrated UAV case.

Algorithm	Method Family	QoE	Battery	Bandwidth	CPU	Total Resource	Efficiency
MAPPO	Centralized learning	0.723 ± 0.060	0.325 ± 0.057	0.622 ± 0.113	0.686 ± 0.122	0.542 ± 0.097	0.398 ± 0.097
MADDPG	Centralized learning	0.973 ± 0.003	0.355 ± 0.013	0.681 ± 0.033	0.752 ± 0.031	0.593 ± 0.025	0.617 ± 0.015
MATD3	Centralized learning	0.970 ± 0.006	0.355 ± 0.011	0.676 ± 0.026	0.747 ± 0.024	0.590 ± 0.020	0.615 ± 0.017
Heuristic	Heuristic method	0.977 ± 0.001	0.349 ± 0.000	0.666 ± 0.000	0.738 ± 0.000	0.582 ± 0.000	0.628 ± 0.001
Random	Non-learning	0.976 ± 0.001	0.323 ± 0.001	0.607 ± 0.003	0.682 ± 0.003	0.535 ± 0.003	0.655 ± 0.002
FedAvg-PPO (noMixer)	Federated learning	0.982 ± 0.007	0.322 ± 0.015	0.606 ± 0.033	0.679 ± 0.031	0.533 ± 0.026	0.662 ± 0.011
FedAvg-PPO (Mixer)	Federated learning	0.982 ± 0.009	0.317 ± 0.012	0.597 ± 0.029	0.669 ± 0.027	0.525 ± 0.023	0.667 ± 0.013
FedAdam-PPO (noMixer)	Federated learning	0.985 ± 0.006	0.320 ± 0.002	0.603 ± 0.004	0.675 ± 0.005	0.531 ± 0.004	0.667 ± 0.008
FedAdam-PPO (Mixer)	Federated learning	0.984 ± 0.006	0.320 ± 0.003	0.604 ± 0.005	0.675 ± 0.006	0.531 ± 0.004	0.666 ± 0.008
FedYogi-PPO (noMixer)	Federated learning	0.983 ± 0.008	0.322 ± 0.007	0.608 ± 0.017	0.680 ± 0.016	0.535 ± 0.013	0.662 ± 0.012
FedYogi-PPO (Mixer)	Federated learning	0.983 ± 0.008	0.319 ± 0.010	0.601 ± 0.023	0.672 ± 0.022	0.529 ± 0.018	0.666 ± 0.013
FedR-VMAPPO (noMixer)	Federated learning	0.968 ± 0.002	0.244 ± 0.016	0.453 ± 0.041	0.515 ± 0.035	0.402 ± 0.030	0.726 ± 0.020
FedR-VMAPPO (Mixer)	Federated learning	0.961 ± 0.012	0.237 ± 0.081	0.454 ± 0.151	0.502 ± 0.171	0.396 ± 0.133	0.724 ± 0.070

QoE: Quality of Experience, CPU: Central Processing Unit. Mixer and noMixer: Identify the benchmark configurations with and without CAR-Mixer coordinated value estimation, respectively.

Using the recorded benchmark outputs, a post hoc sensitivity analysis examined how the combined reporting results changed under alternative resource coefficients. Relative to the baseline resource weights (0.34, 0.33, 0.33), emphasizing battery with (0.44, 0.28, 0.28), bandwidth with (0.29, 0.43, 0.28), or CPU with (0.29, 0.28, 0.43) produced maximum absolute efficiency changes of 0.022, 0.008, and 0.014, respectively, across the benchmarked methods. The corresponding Spearman rank correlations with the baseline efficiency ordering were 1.000, 1.000, and 1.000. Changing the efficiency penalty from the baseline $\lambda_R = 0.60$ to 0.40 or 0.80 produced a maximum absolute efficiency change of 0.119. The corresponding rank correlations were 0.989 and 0.995, respectively. Thus, the absolute efficiency values respond to the selected resource penalty as expected, whereas the relative benchmark ordering remains highly stable under the tested perturbations. For the QoE weights, the linear form of equation (6) gives a maximum change of 0.10 under either tested priority-weight perturbation because the three delivery rates are bounded in [0,1]. The individual critical, important, and normal delivery rates are recorded by the simulator and used to calculate QoE, while table 7 presents their weighted aggregate.

The five-seed benchmark also showed statistically detectable differences among the evaluated methods under the common testbed configuration. Friedman tests remained significant after Holm correction for all six reported metrics, with the adjusted p-values not exceeding 0.000141 and Kendall's W values ranging from 0.678 to 0.851. For the three combined benchmark measures, the results were $\chi^2(12) = 44.782$, $p_{\text{Holm}} = 0.000056$, and $W = 0.746$ for QoE; $\chi^2(12) = 41.486$, $p_{\text{Holm}} = 0.000141$, and $W = 0.691$ for total resource use; and $\chi^2(12) = 51.033$, $p_{\text{Holm}} = 0.000006$, and $W = 0.851$ for efficiency. These omnibus results indicate that the benchmark pipeline captures systematic variation among the method behaviors across the matched seed conditions without requiring pairwise algorithm-ranking claims.

4.4. Profile-Extension Validation Results

The profile-extension validation examined whether the simulator could represent broader mobile IoT scenarios through the profile layer. Table 8 summarizes the supported profile types, their application interpretation, calibration status, and main simulation role.

Table 8: Mobile IoT node profile configuration summary.

Node profile	Application interpretation	Calibration status	Main simulation role
UAV / drone	Aerial monitoring and image uplink	Dataset-calibrated validated case	Main validated scenario
Vehicle	Vehicular sensing, road monitoring, and V2X-style uplink	Configurable profile extension	Tests road-mobile IoT assumptions
Wearable	Health, activity, and alert monitoring	Configurable profile extension	Tests low-power priority-sensitive devices
Ship	Maritime monitoring, navigation, and safety reporting	Configurable profile extension	Tests bandwidth-constrained maritime IoT assumptions
Mobile robot	Industrial inspection, warehouse automation, and robotics	Configurable profile extension	Tests compute-pressure mobile robotic workloads

IoT: Internet of Things, UAV: Unmanned Aerial Vehicle; V2X: Vehicle-to-Everything.

The profile-validation checks showed that the homogeneous UAV, vehicle, wearable, ship, mobile-robot, and mixed-node cases preserved the expected observation and action dimensions. The checked cases also passed the finite-output, resource-bound, and delivery/loss-bound checks.

4.5. Lightweight Large-Scale Smoke-Test Results

The large-scale smoke test examined simulator behavior as the number of mixed mobile IoT nodes increased. Table 9 reports representative results from 10 to 100,000 nodes using lightweight mode. All tested populations completed environment creation, reset, and one-step execution. The table reports the observation dimension, action dimension, environment creation time, reset time, one-step execution time, process-memory growth, approximate memory per node, and node-step execution throughputs.

Table 9: Large-scale simulator profiling in lightweight mode.

Nodes	Observation dimension	Action dimension	Create time (s)	Reset time (s)	Step time (s)	Memory increase (MiB)	Memory/node (KiB)	Node-step/s
10	70	30	0.469	0.003	0.007	74.363	7614.800	1514.968
50	350	150	0.470	0.009	0.014	68.453	1401.920	3554.418
100	700	300	0.439	0.014	0.034	66.426	680.200	2964.896
250	1750	750	0.512	0.023	0.046	66.039	270.496	5464.493
500	3500	1500	0.639	0.057	0.146	63.449	129.944	3433.290
1000	7000	3000	0.834	0.106	0.186	63.828	65.360	5373.204
10000	70000	30000	5.793	1.099	2.482	197.742	20.249	4029.563
50000	350000	150000	27.495	6.499	12.644	337.402	6.910	3954.434
100000	700000	300000	52.851	12.129	19.475	1633.512	16.727	5134.722

s: Seconds, MiB: Mebibyte, KiB: Kibibyte.

At 100,000 nodes, the simulator produced an observation dimension of 700,000 and an action dimension of 300,000. Environment creation took 52.851 s, reset took 12.129 s, and one-step execution took 19.475 s. The additional process memory was 1633.512 MiB, corresponding to approximately 16.727 KiB per node. The simulator recorded 5134.722 node-step updates per second.

Figure 9 presents the runtime, memory, and execution-throughput profile for the lightweight large-scale smoke test.

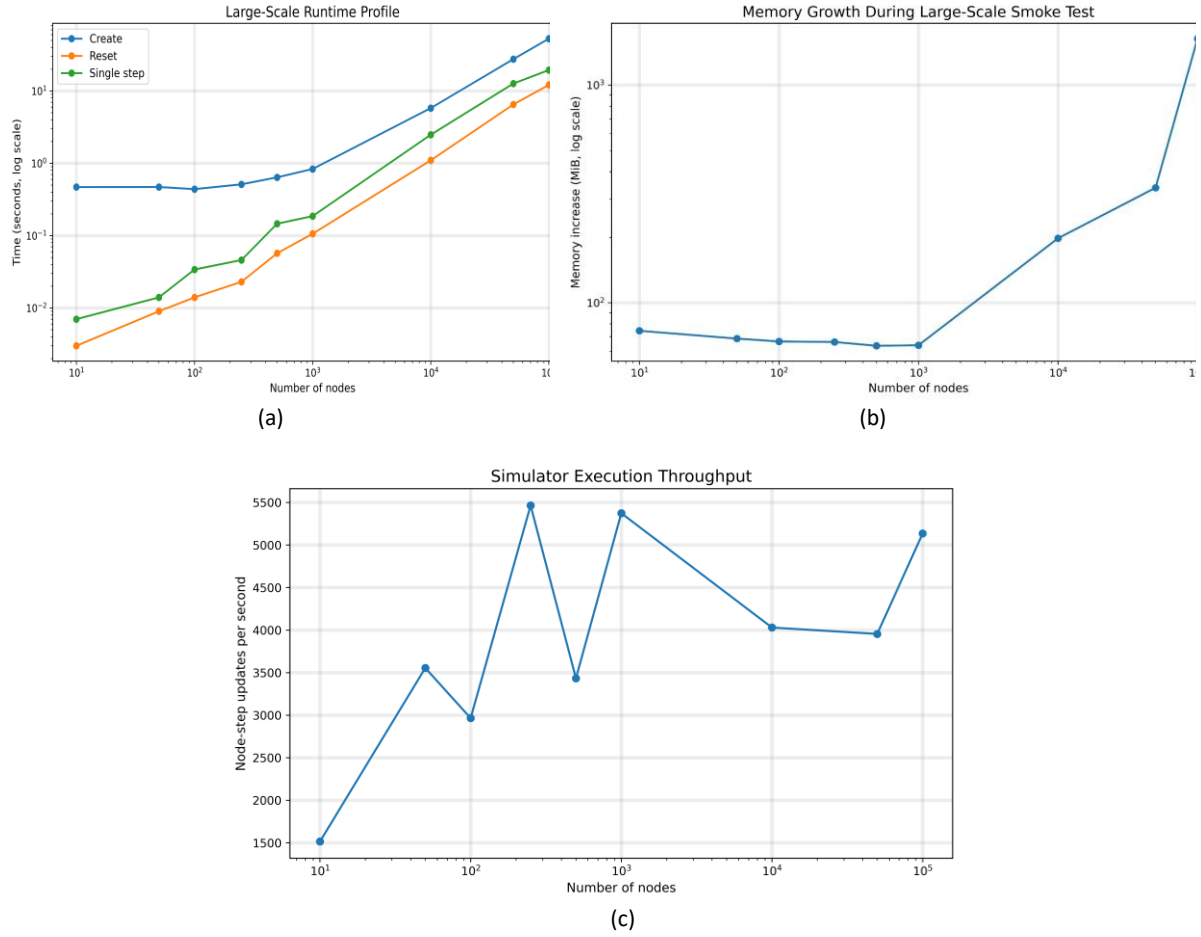

Figure 9: Lightweight simulator scalability profile: (a) environment creation, reset, and single-step execution time; (b) process-memory growth; and (c) node-step execution throughput.

5. Discussion

The results of this study show that the proposed simulation testbed can provide a repeatable and controlled environment for evaluating priority-aware resource allocation in mobile IoT networks. As shown earlier in figure 1, the testbed connects empirical inputs, profile abstraction, simulation execution, benchmarking, validation, and reporting within one workflow. This integrated structure is significant because mobile IoT resource allocation is affected by communication variation, battery limitation, bandwidth availability, CPU capacity, workload priority, and service-quality requirements. Therefore, the proposed testbed addresses an evaluation need rather than a single algorithmic task only.

The dataset-calibration results provide the first evidence that the validated UAV/drone case is connected to empirical behavior rather than being only a manually configured simulation. As shown in figure 5, the simulator uses trace-driven communication behavior to represent changes in RTT and packet-loss probability during execution. More specifically, table 6 shows mean and standard-deviation differences ranging from 0.475% to 1.833%, matching the P10 and median values for both communication metrics, and empirical-distribution KS distances of 0.004071 for RTT and 0.002283 for packet-loss probability. These results indicate the preservation of both descriptive and distributional trace characteristics and should not be interpreted as an exact prediction of every real wireless event. In practical terms, this means that the simulator can expose evaluated methods to communication variations grounded in the recorded communication dataset [33] while preserving repeatable execution, as demonstrated by the validation results in table 6. This supports the purpose of trace-driven simulation, where empirical records are used to make the environment more realistic without losing experimental control.

The workload-calibration result also strengthens the UAV/drone case because image-size information is used to influence transmission cost. As shown in figure 6, the UAV image workload is converted into representative and capped size values, which are then used to derive resource-cost multipliers for critical, important, and normal traffic. This is important because priority-aware image transmission should not treat traffic classes only as labels. The classes should also affect resource consumption during execution. By connecting image-workload statistics to battery, bandwidth, and CPU costs, the testbed makes workload behavior visible in both the service and resource metrics. However, this calibration remains specific to the UAV image dataset used in this study. Other mobile IoT domains may have different workload patterns [4-6], so future datasets are needed before the same level of empirical interpretation can be claimed for the vehicle, wearable, ship, or mobile-robot profiles.

The QoE-resource benchmark demonstrates why priority-aware service evaluation should not depend on QoE alone. As presented in table 7 and figure 8, centralized learning, federated learning, heuristic, and random methods were executed through the same simulator pipeline and reported using QoE, battery usage, bandwidth usage, CPU usage, total resource usage, and efficiency. The combined reporting shows that a method may achieve high QoE by consuming excessive resources, which may not be suitable for mobile IoT devices with limited energy and communication capacity. Battery, bandwidth, CPU, total resource use, and efficiency are calculated consistently across the method families, including method-specific communication overhead where applicable. Previous QoE-oriented studies [9-11] have shown that service usefulness can be assessed through multiple service and network factors rather than raw throughput alone. The present testbed follows this direction but adds explicit multi-resource reporting and matched method-family evaluation. Therefore, table 7 should not be read as an algorithm ranking only. It should be read as evidence that the testbed can expose service-resource trade-offs under common evaluation conditions.

The profile-extension results show that the testbed is not restricted to one fixed UAV setting. As presented in table 8, the simulator supports UAV/drone, vehicle, wearable, ship, and mobile-robot profiles. These profiles represent different application interpretations, resource capacities, and simulation roles, consistent with heterogeneous device modeling in existing edge/IoT simulators [17], while preserving the common observation and action dimensions defined in table 5. This supports the design choice of using a profile layer instead of building a separate simulator for every mobile IoT device type. At the same time, the interpretation must remain careful. Table 8 also clarifies that the UAV/drone profile is the dataset-calibrated validated case, while the other profiles are configurable scenario extensions. This distinction matters because mobile IoT domains differ when it comes to mobility behavior, communication conditions, traffic characteristics, and resource constraints. Therefore, the non-UAV profiles show extensibility, but do not yet establish empirical validity for those domains.

The lightweight large-scale smoke test provides evidence for simulator execution scalability, but the meaning of this result is limited to the tested execution mode. As presented in table 9, the simulator executed mixed-node populations ranging from 10 to 100,000 nodes in lightweight mode. At 100,000 nodes, the simulator recorded 1,633.512 MiB additional memory and 5,134.722 node-step updates per second. Figure 9 also shows the runtime, memory, and execution-throughput profile of this large-scale test. These results indicate that environment creation, reset, observation construction, action handling, and one-step execution can remain feasible at large node counts when repeated per-node image-cost scanning is disabled and communication traces are shared. However, this should not be interpreted as proof of full learning scalability, real wireless-network scalability, or physical deployment readiness. Large-scale learning requires additional tests involving training convergence, multi-round update time, communication overhead, and algorithm stability.

The proposed testbed is complementary to existing simulation tools rather than a replacement for them. Network simulators such as ns-3 [29] and OMNeT++ [30] support communication-network and protocol-level studies, whereas iFogSim [14], iFogSim2 [15], YAFS [16], and PureEdgeSim [17] support resource management, service placement, and orchestration across IoT, edge, fog, and cloud environments. SimulateIoT-Services further addresses IoT service and resource-allocation simulation [31]. Related resource-allocation studies [22–28] examine specific optimization, learning, or federated approaches under their respective evaluation settings. The proposed testbed addresses a more focused

methodological requirement by combining priority-aware transmission decisions, dataset-calibrated communication and workload behavior, battery-bandwidth-CPU resource accounting, and benchmarking across different method families in one common interface. This distinction identifies the contribution of the proposed testbed without implying overall performance superiority over existing simulation tools or allocation methods.

Several limitations remain. The current simulator abstracts physical-layer interference, detailed mobility dynamics, weather effects, exact routing behavior, and deployment-specific communication protocols. The UAV/drone case is the only empirically calibrated scenario, while the vehicle, wearable, ship, and mobile-robot profiles remain configurable extensions. The benchmark results are also influenced by the selected datasets, random seeds, episode settings, and evaluated methods. Future work should extend the dataset calibration to additional mobile IoT domains, evaluate longer learning runs under larger populations, and connect the testbed with packet-level or edge/fog simulators. This integration would allow future studies to combine the priority-aware resource-allocation interface developed in this work with more detailed communication and service-orchestration behavior.

6. Conclusions

This paper presented a trace-driven and dataset-calibrated simulation testbed for priority-aware resource allocation in mobile IoT networks. The testbed addresses the need for a repeatable evaluation environment that represents communication variation, traffic priority, and constrained battery, bandwidth, and CPU resources in one unified evaluation interface. The validated UAV/drone case integrates communication traces, flight/energy records, and UAV image-workload data, while the profile layer supports configurable extension to other mobile IoT node types.

The communication-trace validation showed that the simulator preserved the main statistical and distributional characteristics of the empirical input. The benchmark evaluation demonstrated that centralized learning, federated learning, heuristic, and Random methods could be compared through a common environment and reporting interface using QoE, resource-consumption, and efficiency measures. The simulator also supports detailed delivery, loss, latency, and throughput recording during execution. Furthermore, the profile-extension and lightweight scalability experiments demonstrated the flexibility of the simulation environment in representing different mobile IoT node configurations and executing larger mixed-node populations under the defined lightweight conditions.

The contribution of this work is a reusable simulation testbed rather than a physical deployment system or packet-level network simulator. The empirical interpretation is strongest for the UAV/drone case, while the non-UAV profiles remain configurable extensions that require future dataset calibration. Future work will extend the testbed with additional empirical datasets, longer learning experiments under larger heterogeneous populations, and possible integration with packet-level or edge/fog simulators. These extensions would strengthen the role of the testbed as a reusable evaluation environment for mobile IoT resource-allocation studies.

Author Contributions: Mohammad Esmail Ahmad: Conceptualization, Data curation, Formal Analysis, Investigation, Methodology, Resources, Software, Validation, Visualization, Writing – original draft. **Miran Taha Abdullah:** Supervision, Project administration, Methodology, Validation, Writing – review & editing

Data availability: The public datasets analyzed during the current study are available in the following repositories: Drone Communication Dataset, Kaggle, doi: 10.34740/KAGGLE/DSV/10702965; Data Collected with Package Delivery Quadcopter Drone, Carnegie Mellon University, doi: 10.1184/R1/12683453.v1; and UAV-Based Agricultural Image Dataset, Kaggle, https://www.kaggle.com/datasets/ziya07/uav-based-agricultural-image-dataset. The implementation code and generated simulation outputs are available from the corresponding author upon reasonable request, subject to institutional and doctoral-project requirements.

Conflicts of interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding: The authors did not receive support from any organization for the conducting of the study.

References

[1] S. Tuli *et al.*, "AI augmented edge and fog computing: Trends and Challenges," *Journal of Network and Computer Applications*, vol. 216, Art. no. 103648, 2023, doi: 10.1016/j.jnca.2023.103648.

[2] M. H. Alsharif *et al.*, "A comprehensive survey of energy-efficient computing to enable sustainable massive IoT networks," *Alexandria Engineering Journal*, vol. 91, pp. 12–29, 2024, doi: 10.1016/j.aej.2024.01.067.

[3] X. Xia, S. M. M. Fattah, and M. A. Babar, "A survey on UAV-enabled edge computing: Resource management perspective," *ACM Computing Surveys*, vol. 56, no. 3, Art. no. 78, pp. 1–36, 2024, doi: 10.1145/3626566.

[4] Naren, A. K. Gaurav, N. Sahu, A. P. Dash, G. S. S. Chalapathi, and V. Chamola, "A survey on computation resource allocation in IoT enabled vehicular edge computing," *Complex & Intelligent Systems*, vol. 8, pp. 3683–3705, 2022, doi: 10.1007/s40747-021-00483-x.

[5] C. Li, J. Wang, S. Wang, and Y. Zhang, "A review of IoT applications in healthcare," *Neurocomputing*, vol. 565, Art. no. 127017, 2024, doi: 10.1016/j.neucom.2023.127017.

[6] S. Aslam, M. P. Michaelides, and H. Herodotou, "Internet of ships: a survey on architectures, emerging applications, and challenges," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9714–9727, 2020, doi: 10.1109/JIOT.2020.2993411.

[7] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019, doi: 10.1109/JSAC.2019.2904348.

[8] X. Zheng, Y. Wu, L. Zhang, M. Tang, and F. Zhu, "Priority-aware path planning and user scheduling for UAV-mounted MEC networks: A deep reinforcement learning approach," *Physical Communication*, vol. 62, Art. no. 102234, 2024, doi: 10.1016/j.phycom.2023.102234.

[9] R. Herrero, "Multifactor QoE scores in IoT topologies," *Internet of Things*, vol. 24, Art. no. 100892, 2023, doi: 10.1016/j.iot.2023.100892.

[10] M. Taha, A. Ali, J. Lloret, P. R. L. Gondim, and A. Canovas, "An automated model for the assessment of QoE of adaptive video streaming over wireless networks," *Multimedia Tools and Applications*, vol. 80, no. 17, pp. 26833–26854, 2021, doi: 10.1007/s11042-021-10934-9.

[11] M. Taha, A. Canovas, J. Lloret, and A. Ali, "A QoE adaptive management system for high definition video streaming over wireless networks," *Telecommunication Systems*, vol. 77, no. 1, pp. 63–81, 2021, doi: 10.1007/s11235-020-00741-2.

[12] C. Sonmez, A. Ozgovde, and C. Ersoy, "EdgeCloudSim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, Art. no. e3493, 2018, doi: 10.1002/ett.3493.

[13] A. G. Ramonet, T. Pecorella, B. Picano, and K. Kinoshita, "Perspectives on IoT-oriented network simulation systems," *Computer Networks*, vol. 253, Art. no. 110749, 2024, doi: 10.1016/j.comnet.2024.110749.

[14] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in Internet of Things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017, doi: 10.1002/spe.2509.

[15] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments," *Journal of Systems and Software*, vol. 190, Art. no. 111351, 2022, doi: 10.1016/j.jss.2022.111351.

[16] I. Lera, C. Guerrero, and C. Juiz, "YAFS: A simulator for IoT scenarios in fog computing," *IEEE Access*, vol. 7, pp. 91745–91758, 2019, doi: 10.1109/ACCESS.2019.2927895.

[17] C. Mechalikh, H. Taktak, and F. Moussa, "PureEdgeSim: A simulation framework for performance evaluation of cloud, edge and mist computing environments," *Computer Science and Information Systems*, vol. 18, no. 1, pp. 43–66, 2021, doi: 10.2298/CSIS200301042M.

[18] C. Garrido-Hidalgo, L. Roda-Sanchez, F. J. Ramírez, A. Fernández-Caballero, and T. Olivares, "Efficient online resource allocation in large-scale LoRaWAN networks: A multi-agent approach," *Computer Networks*, vol. 221, Art. no. 109525, 2023, doi: 10.1016/j.comnet.2022.109525.

[19] E. Al-Masri, A. Souri, H. Mohamed, W. Yang, J. Olmsted, and O. Kotevska, "Energy-efficient cooperative resource allocation and task scheduling for internet of things environments," *Internet of Things*, vol. 23, Art. no. 100832, 2023, doi: 10.1016/j.iot.2023.100832.

[20] W. S. Costa *et al.*, "Planning and resource allocation of a hybrid IoT network using artificial intelligence," *Internet of Things*, vol. 26, Art. no. 101225, 2024, doi: 10.1016/j.iot.2024.101225.

[21] Y. Peng *et al.*, "An intelligent resource allocation strategy with slicing and auction for private edge cloud systems," *Future Generation Computer Systems*, vol. 160, pp. 879–889, 2024, doi: 10.1016/j.future.2024.06.045.

[22] X. Zhang, Z. Huang, L. Huang, and H. Yang, "Resource allocation strategy based on service function chaining in multi-access edge computing network," *Journal of King Saud University - Computer and Information Sciences*, vol. 36, Art. no. 102001, 2024, doi: 10.1016/j.jksuci.2024.102001.

[23] X. Li, P. Xiao, D. Tang, X. Li, Q. Wang, and D. Che, "UAVs-assisted QoS guarantee scheme of IoT applications for reliable mobile edge computing," *Computer Communications*, vol. 223, pp. 55–67, 2024, doi: 10.1016/j.comcom.2024.05.010.

[24] Z. Li, C. Xu, Z. Zhang, and R. Wu, "Deep reinforcement learning based trajectory design and resource allocation for task-aware multi-UAV enabled MEC networks," *Computer Communications*, vol. 213, pp. 88–98, 2024, doi: 10.1016/j.comcom.2023.11.006.

[25] Y. Zhang, Z. Na, Z. Wen, A. Nallanathan, and W. Lu, "Joint service caching, computation offloading and resource allocation for dual-layer aerial Internet of Things," *Computer Networks*, vol. 257, Art. no. 110974, 2025, doi: 10.1016/j.comnet.2024.110974.

[26] F. Liang, W. Yu, X. Liu, D. W. Griffith, and N. Golmie, "Toward deep Q-network-based resource allocation in industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9138–9150, 2022, doi: 10.1109/JIOT.2021.3093346.

[27] D. Loconte, S. Ieva, A. Pinto, G. Loseto, F. Scioscia, and M. Ruta, "Expanding the cloud-to-edge continuum to the IoT in serverless federated learning," *Future Generation Computer Systems*, vol. 155, pp. 447–462, 2024, doi: 10.1016/j.future.2024.02.024.

[28] A. S. M. S. Sagar, A. Haider, and H. S. Kim, "A hierarchical adaptive federated reinforcement learning for efficient resource allocation and task scheduling in hierarchical IoT network," *Computer Communications*, vol. 229, Art. no. 107969, 2025, doi: 10.1016/j.comcom.2024.107969.

[29] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Berlin, Germany: Springer, 2010, pp. 15–34, doi: 10.1007/978-3-642-12331-3_2.

[30] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in Proc. 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops (Simutools), Marseille, France, Art. no. 60, pp. 1–10, 2008, doi: 10.4108/ICST.SIMUTOOLS2008.3027.

[31] F. Díaz Cantero, J. Á. Barriga Corchero, M. Á. Pérez-Toledano, and P. J. Clemente, "A simulation framework for assessing and optimizing IoT service and resource allocation: SimulateIoT-Services," *Internet of Things*, vol. 34, Art. no. 101736, 2025, doi: 10.1016/j.iot.2025.101736.

[32] W. Chen, X. Qiu, T. Cai, H.-N. Dai, Z. Zheng and Y. Zhang, "Deep reinforcement learning for Internet of Things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1659–1692, 2021, doi: 10.1109/COMST.2021.3073036.

[33] Zenodi, "Drone communication dataset," *Kaggle*, 2025, doi: 10.34740/KAGGLE/DSV/10702965. [Accessed: Nov. 5, 2025]

[34] T. A. Rodrigues *et al*., "Data collected with package delivery quadcopter drone," *Carnegie Mellon University*, 2020, doi: 10.1184/R1/12683453.v1.

[35] T. A. Rodrigues *et al*., "In-flight positional and energy use data set of a DJI matrice 100 quadcopter for small package delivery," *Scientific Data*, vol. 8, Art. no. 155, 2021, doi: 10.1038/s41597-021-00930-x.

[36] Kaggle, "UAV-Based agricultural image dataset," 2025. [Online]. Available: https://www.kaggle.com/datasets/ziya07/uav-based-agricultural-image-dataset. [Accessed: Dec. 3, 2025]

[37] C. Yu *et al*., "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS 2022)*, New Orleans, USA, 2022, Art. no. 1787, pp. 1–14, doi: 10.5555/3600270.3602057.

[38] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, USA, 2017, pp. 6382–6393, doi: 10.5555/3295222.3295385.

[39] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, 2018, vol. 80, pp. 1587–1596, [Online]. Available: https://proceedings.mlr.press/v80/fujimoto18a.html.

[40] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data", in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, Florida, USA, 2017, vol. 54, pp. 1273–1282, [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html.

[41] X. Liu, Y. Deng, A. Nallanathan, and M. Bennis, "Federated learning and meta learning: Approaches, applications, and directions," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 1, pp. 571–618, 2024, doi: 10.1109/COMST.2023.3330910.

[42] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, USA, 2018, vol. 32, no. 1, pp. 3207–3214, doi: 10.1609/aaai.v32i1.11694.

[43] A. Patterson, S. Neumann, M. White, and A. White, "Empirical design in reinforcement learning," *Journal of Machine Learning Research*, vol. 25, Art. no. 318, pp. 1–63, 2024, [Online]. Available: https://jmlr.org/papers/v25/23-0183.html.