future internet

MDPI

Article

Adaptive Service Migration in Hybrid MEC–Cloud Environments: A Queueing-Theoretic Framework for Split-User Offloading

Anna Kushchazli [1], Kseniia Leonteva [1], Darina Shiyapova [1], Alexandr Priscepov [1] and Irina Kochetkova [1,2,*]

[1] Department of Probability Theory and Cyber Security, RUDN University, 117198 Moscow, Russia; kushchazli-ai@rudn.ru (A.K.); leontyeva-ka@rudn.ru (K.L.); 1132226458@rudn.ru (D.S.); 1032255843@rudn.ru (A.P.)

[2] Federal Research Center "Computer Science and Control" of The Russian Academy of Sciences, 119333 Moscow, Russia

* Correspondence: kochetkova-ia@rudn.ru

Abstract

Resource-constrained Multi-Access Edge Computing (MEC) nodes cannot fully replace cloud infrastructure, yet existing service placement models treat edge hosting as an all-or-nothing decision. This paper proposes a queueing-theoretic framework for split-user offloading in hybrid MEC–cloud environments. The system is modeled as a Continuous-Time Markov Chain (CTMC) over a load-vector state space that admits a product-form stationary distribution. A delay-aware greedy orchestration policy determines, at every arrival and departure event, which service occupies the MEC node and how many of its users are offloaded from the cloud. Closed-form expressions are derived for average end-to-end (E2E) delay, MEC occupancy and saturation probabilities, per-service hosting probabilities, and delay-saving indicators. Numerical analysis of a five-service industrial scenario shows that the proposed split-user mechanism keeps the MEC node occupied for most of the observation time (around 97% at the baseline load), naturally prioritizes services with the largest aggregate latency benefit, and substantially reduces the average delay compared with a cloud-only configuration. The analytical results are validated by discrete-event simulation, which matches the CTMC values with relative discrepancy below 1% under the Poisson/exponential assumptions; additional simulations quantify the sensitivity to alternative arrival and service-time distributions. The framework provides analytically tractable, interpretable decision logic with negligible runtime overhead, making it a suitable analytical foundation for cloud service orchestration platforms that must meet strict QoS targets in next-generation edge networks.

Keywords: cloud-edge orchestration; split-user offloading; service migration; MEC–cloud systems; queueing theory; CTMC; delay optimization; resource allocation; next-generation networks

Check for updates

Academic Editors: Efthimia Mavridou and Eleni Vrochidou

Received: 30 March 2026
Revised: 30 April 2026
Accepted: 8 May 2026
Published: 14 May 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Networks will fundamentally transform digital ecosystems by delivering unprecedented data rates, ultra-low latency, and intelligent resource orchestration. To support emerging applications such as extended reality, autonomous systems, and real-time industrial automation, multi-access edge computing (MEC) has emerged as a critical architectural paradigm that extends cloud-like capabilities to the network edge [1,2]. As communication

infrastructures evolve beyond 5G, decentralization of computational resources becomes increasingly essential to meet stringent quality of service (QoS) requirements [2,3]. By deploying computing resources in close proximity to end-users, MEC significantly reduces latency and enhances overall network efficiency [3].

The fundamental distinction between MEC and traditional cloud computing lies in their topological organization. Cloud computing relies on centralized, often remote data centers, where service provisioning management involves sophisticated semantic and data-driven frameworks [4,5]. MEC, by contrast, employs a distributed network of edge nodes geographically near users. This proximity is indispensable for latency-sensitive applications, including immersive media, smart city services, and mission-critical industrial operations [2,6]. Service migration—the process of relocating a running service or its state between computational nodes—serves as a key mechanism to maintain QoS adherence and optimize resource utilization under dynamic conditions [7,8]. Unlike in cloud environments, where migration primarily targets load balancing and cost reduction, MEC migration must prioritize latency reduction and real-time performance guarantees [7,9].

Recent studies have explored adaptive migration strategies using machine learning and deep reinforcement learning (DRL) [10–12]. However, most existing models adopt an all-or-nothing approach, assuming that a service resides entirely either in the cloud or at the edge at any given time [7,8]. This assumption overlooks the benefits of split-user offloading, wherein a service's user base is dynamically partitioned between the edge and the cloud. Furthermore, DRL-based approaches operate as black boxes and lack closed-form performance guarantees, which hinders their deployment in systems requiring auditable, low-overhead control logic. As data-driven decision support approaches gain traction in cloud-based service provisioning [5], the need for rigorous analytical models of hybrid edge–cloud systems becomes particularly acute. Queueing theory provides a powerful mathematical foundation for modeling and optimizing such systems [13,14], offering tractable tools to evaluate performance metrics and design efficient resource management strategies under uncertainty. To the best of our knowledge, no existing work provides a continuous-time Markov chain (CTMC)-based model that simultaneously captures split-user offloading and an explicit delay-aware migration policy in a hybrid MEC–cloud system—the precise gap this paper addresses.

A related but different line of research considers task-level offloading between mobile devices and MEC servers. In such models, a computational task generated by a user may be executed locally, offloaded to the MEC, or split between local and edge resources, often under wireless-channel and energy constraints. Examples include energy-latency trade-off optimization in MEC networks, IRS-aided binary offloading, and RIS-aided cooperative MEC systems [15–17]. These studies operate mainly at the device–edge computation layer. In contrast, the present paper considers the MEC–cloud orchestration layer: individual user sessions are not divided into local and edge subtasks; instead, the population of users of the same service is split between a local MEC node and the corresponding cloud path. Therefore, the local/MEC split and the MEC/Cloud split address complementary levels of the offloading problem.

In this paper, we propose a queueing-theoretic framework for adaptive service migration in hybrid MEC–cloud environments with split-user offloading capability. The system comprises a resource-constrained MEC node and multiple remote cloud instances, each permanently hosting a dedicated service. A distinctive feature of the model is the ability to dynamically offload a subset of a service's users from the cloud to the MEC node, while the MEC node hosts users from only one service at a time. Building upon our earlier work [18,19], this paper establishes a rigorous mathematical foundation for analyzing hybrid MEC–cloud systems with adaptive split-user migration. Practical deployments require

consideration of additional factors such as migration overhead, heterogeneous network conditions, and multi-node edge environments; however, a solid analytical framework is essential for systematic exploration of fundamental system properties before addressing these applied challenges.

The main contributions of our study are as follows:

1. We propose a hybrid MEC–cloud queueing model that formalizes split-user offloading and a delay-aware greedy orchestration policy, which determines at every arrival and departure event which service is active at the MEC node and how many of its users are offloaded from the cloud.
2. We prove that the stationary distribution of the underlying CTMC admits a product-form solution that decouples across services, and derive closed-form expressions for the mean end-to-end (E2E) delay, per-service MEC hosting probability, MEC occupancy/saturation, and delay-saving indicators.
3. We validate the closed-form expressions using a discrete-event simulation (DES), extend the numerical evaluation to a heterogeneous five-service industrial MEC–cloud scenario, compare the proposed split-user offloading policy with a cloud-only baseline, and examine the sensitivity of the results to alternative arrival and service-time distributions.

In contrast to monolithic migration, where a service is moved to the edge as a whole, the proposed split-user mechanism allows only a capacity-limited subset of users of the selected service to be served at the MEC, while the remaining users continue to be served by the cloud.

The remainder of this paper is organized as follows: Section 2 reviews related work on MEC service migration, orchestration policies, and analytical queueing models. Section 3 details the system model and the proposed migration policy. Section 4 develops the queueing model and derives closed-form performance metrics. Section 5 presents numerical results and discusses practical implications and limitations. Section 6 concludes this paper and outlines future research directions.

2. Related Work

In this section, we review the literature relevant to the proposed framework across three thematic areas: service migration in MEC–cloud architectures, orchestration and offloading policies based on artificial intelligence (AI) and game-theoretic methods, and analytical queueing models for edge computing performance evaluation. Each area is examined in turn, culminating in the identification of the gap that the present work addresses.

2.1. MEC Service Migration

The foundational principle of MEC involves decentralizing computational resources from centralized data centers to the network edge [1,2]. This shift is critical for meeting the ultra-low latency and high-reliability demands of applications such as autonomous vehicles, the industrial Internet of Things (IoT), and extended reality [2,6]. The European Telecommunications Standards Institute (ETSI) has standardized the MEC framework and reference architecture [1] and has defined an Application Mobility Service (AMS) application programming interface (API) to support service continuity under user mobility [9].

While MEC primarily focuses on a flat architecture with computing resources deployed at the network edge close to base stations [2], fog computing extends this concept by introducing a hierarchical, multi-tier structure that places intermediate fog nodes between end devices and the cloud. This multi-layer approach enables more flexible resource distribution, better scalability for massive IoT deployments, and enhanced support for location-aware, delay-sensitive services [20,21]. Fog architectures are particularly advanta-

geous for applications requiring distributed intelligence, such as telepresence, vehicular networks, and industrial IoT [21,22]. Recent works have explored fog-enabled testbeds for next-generation microservices [23] and cross-layer multipath routing for live microservice migration [24].

Within this architecture, service migration emerges as a primary mechanism for maintaining performance under dynamic conditions such as user mobility and fluctuating load [7,8,25]. A comprehensive survey of service migration strategies in MEC, covering trigger mechanisms, migration targets, and performance trade-offs, is provided in [7]. Early migration strategies were often monolithic, treating a service as an indivisible unit that must reside entirely either in the cloud or at a single edge node [8]. While this simplifies management, it causes inefficient resource utilization when a resource-constrained edge node cannot accommodate an entire service but could beneficially host a subset of its users. More recent works have begun to challenge this paradigm: a hysteresis-based approach to migration decisions is presented in [26], application-aware migration of video services in MEC environments is examined in [27], and a proactive migration strategy for 5G-enabled vehicular networks is introduced in [25]. These works collectively motivate the need for analytical models that go beyond monolithic placement.

2.2. Orchestration Policies

Orchestration in hybrid edge–cloud environments involves complex decision-making: when to migrate, which service to prioritize, and where to place computational load. Research has increasingly turned to data-driven and AI-based methods to tackle this complexity [28]. Dynamic resource allocation and network slicing strategies have been studied for cloud-radio access network (RAN) environments [29], and federated learning has been applied to joint radio/MEC resource management in open RAN architectures [30].

A significant body of work employs RL and DRL to create adaptive policies that react to changing network states. A DRL-based approach for service migration in MEC is developed in [10]; online service placement with joint computation resource allocation using DRL is studied in [11]; and joint service migration and resource allocation for edge IoT systems using a long short-term memory-based proximal deep Q-network is proposed in [12]. Other approaches formulate the problem as a Markov decision process [31] or apply game-theoretic frameworks to model strategic interactions among edge nodes and users [32]. Multi-agent fog service placement approaches have extended these ideas to hierarchical edge-fog topologies [33], and exact algorithms for task offloading with service caching and dependency have been developed [34]. While these methods show promise in handling complexity and uncertainty, they often operate as black boxes, lacking interpretability and closed-form performance guarantees, while their training and inference overhead may be prohibitive for resource-constrained edge nodes. In contrast, the policy proposed in this paper is an interpretable, delay-aware greedy rule that provides analytically defined decision logic with a minimal computational footprint—properties directly relevant to cloud orchestration frameworks requiring auditable, low-overhead control logic.

2.3. Analytical Queueing Models for MEC

Queueing theory provides a rigorous mathematical foundation for analyzing MEC systems, where user arrivals and service times are inherently random. A Markovian queueing model with reneging for delay-constrained data offloading in an integrated cloud–fog–edge system is presented in [14], deriving closed-form expressions for blocking probability and mean delay. Performance modeling of vehicular edge computing with bursty task arrivals is addressed in [35] using a Markov Modulated Poisson Process (MMPP), while a worst-case performance analysis using stochastic network calculus is developed in [36],

providing delay bounds for mobile vehicular applications. An analytical queueing model for computation offloading with multiple heterogeneous MEC servers is studied in [13], the most closely related to the present work: it derives exact performance metrics for a multi-server MEC setting but does not consider service migration or split-user offloading. Non-product-form queueing network models for multi-class 5G chains are analyzed in [37]; by contrast, our model achieves a product-form stationary distribution, enabling more efficient closed-form computation.

Existing queueing models for MEC either retain the monolithic service assumption or model migration as a complete transfer of all users belonging to a service. This paper integrates insights from all three strands reviewed above: it formalizes the split-user offloading concept from the architectural perspective, achieves delay minimization through a lightweight analytical policy rather than complex AI, and extends the CTMC framework to a novel state space that admits a product-form stationary distribution. To the best of our knowledge, no prior work provides a CTMC-based analytical model that simultaneously captures split-user offloading and an explicit delay-aware migration policy in a hybrid MEC–cloud system—a gap the present paper fills.

3. System Model

In this section, we formalize the hybrid MEC–cloud architecture and define the migration policy that governs user placement. We first describe the system architecture and its key assumptions, then introduce the delay model and the QoS objective that motivates the migration decisions, and finally specify the delay-aware greedy orchestration policy, including its triggering conditions and migration logic.

3.1. System Architecture

This work considers a hybrid computing system consisting of a single resource-constrained MEC node and K remote cloud instances, one per service class. Let $\mathcal{K} = \{1, \ldots, K\}$ denote the set of services. Each service $k \in \mathcal{K}$ is permanently hosted by a dedicated cloud instance with bandwidth capacity C_k.

This architecture is used as a focused abstraction of an industrial edge deployment in which one local MEC node is attached to several existing back-end cloud services, such as control, telemetry, rendering, or analytics services. The single-MEC assumption isolates the main bottleneck considered in this paper: how a limited edge resource should be shared between heterogeneous service classes. A multi-MEC topology with inter-edge migration is an important extension, but it would introduce additional routing and association decisions beyond the scope of the present CTMC model.

The dedicated-cloud-per-service representation should be interpreted as a per-class cloud path with its own capacity and effective delay, rather than as a restriction that a physical cloud platform can run only one service. In practice, these parameters can represent logically separated service backends or isolated cloud slices. A shared elastic cloud can be incorporated in future work by making C_k and d_k state-dependent.

The MEC node has a limited bandwidth capacity C_0 and can simultaneously serve users of only one active service $s \in \{0\} \cup \mathcal{K}$, where $s = 0$ denotes the idle state. For the active service s, exactly m of its users are placed on the MEC node, while the remaining $n_s - m$ users of that service continue to be served by the corresponding cloud instance; users of all other services $k \neq s$ reside entirely in their respective clouds. This splitting of a service's user base between the edge and the cloud is referred to as the split-user offloading mechanism.

Users of each service k arrive according to a Poisson process with rate λ_k 1/s and have exponentially distributed service times with mean $1/\mu_k$ s, independently of their

placement. When placed on the MEC, each user of service k occupies a fixed bandwidth b_k bps; cloud users consume no MEC resources. The maximum number of users that can be simultaneously hosted on the MEC for service s is $M_s = \lfloor C_0/b_s \rfloor$, and the maximum cloud capacity for service k is $N_k = \lfloor C_k/b_k \rfloor$. The cloud always retains sufficient capacity to accept users already admitted to the system when they are reassigned from the MEC back to the cloud. New arrivals, however, are admitted only while $n_k < N_k$; otherwise, they are rejected by the finite admission bound introduced in assumption (A3). The main notation is summarized in Table 1.

Table 1. Main notation.

Parameter	Description
K	Number of services and cloud instances
$\mathcal{K} = \{1, \ldots, K\}$	Set of services
C_0	Bandwidth capacity of the MEC node, bps
C_k	Bandwidth capacity of the k-th cloud, bps
b_k	Bandwidth requirement per user for service k, bps
$M_s = \lfloor C_0/b_s \rfloor$	Maximum MEC capacity for service s
$N_k = \lfloor C_k/b_k \rfloor$	Maximum cloud capacity for service k
λ_k	Arrival rate for service k, 1/s
μ_k	Service rate for a user of service k, 1/s
d_0	E2E delay for a user served at the MEC node, s
d_k	E2E delay for a user of service k served in the cloud, s
$\Delta d_k = d_k - d_0$	Delay difference: benefit of MEC placement for service k, s
n_k	Total number of active users of service k in the system
$\mathbf{n} = (n_1, \ldots, n_K)$	Total number of active users in the system
m	Number of users of the active service placed at the MEC node
s	Index of the service currently hosted on the MEC node ($s = 0$: idle)
$(\mathbf{n}, m, s)$	Extended system state
$\mathcal{X}$	Extended admissible state space
$\mathcal{N}$	Effective state space reachable under the migration policy
$s(\mathbf{n})$	Policy function: index of the service selected for MEC placement
$m(\mathbf{n})$	Policy function: number of users placed at the MEC under $s(\mathbf{n})$
$d(\mathbf{n}, m, s)$	Total system E2E delay in state $(\mathbf{n}, m, s)$, s

3.2. Modeling Assumptions

For clarity and to make the scope of the analytical model explicit, we summarize the modeling assumptions used throughout this paper.

(A1) **Poisson arrivals.** Users of service class k arrive according to an independent Poisson process with rate λ_k. This assumption makes the vector of active users Markovian and is used in the product-form derivation.

(A2) **Exponential service times.** The service time of each user of class k is exponentially distributed with mean $1/\mu_k$ and is independent of the user's placement. Placement at the MEC or in the cloud changes the experienced end-to-end delay, but not the service completion rate. This assumption is required for the CTMC representation and is relaxed numerically through alternative service-time distributions.

(A3) **Finite admission capacity.** For each service class k, the total number of active users is bounded by N_k. An arriving user of class k is admitted only when $n_k < N_k$; otherwise, the arrival is rejected by the finite admission bound. This bound may represent a finite cloud-side admission limit or a numerical truncation of the state space. Users already admitted to the system are never lost when they are reassigned between the MEC and the cloud.

(A4) **Single active service at the MEC.** At any time, the MEC node hosts users of at most one service class $s \in \{0, 1, \ldots, K\}$, where $s = 0$ denotes the idle state. This captures

the case where a resource-constrained MEC node runs one active service container or network function at a time.

(A5) **Instantaneous and reversible reconfiguration.** Changes in the active service s and in the number of MEC-hosted users m occur instantaneously and do not consume additional bandwidth in the baseline CTMC model. The baseline CTMC therefore provides an idealized lower-delay reference case; explicit migration and reconfiguration overhead is discussed as a modeling limitation and left for future migration-aware extensions.

(A6) **Class-dependent effective delays.** A user served at the MEC experiences delay d_0, while a user of service class k served in the cloud experiences delay $d_k > d_0$. These values are interpreted as effective mean end-to-end delays that aggregate radio transmission, transport, processing, buffering, and protocol overheads. They can be obtained from measurements or from a lower-layer channel-aware model.

3.3. Delay Model and QoS Objectives

The E2E delay represents the total round-trip time experienced by a user, encompassing radio-channel transmission, processing at the serving node, buffering, encoding/decoding, and the return path. A user placed on the MEC node experiences a fixed delay d_0 due to physical proximity and the absence of a long-haul transmission segment. A user of service k served in the cloud experiences a significantly higher delay d_k, which includes inter-domain transmission, remote processing, and the return path. The quantity $\Delta d_k = d_k - d_0 > 0$ measures the per-user latency benefit of MEC placement for service k and serves as the primary criterion for migration decisions: a larger Δd_k implies a greater gain from moving users to the edge.

The QoS objective of the orchestration layer is to minimize the instantaneous total system delay, defined as the sum of E2E delays over all users present in the system:

$$d(\mathbf{n}, m, s) = m d_0 + \sum_{k \in \mathcal{K} \setminus \{s\}} n_k d_k + (n_s - m) d_s, \tag{1}$$

where $\mathbf{n} = (n_1, \ldots, n_K)$ is the vector of total user counts per service, m is the number of active-service users at the MEC, and s is the index of the active service. When $s = 0$ (MEC idle), the formula reduces to $d(\mathbf{n}, 0, 0) = \sum_{k \in \mathcal{K}} n_k d_k$, since all users reside in their respective clouds. Minimizing (1) at each event is the basis of the orchestration policy described next.

3.4. Migration Orchestration Policy

User placement is governed by a delay-aware greedy orchestration policy that operates in an event-driven manner: a placement decision is made upon every user arrival and every service completion. At each event, the policy selects the service s and the number of edge-hosted users m that minimize the total delay (1) subject to the capacity constraints $m b_s \leq C_0$ and $n_k b_k \leq C_k$ for all $k \in \mathcal{K}$. Priority is given to the service with the largest delay difference Δd_k, since its users derive the greatest benefit from MEC placement.

The policy distinguishes three outcomes after each event. If the currently active service s remains optimal, the MEC placement is updated in place: m increases by one on an arrival to s (provided MEC capacity permits) or decreases by one on a departure from s. If an alternative service $l \neq s$ becomes preferable, full migration occurs when $n_l \leq M_l$, switching the active service to l and placing all n_l of its users on the MEC; split-user offloading occurs when $n_l > M_l$, placing M_l users of service l on the MEC and returning the remaining $n_l - M_l$ to the cloud. The conditions triggering each outcome are summarized in Table 2. Migration is instantaneous and costless in this model—an idealization discussed in Section 5—and does not cause loss of users already admitted to the system since cloud-side reassignment is assumed feasible under (A3).

The reconfiguration is interpreted at the session-routing level. The model does not assume that a partially executed task is checkpointed and moved between the cloud and the MEC; instead, subsequent requests of the affected session are served according to the updated placement. Any additional handover or warm-up delay is outside the baseline CTMC model and is discussed as a limitation.

Table 2. Summary of migration conditions.

Transition Type	Migration Condition
Remain at service s	$m\Delta d_s \geq \max_{l \neq s} \min(n_l, M_l)\Delta d_l$
Full migration to service $l \neq s$, if $n_l \leq M_l$	$n_l \Delta d_l > m'^{\,1} \Delta d_s$
Split-user offloading to service $l \neq s$, if $n_l > M_l$	$M_l \Delta d_l > m'^{\,1} \Delta d_s$

[1] Here, m' denotes the updated value of m after the current arrival or departure event.

4. Queueing Model

In this section, we develop the queueing-theoretic model for the system. We define the CTMC state space and embed the migration policy into the stochastic formulation, specify the transition rates governing system dynamics, establish the product-form stationary distribution, and derive closed-form expressions for the key performance metrics: average E2E delay, per-service hosting probability, and MEC utilization.

4.1. CTMC Formulation and State Space

To describe the system dynamics, we introduce a CTMC $\{\mathbf{X}(t),\ t \geq 0\}$, where $\mathbf{X}(t) = (\mathbf{N}(t), M(t), S(t))$ captures the total number of active users, the number of users of the active service placed at the MEC node, and the active service index at time t. The extended state space $\mathcal{X}$ consists of all triples $(\mathbf{n}, m, s)$ satisfying the resource capacity constraints:

$$\begin{aligned}\mathcal{X} = &\{(\mathbf{0}, 0, 0)\} \\ &\cup \{(\mathbf{n}, m, s): \quad n_s = m,\ \ 0 < m < M_s,\ \ 0 \leq n_k \leq N_k,\ \ k \in \mathcal{K}\backslash\{s\},\ \ s \in \mathcal{K}\} \\ &\cup \{(\mathbf{n}, M_s, s): \quad M_s \leq n_s \leq N_s,\ \ 0 \leq n_k \leq N_k,\ \ k \in \mathcal{K}\backslash\{s\},\ \ s \in \mathcal{K}\}.\end{aligned}$$

State transitions in $\mathcal{X}$ occur at user arrival and service completion epochs. Migration is not an independent event type: it occurs instantaneously at the same epoch, whenever the policy requires a change in s or m. The transition structure is as follows: Let $\mathbf{e}_k$ denote the unit vector with a one in position k.

Arrivals. When a user of service k arrives in state $(\mathbf{n}, m, s)$ with $n_k < N_k$, the load vector updates to $\mathbf{n} + \mathbf{e}_k$ and the policy is re-evaluated:

- If $k = s$ and $m < M_s$: no migration; new state $(\mathbf{n} + \mathbf{e}_k, m + 1, s)$.
- If $k = s$ and $m = M_s$: MEC saturated, user goes to cloud; new state $(\mathbf{n} + \mathbf{e}_k, M_s, s)$.
- If $k \neq s$ and service k does not become preferable: no migration; new state $(\mathbf{n} + \mathbf{e}_k, m, s)$.
- If $k \neq s$ and full migration to k (i.e., $n_k + 1 \leq M_k$ and $(n_k + 1)\Delta d_k > m'\Delta d_s$): new state $(\mathbf{n} + \mathbf{e}_k, n_k + 1, k)$.
- If $k \neq s$ and split-user offloading to k (i.e., $n_k + 1 > M_k$ and $M_k \Delta d_k > m'\Delta d_s$): new state $(\mathbf{n} + \mathbf{e}_k, M_k, k)$.

Departures. When a user of service k completes in state $(\mathbf{n}, m, s)$ with $n_k > 0$, the load vector updates to $\mathbf{n} - \mathbf{e}_k$ and the policy is re-evaluated:

- If $k = s$ and $m > 0$: user was at MEC; new state $(\mathbf{n} - \mathbf{e}_k, m - 1, s)$, followed by possible migration if another service l becomes preferable.

- If $k \neq s$: user was in the cloud of service k; new state $(\mathbf{n} - \mathbf{e}_k, m, s)$, with possible migration if service s is no longer optimal.

In all cases, the post-event migration step applies the conditions of Table 2 with m' equal to the updated value of m after the arrival or departure, and selects the state $(\mathbf{n}', m', s')$ that minimizes $d(\mathbf{n}', m', s')$ from (1).

The migration policy defined in Section 3 uniquely maps every vector $\mathbf{n}$ to a placement decision $(m(\mathbf{n}), s(\mathbf{n}))$ via:

$$s(\mathbf{n}) = \underset{k \in \mathcal{K}}{\operatorname{argmax}} \min(n_k, M_k)\Delta d_k, \tag{2}$$

$$m(\mathbf{n}) = \min\left(n_{s(\mathbf{n})}, M_{s(\mathbf{n})}\right). \tag{3}$$

Consequently, the extended state $(\mathbf{n}, m, s)$ is fully determined by $\mathbf{n}$ alone, and the effective state space reduces to:

$$\mathcal{N} = \{\mathbf{n} : \quad 0 \leq n_k \leq N_k, \;\; k \in \mathcal{K}\}. \tag{4}$$

The process $\{\mathbf{N}(t), \;\; t \geq 0\}$ is therefore a well-defined CTMC on $\mathcal{N}$ with generator $\mathbf{Q}$. The CTMC is irreducible: the zero state $\mathbf{n} = \mathbf{0}$ is reachable from any state via a finite sequence of departures, and any state is reachable from $\mathbf{n} = \mathbf{0}$ via a finite sequence of arrivals. Irreducibility guarantees the existence and uniqueness of the stationary distribution. Because migration is an internal reconfiguration that does not alter the arrival or service-completion rates of any service class, the stationary distribution over $\mathcal{N}$ is independent of the specific migration policy.

4.2. Transition Rates

Projected onto the effective state space $\mathcal{N}$, the dynamics of the CTMC $\mathbf{N}(t)$ are governed by the following transition rates: arrivals increment the vector $\mathbf{n}$ by $\mathbf{e}_k$ at rate λ_k (provided $n_k < N_k$); departures decrement the vector $\mathbf{n}$ by $\mathbf{e}_k$ at rate $n_k\mu_k$ (provided $n_k > 0$):

$$\mathbf{Q}[\mathbf{n}, \mathbf{n} + \mathbf{e}_k] = \lambda_k, \quad n_k < N_k, \tag{5}$$

$$\mathbf{Q}[\mathbf{n}, \mathbf{n} - \mathbf{e}_k] = n_k\mu_k, \quad n_k > 0. \tag{6}$$

The diagonal elements are set in the standard way $\mathbf{Q}[\mathbf{n}, \mathbf{n}] = -\sum_{\mathbf{n}' \neq \mathbf{n}} \mathbf{Q}[\mathbf{n}, \mathbf{n}']$. After each transition $\mathbf{n} \to \mathbf{n}'$, the placement (m, s) is updated instantaneously according to (2) and (3). The transition rates (6) depend only on $\mathbf{n}$ and not on the current placement (m, s), which is the key structural property exploited in the next subsection.

4.3. Product-Form Stationary Distribution

The stationary distribution over the effective state space $\mathcal{N}$ admits a product form. The key observation is that the migration policy changes only the placement of already admitted users between the MEC and the cloud, whereas the load vector $\mathbf{n}$ changes only due to arrivals and service completions.

Proposition 1. *Under assumptions (A1)–(A5), the finite-state CTMC $\{\mathbf{N}(t), t \geq 0\}$ on $\mathcal{N}$ is irreducible and therefore positive recurrent. Its unique stationary distribution is:*

$$\pi(\mathbf{n}) = \frac{1}{G} \prod_{k \in \mathcal{K}} \frac{\rho_k^{n_k}}{n_k!}, \quad \mathbf{n} \in \mathcal{N}, \tag{7}$$

where $\rho_k = \lambda_k/\mu_k$ is the offered load of service class k, and:

$$G = \sum_{\mathbf{n}\in\mathcal{N}} \prod_{k\in\mathcal{K}} \frac{\rho_k^{n_k}}{n_k!} \tag{8}$$

is the normalization constant.

Proof. First,the state space $\mathcal{N} = \{\mathbf{n} : 0 \le n_k \le N_k,\ k \in \mathcal{K}\}$ is finite. From any state $\mathbf{n} \in \mathcal{N}$, the zero state can be reached through a finite sequence of service completions. Conversely, any state in $\mathcal{N}$ can be reached from the zero state through a finite sequence of admissible arrivals. Since the corresponding transition rates are positive on these edges, the CTMC is irreducible. Finiteness then implies positive recurrence and uniqueness of the stationary distribution.

Second, under the greedy policy (2) and (3), the placement variables (m, s) are deterministic functions of the load vector $\mathbf{n}$. A migration or reconfiguration event changes only this placement; it does not create or remove users. Therefore, the marginal process $\{\mathbf{N}(t)\}$ evolves only through the birth–death transitions:

$$\mathbf{n} \to \mathbf{n} + \mathbf{e}_k \quad \text{at rate } \lambda_k, \qquad n_k < N_k,$$

$$\mathbf{n} \to \mathbf{n} - \mathbf{e}_k \quad \text{at rate } n_k\mu_k, \qquad n_k > 0.$$

These are exactly the transition rates defined in Section 4.2 and they do not depend on the current MEC placement (m, s). This is the structural reason why the stationary distribution of the load vector is independent of the placement policy, provided that the policy only reassigns already admitted users and does not modify the class-wise arrival and service completion dynamics.

Third, the distribution (7) satisfies the detailed-balance equations on every admissible edge of the state space. Indeed, for any $\mathbf{n} \in \mathcal{N}$ such that $n_k < N_k$:

$$\lambda_k \pi(\mathbf{n}) = (n_k + 1)\mu_k \pi(\mathbf{n} + \mathbf{e}_k), \quad k \in \mathcal{K}. \tag{9}$$

Substituting (7) into both sides gives:

$$\lambda_k \frac{1}{G} \frac{\rho_k^{n_k}}{n_k!} \prod_{j\neq k} \frac{\rho_j^{n_j}}{n_j!} = (n_k + 1)\mu_k \frac{1}{G} \frac{\rho_k^{n_k+1}}{(n_k+1)!} \prod_{j\neq k} \frac{\rho_j^{n_j}}{n_j!}, \tag{10}$$

which holds because $\rho_k = \lambda_k/\mu_k$. Hence, the detailed-balance relations hold for all neighboring states, and therefore $\pi\mathbf{Q} = \mathbf{0}$. By uniqueness of the invariant distribution on the finite irreducible CTMC, $\pi(\mathbf{n})$ in (7) is the stationary distribution.

The migration policy affects performance metrics through the deterministic projection $\mathbf{n} \mapsto (m(\mathbf{n}), s(\mathbf{n}))$, but it does not affect the stationary distribution of $\mathbf{N}(t)$ itself. Consequently, delay, MEC hosting probabilities, and saturation metrics are computed by weighting the policy-induced placement decisions with the product-form probabilities (7), without enumerating the full extended state space $\mathcal{X}$. □

4.4. Performance Metrics

Using the stationary distribution (7), we derive closed-form expressions for the key QoS indicators evaluated at the user-session level.

The hosting probability of service k—the fraction of time the MEC node is assigned to service k—is:

$$\pi_k = \sum_{\mathbf{n}\in\mathcal{N}_k} \pi(\mathbf{n}), \quad k \in \mathcal{K}, \tag{11}$$

$$\mathcal{N}_k = \{\mathbf{n} \in \mathcal{N} : \quad s(\mathbf{n}) = k\}. \tag{12}$$

The average number of users of service k at the MEC node is:

$$\bar{m}_k = \sum_{\mathbf{n}\in\mathcal{N}_k} m(\mathbf{n})\pi(\mathbf{n}), \quad k \in \mathcal{K}. \tag{13}$$

The mean total system delay is the expected sum of E2E delays over all users in the system:

$$\bar{D} = \sum_{\mathbf{n}\in\mathcal{N}} \left[\sum_{k\in\mathcal{K}} n_k d_k - m(\mathbf{n})\Delta d_{s(\mathbf{n})} \right] \pi(\mathbf{n}). \tag{14}$$

The average delay saving due to MEC placement for service k is:

$$\delta_k = \bar{m}_k \Delta d_k, \quad k \in \mathcal{K}, \tag{15}$$

and the overall delay saving is $\delta = \sum_{k\in\mathcal{K}} \delta_k$. Finally, the MEC saturation probability is:

$$p_{\text{MEC}} = \sum_{\mathbf{n}\in\bar{\mathcal{N}}} \pi(\mathbf{n}), \tag{16}$$

$$\bar{\mathcal{N}} = \{\mathbf{n} \in \mathcal{N} : \quad m(\mathbf{n}) = M_{s(\mathbf{n})}\}. \tag{17}$$

5. Numerical Results and Discussion

In this section, we present the numerical results for the proposed hybrid MEC–cloud framework. We describe the industrial scenario and parameter baseline, analyze the impact of traffic load on all key metrics, examine the sensitivity of mean delay to cloud path quality and service duration, and discuss modeling limitations.

5.1. Scenario Description

The numerical study is carried out for a five-service industrial MEC–cloud scenario whose architecture is illustrated in Figure 1. The five service classes represent qualitatively distinct industrial applications with heterogeneous latency requirements, traffic intensities, and service durations. The default system parameters are summarized in Table 3. The MEC delay is fixed at $d_0 = 5$ ms for all services, whereas the cloud delays range from 60 ms to 150 ms. Therefore, the delay benefit of MEC placement, $\Delta d_k = d_k - d_0$, is service-dependent and ranges from 55 ms for Service 3 to 145 ms for Service 4.

Table 3. Default system parameters for the five-service industrial MEC–cloud scenario.

Component	Description	Parameter	Value	Units
Service 1 (Crane control)	User arrival rate	λ_1	0.8	1/s
	Mean service rate	μ_1	2.0	1/s
	Max. cloud capacity	N_1	5	users
Service 2 (AGV telemetry)	User arrival rate	λ_2	1.5	1/s
	Mean service rate	μ_2	1.5	1/s
	Max. cloud capacity	N_2	4	users

Table 3. *Cont.*

Component	Description	Parameter	Value	Units
Service 3 (Pred. maintenance)	User arrival rate	λ_3	1.2	1/s
	Mean service rate	μ_3	2.5	1/s
	Max. cloud capacity	N_3	4	users
Service 4 (Remote rendering)	User arrival rate	λ_4	0.6	1/s
	Mean service rate	μ_4	1.0	1/s
	Max. cloud capacity	N_4	4	users
Service 5 (Video analytics)	User arrival rate	λ_5	2.0	1/s
	Mean service rate	μ_5	1.8	1/s
	Max. cloud capacity	N_5	4	users
MEC node	MEC capacity, Service 1	M_1	3	users
	MEC capacity, Service 2	M_2	2	users
	MEC capacity, Service 3	M_3	4	users
	MEC capacity, Service 4	M_4	2	users
	MEC capacity, Service 5	M_5	3	users
	E2E delay at MEC	d_0	5	ms
Cloud servers	Cloud delay, Service 1	d_1	80	ms
	Cloud delay, Service 2	d_2	120	ms
	Cloud delay, Service 3	d_3	60	ms
	Cloud delay, Service 4	d_4	150	ms
	Cloud delay, Service 5	d_5	100	ms

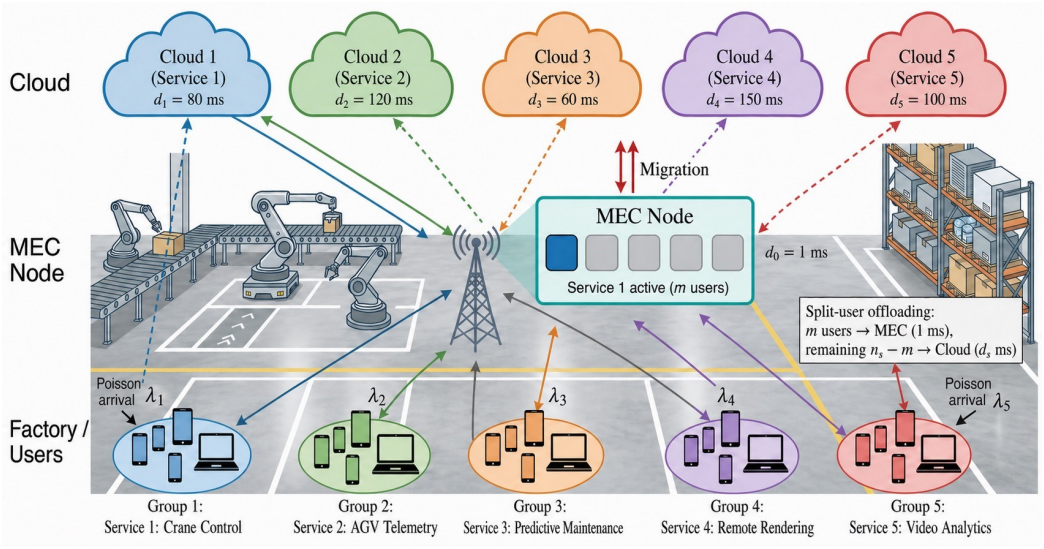


Figure 1. Hybrid MEC–cloud architecture with split-user offloading for five classes of industrial services: crane control (Service 1), AGV telemetry (Service 2), predictive maintenance (Service 3), remote rendering (Service 4) , and video analytics (Service 5).

The MEC delay is $d_0 = 5$ ms for all service classes; therefore, the corresponding delay benefits are $\Delta d = (75, 115, 55, 145, 95)$ ms for Services 1–5, respectively. The offered loads are computed as $\rho_k = \lambda_k/\mu_k$, giving $\rho_1 = 0.8/2.0 = 0.400$, $\rho_2 = 1.5/1.5 = 1.000$, $\rho_3 = 1.2/2.5 = 0.480$, $\rho_4 = 0.6/1.0 = 0.600$, and $\rho_5 = 2.0/1.8 \approx 1.111$.

The parameter set deliberately combines service classes with different roles. Service 4 is the most delay-critical class: it has the largest cloud delay ($d_4 = 150$ ms) and the largest MEC placement benefit ($\Delta d_4 = 145$ ms), although its offered load is moderate ($\rho_4 = 0.600$). Service 5 is the most load-intensive class, with the highest offered load ($\rho_5 \approx 1.111$) and a substantial delay benefit ($\Delta d_5 = 95$ ms). Service 2 combines unit offered load ($\rho_2 = 1.000$) with a large delay benefit ($\Delta d_2 = 115$ ms), making it another strong candidate for MEC

placement. By contrast, Service 1 has moderate load ($\rho_1 = 0.400$) and moderate latency gain ($\Delta d_1 = 75$ ms), whereas Service 3 has the smallest cloud delay ($d_3 = 60$ ms), the smallest MEC placement benefit ($\Delta d_3 = 55$ ms), and offered load $\rho_3 = 0.480$, making it the least latency-critical class in the considered scenario.

Under the policy (2) and (3), the MEC placement priority is not determined by Δd_k alone. Instead, the active service is selected according to the aggregate instantaneous delay saving $\min(n_k, M_k)\Delta d_k$. Therefore, the actual MEC occupancy depends jointly on the delay benefit, the offered load, and the service-specific MEC capacity M_k. In the baseline configuration, Services 2, 4, and 5 are expected to be the main competitors for MEC placement. The sensitivity analyses that follow vary the load scaling factor a (all λ_k scaled by $a \in [0.5, 1.5]$), the cloud delay ratio d_k/d_0, and the service time scale $1/\mu$ multiplier; for each experiment, only the indicated parameter deviates from the baseline in Table 3.

5.2. Impact of Traffic Load

We first examine how the main performance metrics evolve as the traffic intensity is uniformly scaled. The load scaling factor a multiplies all arrival rates simultaneously, $\lambda_k \to a\lambda_k$, so that the offered loads $\rho_k = a\lambda_k/\mu_k$ grow proportionally, while the service rates, capacity constraints, and delay parameters remain fixed.

Figure 2 shows the probability π_k that service k is active at the MEC node. At the baseline load, the MEC node is occupied for approximately 97.4% of the time, with idle probability only about 2.6%. The hosting probabilities are highly uneven: $\pi_1 = 0.0421$, $\pi_2 = 0.3512$, $\pi_3 = 0.0268$, $\pi_4 = 0.2826$, and $\pi_5 = 0.2714$. Thus, MEC occupancy is mainly shared by Services 2, 4, and 5. Service 2 is selected most frequently because it combines unit offered load with a large delay benefit $\Delta d_2 = 115$ ms. Service 4 is also frequently selected owing to the largest per-user delay benefit $\Delta d_4 = 145$ ms, while Service 5 receives a substantial share of MEC hosting time because it has the highest offered load, $\rho_5 \approx 1.111$. Services 1 and 3 are selected only rarely since their aggregate instantaneous delay saving $\min(n_k, M_k)\Delta d_k$ is typically lower.

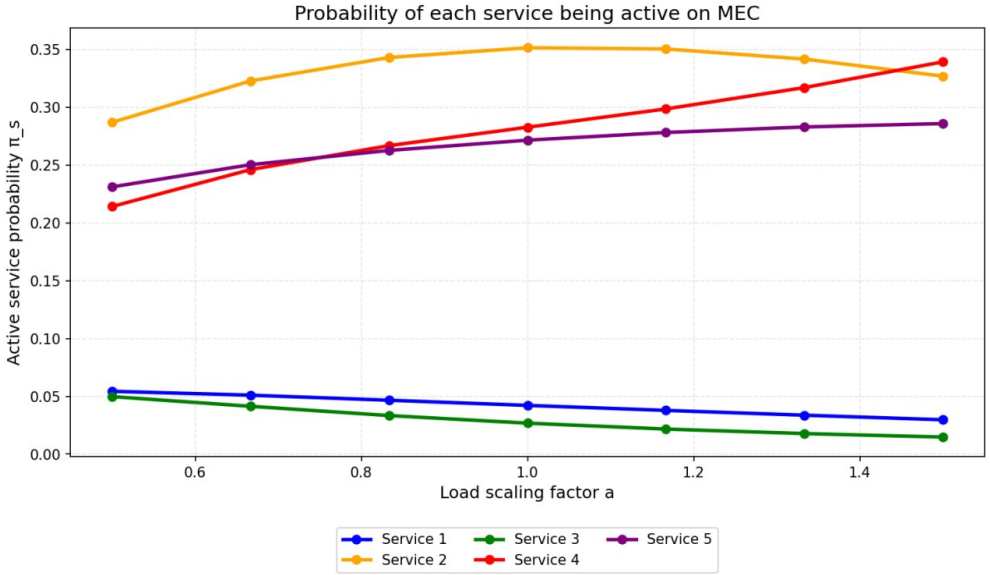


Figure 2. Probability π_s that service s is active at the MEC node vs. load scaling factor a for the five-service scenario.

Figure 3 shows the conditional probability that the active service fully occupies the MEC node, i.e., the probability that $n_k \geq M_k$ under the condition $s(\mathbf{n}) = k$. This metric characterizes how often the selected service reaches its service-specific MEC capacity once it is

active. The saturation behavior is service-dependent because the classes differ both in traffic intensity and in MEC capacity. Services with smaller MEC capacities, such as Services 2 and 4 with $M_2 = M_4 = 2$, reach full MEC occupancy more easily once selected, whereas services with larger MEC capacities require more simultaneously active users to saturate the MEC. As the load scaling factor a increases, the conditional saturation probabilities generally increase, reflecting stronger competition for the limited edge resource.

Figure 4 presents the unconditional contribution of each service class to MEC saturation. In contrast to the conditional probabilities in Figure 3, this metric also accounts for how often each service is selected by the orchestration policy. Therefore, a service contributes strongly to unconditional MEC saturation only if it is both frequently active and likely to fill its allocated MEC capacity. The dominant contributions are associated with the service classes that combine high MEC selection probability, limited service-specific MEC capacity, and substantial delay benefit.

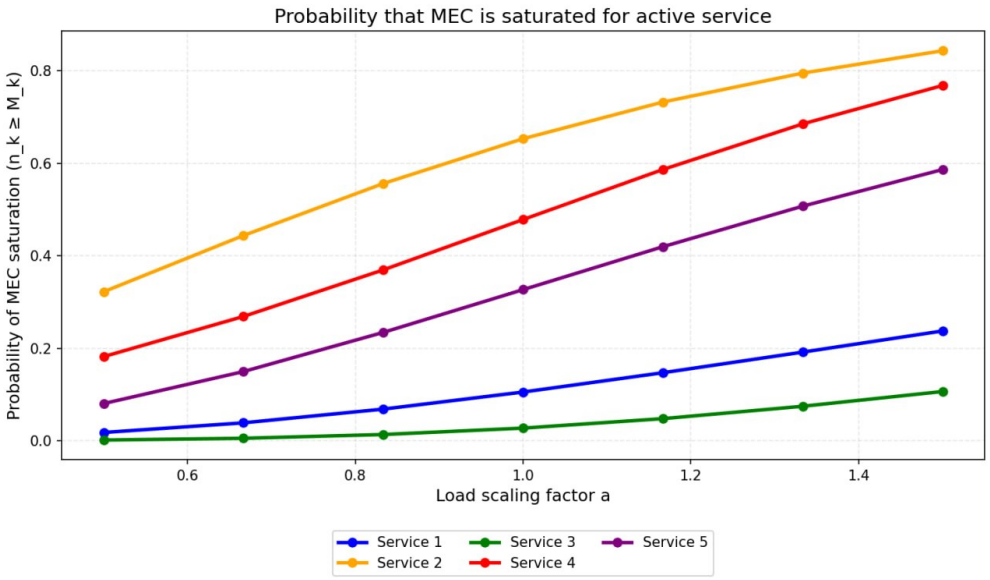


Figure 3. Conditional MEC saturation probability vs. load scaling factor a.

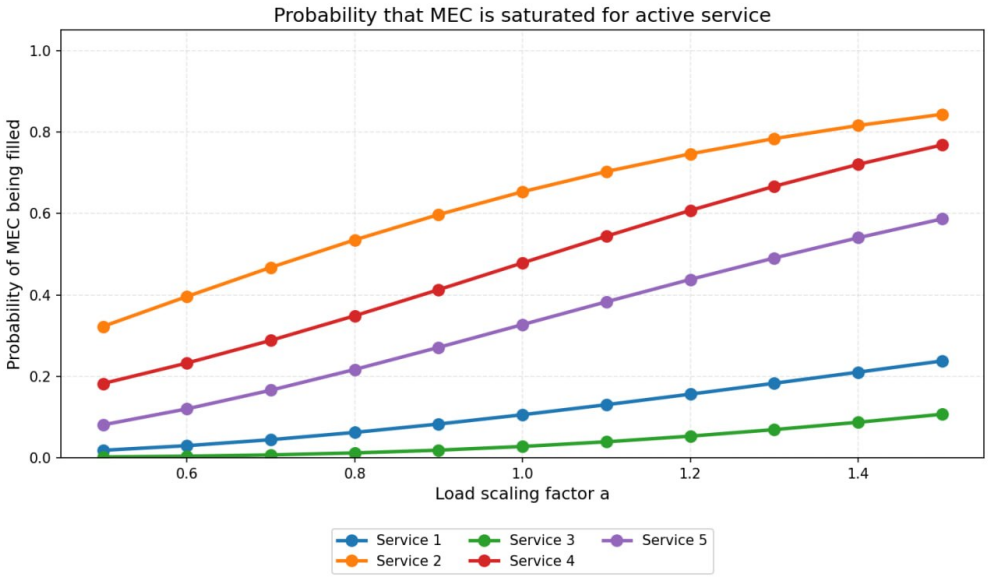


Figure 4. MEC saturation probability vs. load scaling factor a.

Figure 5 shows the average user delay under three cloud-latency scenarios as the load scaling factor a increases. The three curves correspond to low-, medium-, and high-latency cloud configurations, while the MEC delay and the service-capacity parameters remain fixed. In all cases, the average delay increases monotonically with load, because a larger number of active users intensifies competition for the limited MEC capacity and leaves more users served through the cloud path.

The cloud-latency level has a pronounced impact on the resulting delay. At the baseline load, the average user delay is approximately 20 ms in the low-latency cloud scenario, about 50 ms in the medium-latency scenario, and about 87 ms in the high-latency scenario. This confirms that the proposed split-user offloading policy is especially beneficial in regimes where the cloud path delay is large: the larger the gap between MEC and cloud latency, the more important the delay-aware service selection becomes.

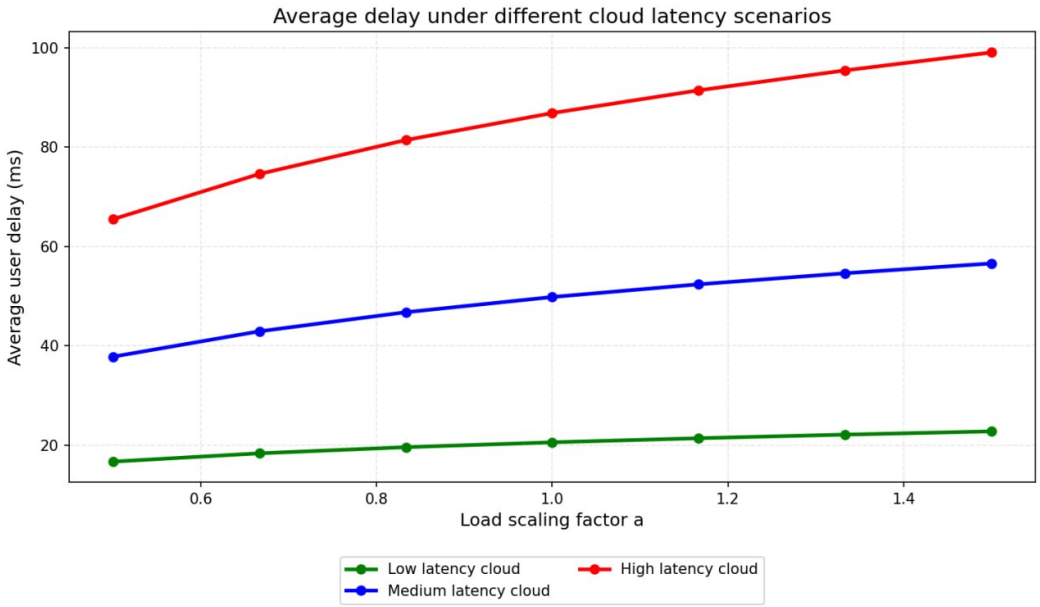


Figure 5. Average user delay (ms) vs. load scaling factor a.

Figure 6 shows the contribution of each service class to the aggregate delay saving achieved by MEC placement. This contribution is determined by two factors: how often the service is selected for MEC hosting and how large its per-user delay benefit Δd_k is. Therefore, the dominant contributors are not necessarily the services with the largest offered load or the largest cloud delay alone, but the services with the largest aggregate effect under the policy criterion $\min(n_k, M_k)\Delta d_k$.

At the baseline load $a = 1$, the largest delay-saving contributions are provided by Services 2 and 5, approximately 2.9 ms and 2.85 ms, respectively, followed by Service 4, with approximately 2.1 ms. Services 1 and 3 contribute only about 0.36 ms and 0.25 ms, respectively. This confirms that the aggregate effect of MEC placement is concentrated in Services 2, 4, and 5: Service 2 is selected most frequently, Service 4 has the largest per-user delay benefit, and Service 5 combines the highest offered load with a substantial delay benefit.

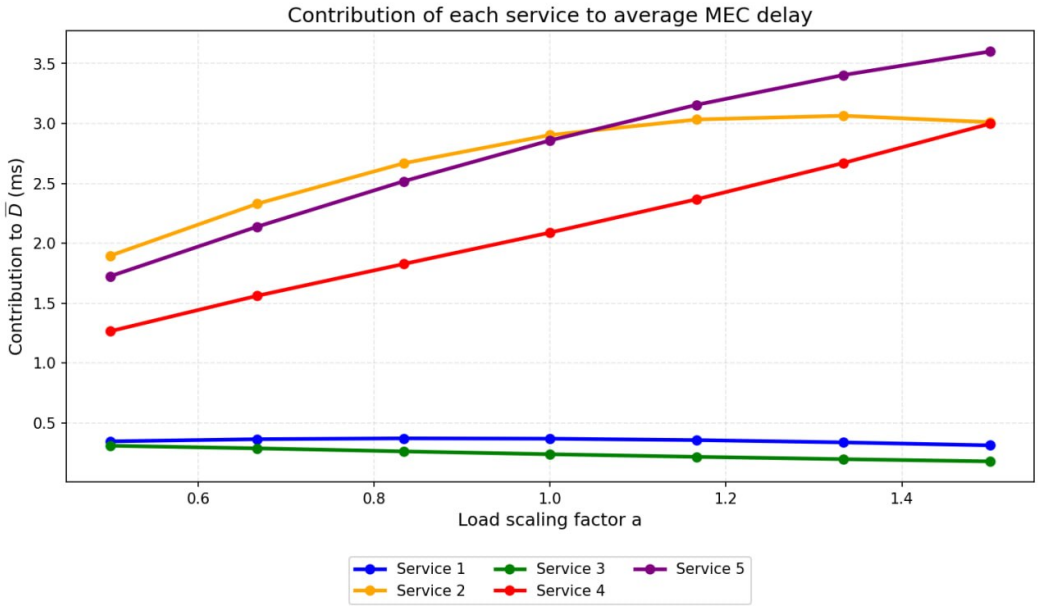


Figure 6. Contribution of each service class to the aggregate delay saving δ vs. load scaling factor a.

5.3. Impact of Cloud Delay and Service Time

We next examine how the delay-related metrics respond to changes in cloud-path latency and service duration. First, we vary the normalized cloud delay ratio d_k/d_0 to quantify how the benefit of MEC placement grows as the cloud path becomes slower relative to the MEC path. Second, we vary the service-time scale multiplier to evaluate the effect of longer sessions on MEC contention.

Figure 7 shows the average delay saving obtained through MEC placement as a function of the normalized cloud delay ratio d_k/d_0. The ratio is varied while the remaining parameters are fixed at the baseline values. The vertical dashed line marks the boundary $d_k/d_0 = 1$, where MEC and cloud delays are equal and edge placement provides no latency benefit.

For all services, the delay saving increases as the cloud path becomes slower relative to the MEC path. Service 5 exhibits the steepest growth, followed by Service 2 and Service 4, because these services combine substantial MEC hosting probability with large latency benefits. Services 1 and 3 grow more slowly, which is consistent with their lower MEC-selection probabilities. This confirms that the proposed policy becomes increasingly beneficial as the cloud-to-MEC delay gap widens.

Figure 8 shows the effect of service-time scaling on the average MEC-related delay. In this experiment, the mean service time of each class is multiplied by the same scale factor, while the arrival rates, capacity limits, and delay parameters remain fixed. As the service time scale increases, users remain active in the system for longer periods, which increases contention for the limited MEC capacity and raises the delay for all service classes.

The sensitivity to service-time scaling is service-dependent. Services 3 and 5 exhibit the steepest growth at large scale factors: Service 3 increases to more than 14 ms at scale factor 2, while Service 5 grows to approximately 13.7 ms. Service 1 shows a moderate increase, reaching about 11.6 ms. Services 2 and 4 remain below 10 ms over the considered range, indicating that their MEC placement remains relatively stable even when service durations are scaled upward. Overall, the experiment confirms that longer sessions amplify competition for the MEC resource and can change the relative delay sensitivity of the service classes.

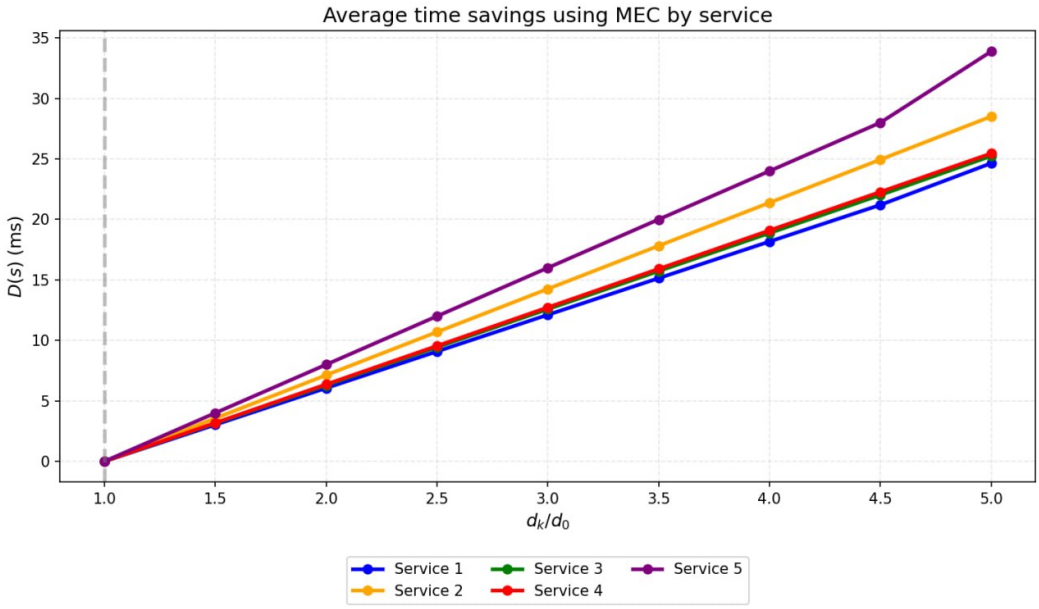


Figure 7. Average delay saving obtained through MEC placement vs. normalized cloud delay ratio d_k/d_0. The grey dashed vertical line indicates the baseline case $d_k/d_0 = 1$, where $d_k = d_0$ and no delay saving is obtained.

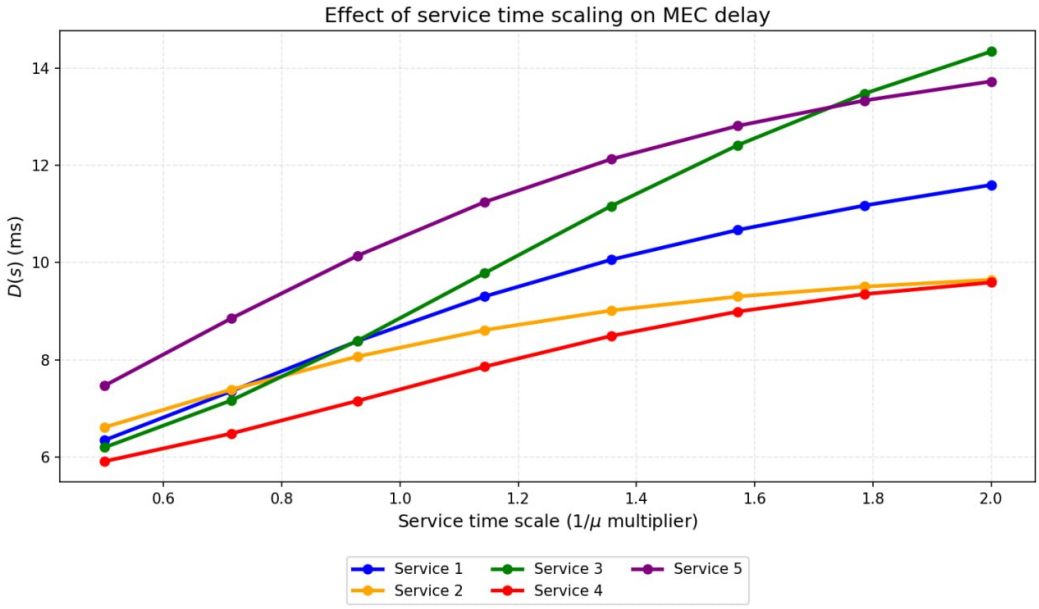


Figure 8. Mean E2E delay vs. service time scale multiplier ($1/\mu$ multiplier).

5.4. Discrete-Event Simulation Validation and Baseline Comparison

To validate the analytical CTMC-based model and to assess the sensitivity of the results to the assumed input distributions, we developed a discrete-event simulation (DES) model of the same hybrid MEC–cloud system. The simulator stores the load vector $\mathbf{n} = (n_1, \ldots, n_K)$, the currently active service at the MEC node s, and the number of users of this service placed at the MEC node, m. After each event, either a user arrival or a service completion, the same greedy decision rule as in the analytical model is applied to update the MEC placement.

In contrast to the analytical CTMC model, the DES does not require all event times to be generated under Markovian assumptions. Instead, inter-arrival times and service times are generated explicitly from specified distributions. This allows us to verify the analytical model in the Markovian case and to evaluate how the system behaves when the arrival

or service-time distribution is changed, while the mean values remain consistent with the baseline parameters in Table 3.

Four distributional configurations are considered. The baseline Markovian configuration, denoted as "pois/exp", uses Poisson arrivals, equivalently exponentially distributed inter-arrival times, and exponential service times. This case provides a direct verification of the analytical CTMC model. The "unif/unif" case uses uniform inter-arrival and service times. The mixed "exp/unif" case combines exponential inter-arrival times with uniform service times, while the reverse "unif/exp" case combines uniform inter-arrival times with exponential service times. The load scaling factor a is shown on the horizontal axis, and the vertical axis reports the mean system delay computed using the same delay metric as in the analytical model. The DES-based sensitivity analysis of the mean system delay with respect to the arrival- and service-time distributions is summarized in Table 4. Values are given in milliseconds.

Table 4. DES-basedsensitivity of the mean system delay to arrival and service-time distributions.

a	Analytical	Pois./Exp.	Unif./Unif.	Exp./Unif.	Unif./Exp.
0.5000	70.59	70.17	77.06	83.18	73.97
0.6667	108.45	108.43	119.20	130.15	113.58
0.8333	149.46	149.77	163.24	182.13	155.66
1.0000	192.57	191.63	208.06	237.00	200.25
1.1667	236.97	236.40	254.13	294.08	245.60
1.3333	282.00	281.30	301.44	350.13	292.07
1.5000	327.22	327.00	348.72	406.67	339.08

The fully Markovian DES configuration closely matches the analytical values over the entire load range. For instance, at the baseline load $a = 1$, the analytical mean system delay is 192.57 ms, while the corresponding DES estimate is 191.63 ms. Across all considered load values, the relative discrepancy remains below 1%, which validates the analytical CTMC formulation under the Poisson/exponential assumptions.

Changing the arrival or service-time distribution modifies the absolute delay level, but the qualitative dependence on the load factor remains the same. The mean system delay increases monotonically with a in all configurations. The largest delays are observed in the "exp/unif" case, reaching 237.00 ms at $a = 1$ and 406.67 ms at $a = 1.5$. The "unif/exp" and "unif/unif" cases remain closer to the Markovian baseline, although they still produce higher delays than the analytical Poisson/exponential model. These results indicate that the Markovian model should be interpreted as an analytically tractable baseline, while the proposed split-user offloading policy preserves its qualitative behavior under distributional perturbations.

Figure 9 presents the line-plot comparison of the mean system delay under different arrival/service-time distribution combinations. Figure 10 complements these results by showing the same data in a pointwise bar-chart form for each load factor. This representation makes it easier to compare the absolute differences between the analytical values and the DES results for each distributional configuration at fixed values of a. The bar-chart view again confirms that the Poisson/exponential DES results are nearly identical to those of the analytical model, while the non-Markovian cases produce a consistently higher mean system delay, especially for the "exp/unif" configuration.

At the baseline load $a = 1$, the bar chart clearly shows the ordering of the considered configurations: the analytical model and the Poisson/exponential DES results are approximately 193 ms and 192 ms, respectively, followed by "unif/exp" at about 200 ms, "unif/unif" at about 208 ms, and "exp/unif" at about 237 ms. The same ordering is pre-

served at higher load levels, with the largest deviation from the analytical baseline observed again for the "exp/unif" case.

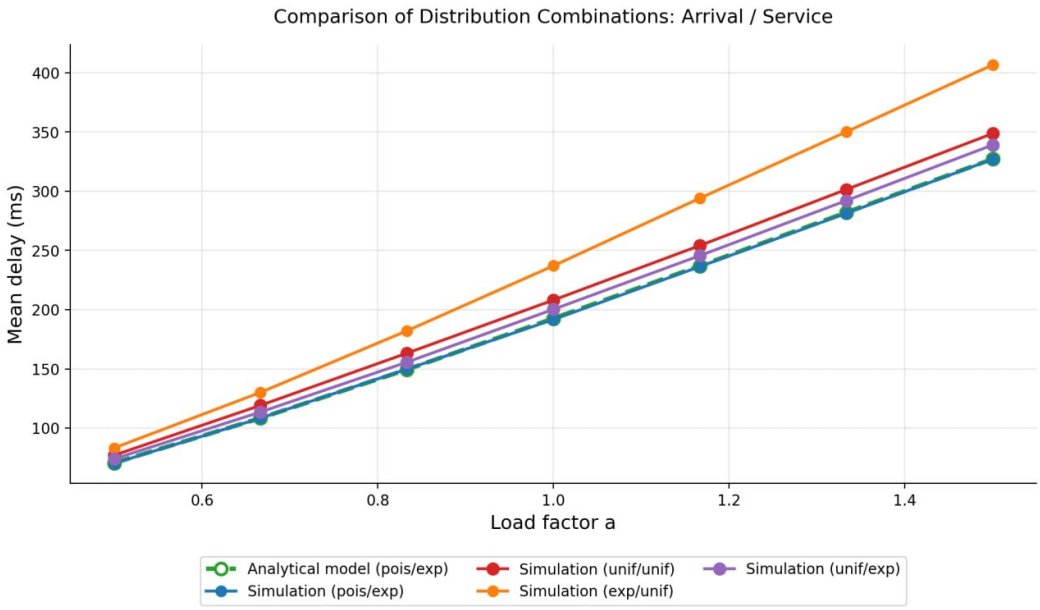


Figure 9. Comparison of mean system delay under different arrival/service-time distribution combinations. The analytical CTMC curve under the Poisson/exponential assumptions closely matches the corresponding DES simulation, while non-Markovian alternatives preserve the same increasing trend with respect to the load scaling factor.

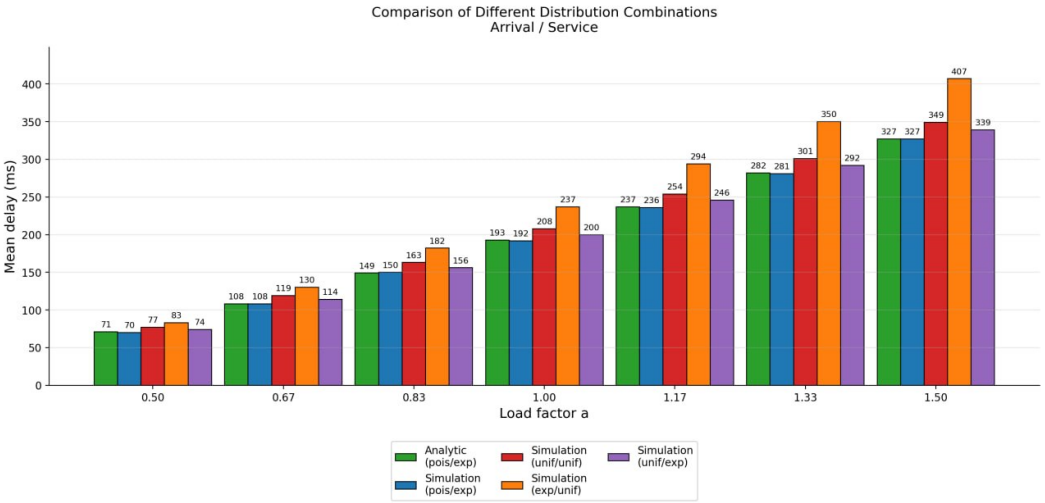


Figure 10. Bar-chart comparison of the mean system delay under different arrival/service-time distribution combinations for each load factor a.

Finally, we compare the proposed MEC-assisted split-user offloading scheme with a cloud-only baseline. In the cloud-only configuration, all users are served through their corresponding cloud paths and the MEC node is not used. In the proposed configuration, the limited MEC capacity is dynamically assigned to the service class that provides the largest aggregate instantaneous delay saving according to (2) and (3).

Figure 11 shows that MEC-assisted split-user offloading substantially reduces the average system delay over the entire load range. The cloud-only delay remains close to 90 ms and increases only moderately with a, whereas the proposed MEC-assisted scheme keeps the average delay below approximately 17 ms even at the highest considered load. The relative delay reduction decreases from about 91.5% at $a = 0.5$ to about 82.4% at

$a = 1.5$, remaining above 80% throughout the experiment. This confirms that even a capacity-constrained MEC node can provide a large delay gain when its resources are allocated selectively according to the proposed delay-aware policy.

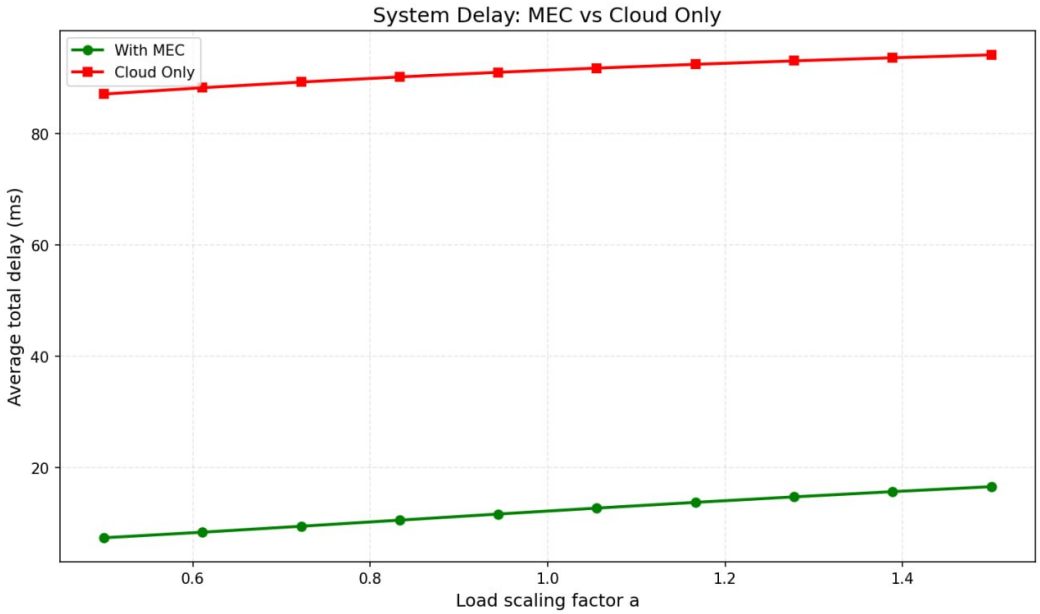


Figure 11. Average system delay with MEC-assisted split-user offloading vs. the cloud-only baseline.

5.5. Discussion

The presented results lead to several structural observations about the behavior of the proposed framework. First, the orchestration policy produces a naturally stratified allocation of MEC resources in the five-service scenario. Service 2 captures the largest share of MEC hosting time at the baseline load ($\pi_2 = 0.3512$), because it combines unit offered load ($\rho_2 = 1.000$) with a large delay benefit ($\Delta d_2 = 115$ ms). Service 4 is also selected frequently ($\pi_4 = 0.2826$), despite its moderate offered load ($\rho_4 = 0.600$), because it has the largest per-user MEC benefit ($\Delta d_4 = 145$ ms). Service 5 receives a substantial share of MEC hosting time ($\pi_5 = 0.2714$) due to the highest offered load ($\rho_5 \approx 1.111$) and a significant delay benefit ($\Delta d_5 = 95$ ms). By contrast, Services 1 and 3 are rarely selected ($\pi_1 = 0.0421$ and $\pi_3 = 0.0268$), which is consistent with their lower aggregate instantaneous delay-saving potential.

This stratification is not a design artifact but an emergent consequence of the greedy delay-minimization criterion (2) and (3). The policy does not prioritize the service with the largest offered load or the largest cloud delay in isolation. Instead, it maximizes the aggregate instantaneous saving $\min(n_k, M_k)\Delta d_k$; so, the MEC assignment depends jointly on the number of active users, the service-specific MEC capacity, and the cloud-to-MEC delay difference. This explains why Services 2, 4, and 5 dominate MEC occupancy, whereas Services 1 and 3 remain marginal under the considered traffic conditions.

Second, the split-user offloading mechanism differs conceptually from monolithic migration [7,8]. In all-or-nothing migration, a service is either placed at the MEC as a whole or remains entirely in the cloud. In contrast, the proposed mechanism allows only a capacity-limited subset of users of the selected service to be served at the MEC. When $n_s \leq M_s$, all users of the active service can be placed at the MEC; when $n_s > M_s$, the policy places M_s users at the MEC, while the remaining $n_s - M_s$ users of the same service continue to be served by the cloud. This configuration is particularly relevant when the MEC node is resource-constrained and cannot host the entire active service population.

The quantitative comparison with the cloud-only baseline in Figure 11 shows that MEC-assisted split-user offloading substantially reduces the average system delay over the considered load range. The relative delay reduction remains above 80%, decreasing from about 91.5% at $a = 0.5$ to about 82.4% at $a = 1.5$, with an improvement of approximately 86% near the baseline load. This confirms that even a capacity-constrained MEC node can provide a large delay gain when its resources are allocated selectively according to the proposed delay-aware policy.

Third, the DES validation confirms that the analytical CTMC model accurately reproduces the event-driven system dynamics under the Markovian assumptions. The Poisson/exponential DES results are nearly indistinguishable from the analytical values across the full load range; at $a = 1$, for example, the analytical mean system delay is 192.57 ms, while the corresponding DES estimate is 191.63 ms. This agreement supports the product-form stationary analysis and validates the closed-form computation of the main performance metrics.

The distributional sensitivity analysis clarifies the role of the Poisson/exponential assumptions. Replacing the arrival or service-time distribution changes the absolute delay level but preserves the monotone growth of the mean system delay with the load factor. The largest delays are observed for the "exp/unif" configuration, which reaches 237.00 ms at $a = 1$ and 406.67 ms at $a = 1.5$. Therefore, the Markovian model should be interpreted as an analytically tractable baseline rather than as a complete representation of all possible industrial traffic patterns. At the same time, the qualitative behavior of the proposed policy remains stable under the considered distributional perturbations.

The present study is subject to several modeling limitations. First, migration and MEC reconfiguration are modeled as instantaneous and costless. This idealization simplifies the CTMC analysis but excludes the latency, bandwidth cost, and control-plane overhead associated with live session transfer, session rerouting, or container warm-up. Therefore, the reported delay values should be interpreted as an idealized lower-delay reference case. Incorporating explicit migration overhead, for example, through a switching cost or a finite-duration migration phase, is a natural direction for follow-up work.

Second, the model uses class-dependent effective delays d_0 and d_k rather than a detailed channel-aware or computation-aware delay decomposition. This is appropriate for the service-orchestration layer considered in this paper, where d_0 and d_k can be interpreted as measured or precomputed mean end-to-end delays. More detailed wireless-channel, task-size, and processing-delay models can be coupled to the proposed framework by periodically updating these effective delay parameters.

Third, the baseline assumptions of Poisson arrivals and exponential service times enable the product-form result, but real industrial traffic may be bursty, correlated, or non-stationary. The DES results with alternative distributions provide a first sensitivity check, while more realistic extensions could use MMPP arrivals [35], phase-type service-time distributions, or stochastic network calculus bounds [36]. Finally, the single-MEC-node scope isolates the split-user offloading decision under one resource bottleneck. Multi-MEC topologies, shared edge resources, and inter-edge migration remain important directions for future investigation, building on multi-server frameworks [13].

6. Conclusions

This paper proposed a queueing-theoretic framework for adaptive service migration in hybrid MEC–cloud environments with split-user offloading. The system was modeled as a CTMC over the vector of active users, while the MEC placement was determined by a delay-aware greedy policy. Under the stated assumptions, the stationary distribution admits a product-form representation, which enables closed-form computation of the main

performance metrics, including service hosting probabilities, MEC saturation, average delay, and delay saving.

The numerical study considered a five-service industrial MEC–cloud scenario with heterogeneous arrival rates, service durations, cloud delays, and MEC capacities. The results showed that the proposed policy produces a naturally stratified allocation of MEC resources. At the baseline load, the MEC node is occupied for approximately 97.4% of the time, and its hosting time is mainly shared by Services 2, 4, and 5. This behavior follows directly from the policy criterion $\min(n_k, M_k)\Delta d_k$: the selected service is not determined by load or latency alone, but by their aggregate instantaneous delay-saving effect.

The split-user mechanism differs from all-or-nothing migration and provides a substantial gain over cloud-only operation. If $n_s \leq M_s$, all users of the active service can be placed at the MEC; if $n_s > M_s$, the MEC hosts M_s users, while the remaining $n_s - M_s$ users of the same service continue to be served by the cloud. This configuration is unavailable under monolithic migration, where a service must either be placed at the MEC entirely or remain in the cloud. The comparison with the cloud-only baseline showed that MEC-assisted split-user offloading reduces the average system delay by more than 80% over the considered load range, with an improvement of approximately 86% near the baseline load.

The analytical results were further validated by discrete-event simulation. Under the Poisson/exponential assumptions, the DES results closely matched the closed-form CTMC values, with relative discrepancy below 1% across the considered load range. Additional simulations with alternative inter-arrival and service-time distributions showed that non-Markovian inputs change the absolute delay level but preserve the qualitative load-dependent behavior of the proposed policy. Thus, the Markovian model serves as a tractable analytical baseline while still capturing the main structural effects of split-user offloading.

Future work will address three main extensions: relaxing the Poisson/exponential assumptions to phase-type or Markovian Arrival Process models; incorporating explicit migration overhead as a finite-duration service phase; and generalizing the architecture to multi-node MEC topologies with shared coverage, building on multi-server analytical frameworks [13] and energy-aware orchestration for next-generation networks [23].

Author Contributions: Conceptualization, I.K.; methodology, A.K. and I.K.; software, A.P.; validation, A.P.; investigation, D.S., A.P., K.L., and A.K.; writing—original draft preparation, D.S. and A.P.; writing—review and editing, A.K. and I.K.; visualization, D.S. and A.P.; supervision, I.K.; funding acquisition, I.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Science and Higher Education of the Russian Federation, project No. 075-15-2024-544.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Acknowledgments: The authors have reviewed and edited the output and take full responsibility for the content of this publication.

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

Abbreviations

The following abbreviations are used in this manuscript:

AGV	Automated Guided Vehicle
AI	Artificial Intelligence

AMS	Application Mobility Service
CTMC	Continuous-Time Markov Chain
DRL	Deep Reinforcement Learning
E2E	End-to-End
ETSI	European Telecommunications Standards Institute
IoT	Internet of Things
MEC	Multi-access Edge Computing
MMPP	Markov-Modulated Poisson Process
QoS	Quality of Service
RAN	Radio Access Network
RL	Reinforcement Learning

References

1. ETSI. *Multi-access Edge Computing (MEC); Framework and Reference Architecture*; Technical Report ETSI GS MEC 003 V4.1.1; European Telecommunications Standards Institute: Valbonne, France, 2025.
2. Taleb, T.; Samdanis, K.; Mada, B.; Flinck, H.; Dutta, S.; Sabella, D. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1657–1681. [CrossRef]
3. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656. [CrossRef]
4. Giakoumis, D.; Mavridou, E.; Votis, K.; Giannoutakis, K.; Tzovaras, D.; Hassapis, G. A Semantic Framework to Support the Management of Cloud-Based Service Provision Within a Global Public Inclusive Infrastructure. *Int. J. Electron. Commer.* **2015**, *20*, 142–173. [CrossRef]
5. Charizanis, G.; Mavridou, E.; Vrochidou, E.; Kalampokas, T.; Papakostas, G.A. Data-Driven Decision Support in SaaS Cloud-Based Service Models. *Appl. Sci.* **2025**, *15*, 6508. [CrossRef]
6. Huang, X.; Yu, R.; Kang, J.; He, Y.; Zhang, Y. Exploring Mobile Edge Computing for 5G-Enabled Software Defined Vehicular Networks. *IEEE Wirel. Commun.* **2017**, *24*, 55–63. [CrossRef]
7. Wang, S.; Xu, J.; Zhang, N.; Liu, Y. A Survey on Service Migration in Mobile Edge Computing. *IEEE Access* **2018**, *6*, 23511–23528. [CrossRef]
8. Frangoudis, P.A.; Ksentini, A. Service Migration Versus Service Replication in Multi-Access Edge Computing. In Proceedings of the 2018 14th International Wireless Communications and Mobile Computing Conference, IWCMC 2018, Limassol, Cyprus, 25–29 June 2018; pp. 124–129. [CrossRef]
9. ETSI. *Multi-Access Edge Computing (MEC); Application Mobility Service API*; Technical Report ETSI GS MEC 021 V2.2.1; European Telecommunications Standards Institute: Valbonne, France, 2022.
10. Wang, H.; Li, Y.; Zhou, A.; Guo, Y.; Wang, S. Service Migration in Mobile Edge Computing: A Deep Reinforcement Learning Approach. *Int. J. Commun. Syst.* **2023**, *36*, e4413. [CrossRef]
11. Liu, T.; Ni, S.; Li, X.; Zhu, Y.; Kong, L.; Yang, Y. Deep Reinforcement Learning Based Approach for Online Service Placement and Computation Resource Allocation in Edge Computing. *IEEE Trans. Mob. Comput.* **2023**, *22*, 3870–3881. [CrossRef]
12. Liu, F.; Yu, H.; Huang, J.; Taleb, T. Joint Service Migration and Resource Allocation in Edge IoT System Based on Deep Reinforcement Learning. *IEEE Internet Things J.* **2024**, *11*, 11341–11352. [CrossRef]
13. Li, K. Computation Offloading Strategy Optimization with Multiple Heterogeneous Servers in Mobile Edge Computing. *IEEE Trans. Sustain. Comput.* 2019, *early access*. [CrossRef]
14. Fantacci, R.; Picano, B. Performance Analysis of a Delay Constrained Data Offloading Scheme in an Integrated Cloud-Fog-Edge Computing System. *IEEE Trans. Veh. Technol.* **2020**, *69*, 12004–12014. [CrossRef]
15. Zhang, J.; Hu, X.; Ning, Z.; Ngai, E.C.H.; Zhou, L.; Wei, J.; Cheng, J.; Hu, B. Energy-Latency Tradeoff for Energy-Aware Offloading in Mobile Edge Computing Networks. *IEEE Internet Things J.* **2018**, *5*, 2633–2645. [CrossRef]
16. Yang, Y.; Gong, Y.; Wu, Y.C. Intelligent-Reflecting-Surface-Aided Mobile Edge Computing With Binary Offloading: Energy Minimization for IoT Devices. *IEEE Internet Things J.* **2022**, *9*, 12973–12983.
17. Liu, Z.; Li, Z.; Gong, Y.; Wu, Y.C. RIS-Aided Cooperative Mobile Edge Computing: Computation Efficiency Maximization via Joint Uplink and Downlink Resource Allocation. *IEEE Trans. Wirel. Commun.* **2024**, *23*, 11535–11550. [CrossRef]
18. Kushchazli, A.; Safargalieva, A.; Kochetkova, I.; Gorshenin, A. Queuing Model with Customer Class Movement Across Server Groups for Analyzing Virtual Machine Migration in Cloud Computing. *Mathematics* **2024**, *12*, 468. [CrossRef]
19. Kushchazli, A.; Leonteva, K.; Kochetkova, I.; Khakimov, A. Evaluating QoS in Dynamic Virtual Machine Migration: A Multi-Class Queuing Model for Edge-Cloud Systems. *J. Sens. Actuator Netw.* **2025**, *14*, 47. [CrossRef]

20. Sopin, E.; Nikita, Z.; Ageev, K.; Shorgin, S. Analysis of the Response Time Characteristics of the Fog Computing Enabled Real-Time Mobile Applications. In *Proceedings of the Lecture Notes in Computer Science, Marrakech, Morocco, 3–5 June 2020*; Springer: Cham, Switzerland, 2020; Volume 12525, pp. 99–109. [CrossRef]
21. Thang, D.V.; Volkov, A.; Muthanna, A.; Koucheryavy, A.; Ateya, A.A.; Jayakody, D.N.K. Future of Telepresence Services in the Evolving Fog Computing Environment: A Survey on Research and Use Cases. *Sensors* **2025**, *25*, 3488. [CrossRef]
22. Abdellah, A.R.; Mohamed, M.A.; Hassan, H.A.; Alsweity, M.; Muthanna, A.; Koucheryavy, A. Deep Learning-Powered Traffic Prediction for Autonomous Vehicles Using Integrated Fog and Multi-Cloud Services. In *Proceedings of the Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2026; Volume 15554, pp. 198–210. [CrossRef]
23. Kuzmina, E.; Tefikova, M.; Volkov, A.; Muthanna, A.; Ateya, A.A.; Koucheryavy, A. Microservice-Based Fog Testbed for 6G Applications. In Proceedings of the International Conference on Advanced Communication Technology, ICACT, Pyeongchang-gun, Republic of Korea, 3–7 February 2024; pp. 174–182. [CrossRef]
24. Tu, N.D.; Elgendy, I.A.; Khakimov, A.; Muthanna, A.; Kochetkova, I.; Samouylov, K.; Koucheryavy, A. Cross-Layer Multipath Routing for Live Microservice Migration in Fog Computing Networks. *IEEE Access* **2026**, *14*, 46671–46686. [CrossRef]
25. Pashazadeh, A.; Nardini, G.; Stea, G. A Proactive Migration Strategy for MEC Applications in 5G-Enabled Vehicular Networks. *IEEE Access* **2026**, *14*, 25486–25502. [CrossRef]
26. Poluektov, D.S.; Khakimov, A.A. Development and Analysis of Models for Service Migration to the MEC Server Based on Hysteresis Approach. *Discret. Contin. Model. Appl. Comput. Sci.* **2022**, *30*, 244–257. [CrossRef]
27. Manariyo, S.; Poluektov, D.; Abdukodir, K.; Muthanna, A.; Makolkina, M. Mobile Edge Computing for Video Application Migration. In *Proceedings of the Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2019; Volume 11660, pp. 562–571. [CrossRef]
28. Patsias, V.; Amanatidis, P.; Karampatzakis, D.; Lagkas, T.; Michalakopoulou, K.; Nikitas, A. Task Allocation Methods and Optimization Techniques in Edge Computing: A Systematic Review of the Literature. *Future Internet* **2023**, *15*, 254. [CrossRef]
29. AlQahtani, S.A. Towards an Optimal Cloud-Based Resource Management Framework for Next-Generation Internet with Multi-Slice Capabilities. *Future Internet* **2023**, *15*, 343. [CrossRef]
30. Martínez-Morfa, M.; Ruiz de Mendoza, C.; Cervelló-Pastor, C.; Sallent-Ribes, S. Federated Learning System for Dynamic Radio/MEC Resource Allocation and Slicing Control in Open Radio Access Network. *Future Internet* **2025**, *17*, 106. [CrossRef]
31. Shao, K.; Lv, B.; Wang, B.; Xu, Y. A Hybrid Meta-Heuristic Algorithm for Joint Edge Caching and Task Scheduling with Mobility Awareness. *Clust. Comput.* **2026**, *29*, 25. [CrossRef]
32. Moura, J.; Hutchison, D. Game Theory for Multi-Access Edge Computing: Survey, Use Cases, and Future Trends. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 260–288. [CrossRef]
33. Šatkauskas, N.; Venčkauskas, A. Multi-Agent Dynamic Fog Service Placement Approach. *Future Internet* **2024**, *16*, 248. [CrossRef]
34. Cui, B.; Zhang, J. Exact and Approximation Algorithms for Task Offloading with Service Caching and Dependency in Mobile Edge Computing. *Future Internet* **2025**, *17*, 255. [CrossRef]
35. Miao, W.; Min, G.; Zhang, X.; Zhao, Z.; Hu, J. Performance Modelling and Quantitative Analysis of Vehicular Edge Computing with Bursty Task Arrivals. *IEEE Trans. Mob. Comput.* **2023**, *22*, 1129–1142. [CrossRef]
36. Miao, W.; Min, G.; Yu, Z.; Zhang, X. Performance Analytical Modeling of Mobile Edge Computing for Mobile Vehicular Applications: A Worst-Case Perspective. *IEEE Trans. Mob. Comput.* **2024**, *23*, 8951–8964. [CrossRef]
37. Di Mauro, M. Performance Assessment of Multi-Class 5G Chains: A Non-Product-Form Queueing Networks Approach. *IEEE Trans. Netw. Serv. Manag.* **2026**, *23*, 789–802. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.