Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.079991

ARTICLE

Proactive Mobility-Aware Fog Service Continuity Using Digital Twins and GRU–EWMA-Based Association Forecasting

Navjeet Kaur[1], **Ayush Mittal**[2] **and Saad Alahmari**[3,*]

[1]Apex Institute of Technology (CSE), Chandigarh University, Mohali, Punjab, India
[2]Strategic Technology Group (STG), Infosys Ltd., Chandigarh, India
[3]Department of Computer Science, Applied College, Northern Border University, Arar, Saudi Arabia

*Corresponding Author: Saad Alahmari. Email: saad.alahmari@nbu.edu.sa

Received: 01 February 2026; Accepted: 24 March 2026; Published: 08 May 2026

ABSTRACT: Mobile fog computing must support latency-sensitive applications under dynamic user mobility and time-varying network conditions. Existing mobility-aware scheduling approaches are largely reactive and often ignore prediction uncertainty, resulting in service disruptions and inefficient task migration. This paper proposes an uncertainty-aware digital twin-based orchestration framework for proactive mobility-aware fog computing. The framework maintains real-time synchronized digital twins of users and fog nodes and integrates a hybrid Gated Recurrent Unit-Exponentially Weighted Moving Average (GRU-EWMA) mobility prediction model with fog-load forecasting to enable joint mobility- and load-aware decision-making. An entropy-based confidence mechanism is introduced to regulate proactive handover and task migration, thereby reducing unnecessary task migrations when predictions are uncertain. The proposed framework is implemented in the MobFogSim simulator and evaluated against state-of-the-art baselines. Experimental results demonstrate that the proposed approach reduces the average task delay by up to 28.1%, decreases energy consumption by up to 9.5%, and improves the task success rate to 99.1%, while incurring only a modest digital-twin computational overhead. These results confirm that integrating uncertainty-aware mobility prediction with digital twin–driven orchestration significantly enhances reliability and efficiency in mobile fog computing environments.

KEYWORDS: Fog computing; mobile edge computing (MEC); digital twin; proactive handoff; task migration; service continuity; gated recurrent unit (GRU); exponentially weighted moving average (EWMA); MobFogSim

1 Introduction

The rapid growth of latency-sensitive mobile applications like AR/VR, interactive analytics, and real-time sensing has intensified the need for reliable computation close to end users. Fog computing addresses this demand by extending cloud capabilities to the network edge through distributed fog nodes, reducing end-to-end latency and improving user experience. However, these benefits primarily hold under low user mobility. As a user moves, the wireless path to the serving fog node may experience degraded link quality and increased propagation delay. This leads to longer response times, service interruption, and missed deadlines if service placement remains static [1]. Consequently, mobility-aware offloading and service migration are essential for sustaining quality of service (QoS) in mobile fog environments [2].

A key difficulty is that mobility and fog congestion evolve simultaneously and unpredictably. Reactive policies that only respond after the user leaves coverage or after queues build up are prone to unavoidable

handoffs, ping-pong effects, and costly migrations. Moreover, decisions made solely from instantaneous measurements, such as RSSI and SNR, can be unreliable due to short-term fluctuations. Decisions made without accounting for near-future fog load may direct users toward congested nodes. Therefore, a practical control plane must (i) predict near-future user association tendencies and locations, (ii) anticipate fog-side congestion, and (iii) quantify uncertainty so that proactive actions are only triggered when predictions are sufficiently reliable.

Digital Twin (DT) technology has recently emerged as a promising paradigm for managing such complex, dynamic systems by maintaining a virtual, continuously synchronised replica of physical entities such as users, fog nodes, and network conditions with data-driven monitoring, simulation, and optimization [3,4]. While DTs were initially adopted in industrial contexts, they are increasingly used in networking and edge/fog computing environments to support latency-aware and context-aware decision-making [5]. Recent studies in [6–9] show that DT-assisted orchestration can improve real-time scheduling and resource allocation when accurate synchronisation and predictive analytics are available. Nevertheless, a gap remains in DT-enabled fog control for mobile users, as existing approaches often treat mobility prediction and fog load prediction separately. This shortcoming triggers proactive actions without an explicit uncertainty signal or lacks a clear hierarchical mechanism to scale decisions across local domains and inter-domain coordination.

This paper proposes a hierarchical DT-enhanced fog orchestration framework in which Local Fog Orchestrators (LFOs) manage fine-grained user and fog twins. The LFO executes real-time control, while a Global Fog Orchestrator (GFO) maintains an abstracted system-wide view for policy coordination and inter-domain support. Within each LFO domain, a predictive control pipeline is introduced that integrates (i) *GRAIN*, a GRU-EWMA hybrid predictor that stabilises noisy measurements via EWMA and encodes short-horizon temporal dynamics via a GRU; (ii) an entropy-based confidence score to quantify the certainty of predicted association tendencies; (iii) EWMA-based forecasting of near-future fog congestion indicators; and (iv) a mobility-load fusion decision rule to select the next serving fog node, followed by a feasibility check using the predicted user location. The resulting predictions and confidence are then used to trigger proactive handoff preparation and task migration once the user is predicted to leave the current fog coverage before task completion. To assess the QoS impact of decisions, a queueing-based response-time abstraction and an energy model for fog execution are employed.

1.1 Research Question, Hypothesis, and Contribution

Considering the challenge discussed above in mobility-aware task scheduling, the proposed work is presented to answer the identified research question: *Can a DT-driven, uncertainty-aware predictive control loop that jointly forecasts user mobility and fog congestion reduce service disruption and latency under mobility while keeping orchestration overhead acceptable?* Further, our hypothesis is that combining (a) noise-stabilised temporal mobility, (b) explicit confidence estimation, and (c) load-aware next-fog selection leads to more robust proactive actions than mobility-only or reactive baselines. Building upon the motivation of mobility-aware fog scheduling and migration, the main contributions of this paper are as follows:

- The proposed work's primary contribution is the design of a digital-twin–enabled fog orchestration framework that maintains synchronised "twin" states for synchronised fog nodes and uses those states to make proactive scheduling decisions. The focus is not merely on offloading but on building an orchestration layer that can continuously track mobility, link conditions, and fog-side congestion and then act in advance to reduce service disruption. The development of the GRAIN prediction component, which is an uncertainty-aware mobility and association predictor, is key to this effort. It combines

EWMA smoothing to stabilise noisy measurements with a GRU-based stable model to learn short-horizon dynamics, producing (i) a probabilistic association tendency over candidate fog nodes and (ii) a predicted future location.

- A third key contribution is the decision policy that fuses predicted mobility with predicted fog congestion to select the next fog node and trigger proactive migration and handoff. The model forecasts fog load using EWMA-based load prediction and combines it with the association tendency to compute a composite next-fog score.
- The proposed model improves average task delay, energy efficiency, and reliability while incurring only a small computational overhead on the Digital Twin (DT).

1.2 Key Novel Contributions beyond Existing Work

While there have been many investigations on the design and performance of mobility-aware scheduling, digital twin-aided orchestration, and learning-enabled offloading in the context of fog and edge computing paradigms, most have addressed each of these areas in isolation without explicitly considering the integration of mobility prediction, resource congestion modeling, and uncertainty-aware decision-making in an integrated manner [10–13]. Although reinforcement learning-based schedulers have been proposed for effective and adaptive offloading decisions in the context of edge computing paradigms, they have been observed to often start proactive decisions without considering the uncertainty in the prediction process [14,15]. Similarly, digital twin-aided frameworks have been proposed for monitoring and accelerating the optimization process in edge computing paradigms, without considering joint modeling and decision-making for mobility and load in an integrated manner [12,13]. The novelty in this work is not in the design and development of individual components but in the proposed uncertainty-aware decision mechanism that incorporates the joint modeling and decision-making in the context of user mobility, resource congestion in the fog environment, and uncertainty in the prediction process in an integrated manner using the digital twin control loop. Unlike existing works that have relied solely on predicting user mobility and uncertainty in the decision-making process, the proposed framework incorporates confidence-gated orchestration and mobility/load fusion to regulate proactive handoff and migration decisions in fog computing paradigms.

Prior work on service continuity in fog and MEC environments has typically addressed *mobility-aware migration*, *resource management*, and *digital-twin-assisted orchestration* as largely separate problems. Mobility-driven methods such as Follow-Me Cloud [16] and PROMO [17] anticipate user movement and support proactive re-association, but they do not explicitly combine mobility prediction with near-future fog congestion estimation in the final service-placement decision. Digital-twin-based approaches, on the other hand, improve monitoring and predictive control, but most do not jointly integrate (i) short-horizon mobility prediction, (ii) fog-side load forecasting, and (iii) uncertainty-aware gating of proactive migration decisions [18–20]. In contrast, the proposed framework combines a synchronized digital twin control loop, a hybrid GRU-EWMA mobility predictor (GRAIN), EWMA-based fog-load forecasting, and entropy-based confidence gating to support proactive and load-aware service continuity. Table 1 summarizes the main differences with representative prior studies.

Table 1: Comparison with representative state-of-the-art approaches.

Study	DT	Mobility	Congestion	Proactive
Taleb et al. [16]	No	Yes	No	Partial
PROMO [17]	No	Yes	No	Yes
Bozkaya [18]	Yes	Yes	Limited	Yes

(Continued)

Table 1 (continued)

Study	DT	Mobility	Congestion	Proactive
Sun et al. [19]	Yes	Limited	Yes	Partial
Wang et al. [20]	Yes	No	Limited	Partial
Proposed Work	Yes	Yes	Yes	Yes

1.3 Organization

The remainder of this paper is organized as follows. Section 2 reviews related work on mobility-aware task scheduling in fog/edge systems, learning-based offloading, and digital-twin–assisted orchestration. Section 3 presents the proposed DT-enhanced proactive scheduling framework, including the considered system architecture presented in Section 3.1, digital-twin construction and synchronization in Section 3.2, the GRAIN-based mobility prediction together with fog-load forecasting and mobility-load fusion for next-fog selection in Section 3.3, and the proactive handoff and task-migration control logic in Section 3.4; the overall scheduling process is summarized in Algorithm 1. The queueing-based execution-delay abstraction and fog execution-energy model used for cost estimation are detailed in Section 3.6. Section 4 formulates the objective function and defines the end-to-end latency and energy surrogates used by the scheduler. Section 5 describes the MobFogSim implementation, simulation settings in Section 5.1, and evaluation metrics in Section 5.3. Section 6 reports and discusses the experimental results. Finally, Section 7 concludes the paper and outlines future research directions, while Appendix A provides a consolidated notation table.

2 Literature Review

This section is divided into two major parts: Section 2.1 discusses the conventional mobility-aware scheduling literature, and the next Section 2.2 presents literature on digital twins.

2.1 Conventional, Learning-Based, and Mobility-Aware Scheduling in Edge Systems

Along with the development of fog and edge computing technologies, cybersecurity has become a significant issue in IoT-based systems. Recent literature emphasises that interpretable and reliable intrusion detection systems must be implemented to ensure transparent, reliable decision-making in complex systems. For example, it has been shown that a reliable intrusion detection system based on hybrid optimisation techniques and Local Interpretable Model-Agnostic Explanations (LIME) achieves high detection rates while maintaining explainability. However, these methods are mostly centralised and cannot be extended to distributed systems and decision-making under uncertainties [21,22].

Task scheduling and resource management in fog and edge computing have been widely studied in recent years. Early works addressed the joint optimisation of communication and computation resources. For example, Mao et al. [23] formulated a mobile-edge computing (MEC) offloading problem that jointly schedules tasks and allocates transmit power. The authors solve the problem via a heuristic that demonstrated the benefits of edge offloading over cloud-only processing. There are subsequent research that increasingly applies machine learning (ML) techniques to tackle the complexity of scheduling under dynamic conditions. Reinforcement learning (RL), in particular, has been popular for edge scheduling because it can learn adaptive policies. Hu et al. [14,24] proposed SPEAR, a deep Q-learning-based scheduler that considers dependency constraints among tasks to optimise their placement in distributed edge-cloud environments, achieving better makespan than heuristic baselines. Further, Zhou et al. [15] employed a deep RL approach

to schedule IoT tasks in a Space-Air-Ground Integrated Network (SAGIN) architecture with the goal of minimizing end-to-end delay. Their RL agent learns an offloading policy across satellite, aerial, and ground nodes, outperforming traditional algorithms in reducing latency. Similarly, Huang et al. [25] addressed deadline-aware task offloading in multi-access edge computing by modeling it as a partially observable Markov decision process and training an RL agent to meet task deadlines under uncertainty. This Partially Observable Markov Decision Process (POMDP)-based approach improved the deadline miss rate compared to greedy policies.

Given the highly dynamic and unpredictable nature of edge environments, such as user mobility and time-varying loads, several works have incorporated mobility awareness and predictive mechanisms into scheduling. Kaur et al. [17] presented PROMO, a PROactive MObility-support model for fog scheduling that anticipates user mobility and proactively assigns latency-sensitive tasks to appropriate fog nodes in advance. By leveraging trajectory prediction techniques, PROMO reduces service disruption as users move. Earlier, Zhu et al. [26] had introduced a "Fog Follow Me" strategy for vehicular fog computing, dynamically migrating or redistributing tasks among fog nodes to maintain low latency and service quality as vehicles travel. Similarly, Maleki and Mashayekhy [27] developed a mobility-aware offloading scheme that uses mobility prediction to decide whether to offload tasks to edge servers or wait, thereby reducing latency by preempting connectivity losses. In addition to software solutions, some researchers considered network handover coordination. Ngo et al. [28] proposed a coordinated container migration and base station handover strategy in MEC to maintain service continuity during user movement. Their method triggers service migration in tandem with cellular handovers, minimizing task interruptions for mobile users.

Another line of work focuses on multi-objective and meta-heuristic solutions for fog scheduling. Traditional scheduling must often balance multiple QoS metrics, leading to NP-hard optimization problems. Kaur et al. [29] formulated fog scheduling as a multi-objective optimization and proposed Task-Resource Adaptive Pairing (TRAP). TRAP uses a batching, ranking, and priority-based heuristic to pair tasks with fog nodes, reducing the search space and simultaneously minimizing delay, energy consumption, and cost. Implemented in iFogSim, TRAP achieved reductions in task delay and energy usage compared to naive scheduling. Liu et al. [30] combined bio-inspired algorithms for efficient scheduling by integrating Particle Swarm Optimisation (PSO) and Genetic Algorithm into an Artificial Bee Colony scheme. Their hybrid algorithm optimizes task allocation across fog clusters, reducing average service latency and energy consumption.

2.2 Digital Twin-Enabled Intelligent Orchestration and Emerging Work

While the above studies improve scheduling through optimization, learning, and mobility prediction, they generally do not use a continuously synchronized digital replica to support online orchestration decisions. In parallel, the concept of the Digital Twin (DT) has emerged as a promising tool for network optimization. A digital twin is a virtual replica of a physical system that is continuously synchronized with real-world data. Tao et al. [31] and Liu et al. [32] surveyed the state of the art in digital twins, highlighting their potential to provide real-time system insights and predictive analytics. The networking community has begun adopting DTs to manage edge resources under uncertainty. Sun et al. [19] introduced the concept of a Digital Twin Edge Network (DITEN) for 6G, where each edge server has a twin model estimating its state, and a system-level twin provides training data for an RL-based offloading decision engine. They formulated a latency minimization problem with migration cost constraints and solved it using an actor-critic deep RL approach with Lyapunov optimization for constraint handling. Wang et al. [20] specifically leveraged a digital twin to accelerate RL convergence for task scheduling. They proposed a DT-assisted Q-learning method where the agent can evaluate multiple actions in parallel within the twin

simulator. To this end, they developed two algorithms, Digital Twin-Assisted Asynchronous Q-learning (DTAQL) and Digital Twin-Assisted Exploring Q-learning (DTEQL), which showed faster convergence than standard Q-learning. Alourani et al. [33] propose a multi-layer, closed-loop smart-city architecture that unifies (i) IoT sensing and data acquisition, (ii) edge-level preprocessing and low-latency AI inference, (iii) privacy-preserving federated learning for distributed model training across heterogeneous devices, and (iv) a synchronized digital twin layer for simulation-driven decision support. The proposed framework is demonstrated on two representative urban domains, i.e., adaptive traffic management and underground pipeline monitoring. Further, Abdallah and Alghamdi [34] present a lightweight, decentralised traffic-signal optimization approach that couples a GRU-based predictor with a DT feedback loop. The GRU module forecasts short-horizon, local congestion events from vehicle-based IoT sensor streams, while the DT validates and adaptively adjusts control actions in response to live roadway-condition changes.

Beyond single-agent systems, multi-agent and federated learning approaches have been proposed to handle distributed edge environments. Zhang et al. [35] presented an adaptive multi-agent deep RL framework with digital twins for vehicular edge networks. In their approach, each vehicle is an agent that learns cooperative offloading policies, while a coordination graph and digital twin of the network help to evaluate joint actions efficiently. This method minimized overall offloading cost and latency by enabling agents to exploit both physical and twin network feedback. Arsalan et al. [36] focused on a federated learning setting for UAV-assisted edge computing. They proposed a DT-driven Federated Deep RL algorithm (DT-AFA) for coordinating task offloading among drones in smart agriculture. Each UAV runs a deep RL agent that decides on task offloading, transmission power, and local execution, while a cloud-based digital twin of the environment enables parallel policy evaluation, and a federated server aggregates the models from the UAVs By incorporating a semantic-aware reward design and leveraging both DT simulation and federated learning, their solution improved task success rates and lowered service migration overhead.

There is also growing interest in integrating security and other QoS aspects into edge scheduling. For instance, Kesavan et al. [37] developed a Secure Edge Enabled Multi-Task Scheduling (SEE-MTS) model for IoE applications using RL. Their framework not only schedules tasks to edge nodes but also employs encryption and dynamic key generation to ensure data security during offloading. A multi-task scheduling mechanism optimizes energy allocation and queue management, and a Q-learning based algorithm minimizes overall task completion time. The result improved energy efficiency and reduced delays while maintaining a high level of security.

Table 2 summarizes the key related works. Notably, most prior works focus on either learning-based scheduling or digital twin simulation, but few combine these with hierarchical control or explicit safety constraints. In our proposed approach, a similar task and resource management problem is addressed, but we introduce a unique combination of features: (i) a hierarchical RL scheduler with meta-controller and low-level optimizer that can efficiently handle large action spaces, (ii) integration of a Digital Twin as a predictive world model for short-horizon planning, and (iii) a constrained RL mechanism with guardrails to enforce hard QoS constraints at runtime. This agentic scheduler continuously adapts to system dynamics and outperforms both purely heuristic baselines and single-level RL in our evaluations. In the following Table 2, we detail how our approach builds on and differentiates from these prior works.

While existing literature addresses mobility prediction, digital twin orchestration, and fog scheduling individually, there is limited work that considers mobility prediction, congestion forecasting, and decision making under uncertainty within a comprehensive control framework.

Table 2: Summary of related work on task scheduling in fog/edge computing.

Ref.	Problem Addressed	Solution Approach	Performance Gains	Tools/Dataset
[23]	• MEC offloading • Limited bandwidth	• Joint task scheduling • Power allocation •Optimization heuristic	• Offloading latency ↓	• Simulation
[14]	• Edge scheduling • Task dependencies	• Deep Q-learning • SPEAR policy	• Makespan ↓ • Waiting time ↓	• Simulation
[15]	• IoT scheduling • SAGIN setting	• Deep RL • DQN-based scheduler	• End-to-end delay ↓	• Simulation
[25]	• Deadline-driven offloading • Partial observability	• Deep RL • POMDP formulation	• Deadline success rate ↑	• Simulation
[29]	• Multi-objective fog scheduling	• TRAP heuristic	• Delay ↓ • Energy ↓	• iFogSim • Synthetic workload
[17]	• Mobility-support fog scheduling	• PROMO proactive allocation • Mobility prediction	• Service latency ↓	• Simulation
[26]	• Vehicular fog (V2X) • Task allocation under mobility	• Dynamic task migration	• Latency ↓	• Vehicular fog testbed
[19]	• 6G edge offloading • Uncertainty-aware control	• Digital Twin Edge Network • Actor–critic RL • Lyapunov optimization	• Offload latency ↓	• MATLAB simulation
[38]	• IoT offloading • Satellite/UAV-assisted	• Learning-based offloading	• Throughput ↑ • Delay ↓	• Theory • Simulation
[35]	• Vehicular edge computing • Large-scale coordination	• Multi-agent deep RL • Digital Twins	• Cost ↓ • Latency ↓	• Veins/MATLAB co-simulation
[20]	• Edge scheduling • Large action space	• DT-assisted Q-learning	• Convergence speed ↑	• Python simulator
[30]	• IoT–fog resource scheduling	• Hybrid meta-heuristic • ABC + PSO + GA	• Delay ↓ • Energy ↓	• Simulation
[36]	• UAV-assisted edge computing • Agriculture deployments	• Federated deep RL • DT-based environment	• Task success ↑ • Delay/Energy ↓	• Python simulator
[37]	• Secure multi-task scheduling • Internet of Everything	• RL-based scheduling • Encryption integration	• Energy efficiency ↑ • Security ↑	• Prototype
[33]	• Infrastructure management • Latency & privacy in IoT	• Hybrid AI-IoT-DT framework • CNN/LSTM + FL • DT simulation	• Response time ↓ • Maintenance cost ↓ • False positives ↓	Simulation
[34]	• Traffic signal optimization • Sustainable urban mobility	• GRU congestion forecasting • DT validation Signal optimisation	• Predictive accuracy ↑ • Latency ↓ • CO_2 emissions ↓	Simulation
Our Work	• DT-enhanced fog scheduling • Mobility-aware QoS reliability	• Digital Twin planning • Confidence-gated prediction • Load-aware control	• Delay ↓ • Energy ↓ • Success rate ↑ • DT overhead ↓	• MobFogSim-based simulation

Further, existing approaches to fog scheduling can be classified as mobility-aware heuristic, learning-based, or digital twin-based [14,15,19,20]. Despite their common goal, significant differences can be observed between these approaches in how they address mobility prediction, resource awareness, and uncertainty handling. A conceptual comparison between some state-of-the-art approaches and the proposed approach is presented in Table 3.

Table 3: Conceptual comparison with state-of-the-art approaches.

Method	Mobility Prediction	Load Prediction	Digital Twin	Uncertainty Modeling	Joint Mobility-Load Decision
TRAP	×	✓	×	×	×
PROMO	✓	×	×	×	×
RL-based scheduling	✓	✓	Partial	×	Partial
DT-based scheduling	Partial	✓	✓	×	Partial
Proposed framework	✓	✓	✓	✓	✓

3 Proposed Framework

3.1 System Overview

The proposed fog computing architecture with an integrated digital-twin (DT) control plane is shown in Fig. 1. The system is organized into three tightly-coupled layers: (i) a *physical execution layer*, (ii) a *digital twin layer*, and (iii) a *hierarchical orchestration layer* comprising a Local Fog Orchestrator (LFO) and a Global Fog Orchestrator (GFO). Let $\mathcal{U}$ denote the set of users mobile devices and $\mathcal{F}$ is the set of fog nodes.

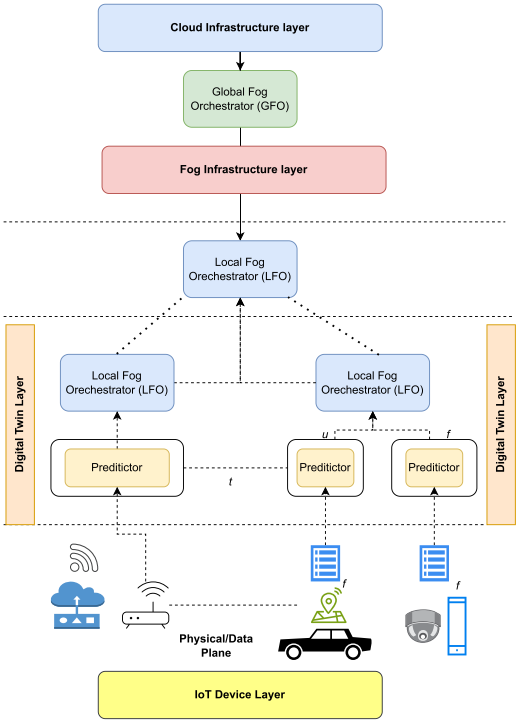


Figure 1: System architecture.

End-user device $\{u_i\}$ where $u_i \in \mathcal{U}$ generate computational tasks $t_i(t)$ over discrete time $t \in \{1, 2, \dots\}$. Each task can be executed on a nearby fog node or offloaded to the cloud data centre. The serving association of user u_i at time t is denoted by $a_i(t) \in \mathcal{F} \cup \{\text{cloud}\}$, and the set of candidate fog nodes within the communication range of user u_i is $\mathcal{N}_i(t) \subseteq \mathcal{F}$. Above the physical layer, the DT layer maintains synchronized virtual replicas of key entities where each physical user u_i is mirrored by a *user twin* $\tilde{u}_i$, and each fog node f_j

is mirrored by a *fog twin* $\tilde{f}_j$. The DT layer is updated through periodic *state synchronization* with a sampling period δ. Concretely, at each discrete time t, the user side and fog side generate telemetry that is mirrored into the corresponding twins, i.e., $\mathbf{o}_i^u(t)$ and $\mathbf{o}_j^f(t)$.

Further, the *Local Fog Orchestrator (LFO)* performs real-time control within its local fog domain, while the *Global Fog Orchestrator (GFO)* provides system-wide policy and coordination. The GFO exchanges *Policy & Coordination* messages with each LFO and interacts with the cloud data centre for long-horizon optimisation, overflow execution, and model lifecycle support. Within each local domain, the LFO: (i) receives synchronized DT states, (ii) derives prediction signals for mobility and fog load, and (iii) issues control actions to the physical layer, including target fog selection for offloading, proactive handoff triggering, and task migration between fog nodes. The DT layer exposes bidirectional coupling between *User Twins* and *Fog Twins*, allowing mobility tendencies to be evaluated jointly with service feasibility. Finally, the LFO forwards compact summaries such as predicted next association, confidence, and predicted fog load indicators, upward to the GFO to enable cross-domain coordination and cloud-assisted decisions when users move beyond a local LFO's coverage.

3.2 Digital Twin Construction and State Synchronization

In the proposed hierarchy given in Fig. 1, the Fog Orchestrator (FO) is realized by (i) a set of LFOs deployed on selected fog nodes, and (ii) a GFO deployed in the cloud. Each LFO manages the digital twins within its domain and executes real-time control. In the digital twin model, each physical entity has a corresponding twin. For each user device $u_i \in \mathcal{U}$, the LFO maintains a user twin $\tilde{u}_i$ with state vector

$$S_i^u(t) = \big(L_i(t),\, v_i(t),\, tl_i(t),\, a_i(t),\, \mathrm{RSSI}_i(t),\, \mathrm{SNR}_i(t)\big), \tag{1}$$

where $L_i(t) \in \mathbb{R}^2$: the geographical location of user u_i at time t, represented as a two-dimensional coordinate. $v_i(t) \in \mathbb{R}^2$ is the velocity vector of user u_i at time t, representing both speed and direction of movement, $tl_i(t)$ is the current task load at user u_i, representing the amount of computation pending execution or offloading, $a_i(t) \in \mathcal{F} \cup \{\text{cloud}\}$ is the the serving association of user u_i at time t, indicating the fog node or cloud instance currently responsible for processing the user's tasks, $\mathrm{RSSI}_i(t)$ is the received signal strength indicator observed by user u_i from its serving node at time t, and $\mathrm{SNR}_i(t)$ is the signal-to-noise ratio experienced by user u_i at time t. Similarly, for each fog node $f_j \in \mathcal{F}$, the LFO maintains a fog twin $\tilde{f}_j$ with state vector

$$S_j^f(t) = \big(\rho_j(t),\, M_j(t),\, B_j(t),\, Q_j(t),\, P_j(t)\big), \tag{2}$$

where $\rho_j(t) \in [0,1)$ is the CPU utilization of fog node f_j at time t, $M_j(t)$ is the available memory at fog node f_j at time t, $B_j(t)$ is the available network bandwidth at fog node f_j for serving connected users at time t, $Q_j(t)$ is the queue length at fog node f_j at time t, representing the number of tasks waiting for execution and directly influencing waiting delay, $P_j(t)$ is the power consumption of fog node f_j at time t, measured in watts. The state synchronization of both twins is performed in discrete time with period δ; thus, at each time t, the LFO holds the mirrored states of the users and fog nodes in its domain.

3.3 GRAIN-Based Mobility and Next-Fog Association Prediction

At each decision epoch t, the Local Fog Orchestrator (LFO) executes the following sequence. First, it forms the user-twin input state $S_i^u(t)$ from synchronized mobility and link-quality features given in Eq. (1). Second, these inputs are smoothed by EWMA via Eq. (3) with factor $\alpha_x = 0.3$ to reduce short-term wireless and mobility noise. Third, the most recent $h = 10$ smoothed states are processed by a one-layer GRU encoder

with hidden dimension $d_h = 64$ to obtain the temporal embedding $\mathbf{h}_i(t)$, given in Eq. (5). Fourth, the model predicts (i) a probability distribution over candidate fog nodes $p_i^{(k)}(t+\Delta)$ using a softmax output layer in Eq. (6) and (ii) the future user location $\widehat{L}_i(t+\Delta)$ given in Eq. (7) over a lookahead horizon of $\Delta = 30$ steps. Fifth, the normalized entropy of the predicted association distribution is converted into a confidence score $c_i(t+\Delta)$, as given in Eq. (8); proactive actions are allowed only when $c_i(t+\Delta) \geq c_{\min}$ with $c_{\min} = 0.6$. In parallel, the LFO forecasts fog-side utilization and queue state using EWMA with $\alpha_z = 0.4$, given in Eq. (9). Finally, the next serving fog is selected by minimizing the composite mobility-load score $\Gamma_{i,k}(t+\Delta)$, as presented in Eq. (10), where the fusion weights are set to $\beta_\rho = 1.0$ and $\beta_Q = 0.5$, and $\widehat{Q}$ is normalized by $Q_{\max} = 50$.

Mobility and fog-service dynamics are modelled in discrete time. At time t, user i is associated with a serving node $a_i(t) \in \mathcal{F} \cup \{\text{cloud}\}$ and has a candidate set $\mathcal{N}_i(t) \subseteq \mathcal{F}$. The Local Fog Orchestrator (LFO) runs GRAIN, which combines (i) EWMA smoothing for noisy mobility/link observations and (ii) a GRU encoder for learning short-term temporal dependencies. In parallel, the LFO applies EWMA over fog-twin states to forecast near-future load indicators.

For each user i, the LFO constructs the input state vector by using synchronized states values from user-twin $S_i^u(t)$ as per Eq. (1). These features jointly characterize both physical movement and communication quality, forming the raw input to the temporal learning module. To suppress short-term fluctuations, each input dimension is smoothed using EWMA, as given in Eq. (3).

$$\tilde{\mathbf{S}}_i^{\mathbf{u}}(t) = \alpha_x \, \mathbf{S}_i^{\mathbf{u}}(t) + (1-\alpha_x)\, \tilde{\mathbf{S}}^{\mathbf{u}}{}_i(t-1), \qquad 0 < \alpha_x \leq 1. \tag{3}$$

where $\tilde{\mathbf{S}}_i^{\mathbf{u}}(t)$ denotes the EWMA-smoothed state vector, and α is the smoothing factor which controls the trade-off between responsiveness and stability. The model assumed a fixed value for α to be 0.3, as it provides a favourable trade-off between responsiveness and noise suppression. Generally, a smaller α_x increases smoothing but introduces lag in handoff and migration decisions, whereas a larger α_x reacts quickly but transmits more measurement noise to the GRU input. Section 6.2 analyzes how α_x and α_z for fog-load EWMA impacts end-to-end QoS and migration behavior.

At time t, the model does not rely only on the latest EWMA-smoothed state $\tilde{\mathbf{S}}^{\mathbf{u}}{}_i(t)$. Instead, it forms a history window of the most recent h smoothed states as shown in Eq. (4).

$$\tilde{\mathbf{S}}_i^{\mathbf{u}}(t-h+1),\ \tilde{\mathbf{S}}_i^{\mathbf{u}}(t-h+2),\ \ldots,\ \tilde{\mathbf{S}}_i^{\mathbf{u}}(t). \tag{4}$$

This window captures the recent evolution of the user's mobility and link conditions, including changes in location and speed, as well as temporal variations in wireless quality. Hence, the GRU operates on a short trajectory segment together with its corresponding link-quality evolution, rather than a single instantaneous snapshot. Let h denote the history length. The GRU encoder produces a latent embedding as given in Eq. (5).

$$\mathbf{h}_i(t) = \phi_\theta\big(\tilde{\mathbf{S}}_i^{\mathbf{u}}(t-h+1{:}t)\big), \tag{5}$$

where $\phi_\theta(\cdot)$ denotes a GRU-based sequence encoder which takes the last h EWMA-smoothed input vectors $\tilde{\mathbf{x}}_i(t-h+1{:}t)$ and compresses them into a fixed-length representation. The parameters θ are the learnable weights of the GRU that are trained from data.

Given $\mathbf{h}_i(t)$, the model predicts a probability distribution $p_i^{(k)}$ over candidate fog nodes to predict the most likely next association of the user by obtaining a comparable score across candidates, presented in Eq. (6).

$$p_i^{(k)}(t+\Delta) = \Pr\ (a_i(t+\Delta) = f_k \mid \mathbf{h}_i(t)) = \text{softmax}_{k \in \mathcal{N}_i(t)}\ (W_o \mathbf{h}_i(t) + \mathbf{b}_o)\,, \tag{6}$$

where $a_i(t+\Delta)$ is the random serving-node association of user i at the future time $t+\Delta$, f_k denotes the k-th candidate fog node in the feasible set $\mathcal{N}_i(t)$, $p_i^{(k)}(t+\Delta)$ is the predicted probability that the future association equals f_k, i.e., $a_i(t+\Delta)=f_k$, conditioned on the current temporal embedding $\mathbf{h}_i(t)$, the term $W_o\mathbf{h}_i(t)+\mathbf{b}_o$ is a linear output layer that maps the embedding $\mathbf{h}_i(t)$ into a vector of unnormalized scores (logits), one score per candidate fog node. Finally, softmax$(\cdot)$ converts these logits into probabilities over the candidate set. In addition, the model predicts the Δ-step-ahead location, given in Eq. (7).

$$\widehat{L}_i(t+\Delta)=\psi_\eta\big(\mathbf{h}_i(t)\big), \tag{7}$$

where $\psi_\eta(\cdot)$ is a regression head with parameters η.

The GRU output in Eq. (6) provides a probability distribution over candidate fog nodes, but the distribution may give high certainty or high uncertainty. Therefore, a scalar confidence score is computed to quantify how reliable the predicted association tendency is. This confidence is useful for (i) deciding whether to trigger proactive handoff or migration, (ii) avoiding unnecessary switching when the prediction is ambiguous, and (iii) enabling risk-aware coordination, e.g., forwarding only high-confidence events to the GFO. Let $\mathbf{p}_i(t+\Delta)=\{p_i^{(k)}(t+\Delta)\}_{f_k\in\mathcal{N}_i(t)}$ denote the probability vector. A highly peaked distribution yields low entropy, indicating strong certainty, whereas a near-uniform distribution yields high entropy, indicating ambiguity. To make entropy comparable across different candidate-set sizes, it is normalised by the maximum possible entropy $\log|\mathcal{N}_i(t)|$. The resulting confidence $c_i(t+\Delta)$ is defined in Eq. (8).

$$c_i(t+\Delta)=1-\frac{H(\mathbf{p}_i(t+\Delta))}{\log|\mathcal{N}_i(t)|},\qquad H(\mathbf{p})=-\sum_{f_k\in\mathcal{N}_i(t)}p^{(k)}\log p^{(k)}. \tag{8}$$

The confidence score $c_i(t+\Delta)$ in Eq. (8) lies in $[0,1]$ and indicates how strongly the predictor favors a single next fog node. The values close to 1 mean the predicted next-fog distribution is very *peaked*, while smaller values mean the prediction is more spread out and uncertain. Let $p_{\max}(t+\Delta)\triangleq\max_{f_k\in\mathcal{N}_i(t)}p_i^{(k)}(t+\Delta)$ be the largest predicted probability among the candidate fog nodes. In the LFO, proactive handoff preparation and task migration are triggered only when the prediction is sufficiently confident, i.e., when $c_i(t+\Delta)\geq c_{\min}$ as in Eq. (24). The value of $c_{\min}$ is set as 0.6 as a conservative operating point, so the control plane reacts only when the prediction is strongly concentrated.

After estimating the user's mobility-driven association tendency and its confidence, the LFO must also ensure that the selected fog node can actually serve the user efficiently. A fog node that is likely to be the next association, i.e., high $p_i^{(k)}$ may still be a poor choice if it is expected to be congested in the near future. Hence, the LFO forecasts near-future fog load indicators from synchronized fog-twin states. These forecasts are later fused with the mobility tendency to make a load-aware next-fog decision. Therefore, for any scalar fog metric $z_j(t)$ like utilization $\rho_j(t)$ or queue length $Q_j(t)$, the LFO maintains an EWMA state $\widehat{z}_j(t)$ and updates the one-step-ahead forecast as given in Eq. (9).

$$\widehat{z}_j(t+1)\;=\;\alpha_z\,z_j(t)+(1-\alpha_z)\,\widehat{z}_j(t),\qquad \alpha_z\in(0,1], \tag{9}$$

here, α_z controls the responsiveness of the forecast where larger α_z reacts faster to sudden load changes, while smaller α_z provides stronger smoothing and greater stability.

So, up to this point, the LFO has obtained two complementary signals: (i) a mobility-driven association tendency $\{p_i^{(k)}(t+\Delta)\}$ from the GRU, as given in Eq. (6), indicating which fog node the user is most likely to move toward, and (ii) near-future congestion forecasts such as $\widehat{\rho}_k(t+\Delta)$ and $\widehat{Q}_k(t+\Delta)$ from fog-level EWMA, as given in Eq. (9), indicating which fog nodes are expected to be heavily loaded. The goal now

is to combine these two signals to make a final, load-aware next-fog decision that avoids both unnecessary handoffs and congested nodes. Hence, for each candidate $f_k \in \mathcal{N}_i(t)$, the LFO defines the mobility-load score $\Gamma_{i,k}$ as given in Eq. (10). Fig. 2 represents the proactive handoff and task migration with digital twin.

$$\Gamma_{i,k}(t+\Delta) \;=\; -\log p_i^{(k)}(t+\Delta) \;+\; \beta_\rho\, \widehat{\rho}_k(t+\Delta) \;+\; \beta_Q\, \widehat{Q}_k(t+\Delta), \tag{10}$$

where $-\log p_i^{(k)}(t+\Delta)$ penalizes candidates that are unlikely according to the GRU. If $p_i^{(k)}$ is high (mobility-preferred), then $-\log p_i^{(k)}$ is small; if $p_i^{(k)}$ is low, the penalty becomes large. Thus, $-\log p_i^{(k)}(t+\Delta)$ encourages selecting fog nodes that match the predicted user movement. $\beta_\rho\, \widehat{\rho}_k(t+\Delta)$ discourages selecting a node predicted to have high utilization. Since queueing delay grows rapidly as utilization approaches 1, this term supports delay-aware decisions. Moreover, $\beta_Q\, \widehat{Q}_k(t+\Delta)$ discourages selecting nodes with predicted queue build-up, capturing short-term congestion that may not be fully reflected by utilization alone. Here, the weights $\beta_\rho, \beta_Q \geq 0$ control the trade-off between following mobility preference and avoiding congestion. Finally, the next serving fog node is chosen by minimizing the composite score:

$$\widehat{\text{nextFog}}(i, t+\Delta) = \arg\min_{f_k \in \mathcal{N}_i(t)} \Gamma_{i,k}(t+\Delta). \tag{11}$$

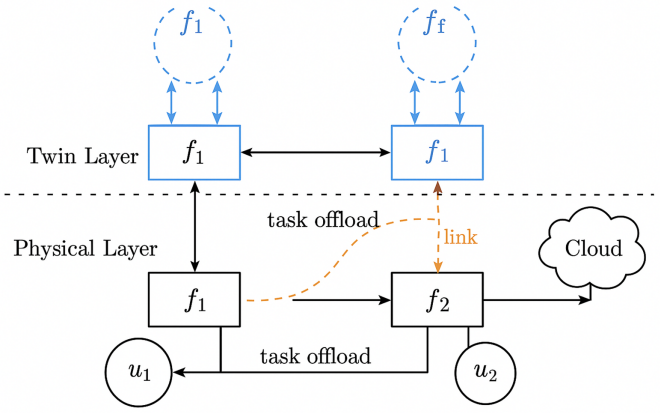


Figure 2: Proactive handoff and task migration under predicted mobility.

This rule selects the candidate that is simultaneously (i) likely from the mobility perspective and (ii) predicted to be less congested from the load perspective.

Finally, the predicted future location $\widehat{L}_i(t+\Delta)$ from Eq. (7) is used for a lightweight feasibility check. The LFO verifies that the selected node $\widehat{\text{nextFog}}(i, t+\Delta)$ provides coverage at $\widehat{L}_i(t+\Delta)$. If the coverage constraint is violated, the LFO falls back to the best-scoring candidate in $\mathcal{N}_i(t)$ that is feasible under the predicted location.

3.4 Mobility-Aware Handoff & Proactive Task Migration Control Logic

The digital-twin layer acts as a cognitive control plane for the fog system by continuously synchronizing user and fog states and enabling proactive decisions. In particular, the previous section (Section 3.3) provides (i) the predicted future location $\widehat{L}_i(t+\Delta)$ and (ii) the predicted next serving fog $\widehat{\text{nextFog}}(i, t+\Delta)$ with confidence. This subsection formalizes how these predictions are used to (i) detect an impending handoff and (ii) trigger task migration when a task is unlikely to finish before the user leaves the current fog coverage.

Let $a_i(t) = f_j$ denote the current serving fog node for user i at time t, where $f_j \in \mathcal{F}$. The user-fog distance is given in Eq. (12).

$$d(i, j, t) = \left\| L_i(t) - L_j \right\|, \tag{12}$$

where L_j is the fixed location of fog node f_j. The wireless link to f_j is feasible only when the user is within the coverage radius R_j. The communication latency is also modelled between user i and fog node f_j as given in Eq. (13).

$$\ell_{i,j}(t) = \begin{cases} \ell_{i,j}^{\text{finite}}(t), & \text{if } d(i, j, t) \le R_j, \\ \infty, & \text{if } d(i, j, t) > R_j, \end{cases} \tag{13}$$

where $\ell_{i,j}^{\text{finite}}(t)$ is the finite latency when coverage holds. Using the mobility forecast $\widehat{L}_i(t + \Delta)$, the LFO checks whether the current serving fog remains feasible at the lookahead horizon, presented by Eq. (14).

$$d(i, j, t + \Delta) = \left\| \widehat{L}_i(t + \Delta) - L_j \right\| > R_j \quad \Rightarrow \quad \text{handoff required within } [t, t + \Delta]. \tag{14}$$

When Eq. (14) holds, the current link to f_j is predicted to become infeasible, i.e., $\ell_{i,j}(t + \Delta) = \infty$, and the system prepares to hand off to the predicted next fog node $\widehat{\text{nextFog}}(i, t + \Delta)$.

Similarly, for proactive task migration, consider a task t_k currently executing on the serving fog $f_j = a_i(t)$. Let $\widehat{T}_{k,j}^{\text{rem}}(t)$ denote the predicted remaining execution time of task t_k on f_j at time t, and let $\widehat{\tau}_i^{(j)}(t)$ denote the predicted remaining dwell time of user i within the coverage of f_j. A proactive migration of task t_k from f_j to the predicted next fog f_ℓ, where $f_\ell = \widehat{\text{nextFog}}(i, t + \Delta)$, is triggered as per Eq. (15), ensuring that migration is initiated only when the task is unlikely to finish before the user leaves the current fog coverage, thereby reducing forced restarts and deadline misses under mobility.

$$m_{k,\, j \to \ell}(t) = 1 \iff \left(\widehat{\text{nextFog}}(i, t + \Delta) = f_\ell \ \wedge \ \widehat{T}_{k,j}^{\text{rem}}(t) > \widehat{\tau}_i^{(j)}(t) - T_{j \to \ell}^{\text{mig}} \right), \tag{15}$$

where $T_{j \to \ell}^{\text{mig}}$ is the migration overhead.

3.5 Algorithm

This section presents the Algorithm 1 of the proposed model. Lines 1–2, the algorithm begins a new control epoch t and immediately synchronizes the digital twins (Line 2). This step ensures the LFO's decisions are based on the latest mirrored user and fog states, which is essential because both mobility and load can change significantly between epochs. In Lines 3–5, the LFO updates near-future fog congestion estimates for each fog node using the EWMA forecasting rule in Eq. (9). This produces lightweight forecasts that approximate the level of busyness each fog node will experience over the next decision horizon. These forecasts are required because choosing the "mobility-preferred" fog alone can lead to high queueing delay when the preferred node is about to become congested. Further, in Lines 6–10, for each user, the LFO first constructs the feasible candidate set $\mathcal{N}_i(t)$ using coverage at the current time (Line 7). If $\mathcal{N}_i(t)$ is empty in Lines 8–10, the algorithm assigns the user to the cloud as a safe fallback, because no local fog can currently serve the user. Next in Lines 11–16, the LFO constructs the instantaneous user feature vector (Line 11; Eq. (1)) and applies EWMA smoothing (Line 12; Eq. (3)) to suppress short-term fluctuations in mobility/link measurements. The smoothed vector is appended to a rolling history buffer (Line 13). If the history length is still shorter than h (Lines 14–16), the GRU encoder cannot yet form a reliable temporal embedding, so the algorithm conservatively keeps the current association until sufficient history

is accumulated. In lines 17–20, the LFO computes the GRU embedding $\mathbf{h}_i(t)$ (Line 17; Eq. (5)), then uses it to predict (i) the association tendency distribution over candidates (Line 18; Eq. (6)) and (ii) the future user location $\widehat{L}_i(t+\Delta)$ (Line 19; Eq. (7)). Finally, it computes the confidence score (Line 20; Eq. (8)). This confidence quantifies whether the predicted association tendency is sharply concentrated, i.e., high certainty or diffuse, i.e., high uncertainty, which is critical for controlling when proactive actions are triggered. In Lines 21–25, for each candidate fog node, the LFO computes the composite mobility-load score (Lines 21–23; Eq. (10)), which penalizes unlikely mobility choices while also penalizing predicted congestion. The next fog is selected as the candidate with the minimum composite score (Line 24; Eq. (11)). The algorithm then performs a feasibility check against the predicted future location (Line 25), i.e., if the chosen fog does not cover $\widehat{L}_i(t+\Delta)$, the algorithm falls back to the best-scoring feasible candidate. The LFO checks whether a handoff is likely within the lookahead horizon using the coverage-loss condition in Eq. (14) (Line 26) and gates the action by confidence ($c_i(t+\Delta) \geq c_{\min}$). If both conditions hold, the LFO triggers proactive handoff preparation (Line 27) toward the selected $\widehat{\text{nextFog}}$. In Line 29–33, for each ongoing task currently executing on the present fog, the LFO checks whether the migration condition holds (Lines 30–31; Eq. (15)). If so, it triggers migration to the predicted next fog (Line 31). This step directly supports service continuity: tasks are migrated only when they are unlikely to complete before the user leaves the current fog's coverage, and only toward the LFO-selected next fog. Finally, in lines 35–36, at the end of the epoch, the LFO may send compact summaries to the GFO, enabling higher-level coordination across domains. The flow chart is also presented via Fig. 3.

Algorithm 1: LFO-side twin-driven control (Minimal, Equation-Referenced).

Require: Synchronized twin states $\{S_i(t)\}_{i\in\mathcal{U}}$, $\{S_j^f(t)\}_{f_j\in\mathcal{F}}$ (sync period δ); history length h, lookahead Δ; EWMA factors (α_x, α_z); fusion weights (β_ρ, β_Q); confidence threshold $c_{\min}$; trained GRAIN components $(\phi_\theta, W_o, \mathbf{b}_o, \psi_\eta)$; per-user history buffers $\{\mathcal{H}_i\}$.
Ensure: Next-fog decision $\widehat{\text{nextFog}}(i, t+\Delta)$, confidence $c_i(t+\Delta)$, and handoff/migration triggers.

1: **for** each decision epoch t **do**
2: **Synchronize twins:** read current $\{S_i(t)\}$ and $\{S_j^f(t)\}$.
(A) Fog-side load forecasting (EWMA)
3: **for** each fog node $f_j \in \mathcal{F}$ **do**
4: Update near-future load forecasts (e.g., $\widehat{\rho}_j$, $\widehat{Q}_j$) using Eq. (9).
5: **end for**
(B) User-side mobility prediction (EWMA + GRU) and decision
6: **for** each user $i \in \mathcal{U}$ **do**
7: Build candidate set $\mathcal{N}_i(t)$ using current coverage (based on user location and fog radii).
8: **if** $\mathcal{N}_i(t) = \varnothing$ **then**
9: Set $\widehat{\text{nextFog}}(i, t+\Delta) \leftarrow$ cloud; **continue**.
10: **end if**
11: Construct feature vector $\mathbf{x}_i(t)$ from user-twin state (Eq. (1)).
12: EWMA-smooth inputs to obtain $\tilde{\mathbf{x}}_i(t)$ (Eq. (3)).
13: Append $\tilde{\mathbf{x}}_i(t)$ to history buffer $\mathcal{H}_i$; keep only the last h samples.
14: **if** $|\mathcal{H}_i| < h$ **then**
15: Use current association $a_i(t)$ (or cloud fallback) until enough history is available; **continue**.
16: **end if**
17: Compute GRU embedding $\mathbf{h}_i(t)$ from $\mathcal{H}_i$ (Eq. (5)).

(Continued)

Algorithm 1 (continued)

Predict association distribution $\{p_i^{(k)}(t+\Delta)\}$ over $\mathcal{N}_i(t)$ (Eq. (6)).
Predict future location $\widehat{L}_i(t+\Delta)$ (Eq. (7)).
Compute confidence $c_i(t+\Delta)$ (Eq. (8)).
for each candidate $f_k \in \mathcal{N}_i(t)$ **do**
    Compute composite mobility–load score $\Gamma_{i,k}(t+\Delta)$ (Eq. (10)).
end for
Select $\widehat{\text{nextFog}}(i, t+\Delta)$ via score minimization (Eq. (11)).
Feasibility check: if selected fog does not cover $\widehat{L}_i(t+\Delta)$, fallback to best feasible candidate; if none, fallback to cloud.

(C) Proactive handoff and migration triggers

if handoff predicted for current serving fog within lookahead (Eq. (14)) **and** $c_i(t+\Delta) \geq c_{\min}$ **then**
    Trigger proactive handoff preparation toward $\widehat{\text{nextFog}}(i, t+\Delta)$.
end if
for each ongoing task t_k executing on current fog $a_i(t)$ **do**
    if migration condition holds (Eq. (15)) **then**
        Trigger task migration $m_{k,j\to\ell}(t) \leftarrow 1$ (to $f_\ell = \widehat{\text{nextFog}}(i, t+\Delta)$).
    end if
end for
end for
Forward compact summaries (nextFog, confidence, trigger flags) to the GFO.
end for

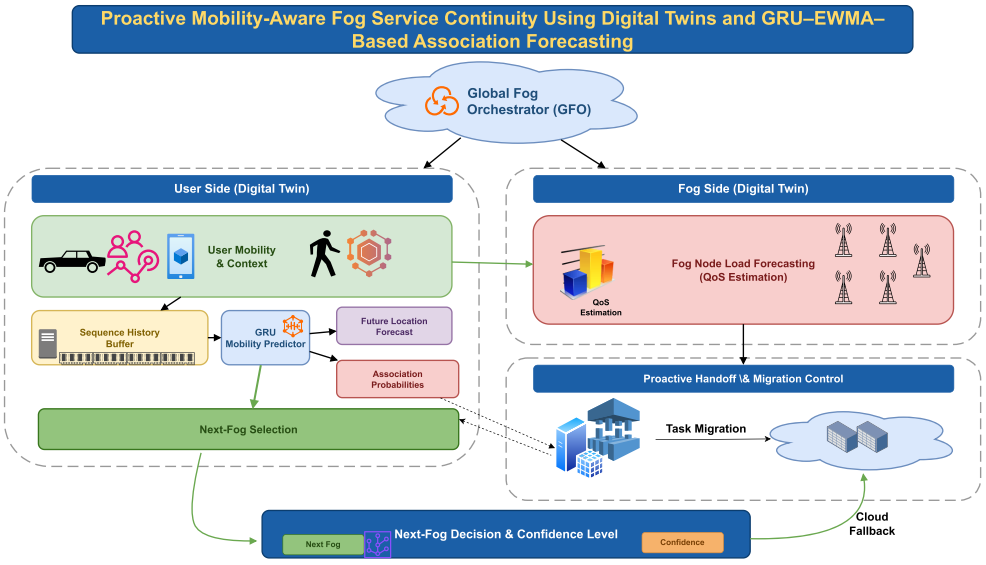


Figure 3: Work flow of the proposed model.

3.6 Fog Execution Delay and Energy Model

Each fog node f_j is modeled as an $M/G/1$ queue to capture load-dependent waiting delay under heterogeneous task sizes. Let λ_j denote the aggregate task arrival rate to f_j (tasks/s) during a control epoch,

and let C_j be the processing capacity of f_j (CPU cycles/s). For a task indexed by v with workload W_v CPU cycles, the service time on f_j is defined as,

$$T_{v,j}^{\text{svc}} = \frac{W_v}{C_j}. \tag{16}$$

Let T_j^{svc} be the generic service-time random variable at node j induced by the workload distribution of tasks assigned to f_j. The utilization of node j is presented

$$\rho_j = \lambda_j \mathbb{E}\left[T_j^{\text{svc}}\right], \qquad 0 \le \rho_j < 1, \tag{17}$$

where ρ_j is dimensionless. The stability condition $\rho_j < 1$ is critical: as $\rho_j \to 1$, the expected queueing delay grows rapidly, reflecting congestion and increased deadline misses.

Using $M/G/1$ allows the scheduler to account for both (i) average load through ρ_j and (ii) workload variability through the second moment $\mathbb{E}\left[(T_j^{\text{svc}})^2\right]$. Variability matters in fog systems because a mix of small and large tasks can lead to long waiting times even when the mean load is moderate.

By the Pollaczek–Khinchine formula, the mean response time, which includes waiting and service time at node j is presented as,

$$\mathbb{E}\left[T_j^{\text{resp}}\right] = \mathbb{E}\left[T_j^{\text{svc}}\right] + \frac{\lambda_j \mathbb{E}\left[(T_j^{\text{svc}})^2\right]}{2(1-\rho_j)}. \tag{18}$$

The first term is the mean processing time, while the second term is the mean waiting-time component that increases with arrival rate λ_j, service-time variability, and proximity to saturation ($\rho_j \to 1$).

For a specific task v, its execution time is approximately f_j as its own processing time plus the node-level mean queueing term,

$$T_{v,j}^{\text{exec}} \approx T_{v,j}^{\text{svc}} + \frac{\lambda_j \mathbb{E}\left[(T_j^{\text{svc}})^2\right]}{2(1-\rho_j)}. \tag{19}$$

This approximation is computationally efficient for online scheduling, where the scheduler can evaluate multiple candidate fog nodes using the current DT-estimated load (λ_j, ρ_j) while still respecting that each task has its own workload W_v.

Further, the energy is computed for executing task v on f_j is modeled as,

$$E_{v,j}^{\text{exec}} = P_j(\rho_j)\, T_{v,j}^{\text{exec}}, \tag{20}$$

i.e., energy equals power times time. Here, $P_j(\rho_j)$ is the average power draw of node j under utilization ρ_j. The model adopts a linear power-utilisation model, presented as,

$$P_j(\rho_j) = P_j^{\text{idle}} + \left(P_j^{\text{peak}} - P_j^{\text{idle}}\right)\rho_j. \tag{21}$$

This model captures a key fog characteristic, even when lightly loaded, edge servers consume non-trivial idle power, and energy increases with utilization. Together, Eqs. (19)–(21) provide a tractable, load-aware cost model used later in the end-to-end latency, energy surrogates and in the scheduling objective.

4 Objective Function

The proposed DT-driven LFO operates in discrete time with synchronization period δ, meaning that at each epoch t it observes an updated virtual mirror of both the users and the fog nodes. This synchronized view provides (i) the current coverage context via user and fog locations and (ii) the current congestion context via fog utilization/queue states. However, under mobility and time-varying load, decisions based only on instantaneous observations can be short-sighted. For this reason, the LFO additionally leverages the GRAIN predictors from Section 3.3 where the future user location $\widehat{L}_i(t+\Delta)$, presented in Eq. (7) is captures and the user is expected to be at the look-ahead horizon, $\widehat{\text{nextFog}}(i, t+\Delta)$, presented in Eq. (11) captures the most suitable next serving node after mobility-load fusion, and the confidence score $c_i(t+\Delta)$, presented in Eq. (8), quantifies whether the association tendency estimate is reliable enough to justify proactive actions. Accordingly, the LFO's control action at epoch t is expressed through two decision structures, i.e., the assignment matrix $\mathbf{X}(t)$ for newly-arrived tasks, and the migration tensor $\mathbf{M}(t)$ for ongoing tasks that may require relocation to preserve service continuity.

To tightly couple the optimization to the DT predictors while keeping the decision variables simple, a small set of binary feasibility indicators is pre-computed. The indicator $\chi_{i,j}(t)$ in Eq. (22) enforces the physical constraint that a task can be offloaded to a fog node only when the user is currently within its coverage. The indicator $\chi_{i,j}(t+\Delta)$ in Eq. (23) extends this logic to the future by using the predicted location $\widehat{L}_i(t+\Delta)$, enabling the model to anticipate imminent coverage loss. Since proactive migration can be harmful when predictions are uncertain, the confidence gate is introduced $g_i(t+\Delta)$ in Eq. (24), which permits proactive actions only when $c_i(t+\Delta)$ exceeds a threshold. Finally, the selector $\pi_{i,\ell}(t+\Delta)$ in Eq. (25) ties migration destinations to the LFO's predicted next fog, presented in Eq. (11), ensuring that migration decisions remain consistent with the mobility-load fusion policy rather than allowing arbitrary destination choices.

$$\chi_{i,j}(t) \triangleq \mathbb{1}\big[d(i,j,t) \le R_j\big], \quad \text{with } d(i,j,t) \text{ from Eq. (12)}, \tag{22}$$

$$\chi_{i,j}(t+\Delta) \triangleq \mathbb{1}\big[\big\|\widehat{L}_i(t+\Delta) - L_j\big\| \le R_j\big], \quad \text{with } \widehat{L}_i(t+\Delta) \text{ from Eq. (7)}, \tag{23}$$

$$g_i(t+\Delta) \triangleq \mathbb{1}[c_i(t+\Delta) \ge c_{\min}], \quad \text{with } c_i(t+\Delta) \text{ from Eq. (8)}, \tag{24}$$

$$\pi_{i,\ell}(t+\Delta) \triangleq \mathbb{1}\big[f_\ell = \widehat{\text{nextFog}}(i, t+\Delta)\big], \quad \text{with } \widehat{\text{nextFog}}(i, t+\Delta) \text{ from Eq. (11)}, \tag{25}$$

where $c_{\min} \in [0,1]$ is a confidence threshold. In this study, $c_{\min}$ is kept *static* across simulation scenarios to keep comparisons reproducible and to isolate the contribution of uncertainty gating.

Given a placement decision, the end-to-end latency of a task consists of communication delay and execution delay given in Eq. (19). When a migration is triggered, an additional migration overhead is incurred, which is explicitly added in the latency surrogate $T_k(t)$ in Eq. (26).

$$T_k(t) = \sum_{j \in \mathcal{V}} x_{k,j}(t)\Big(T^{\text{comm}}_{k,\,i(k)\to j}(t) + T^{\text{exec}}_{k,j}(t)\Big) + \sum_{f_j \in \mathcal{F}} \sum_{f_\ell \in \mathcal{F}} m_{k,\,j\to\ell}(t)\, T^{\text{mig}}_{j\to\ell}. \tag{26}$$

Similarly, the energy surrogate $E_k(t)$ in Eq. (27) accounts for both device-side transmission energy and fog-side execution energy (Eq. (20)).

$$E_k(t) = \sum_{j \in \mathcal{V}} x_{k,j}(t)\Big(E^{\text{tx}}_{k,\,i(k)}(t) + E^{\text{exec}}_{k,j}(t)\Big). \tag{27}$$

These surrogates provide a tractable way to compare candidate decisions while remaining aligned with the underlying physical and queueing models already defined earlier, thereby avoiding repeated definitions.

The primary QoS objective in mobile fog settings is task success, i.e., meeting deadlines despite mobility. This is captured by the failure indicator $z_k(t)$, which penalizes deadline misses and/or disconnections. At the same time, among feasible choices that meet deadlines, the LFO should prefer decisions that reduce user-perceived latency and avoid excessive energy expenditure. Therefore, the composite objective is adopted in Eq. (28), which trades off normalized latency $\widetilde{T}_k(t)$ and normalized energy $\widetilde{E}_k(t)$ (Eq. (29)) while heavily penalizing failures via the w_3 term. The normalization uses task-specific budgets ($L_k^{\max}$ and $E_{i(k)}^{\max}$) so that heterogeneous tasks remain comparable without requiring global min–max statistics.

$$\min_{\mathbf{X}(t),\,\mathbf{M}(t),\,\{z_k(t)\}} J(t) = w_1 \sum_{k\in\mathcal{K}(t)} \widetilde{T}_k(t) + w_2 \sum_{k\in\mathcal{K}(t)} \widetilde{E}_k(t) + w_3 \sum_{k\in\mathcal{K}(t)} z_k(t), \tag{28}$$

with normalization

$$\widetilde{T}_k(t) = \frac{T_k(t)}{L_k^{\max}}, \qquad \widetilde{E}_k(t) = \frac{E_k(t)}{E_{i(k)}^{\max}}. \tag{29}$$

5 Implementation in MobFogSim

This section implements the proposed DT-enhanced orchestration and scheduling pipeline in MobFogSim [39], a simulator designed for fog computing with mobile users and service migration. MobFogSim extends iFogSim to support user mobility traces, wireless handoff events, and module migration across fog nodes, making it suitable for evaluating the proposed mobility-aware offloading and proactive migration logic.

5.1 Simulation Settings

A multi-tier fog environment is implemented, inspired by a smart-city layout. The simulated domain contains $|\mathcal{F}| = 5$ fog nodes and $|\mathcal{U}| = 20$ mobile users, along with a cloud data center considered with high compute capacity, and higher WAN latency. Each fog node f_j is configured with a compute capacity specified in MIPS by MobFogSim; this corresponds to the processing capacity C_j in our model up to a constant unit scaling, and task workloads configured in MI correspond to W_k in cycles. Each fog node is also assigned a wireless access radius, which implements the coverage constraint used in the candidate-set definition and in the feasibility indicators, as also indicated in Eqs. (22) and (23). Device-to-fog links are configured with realistic bandwidth ranges for in-range connectivity, while fog-to-cloud links incur higher latency to represent WAN traversal. Details are given in simulation settings in Table 4.

Table 4: MobFogSim simulation settings for the proposed study.

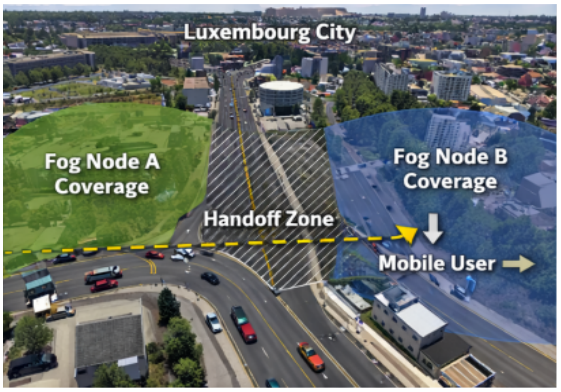
Parameter	Value/Configuration
Simulator	MobFogSim (iFogSim-based; mobility, handoff, and VM/module migration enabled).
Scenario	Smart-city-inspired multi-tier fog with mobile users and cloud.
Fog/Cloud infrastructure	
Number of fog nodes	$\lvert\mathcal{F}\rvert = 5$.
Fog compute capacity	$C_j \in \{2000, 2200, 2500, 2700, 3000\}$ MIPS (per fog).
Fog coverage radius	$R_j = 300$ m (all fog nodes).
Fog–cloud link	Bandwidth = 10 Mbps, one-way latency = 50 ms.
Cloud compute capacity	$C_{\text{cloud}} = 20{,}000$ MIPS (high-capacity data center).

(Continued)

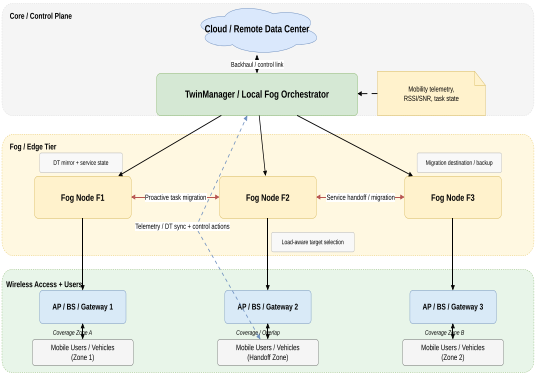
Table 4 (continued)

Parameter	Value/Configuration
Users and mobility	
Number of users	$\lvert\mathcal{U}\rvert = 20$.
Mobility dataset	Luxembourg SUMO Traffic (LuST) trace-driven mobility.
Simulation tick (mobility update)	1 s (user positions updated each tick).
Device–fog wireless link (in coverage)	Uplink bandwidth $B_{i,j} = 15$ Mbps, one-way access latency $\text{prop}_{i,j} = 5$ ms.
Task/Workload model	
Task arrival process	Poisson per user; nominal rate $\lambda_u = 0.2$ tasks/s; sensitivity $\lambda_u \in [0.1, 0.4]$ tasks/s.
Task workload	$W_k \sim \mathcal{U}[500, 2000]$ MI.
Task input size	$D_k \sim \mathcal{U}[0.5, 2]$ MB.
Task deadline	$L_k^{\max} \sim \mathcal{U}[200, 400]$ ms.
Local execution (optional venue)	$C_{\text{local}} = 500$ MIPS.
DT control loop and prediction settings	
Decision epoch /DT synchronization	$\delta = 1$ s.
Lookahead horizon	$\Delta = 30$ steps (≈ 30 s).
GRAIN smoothing (user features)	EWMA factor $\alpha_x = 0.3$ (Eq. (3)).
Fog-load forecasting (EWMA)	EWMA factor $\alpha_z = 0.4$ (Eq. (9)).
GRU history length	$h = 10$ steps (10 s history window).
GRU configuration	1 layer, hidden dimension $d_h = 64$ (offline-trained; online inference).
Confidence threshold	$c_{\min} = 0.6$ (Eq. (8)).
Mobility–load fusion and migration	
Fusion weights	$\beta_\rho = 1.0$, $\beta_Q = 0.5$ (Eq. (10)); $\widehat{Q}$ normalized by $Q_{\max} = 50$.
Migration overhead	$T_{j\to\ell}^{\text{mig}} = 50$ ms (constant per migration).
Capacity safety margin	$\epsilon = 0.1$.
Experimental protocol	
Baselines	TRAP [29], PROMO [17], Proposed DT-enhanced.
Simulation duration	30 min per run.
Number of replications	10 runs; results averaged.

To drive user mobility, the Luxembourg SUMO Traffic (LuST) mobility dataset [40] is utilized. LuST is a representative outdoor smart-city mobility workload that provides realistic user routes and speed variations. In MobFogSim, each user's location is updated at each simulation tick according to the trace, and the simulator generates handoff events when the user crosses fog coverage boundaries. However, evaluating additional mobility models and indoor traces is left for future work. Fig. 4 provides a visual reference for the real-world-derived LuST urban environment and the corresponding fog deployment topology considered in the experiments.



(a) Luxembourg urban mobility environment used in the simulations.



(b) Corresponding fog topology with labeled fog nodes, users, and cloud tier.

Figure 4: Visual representation of the evaluated environment. (**a**) Real-world-derived LuST urban road layout used for user mobility generation, with example trajectories/coverage regions annotated. (**b**) Labelled deployment topology used in MobFogSim, showing mobile users, fog nodes, wireless access points, and the cloud backend.

5.2 The Digital-Twin Control Plane in MobFogSim

MobFogSim does not contain any built-in "digital twin layer" module; therefore, it is implemented logically as a control module that emulates the LFO functionality in our architecture. Specifically, a `TwinManager` module is introduced that runs alongside the MobFogSim broker/controller and maintains the synchronized user-twin and fog-twin state structures as given in Eqs. (1) and (2). At each decision epoch t, `TwinManager` pulls the latest simulation state for all users and fog nodes in the domain, thereby producing the LFO-available mirrored states $\{S_i(t)\}$ and $\{S_j^f(t)\}$. In this experiment, a single LFO domain is focused on; the GFO functionality is not explicitly simulated but is conceptually represented by the optional aggregation of domain summaries.

At each epoch t, `TwinManager` executes the same conceptual pipeline as Algorithm 1. First, it constructs the candidate set $\mathcal{N}_i(t)$ for each user from coverage and identifies feasible fog nodes. Second, it runs the GRAIN mobility predictor that forms the user feature vector as per Eq. (1), applies EWMA smoothing as Eq. (3), and executes a GRU forward pass over the most recent h smoothed states as per Eq. (5) to obtain (i) the association tendency distribution over candidates given in Eq. (6) and (ii) the predicted future user location given in Eq. (7). Third, it computes the confidence score via normalized entropy given in Eq. (8). In parallel, it updates EWMA-based forecasts of fog-side load indicators using fog-twin states given in Eq. (9). Finally, it selects the next fog by mobility-load fusion given in Eqs. (10) and (11) and applies a feasibility check using the predicted location to avoid selecting a fog node that is unlikely to cover the user at the lookahead horizon. Since DT synchronization is emulated as a local state read in MobFogSim, the simulator does not explicitly capture mobile-device battery drain due to DT telemetry. The assignment matrix $\mathbf{X}(t)$ is realized by selecting the execution venue—fog node, cloud, or local device—for each new task upon arrival, i.e., setting the target fog device for the task's module placement in MobFogSim. The migration tensor $\mathbf{M}(t)$ is realised by triggering module migration between fog nodes for ongoing services when the migration condition holds as given also in Eq. (15), approximating state transfer by the configured migration delay $T^{\text{mig}}_{j\to\ell}$. Further, task failures are detected from simulator logs when a task misses its deadline or is dropped due to loss of connectivity before completion. The GRU component is pre-trained offline using mobility traces and then used for inference during simulation runs. To avoid optimistic bias, the training data are separated from evaluation runs by using a time set in the simulation; the LFO performs inference only on the segment. During the simulation, the LFO performs inference only at each epoch.

MobFogSim provides a best-case DT deployment model in which the TwinManager reads the state locally from the simulator. As a result, synchronization delay is negligible in our reported experiments, and the DT update period equals the decision epoch, i.e., $\delta = 1$ s. In real deployments, non-zero telemetry and twin-update latency can reduce the effective lookahead of proactive migration. Proactive decisions are most beneficial when the DT update interval and communication delay are small relative to the user's residence time in a fog coverage region.

5.3 Evaluation Metrics

The evaluation metrics are chosen to align directly with the objective components in Eq. (28) and to characterize the operational behavior of mobility handling. First, the average end-to-end delay is computed using the per-task latency surrogate in Eq. (26) and averaged across all tasks. This delay aggregates communication delay, load-dependent execution delay, and migration overhead. In addition to the mean, we report tail latency, e.g., 95th percentile, (e.g., the 95th percentile) to reflect reliability under congestion and mobility. Second, average energy is computed using the per-task energy surrogate in Eq. (27), aggregated over tasks and reported as energy per task. Third, the task success rate measures the fraction of tasks completed within the deadline without being dropped. This aligns with the failure indicator $z_k(t)$ used in Eq. (28) and is computed as $\text{SR} = 100\% \cdot \left(1 - \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} z_k\right)$.

Fourth, DT overhead quantifies the additional computation and communication costs introduced by the DT-enabled control loop. Therefore, (i) CPU utilization time attributable to EWMA updates, GRU inference, confidence computation, fog EWMA forecasting, and fusion-based decision making, and (ii) additional bytes transmitted for twin synchronization messages, are measured.

Finally, fifth, migration count is the total number of migrations triggered, and service migration rate is the fraction of tasks that undergo at least one migration. These metrics help identify whether a scheme is overly reactive, i.e., excessive migrations or insufficiently adaptive, i.e., too few migrations, and they contextualize delay and energy trade-offs.

6 Results and Discussion

The metrics in Section 5.3 are evaluated across scenarios with varying user speeds and task arrival rates to stress the system under different mobility and workload intensities. We compare three configurations: (i) *TRAP* [29], which performs multi-objective scheduling without explicit mobility handling; (ii) *PROMO* [17], which supports mobility-aware migration based on reactive/threshold triggers without DT-driven prediction and load-aware fusion; and (iii) The *proposed DT-enhanced* scheduler, which uses DT-synchronised states, GRAIN-based mobility prediction, confidence estimation, fog-load EWMA forecasting, and mobility-load fusion for next-fog selection. The reported values are averaged over 10 independent runs of a 30-minute simulation scenario with different random seeds.

Table 5 summarizes the objective-aligned metrics used in Eq. (28) that are mean delay through $T_k(t)$ in Eq. (26), energy per task through $E_k(t)$ in Eq. (27), and service reliability captured by the failure indicator $z_k(t)$ reported here as success rate = 100% − failure rate. DT overhead is reported alongside these objective terms to quantify the additional compute cost of the proposed DT control loop. Further, we also include statistical analysis to quantify run-to-run variability. Therefore, aggregate metrics are reported as mean ± standard deviation over 10 independent runs, and the corresponding 95% confidence interval for $n = 10$ is computed as $\bar{x} \pm t_{0.975,9}\, s/\sqrt{10}$, where $t_{0.975,9} = 2.262$.

Table 6 summarizes the variability of the observed improvements across low-, medium-, and high-load regimes. The proposed method shows consistently positive gains across all tested loads, with especially stable

improvements for prediction-quality metrics such as MAE, RMSE, and MAPE, while metrics such as DMR exhibit larger variability due to their stronger sensitivity to congestion severity.

Table 5: Performance comparison (TRAP vs. PROMO vs. proposed DT-enhanced).

Approach	Avg. Task Delay (ms)	Energy/Task (J)	Success Rate (%)	DT Overhead (% of CPU)
TRAP	250.3	1.50	80.4	0
PROMO	158.7	1.42	95.0	0
Proposed	139.6	1.32	99.1	5.1

Table 6: Improvement robustness across load levels (Low, Medium, High).

Metric	Low (%)	Medium (%)	High (%)	Mean ± SD (%)	95% CI (%)
Latency mean reduction	15.44	14.08	23.97	17.83 ± 5.37	17.83 ± 13.33
Latency p95 reduction	14.36	16.62	26.56	19.18 ± 6.49	19.18 ± 16.12
DMR reduction	23.45	57.08	79.56	53.36 ± 28.24	53.36 ± 70.15
Throughput gain	4.42	5.10	6.21	5.24 ± 0.90	5.24 ± 2.24
Total energy reduction	2.92	13.05	9.30	8.42 ± 5.12	8.42 ± 12.72
Jain fairness gain	4.99	3.98	10.48	6.48 ± 3.50	6.48 ± 8.69
Normalized cost reduction	17.00	21.16	19.39	19.18 ± 2.09	19.18 ± 5.19
MAE reduction	47.91	48.24	48.76	48.30 ± 0.43	48.30 ± 1.07
RMSE reduction	54.67	50.54	51.07	52.10 ± 2.25	52.10 ± 5.58
MAPE reduction	45.23	48.95	43.08	45.76 ± 2.97	45.76 ± 7.38

Note: Values are computed from the uploaded improvement data across the three load regimes (Low, Medium, High). The 95% confidence interval is computed as $\bar{x} \pm t_{0.975,2}\, s/\sqrt{3}$, where $t_{0.975,2} = 4.303$.

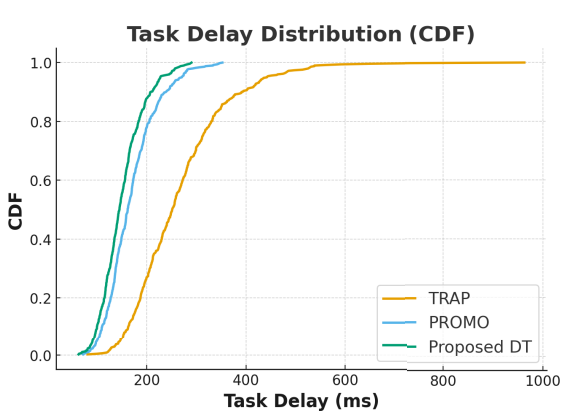
Relative to the static baseline TRAP, PROMO reduces the average task delay from 250.3 to 158.7 ms, corresponding to a 36.6% delay reduction. PROMO also improves reliability from 80.4% to 95.0% success rate; equivalently, the failure rate drops from 19.6% to 5.0%, i.e., a 74.5% reduction in failures, which directly reduces the $\sum_k z_k(t)$ penalty in Eq. (28). Energy per task decreases modestly from 1.50 to 1.42 J, means 5.3% reduction, consistent with fewer mobility-induced disruptions and reduced time spent under poor connectivity.

The proposed DT-enhanced scheduler yields further gains over PROMO by coupling GRAIN-based mobility prediction with fog-load forecasting and mobility-load fusion, explained in Eqs. (10) and (11). Specifically, the mean delay decreases from 158.7 to 139.6 ms, i.e., an additional 12.0% reduction vs. PROMO and a 44.2% reduction vs. TRAP. Energy per task decreases from 1.42 to 1.32 J, i.e., a 7.0% reduction vs. PROMO and a 12.0% reduction vs. TRAP. Reliability improves from 95.0% to 99.1% success rafavourablete, which is a +4.1 percentage-point increase. In terms of failures, the failure rate drops from 5.0% to 0.9%, i.e., an 82.0%, reduction in failures vs. PROMO and a 95.4% reduction vs. TRAP, from 19.6% to 0.9%. These improvements indicate that the DT-enhanced method not only migrates proactively but also avoids migrating users toward nodes predicted to become congested, thereby reducing queueing delay and deadline misses. These QoS improvements incur a modest DT compute overhead of 5.1% CPU. Importantly, this overhead reflects the added control-plane intelligence central to the proposed theme of twin synchronization, EWMA updates, GRU inference, confidence computation, fog EWMA forecasting, and fusion scoring. Given

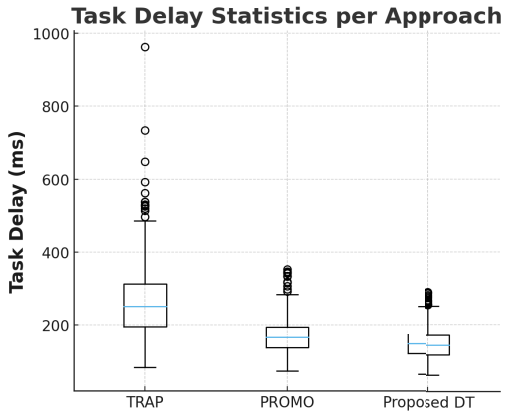
the observed 12.0% mean-delay reduction and 82.0% failure reduction relative to PROMO, the overhead represents a favorable trade-off in the tested setting.

The gains in Table 5 are not due to a single factor, but to the joint effect of *prediction*, *load awareness*, and *uncertainty gating*. TRAP is primarily reactive and does not explicitly anticipate user movement, so tasks may remain attached to a fog node even as the user moves away, increasing communication delays and the risk of deadline misses. PROMO improves on TRAP by anticipating mobility, but its decisions are driven primarily by movement tendencies and do not explicitly account for whether the target fog will remain lightly loaded over the decision horizon. In contrast, the proposed DT-enhanced method combines GRAIN-based association prediction with EWMA-based fog-load forecasting and the fusion score in Eqs. (10) and (11). This allows the controller to migrate not simply to the *next likely* fog, but to the *next reliable and less congested* fog.

The plots in Figs. 5–10 provide evidence beyond the mean values in Table 5. Collectively, they show that the proposed DT-enhanced method is *productive* in the sense that it (i) reduces not only *average* delay but also *tThe ail* delay, (ii) improves reliability under increasing load, (iii) improves the energy–delay operating point, and (iv) does so with bounded DT overhead and *fewer* migrations than a reactive migration baseline. In other words, the gains are not obtained by "over-migrating" or shifting cost from one metric to another; they come from earlier, more selective, and load-aware decisions.



(a) CDF of task delay for TRAP, PROMO, and the proposed DT-enhanced scheduler.



(b) Boxplot of task delay statistics for the three approaches.

Figure 5: Task delay behavior under TRAP, PROMO, and the proposed DT-enhanced model.

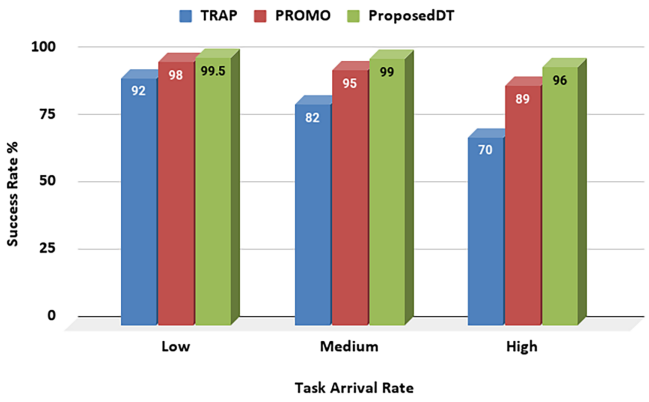


Figure 6: Success rate vs. task arrival rate.

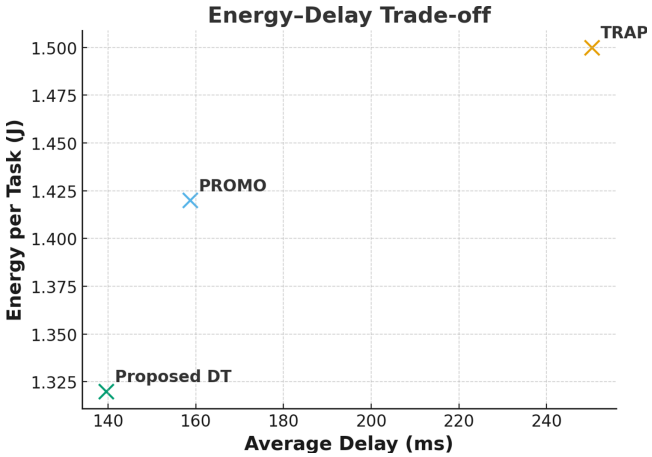
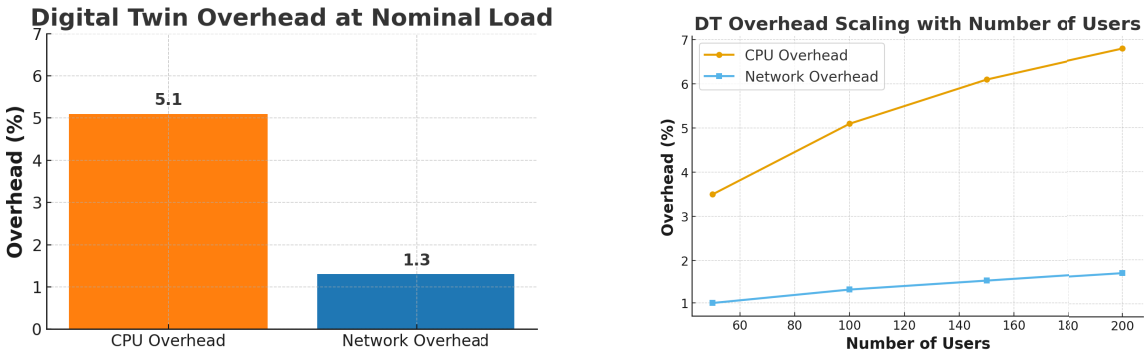


Figure 7: Joint behavior of average task delay and energy per task.



(a) Digital twin overhead at nominal load.

(b) CPU and network overhead vs. number of users.

Figure 8: Digital-twin overhead characteristics.

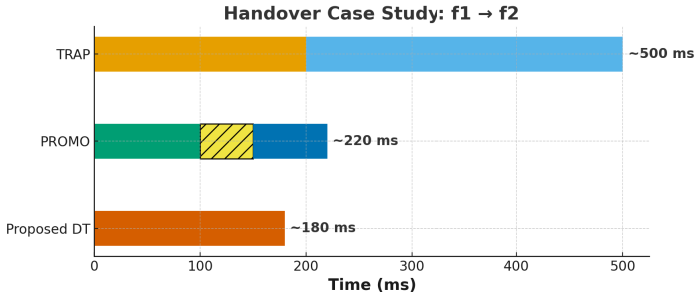
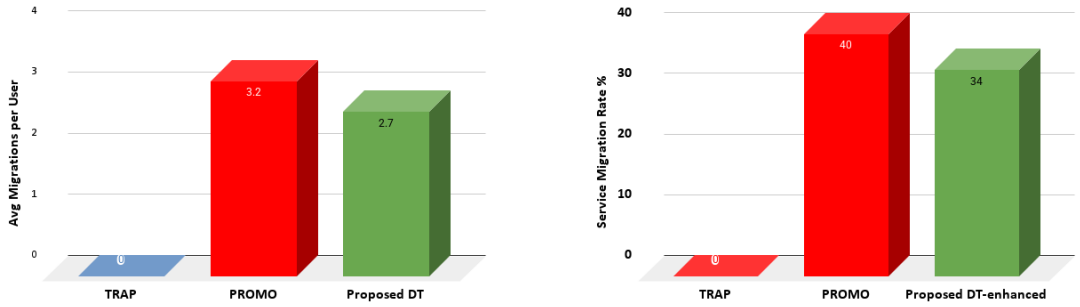


Figure 9: Illustrative handover case where a user moves from fog node f_1 towards f_2.



(a) Average migrations per user (30 min). (b) Service migration rate per approach.

Figure 10: Migration behavior under TRAP, PROMO, and the proposed DT-enhanced scheduler.

Fig. 5a shows the empirical CDF of task delay. A curve that is further left indicates that a larger fraction of tasks finish within a smaller delay budget. The proposed DT-enhanced curve is consistently left-shifted relative to PROMO, and both are substantially left of TRAP, implying that DT-enhanced scheduling improves delay for *most* tasks, not only on average. Importantly, the right tail under TRAP is much heavier: the CDF approaches 1 onlynear ~900–1000 ms, indicating rare but severe latency spikes. In contrast, PROMO saturates near ~350 ms, and the proposed DT-enhanced method saturates near ~280–300 ms, showing that the DT-enhanced policy significantly suppresses worst-case delays. Fig. 5b reinforces this interpretation: TRAP exhibits a higher median and a much wider spread with many high outliers, whereas PROMO reduces both the median and the interquartile range (IQR). The proposed DT-enhanced method has the lowest median and a tighter IQR, indicating more stable QoS in addition to lower mean delay. This is aligned with the objective term based on $T_k(t)$ in Eq. (26), because reducing tail latency also reduces the probability of deadline violations.

Fig. 6 evaluates the success rate as the task arrival rate increases. The proposed DT-e,nhanced method consistently maintains the highest success rate, and the gap widens at higher load, which is where Congestion and mobility jointly increase deadline misses. Concretely, at *High* arrival rate, success improves from 70% in TRAP and 89% in PROMO to 96% in DT-enhanced, i.e., a +26 percentage-point gain over TRAP and a +7 percentage-point gain over PROMO. At *Medium* arrival rate, the proposed method achieves 99% vs. 95% for PROMO and 82% for TRAP. These results indicate that the DT-enhanced strategy improves the failure indicator term $\sum_k z_k(t)$ in Eq. (28), and that the improvement is robust to increased load rather than limited to a single operating point.

Fig. 7 summarizes whether delay improvements are achieved at the cost of higher energy. The proposed DT-enhanced method moves toward the bottom-left region, indicating a genuine improvement in the multi-objective sense, as in Eq. (28). Compared to PROMO, the proposed method reduces delay from ~159 to ~140 ms while also reducing energy per task from ~1.42 to ~1.32 J. Compared to TRAP, it reduces both delay and energy substantially. This confirms that the DT-enhanced approach does not merely trade energy for latency; it improves both, consistent with the intended mobility–load fusion behavior.

Fig. 8a quantifies the additional cost of the DT control loop at nominal load: ~5.1% CPU overhead and ~1.3% network overhead. Fig. 8b shows how this overhead scales with the number of users, where CPU overhead increases from ~3.5% at 50 users to ~6.8% at 200 users, while network overhead increases from ~1.0% to ~1.7%. Two points are important here. First, the absolute overhead remains modest, below 7% CPU and below 2% network in the tested range. Second, the growth is gradual, indicating that the control-plane

cost scales reasonably with user population. Therefore, the DT layer appears *practically viable* because it introduces bounded overhead while enabling measurable gains in QoS and reliability.

The handover case study in Fig. 9 illustrates the causal mechanism behind the aggregate improvements. Under TRAP, the task remains anchored to f_1 even as the user moves toward f_2, resulting in a long completion time of ~500 ms. PROMO mitigates this by migrating near the handoff boundary, reducing completion time to ~220 ms, but the migration incurs a visible overhead component (the hatched segment). The proposed DT-enhanced method completes in ~180 ms by anticipating the short dwell time near f_1 and proactively transitioning service earlier, thereby avoiding or minimizing live migration overhead. Relative to PROMO, this is an approximate 18% reduction in the case-study delay, i.e., 220 → 180 ms, and relative to TRAP, it is an approximate 64% reduction from 500 → 180 ms.

Fig. 10a shows average migrations per user where PROMO performs ~3.2 migrations/user, while the proposed DT-enhanced method performs ~2.7 migrations/user, which is ~15.6% reduction, and TRAP performs none because it does not migrate. Fig. 10b shows the service migration rate where PROMO migrates ~40% of services/tasks, while the proposed DT-enhanced method migrates ~34%, which is a ~15% reduction. These results are significant because they rule out a common "false win" in mobility studies: achieving lower delay simply by migrating more aggressively. The proposed method achieves *lower delay and higher success rate with fewer migrations than PROMO*, indicating more selective and better-timed mobility handling.

6.1 Ablation Study and Component Contribution Analysis

To quantify the contribution of individual components in the proposed framework, we conducted an ablation study by selectively removing key modules. Unlike TRAP and PROMO, which are full external baseline schedulers, the rows in Table 7 are internal ablation variants of the proposed method obtained by disabling one module at a time.

Table 7: Ablation study of the proposed scheduling framework.

Model Variant	Avg. Delay (ms)	Energy (J)	Success Rate (%)
Full model (DT + GRU + EWMA + fusion)	139.6	1.32	99.1
Without Digital Twin	164.8	1.40	94.5
Without GRU (EWMA only)	171.2	1.44	93.1
Without confidence gating	153.6	1.37	96.2
Without load prediction	159.9	1.39	95.4

The results indicate that the digital twin and confidence-gated decision mechanism contribute most significantly to reliability improvements, while GRU-based mobility prediction primarily reduces latency. This confirms that the proposed framework's performance gains arise from the synergistic interaction of multiple components rather than a single algorithmic improvement.

6.2 Sensitivity of EWMA Hyperparameters

The EWMA factors α_x which is a user-feature smoothing as given in Eq. (3) and α_z which is a fog-metric forecasting in Eq. (9) control the robustness trade-off in the GRAIN loop. Since, proactive handoff or migration depends on both the association tendency $\mathbf{p}_i(t+\Delta)$ and the predicted fog metrics $(\widehat{\rho}_j, \widehat{Q}_j)$, α directly impacts (i) the trigger time of proactive actions, (ii) stability of the selected next fog, and (iii) migration churn.

With decision epoch δ, EWMA weights decay as $(1-\alpha)^k$, giving an effective memory $S_{\text{eff}} \approx \frac{2}{\alpha} - 1$ samples and a time constant $\tau = -\delta/\ln(1-\alpha)$. The consideration of small value for α suppresses noise but increases lag in decisions, whereas large α value reacts quickly but can propagate signal noise ratio and queue jitter. Fig. 11 illustrates this effect via the EWMA step response.

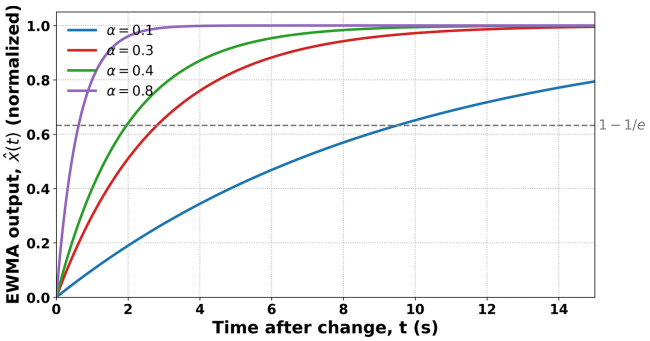


Figure 11: EWMA step response for different smoothing factors α. The dashed line marks the $1 - 1/e \approx 0.632$ level used to visualize the time constant.

6.3 Complexity Analysis

Proactive mobility-aware schemes such as TRAP and PROMO achieve gains only when near-future mobility predictions are reliable. In a worst-case setting where the prediction input becomes intentionally incorrect or highly volatile, proactive control can trigger unnecessary migrations, increasing churn and overhead.

In our proposed framework, unreliable predictions typically produce a more spread-out next-fog probability distribution, which increases the normalized entropy and lowers the confidence score. Therefore, the entropy-based confidence gate suppresses proactive handoff preparation and proactive migrations when predictions are unreliable. Hence, in the worst case, the system degrades conservatively toward baseline reactive behaviour rather than amplifying oscillations. A full adversarial evaluation is beyond the scope of the present simulation and is left for future work.

Further, computational and communication complexity is analyzed. Let $|\mathcal{N}_i(t)|$ be the number of nearby candidate fog nodes for user i at epoch t, and let K be the number of tasks considered for migration. At each epoch, the controller performs: (i) one forward pass of the GRAIN predictor, (ii) entropy computation over $|\mathcal{N}_i(t)|$ candidates, and (iii) migration-rule evaluation for each task across these candidates. Thus, the online time complexity per epoch is

$$\mathcal{O}\big(C_{\text{GRU}} + |\mathcal{N}_i(t)| + K \cdot |\mathcal{N}_i(t)|\big) = \mathcal{O}\big(C_{\text{GRU}} + (K+1)|\mathcal{N}_i(t)|\big), \tag{30}$$

where C_{GRU} is the fixed cost of a GRU forward pass for a configured hidden size and sequence length. In a real-world scenario, $|\mathcal{N}_i(t)|$ is small because it is limited to coverage neighbours, keeping the decision overhead lightweight. Furthermore, the communication overhead is dominated by periodic telemetry updates. With per-user update interval δ, the uplink reporting cost scales as $\mathcal{O}(U/\delta)$ for U active users.

6.4 Scalability Analysis

To evaluate the scalability of the proposed framework, the simulation was extended to include large-scale scenarios with varying numbers of mobile users and fog nodes. Scalability is a basic requirement of fog and edge computing systems, as it is anticipated that a considerable number of heterogeneous devices will be

involved in a fog computing system, where a wide variety of mobility patterns will be encountered [10,11]. As expected, with increased system scale, all suggested approaches will exhibit higher latency due to increased competition for fog resources and communication overhead, as reported in previous studies on large-scale edge computing systems [41]. At the same time, it was observed that the suggested framework continues to achieve a lower average task delay than the TRAP and PROMO approaches across all simulation scenarios. This shows that the suggested framework, integrating mobility prediction and load-aware scheduling, can achieve efficient resource allocation as workload and mobility levels increase. In addition, the proposed framework's task success rate remains higher than that of other approaches, particularly in large-scale scenarios. This observation is consistent with recent studies reporting that predictive and mobility-aware orchestration can significantly improve task reliability in edge and fog computing systems [41,42]. In terms of overhead, the digital twin control loop incurs additional computational overhead that scales with system size. However, it was observed that the increase in overhead with system scale is gradual, indicating that digital twin-based orchestration can be implemented with reasonable control plane overhead, as reported in recent studies on digital twin-based fog computing systems [12,13]. It was observed that the proposed framework maintains superior performance in terms of quality of service and feasibility as the system scale increases, indicating its applicability to large-scale fog computing systems, as shown in Table 8.

Table 8: Impact of user and fog node scaling on delay and success rate.

Users	Fog Nodes	Delay (ms) Proposed	Delay PROMO	Delay TRAP	Success Rate (%)
20	5	139.6	158.7	250.3	99.1
50	5	152.4	180.6	290.2	97.9
100	10	168.9	210.4	330.5	96.6
200	20	192.7	255.8	390.9	94.3

The reported results should be interpreted in light of several experimental assumptions. First, the study is simulation-based in MobFogSim and therefore inherits the simulator's abstraction level for wireless links, processing delays, and migration costs, rather than a real hardware deployment. Second, user mobility is driven only by the Luxembourg SUMO Traffic (LuST) dataset, which is representative of outdoor urban vehicular movement but does not cover indoor, pedestrian, or dense industrial IoT mobility regimes. Third, the DT control plane is emulated as a co-located `TwinManager` that reads simulator state locally, corresponding to a best-case edge deployment; thus, mobile-device battery drain and large non-zero synchronization delays are not explicitly measured in the reported runs. Finally, the experiments focus on a single LFO domain and do not evaluate inter-domain coordination. These limitations may affect the generalizability of the absolute performance values, but the relative comparison among TRAP, PROMO, and the proposed method remains meaningful because all schemes are evaluated under the same controlled conditions.

7 Conclusion

The present paper proposes a digital twin–based, uncertainty-aware framework for proactive fog scheduling under dynamic user mobility and fog congestion. By integrating confidence-gated mobility prediction with load-aware decision-making, the framework enables anticipatory handoff and task migration while avoiding unnecessary control actions. Empirical results demonstrate that the proposed approach consistently outperforms baseline methods, achieving lower latency and energy consumption, higher service reliability, and only modest digital-twin overhead. These findings confirm that uncertainty-aware digital

twin orchestration is not merely an incremental enhancement but a critical mechanism for ensuring robust service continuity in mobile fog environments. Overall, this work establishes a principled foundation for next-generation proactive edge and fog control, paving the way for scalable, intelligent, and reliability-aware orchestration in highly dynamic edge systems.

Acknowledgement: The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, Saudi Arabia, for funding this research work.

Funding Statement: This research was funded by the Deanship of Scientific Research at Northern Border University. Arar, Saudi Arabia, under grant number NBU-FFR-2026-451-5.

Author Contributions: Navjeet Kaur conceptualizes, investigates, and writes the original draft. Ayush Mittal validates and implements the proposed work. Saad Alahmari provided project supervision, methodological guidance, and critical manuscript revision. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data and materials used in this study are available from the corresponding author upon request.

Ethics Approval: This study did not involve human participants, human data, or animals. Therefore, ethics approval was not required.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A Notation Table

Table A1: Notation used in the DT synchronization, mobility prediction, association, load forecasting, offloading, handoff/migration, objective, and queueing/energy models.

Symbol	Meaning
i	User (mobile device) index; device u_i.
j, ℓ	Fog node indices (current serving fog f_j, predicted/target next fog f_ℓ).
k	Task index; also used as candidate-index in $\mathcal{N}_i(t)$ where stated (context-dependent).
v	Generic task index used in queueing/energy derivations (to distinguish from scheduling task index k).
t	Discrete time (control epoch) index.
δ	DT synchronization period / decision epoch duration.
Δ	Lookahead / prediction horizon (steps ahead).
$\mathcal{U}$	Set of users (devices).
$\mathcal{F}$	Set of fog nodes.
$\mathcal{V}$	Set of execution venues (typically $\mathcal{F} \cup \{\text{cloud}\}$; can be extended if local execution is modeled).
$\mathcal{K}(t)$	Set of tasks considered/arriving at epoch t for scheduling.
$\mathcal{K}$	Set of tasks in an evaluation window (used for reporting mean/tail metrics).
$\mathcal{N}_i(t)$	Candidate fog-node set feasible for user i at time t (in-range/eligible).
$i(k)$	Mapping from task k to its generating user/device index.
$\tilde{u}_i, \tilde{f}_j$	Digital twins of user u_i and fog node f_j.
$S_i(t)$	User-twin state vector (Eq. (1)).
$S_j^f(t)$	Fog-twin state vector (Eq. (2)).

(Continued)

Table A1 (continued)

Symbol	Meaning
$a_i(t)$	Serving association of user i at time t; $a_i(t) \in \mathcal{F} \cup \{\text{cloud}\}$.
$L_i(t)$	Location of user i at time t.
$v_i(t)$	Velocity (or speed) of user i at time t.
$tl_i(t)$	Current task/traffic load at user i (pending computation/offload demand).
$\text{RSSI}_i(t)$, $\text{SNR}_i(t)$	Link indicators for user i on its current/serving link (as used in $S_i(t)$).
$\text{RSSI}_{i,j}(t)$, $\text{SNR}_{i,j}(t)$	Link indicators between user i and fog node f_j (as used in $\mathbf{o}_i^u(t)$).
$\mathbf{x}_i(t)$	Instantaneous GRAIN input feature vector derived from DT states (Eq. (1)).
$\tilde{\mathbf{x}}_i(t)$	EWMA-smoothed user feature vector (Eq. (3)).
α_x	EWMA smoothing factor for user-feature smoothing (Eq. (3)).
h	History length (number of past steps used in GRU).
$\phi_\theta(\cdot)$	GRU-based sequence encoder with parameters θ (Eq. (5)).
$\mathbf{h}_i(t) \in \mathbb{R}^{d_h}$	GRU hidden state/latent embedding at time t; d_h is embedding dimension.
f_k	k-th candidate fog node in $\mathcal{N}_i(t)$.
$p_i^{(k)}(t+\Delta)$	Probability that user i associates with f_k at time $t+\Delta$ (Eq. (6)).
W_o, $\mathbf{b}_o$	Output-layer parameters for association prediction (Eq. (6)).
$\psi_\eta(\cdot)$	Regression head with parameters η (Eq. (7)).
$\widehat{L}_i(t+\Delta)$	Predicted future location of user i at horizon Δ (Eq. (7)).
$\mathbf{p}_i(t+\Delta)$	Probability vector over candidates at horizon Δ; $\{p_i^{(k)}(t+\Delta)\}_{f_k \in \mathcal{N}_i(t)}$.
$H(\mathbf{p})$	Shannon entropy of distribution $\mathbf{p}$ (Eq. (8)).
$c_i(t+\Delta)$	Normalized-entropy confidence score in $[0,1]$ (Eq. (8)).
$c_{\min}$	Confidence threshold used for gating proactive actions (Eq. (24)).
$Q_j(t)$	Queue length at fog node f_j at time t.
$\rho_j(t)$	Utilization of fog node f_j at time t.
$\lambda_j(t)$	Aggregate task arrival rate to f_j at time t (tasks/s).
$N_j(t)$	Number of active sessions/users served by f_j at time t (recommended to avoid conflict with memory).
$M_j(t)$	Available memory at fog node f_j at time t (as in $S_j^f(t)$).
$B_j(t)$	Available network bandwidth at fog node f_j at time t (as in $S_j^f(t)$).
$P_j(t)$	Instantaneous/measured power of fog node f_j at time t as reported by DT (if used).
$z_j(t)$	Generic scalar fog metric used for EWMA forecasting (e.g., $\rho_j(t)$ or $Q_j(t)$).
$\widehat{z}_j(t)$	EWMA forecast/state of $z_j(\cdot)$ at time t (Eq. (9)).
α_z	EWMA smoothing factor for fog-metric forecasting (Eq. (9)).
β_ρ, β_Q	Fusion weights for utilization and queue forecasts (Eq. (10)).
$\Gamma_{i,k}(t+\Delta)$	Composite mobility–load score for selecting candidate f_k (Eq. (10)).
$\widehat{\text{nextFog}}(i, t+\Delta)$	Predicted next serving fog node for user i (Eq. (11)).
L_j	Fixed location of fog node f_j.
R_j	Coverage radius of fog node f_j.
$d(i,j,t)$	Euclidean distance between user i and fog node f_j at time t (Eq. (12)).
$\ell_{i,j}(t)$	Link-latency feasibility surrogate (piecewise finite/∞; Eq. (13)).
$\chi_{i,j}(t)$	Current coverage feasibility indicator $\mathbb{1}[d(i,j,t) \le R_j]$ (Eq. (22)).

(Continued)

Table A1 (continued)

Symbol	Meaning
$\chi_{i,j}(t+\Delta)$	Predicted future feasibility indicator using $\widehat{L}_i(t+\Delta)$ (Eq. (23)).
$g_i(t+\Delta)$	Confidence gate indicator $\mathbb{1}[c_i(t+\Delta) \geq c_{\min}]$ (Eq. (24)).
$\pi_{i,\ell}(t+\Delta)$	Destination selector consistent with $\widehat{\text{nextFog}}(i, t+\Delta)$ (Eq. (25)).
t_k	Task indexed by k.
W_k	Task workload (CPU cycles).
D_k	Task input data size (bytes).
$L_k^{\max}$	Task deadline / maximum latency requirement.
$R_{i,j}(t)$	Uplink transmission rate (bits/s) from user i to fog f_j at epoch t (if modeled explicitly).
$B_{i,j}$	Uplink bandwidth/throughput parameter between user i and fog f_j.
$\text{prop}_{i,j}$	Propagation/access delay between user i and fog f_j.
$T_{k,i\to j}^{\text{comm}}(t)$	Communication time to offload task k from user $i(k)$ to fog f_j at epoch t.
$T_{k,j}^{\text{exec}}(t)$	Execution/response-time proxy for task k at fog node f_j at epoch t (queueing-based, e.g., Eq. (19)).
$T_{j\to\ell}^{\text{mig}}$	Migration overhead from f_j to f_ℓ (checkpoint, transfer, resume).
$T_k(t)$	End-to-end latency surrogate (communication + execution + migration overhead; Eq. (26)).
P_i^{tx}	Transmit power of user device i.
$E_{k,j}^{\text{exec}}(t)$	Fog-side execution energy for task k at node f_j (Eq. (20)).
$E_k(t)$	Per-task energy surrogate (device transmission + fog execution; Eq. (27)).
$E_i^{\max}$	Energy budget (constraint) for user device i (used as $E_{i(k)}^{\max}$ in normalization).
$x_{k,j}(t)$	Binary assignment decision; 1 if task k is assigned to venue $j \in \mathcal{V}$ at epoch t, else 0.
$\mathbf{X}(t)$	Assignment matrix collecting $\{x_{k,j}(t)\}$.
$m_{k,\,j\to\ell}(t)$	Migration indicator; 1 if task k migrates from f_j to f_ℓ at epoch t, else 0 (Eq. (15)).
$\mathbf{M}(t)$	Migration decision tensor collecting $\{m_{k,\,j\to\ell}(t)\}$.
$\widehat{T}_{k,j}^{\text{rem}}(t)$	Predicted remaining execution time of task k on current fog f_j at epoch t.
$\widehat{\tau}_i^{(j)}(t)$	Predicted remaining dwell time of user i within coverage of current fog f_j at epoch t.
$z_k(t)$	Failure indicator (e.g., deadline miss/drop) for task k at epoch t.
w_1, w_2, w_3	Objective weights for normalized latency, normalized energy, and failures (Eq. (28)).
$\widetilde{T}_k(t)$	Normalized latency: $\widetilde{T}_k(t) = T_k(t)/L_k^{\max}$ (Eq. (29)).
$\widetilde{E}_k(t)$	Normalized energy: $\widetilde{E}_k(t) = E_k(t)/E_{i(k)}^{\max}$ (Eq. (29)).
$J(t)$	Objective value at epoch t (Eq. (28)).
C_j	Processing capacity of fog node f_j (cycles/s).
W_v	Workload (CPU cycles) of a generic task v (used in queueing derivations).
$T_{v,j}^{\text{svc}}$	Service/processing time of task v on node f_j: $T_{v,j}^{\text{svc}} = W_v/C_j$.
T_j^{svc}	Generic service-time random variable at node f_j.
ρ_j	Utilization of node f_j: $\rho_j = \lambda_j \mathbb{E}[T_j^{\text{svc}}]$ (Eq. (17)).
$\mathbb{E}[T_j^{\text{resp}}]$	Mean response time at node f_j (Pollaczek–Khinchine; Eq. (18)).
$T_{v,j}^{\text{exec}}$	Task-level execution-time approximation on f_j (Eq. (19)).
$P_j(\rho_j)$	Power consumption model at f_j as a function of utilization (Eq. (21)).

(Continued)

Table A1 (continued)

Symbol	Meaning
P_j^{idle}, P_j^{peak}	Idle and peak power of fog node f_j.
$E_{v,j}^{\text{exec}}$	Compute-side energy to execute task v at node f_j (Eq. (20)).

References

1. Joseph AG, Gokhale MM, Mani JMJ, Veni T. Mobility aware computation offloading in fog devices using virtual machine migration. In: 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT). Piscataway, NJ, USA: IEEE; 2023. p. 1–6.
2. Araújo MC, Bittencourt LF. CMFogV: proactive content migration for multi-level fog computing. Pervasive Mob Comput. 2024;102:101933.
3. Li Z, Mei X, Sun Z, Xu J, Zhang J, Zhang D, et al. A reference framework for the digital twin smart factory based on cloud-fog-edge computing collaboration. J Intell Manuf. 2025;36(5):3625–45. doi:10.1007/s10845-024-02424-0.
4. Hayawi K, Sajid J, Malik AW, Mathew SS. Digital twin assisted task offloading for workload management at fog nodes. IEEE Internet Things J. 2025;12(13):23061–72. doi:10.1109/jiot.2025.3550832.
5. Liang Y, Li G, Zhang G, Guo J, Liu Q, Zheng J, et al. Latency reduction in immersive systems through request scheduling with digital twin networks in collaborative edge computing. ACM Trans Sens Netw. 2024;8:12679. doi:10.1145/3701562.
6. El-Khatib RF, Elsayed SA, Zorba N, Hassanein HS. Proactive task allocation in extreme edge computing for digital twin services. IEEE Internet Things J. 2025;12(10):14051–66. doi:10.1109/jiot.2024.3524868.
7. Zhao Z, Wang Y, Xie X. Advancing traffic resource scheduling with cloud-edge collaboration: a virtualized digital twin perspective. IEEE Internet Things J. 2025;12(19):39625–39. doi:10.1109/jiot.2025.3588153.
8. Jeremiah SR, Yang LT, Park JH. Digital twin-assisted resource allocation framework based on edge collaboration for vehicular edge computing. Future Gener Comput Syst. 2024;150(3):243–54. doi:10.1016/j.future.2023.09.001.
9. Wang Y, Su Z, Guo S, Dai M, Luan TH, Liu Y. A survey on digital twins: architecture, enabling technologies, security and privacy, and future prospects. IEEE Internet Things J. 2023;10(17):14965–87.
10. Xie B, Cui H. Deep reinforcement learning-based dynamical task offloading for mobile edge computing. J Supercomput. 2025;81(1):35. doi:10.1007/s11227-024-06603-x.
11. Sheng S, Chen P, Chen Z, Wu L, Yao Y. Deep reinforcement learning-based task scheduling in IoT edge computing. Sensors. 2021;21(5):1666. doi:10.3390/s21051666.
12. Chen X, Cao J, Liang Z, Sahni Y, Zhang M. Digital twin-assisted reinforcement learning for resource-aware microservice offloading in edge computing. In: 2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS). Piscataway, NJ, USA: IEEE; 2023. p. 28–36.
13. Zhou X, Peng Y, Lan T, Zhang Z, Tang B, Guan X. Digital twin empowered task offloading for mobile edge computing in 6G internet of vehicles. IEEE Internet Things J. 2025;12(15):29189–202. doi:10.1109/jiot.2025.3575466.
14. Hu Z, Tu J, Li B. SPEAR: optimized dependency-aware task scheduling with deep reinforcement learning. In: Proceedings of the 2019 39th IEEE International Conference on Distributed Computing Systems. Piscataway, NJ, USA: IEEE; 2019. p. 2037–46.
15. Zhou C, Wu W, He H, Yang P, Lyu F, Zhang N, et al. Deep reinforcement learning for delay-oriented IoT task scheduling in SAGIN. IEEE Trans Wireless Commun. 2021;20(2):911–25. doi:10.1109/twc.2020.3029143.
16. Taleb T, Ksentini A, Frangoudis PA. Follow-Me cloud: when cloud services follow mobile users. EEE Trans Cloud Comput. 2019;7(2):369–82.
17. Kaur N, Kumar A, Kumar R. PROMO: PROactive mObility-support model for task scheduling in fog computing. Int J Comput Appl. 2022;44(11):1092–101.
18. Bozkaya E. Digital twin-assisted and mobility-aware service migration in mobile edge computing. Comput Netw. 2023;231(7):109798. doi:10.1016/j.comnet.2023.109798.

19. Sun W, Zhang H, Wang R, Zhang Y, Shen X, Dai H. Reducing offloading latency for digital twin edge networks in 6G. IEEE Trans Veh Technol. 2020;69(10):12240–51. doi:10.1109/tvt.2020.3018817.
20. Wang X, Ma L, Li H, Yin Z, Luan TH, Cheng N. Digital twin-assisted efficient reinforcement learning for edge task scheduling. arXiv:2208.01781. 2022.
21. Ogunseyi TB, Thiyagarajan G. An explainable LSTM-based intrusion detection system optimized by firefly algorithm for IoT networks. Sensors. 2025;25(7):2288. doi:10.3390/s25072288.
22. Lazzarini R, Tianfield H, Charissis V. Federated learning for IoT intrusion detection. AI. 2023;4(3):509–30. doi:10.3390/ai4030028.
23. Mao Y, Zhang J, Letaief KB. Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems. In: Proceedings of the 2017 IEEE Wireless Communications and Networking Conference. Piscataway, NJ, USA: IEEE; 2017. p. 1–6.
24. Luo J, Song Q, Guo F, Wu H, Som HM, Alahmari S, et al. Joint deep reinforcement learning strategy in MEC for smart internet of vehicles edge computing networks. Sustain Comput Inform Syst. 2025;46(4):101121. doi:10.1016/j.suscom.2025.101121.
25. Huang H, Ye Q, Zhou Y. Deadline-aware task offloading with partially-observable deep reinforcement learning for MEC. IEEE Trans Netw Sci Eng. 2021;9(6):3870–85. doi:10.1109/tnse.2021.3115054.
26. Zhu C, Pastor G, Xiao Y, Liu C, Fu S. Fog following me: latency and quality balanced task allocation in vehicular fog computing. In: Proceedings of the 2018 15th Annual IEEE International Conference on Sensing, Communication and Networking. Piscataway, NJ, USA: IEEE; 2018. p. 1–9.
27. Maleki EF, Mashayekhy L. Mobility-aware computation offloading in edge computing using prediction. In: Proceedings of the 2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC). Piscataway, NJ, USA: IEEE; 2020. p. 69–74.
28. Ngo MV, Luo TK, Huynh HT, Nguyen HT, Dutkiewicz E. Coordinated container migration and base station handover in mobile edge computing. In: Proceedings of the 2020 IEEE Global Communications Conference. Piscataway, NJ, USA: IEEE; 2020. p. 1–6.
29. Kaur N, Kumar A, Kumar R. TRAP: task-resource adaptive pairing for efficient scheduling in fog computing. Clust Comput. 2022;25(6):4257–73. doi:10.1007/s10586-022-03641-z.
30. Liu W, Li C, Zheng A, Zheng Z, Zhang Z, Xiao Y. Fog computing resource-scheduling strategy in IoT based on artificial bee colony algorithm. Electronics. 2023;12(7):1511. doi:10.3390/electronics12071511.
31. Tao F, Zhang H, Liu A, Nee AYC. Digital twin in industry: state-of-the-art. IEEE Trans Ind Inf. 2019;15(4):2405–15. doi:10.1109/tii.2018.2873186.
32. Liu M, Fang S, Dong H, Xu C. Review of digital twin about concepts, technologies, and industrial applications. J Manuf Syst. 2021;58(Pt B):346–61. doi:10.1016/j.jmsy.2020.06.017.
33. Alourani A, Alam M, Ali A, Khan IR, Samal CK. Hybrid AI-IoT framework with digital twin integration for predictive urban infrastructure management in smart cities. Comput Mater Contin. 2025;86(1):1–32. doi:10.32604/cmc.2025.070161.
34. Abdallah W, Alghamdi M. Digital twin-enabled AI for sustainable traffic management: real-time urban mobility optimization in smart cities. PeerJ Comput Sci. 2026;12(7):e3574. doi:10.7717/peerj-cs.3574.
35. Zhang K, Cao J, Zhang Y, Shen S, Wu D. Adaptive digital twin and multiagent deep reinforcement learning for vehicular edge computing and networks. IEEE Trans Ind Inf. 2022;18(2):1405–13. doi:10.1109/tii.2021.3088407.
36. Arsalan A, Umer T, Rehman RA, Bilal M, Mumtaz S. Digital twin-driven federated deep reinforcement learning for mobility-aware UAV-IoT coordination in smart agriculture. SSRN. 2025. doi:10.2139/ssrn.5986035.
37. Kesavan TV, Venkatesan R, Wong WK, Ng PK. Reinforcement learning based secure edge enabled multi-task scheduling model for internet of everything applications. Sci Rep. 2025;15(1):6254. doi:10.1038/s41598-025-89726-2.
38. Cheng N, Lyu F, Quan W, Foh CH, Zhang H, Jia W, et al. Space/aerial-assisted computing offloading for IoT applications: a learning-based approach. IEEE J Sel Areas Commun. 2019;37(5):1117–29. doi:10.1109/jsac.2019.2906789.

39. Puliafito C, Gonçalves DM, Lopes MM, Martins LL, Madeira E, Mingozzi E, et al. MobFogSim: simulation of mobility and migration for fog computing. Simul Model Pract Theory. 2020;101:102062.
40. Codecà L. LuSTScenario: Luxembourg SUMO traffic (LuST) scenario. 2016. GitHub Repository, Version 2.0 [cited 2026 Jan 23]. Available from: https://github.com/lcodeca/LuSTScenario.
41. Chen X, Jiao L, Li W, Fu X. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Netw. 2015;24(5):2795–808. doi:10.1109/tnet.2015.2487344.
42. Oueis J, Strinati EC, Barbarossa S. The fog balancing: load distribution for small cell cloud computing. In: 2015 IEEE 81st Vehicular Technology Conference (VTC Spring). Piscataway, NJ, USA: IEEE; 2015. p. 1–6.