Journal of
Sensor and Actuator Networks

MDPI

Article

Evaluating QoS in Dynamic Virtual Machine Migration: A Multi-Class Queuing Model for Edge-Cloud Systems

Anna Kushchazli [1], Kseniia Leonteva [1], Irina Kochetkova [1,2,*] and Abdukodir Khakimov [1]

[1] Institute of Computer Science and Telecommunications, RUDN University, 117198 Moscow, Russia; kushchazli-ai@rudn.ru (A.K.); leontyeva-ka@pfur.ru (K.L.); khakimov-aa@rudn.ru (A.K.)
[2] Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, 119333 Moscow, Russia
* Correspondence: kochetkova-ia@rudn.ru

Abstract: The efficient migration of virtual machines (VMs) is critical for optimizing resource management, ensuring service continuity, and enhancing resiliency in cloud and edge computing environments, particularly as 6G networks demand higher reliability and lower latency. This study addresses the challenges of dynamically balancing server loads while minimizing downtime and migration costs under stochastic task arrivals and variable processing times. We propose a queuing theory-based model employing continuous-time Markov chains (CTMCs) to capture the interplay between VM migration decisions, server resource constraints, and task processing dynamics. The model incorporates two migration policies—one minimizing projected post-migration server utilization and another prioritizing current utilization—to evaluate their impact on system performance. The numerical results show that the blocking probability for the first VM for Policy 1 is 2.1% times lower than for Policy 2 and the same metric for the second VM is 4.7%. The average server's resource utilization increased up to 11.96%. The framework's adaptability to diverse server–VM configurations and stochastic demands demonstrates its applicability to real-world cloud systems. These results highlight predictive resource allocation's role in dynamic environments. Furthermore, the study lays the groundwork for extending this framework to multi-access edge computing (MEC) environments, which are integral to 6G networks.

Keywords: blocking probability; cloud system; continuous-time Markov chain; edge-cloud system; migration probability; migration; overloaded server; queuing system; server utilization; virtual machine

check for updates

Academic Editor: Lei Shu

Received: 13 March 2025
Revised: 17 April 2025
Accepted: 21 April 2025
Published: 25 April 2025

Citation: 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. https://doi.org/10.3390/jsan14030047

Copyright: © 2025 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 (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The sixth generation of communication networks (6G) is poised to revolutionize digital infrastructure by enabling unprecedented data transfer speeds, reliability, and enhanced cloud resource management capabilities. As organizations increasingly adopt digital transformation strategies, cloud computing services have become indispensable. These services, categorized into Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), offer varying levels of resource management and control [1]. Among these, IaaS provides the greatest flexibility but requires deeper architectural understanding, making efficient resource allocation and migration critical for optimal performance.

Virtual machine (VM) migration is a cornerstone of modern cloud computing, enabling dynamic resource allocation, load balancing, and fault tolerance. However, the growing complexity of cloud environments and the emergence of latency-sensitive applications,

such as those in 6G networks, demand more sophisticated migration strategies. These strategies must minimize downtime, reduce network overhead, and ensure seamless service continuity, making VM migration a highly relevant and active area of research [2].

Current VM migration techniques face several challenges. Traditional methods often rely on static rules or heuristic approaches, which may not adapt well to dynamic workloads or real-time resource demands. For instance, while containerization offers lightweight application isolation, it lacks robust live migration capabilities compared to VMs [3]. Existing VM migration algorithms, though effective in certain scenarios, struggle with balancing resource utilization, minimizing migration costs, and ensuring quality of service (QoS) under fluctuating workloads.

Moreover, many approaches do not fully account for the interplay between task arrival patterns, resource requirements, and server capacities. This can lead to suboptimal migration decisions, increased downtime, or resource contention. While predictive analysis and adaptive management technologies have been proposed to address these issues, their integration into practical systems remains limited, highlighting the need for more comprehensive and flexible solutions.

This paper addresses these challenges by proposing a novel VM migration model that integrates resource utilization and task processing dynamics [4]. Our approach leverages queuing theory and stochastic processes to optimize migration decisions, ensuring efficient load balancing and minimal service disruption. Specifically, we focus on three key aspects of VM migration: (1) when to migrate, (2) which VM to migrate, and (3) where to migrate. By considering both current and future resource states, our model minimizes the number of tasks affected during migration while maintaining high QoS. This research is related to sensor and actuator networks (SANs) by addressing key challenges in the dynamic management of resources for edge-cloud infrastructures. As SANs rely more on distributed computing models, such as cloud-edge coordination for real-time data processing in smart cities, Industry 4.0, and tactile internet applications, efficient VM migration becomes crucial for ensuring low-latency communication, scalability, and QoS.

The main contributions of our study are as follows:

- We developed a queuing-based model that captures the stochastic nature of task arrivals and processing, enabling dynamic and efficient VM migration decisions.
- Our approach explicitly considers server resource capacities and VM resource requirements, ensuring balanced utilization and minimizing migration overhead.
- The proposed model is adaptable to various cloud architectures, including edge computing environments, and can be extended to support containerized applications.

The rest of the paper is organized as follows. Section 2 provides an overview of related work in VM migration. Section 3 details the system model, including resource constraints and migration policies. Section 4 presents the queuing model for task processing and derives key performance metrics. Section 5 presents numerical results and discussion. Finally, Section 6 concludes the paper and outlines future research directions.

2. Background and Related Work

In this section, we explore the fundamental aspects of VM migration, focusing on the challenges, methodologies, and state-of-the-art solutions in the field. We begin by examining the core goals of VM migration, which include optimizing resource utilization, minimizing service disruptions, and ensuring high QoS. Next, we discuss the triggering mechanisms that initiate migration, highlighting the importance of timely and efficient decision-making. We then delve into the selection of VMs for migration. Finally, we analyze the server selection process.

2.1. Migration Goals

VM migration, the process of transferring running VMs between physical servers, is a critical mechanism for optimizing resource utilization and maintaining QoS in modern data centers. Effective migration strategies must address three fundamental questions:

- When to migrate? Determining the optimal timing is crucial to minimize service disruption and migration overhead. Premature migrations waste resources, while delayed migrations risk QoS degradation.
- Which VM to migrate? Selecting the appropriate VM involves balancing factors such as resource requirements, service criticality, and migration costs. The chosen VM should alleviate server overload without introducing new bottlenecks.
- Where to migrate? Identifying suitable target servers requires analyzing current and projected resource utilization to ensure stable operation post-migration.

These decisions are guided by the need to balance server loads, both in the present and future. Overloaded servers, which cause resource contention and potential service disruptions, are prioritized for VM offloading. Underloaded servers, while less critical, still require attention to prevent resource wastage. As highlighted by [5], overutilization is the primary driver for rebalancing procedures, as it directly impacts system performance and user experience.

Effective migration strategies must also consider the trade-offs between migration frequency and system stability. Frequent migrations can optimize resource utilization but may introduce excessive overhead, while infrequent migrations risk prolonged periods of suboptimal performance. The challenge lies in identifying the right balance to maintain high QoS with minimal disruption.

2.2. Migration Triggering

Load balancing in server environments is essential for maintaining system stability and performance. It can be triggered by various events, such as system malfunctions, the launch of new VMs, or significant changes in workload patterns. Load balancing algorithms can be categorized based on their initiation point: sender-initiated or receiver-initiated [6].

Sender-initiated algorithms proactively redistribute workloads from overloaded servers to underutilized ones. They monitor server loads and initiate migrations when thresholds are exceeded, ensuring that no single server becomes a bottleneck. Receiver-initiated algorithms focus on underloaded servers, redistributing their workloads to reduce the number of active servers. While effective in consolidating resources, this approach may incur additional costs due to increased migration overhead.

Load balancing methods are further divided into static and dynamic approaches [6–8]. Static methods rely on predefined rules and schedules. Round-Robin (RR) distributes VMs sequentially without considering their resource requirements. Weighted Round-Robin (WRR) accounts for VM characteristics, such as resource demands, to achieve a more balanced distribution. Opportunistic load balancing assigns VMs randomly, as proposed in [9]. Dynamic methods adapt to real-time system conditions, including Min-Max and Max-Min algorithms. They optimize task execution times by prioritizing either the shortest or longest tasks.

Effective load balancing requires continuous monitoring of system states. Common approaches include the following. Random distribution assumes that requests and resources are allocated randomly, providing simplicity but lacking optimization. By scheduling resources to be assigned based on current load conditions, as described in [10]. Minimum connections selects the server with the fewest active connections for migration, as suggested in [11]. Blockchain integration enhances system resilience by connecting servers

through blockchain technology, as proposed in [12]. This approach is particularly relevant for service migration in 6G networks [13].

Live migration ensures minimal service disruption by transferring VM states without interrupting access. Pre-copy migration continuously transfers memory pages from the source machine, minimizing downtime. This method is generally faster than post-copy migration [14–16]. Post-copy migration transfers memory pages after the VM has been restarted on the target server, reducing initial downtime but potentially increasing overall migration time. Finally, stop-and-copy temporarily halts the VM and transfers only frequently used memory pages, significantly reducing downtime and data transfer operations. After the initial transfer, the VM is quickly restarted on the target server.

These techniques and strategies collectively enable efficient load balancing and migration, ensuring high system performance and resource utilization.

2.3. Virtual Machine Selection

Effective VM selection is crucial for optimizing dynamic migration processes and ensuring balanced resource utilization. This involves identifying and addressing imbalances in computing resources, particularly by prioritizing overloaded servers and developing strategies to optimize underutilized ones. Migration is typically triggered when there is a risk of resource degradation, which could lead to system failures. Overloaded servers often cause resource contention, while underutilized servers, though less risky, represent inefficiencies in resource allocation [5,17].

When selecting a VM for migration, the following factors are considered: resource utilization, when VMs with lower CPU and RAM utilization are preferred, as they are easier to migrate with minimal impact on performance; migration cost, when VMs requiring fewer resources to migrate are prioritized to reduce overhead and downtime; and maximum potential growth policy that aims to maximize server utilization without incurring additional migration costs, ensuring efficient resource allocation [18,19].

Dynamic consolidation of VMs leverages user-provided information to enhance energy efficiency and resource management. Specifically, users indicate the expected termination time of their services, allowing the system to plan migrations and consolidations more effectively. This user-centric approach not only improves energy efficiency but also ensures that migrations are aligned with service requirements, minimizing disruptions [20].

Thereby, overloaded servers can lead to performance degradation, making it essential to identify and migrate VMs that alleviate contention. Underutilized servers, while less risky, contribute to energy inefficiencies. Dynamic consolidation helps address this by reducing the number of active servers. Migrations must be carefully planned to minimize disruptions to user services, particularly in environments with strict QoS requirements. By incorporating these criteria and strategies, the VM selection process ensures efficient resource utilization, minimizes migration costs, and maintains high system performance. This approach is particularly valuable in dynamic environments where workloads and resource demands fluctuate significantly.

2.4. Server Selection

Server selection is a critical component of VM migration, as it directly impacts resource utilization, power consumption, and QoS [21]. Efficient server selection ensures that VMs are migrated to servers that can accommodate their resource demands without causing overloading or underutilization. This process involves identifying both underloaded and overloaded servers, where underloaded servers represent inefficient resource usage, and overloaded servers exceed predefined thresholds for CPU, RAM, or network bandwidth [5,17].

The optimization goal may vary depending on the system's requirements, such as minimizing the number of active servers to reduce power consumption or maximizing resource utilization to improve performance [22]. For complex problems like hosting NP-hard VMs, heuristic and metaheuristic algorithms are often employed. These methods provide near-optimal solutions within reasonable computational time. Heuristic algorithms focus on specific problem characteristics to find feasible solutions quickly. Metaheuristic algorithms use higher-level strategies to explore the solution space more effectively, as demonstrated in [23,24].

AI-based methods have become increasingly important for server selection, enabling real-time monitoring and analysis of resource metrics. Reinforcement Learning (RL) optimizes decision-making processes using Markov decision processes, as discussed in [25]. Deep learning analyzes large datasets to identify patterns and trends, improving demand forecasting and resource allocation [26]. Linear regression predicts CPU consumption to prevent SLA violations, ensuring compliance with contractual obligations. The application of AI methods, as highlighted in [27], enhances cloud computing performance by optimizing dynamic planning, load balancing, and resource migration. These techniques train models to organize cloud resources efficiently, improving infrastructure management and reducing operational costs.

Markov chains are also utilized for server selection, providing a probabilistic framework for modeling system states and transitions. This approach, as explored in [28], offers a robust method for addressing similar optimization problems in cloud environments. By integrating these advanced techniques, server selection becomes a data-driven process that maximizes efficiency, minimizes costs, and ensures high QoS for all hosted services. In particular, there are two types of server selection. Firstly, the target server is the one that has space for the VM at the time of migration [29]. The other is when the system is looking at future utilization in order to minimize energy consumption [30,31]. Both types are used to find a good decision for optimizing the cloud infrastructure. Our research also uses these policies.

3. System Model

In this section, we formalize a cloud computing system with VM migration capabilities, focusing on resource-constrained edge-cloud environments. The model addresses dynamic task scheduling and server allocation for latency-sensitive applications such as 16K UHD video streaming, extended reality (XR), and holographic services. We begin by defining the system's core components and operational assumptions, followed by a detailed description of the migration policies that govern VM relocation.

3.1. General Assumptions

We model a cloud computing system consisting of S physical servers $\mathcal{S} = \{1, \ldots, S\}$, each with a fixed bandwidth capacity C_s (bps). These servers host V VMs $\mathcal{V} = \{1, \ldots, V\}$, which provide dedicated computational services to users. As part of the system, the user can determine the size, number, and type of VMs, the server storage volume, and their number and network settings. Performance depends on the configuration set by the user.

Each VM v is characterized by its resource demand b_v (bps) and task processing rate μ_v (1/s), with task execution times following an exponential distribution. Task arrivals to each VM are modeled as independent Poisson processes with rates λ_v (1/s), reflecting the stochastic nature of user requests.

The system enforces strict capacity constraints to ensure stable operation. Specifically, the total bandwidth demand on any server s must not exceed its capacity C_s. VMs are not statically bound to servers; they can migrate between servers to optimize resource

utilization while maintaining service continuity. This dynamic placement capability allows the system to adapt to changing workloads and prevent server overloads.

During the dynamic migration process, a VM is transferred between servers without interrupting its operation. The migration procedure is initiated by the control system (controller) after identifying the target server with the necessary computational resources. The process is coordinated by a special controller that monitors the resource usage in real time. During the migration, a sequential copy of the data is generated from the source server to the target server. This mechanism ensures the uninterrupted operation of the VM, eliminating interruptions that are noticeable to the end user. Upon completion of the transfer, the VM continues to operate on the new server, and the resources of the source server are freed for subsequent use. The migration process duration is minimized and does not significantly affect the performance of the VM, ensuring its continuous operation from the end user's perspective.

Table 1 provides formal definitions of all system parameters and state variables, ensuring consistency throughout the analysis.

Table 1. Main notation.

Parameter	Description
$\mathcal{S} = \{1, \ldots, S\}$	Set of servers (physical nodes) in the system
C_s	Bandwidth capacity of server s, bps
$\mathcal{V} = \{1, \ldots, V\}$	Set of VMs hosted on servers
b_v	Bandwidth requirement per task for VM v, bps
λ_v	Arrival rate of tasks for VM v, 1/s
μ_v^{-1}	Average processing time per task on VM v, s
n_v	Number of tasks currently being processed by VM v
s_v	Server hosting VM v
$\mathbf{n} = (n_1, \ldots, n_V)$	Vector of active tasks across all VMs
$\mathbf{s} = (s_1, \ldots, s_V)$	Vector of server assignments for all VMs
$\mathbf{x} = (\mathbf{n}, \mathbf{s})$	System state
$\mathcal{X}$	State space of the system
$\mathbf{Q}[\mathbf{x}, \mathbf{x}']$	Transition rate from state $\mathbf{x}$ to state $\mathbf{x}'$
$\boldsymbol{\pi}$	Stationary probability distribution
$c_s(\mathbf{x})$	Occupied bandwidth on server s in state $\mathbf{x}$, bps
$\mathcal{X}_v$	Set of states where new task arrivals for VM v are rejected on the current hosting server
$\mathcal{S}_v(\mathbf{x})$	Set of target servers available for migrating VM v in state $\mathbf{x}$
$s_v^*(\mathbf{x})$	Target server for migrating VM v in state $\mathbf{x}$ (the superscript * explicitly denotes that this server is the optimal or designated target for migration)
$\mathcal{M}_v$	Set of states triggering migration of VM v after a new task arrival
$i = 1, 2$	Migration policy identifier

3.2. Migration Policies

Effective VM migration requires balancing service continuity with resource efficiency. Our framework optimizes this balance through three decision phases, activated exclusively during task arrivals that threaten server overload.

Migration initiates when a new task arrival to VM v would exceed its host server's capacity. This event-driven approach minimizes unnecessary migrations, as static periods without task arrivals preserve system stability. The triggering condition ensures that migration occurs only when absolutely necessary, reducing both energy consumption and potential service disruptions.

The target VM for migration is uniquely determined by the overload trigger: the VM receiving the new task (v) becomes the migration candidate. This choice prevents cascading migrations that could disrupt other services, as relocating unrelated VMs would unnecessarily increase overhead. By focusing on the VM directly affected by the new task, we ensure that only the necessary resources are reallocated, maintaining the QoS for other VMs.

Two optimization strategies govern target server selection. Policy 1 (future utilization minimization) selects the server minimizing post-migration utilization. This policy aims to balance the load across servers by considering the future state of the system after migration. It is particularly useful in environments with predictable workloads, where future resource demands can be anticipated.

Policy 2 (current utilization minimization) selects the server with the lowest current utilization. This policy focuses on the immediate state of the system, making it suitable for highly dynamic environments where future resource demands are uncertain. It aims to quickly alleviate current overload conditions without considering long-term implications.

Tie-breaking selects the smallest-indexed server when multiple candidates exist. Figure 1 illustrates this process through a three-server scenario where a new task to VM-2 triggers migration. From the left to the horizontal gray arrow, we see the initial state with Server 2 overload risk. To the right after the horizontal gray arrow, we see the post-migration state: VM-2 is relocated to Server 3, freeing capacity for a new task. The two arrows above the server blocks indicate the migration direction. The blue arrow indicates the arrival of a new task on the server.

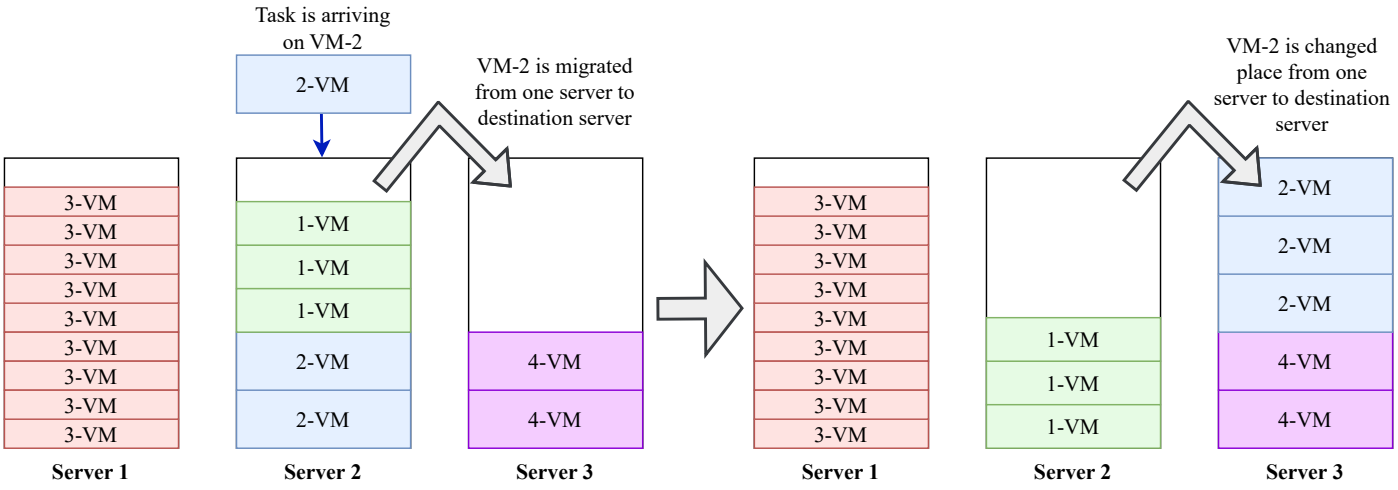


Figure 1. Illustration of the VM migration.

The system permits migration only if migration reduces future overload probability (Policy 1) or current imbalance (Policy 2). These constraints prevent thrashing behavior while ensuring QoS for all hosted services. Section 4 quantifies their impact through Markovian state transitions and performance metrics.

4. Queuing Model

In this section, we develop a queuing model that captures task processing and VM migration dynamics in edge-cloud systems. The model formalizes the interplay between task arrivals, resource allocation, and migration decisions using a continuous-time Markov chain (CTMC). We introduce the CTMC state space and transition structure, detailing how the system evolves through task arrivals, completions, and migrations. Two migration policies are presented: one minimizes future resource utilization, while the other focuses on

current utilization. The transition rate matrix is formalized to quantify system dynamics, and performance metrics are defined to evaluate QoS and migration efficiency.

4.1. Continuous-Time Markov Chain

The system is modeled as a multi-class queuing system where VMs act as mobile service centers capable of migrating between physical servers. This architecture comprises S server groups, each with a fixed bandwidth capacity C_s, and V distinct task classes represented by VMs. Each VM is permanently associated with a specific task class, processing requests that require b_v per task. Crucially, all tasks from a given VM must be processed on a single server group, enforcing a strict coupling between task classes and physical resources. VM migration enables dynamic reassignment of these task classes to different server groups, allowing the system to adapt to fluctuating workloads. Figure 2 illustrates this queuing architecture, where the two-way red arrows depict both task flows and migration paths between servers and the red dotted boxes represent the estimated new location of the migrated VM.

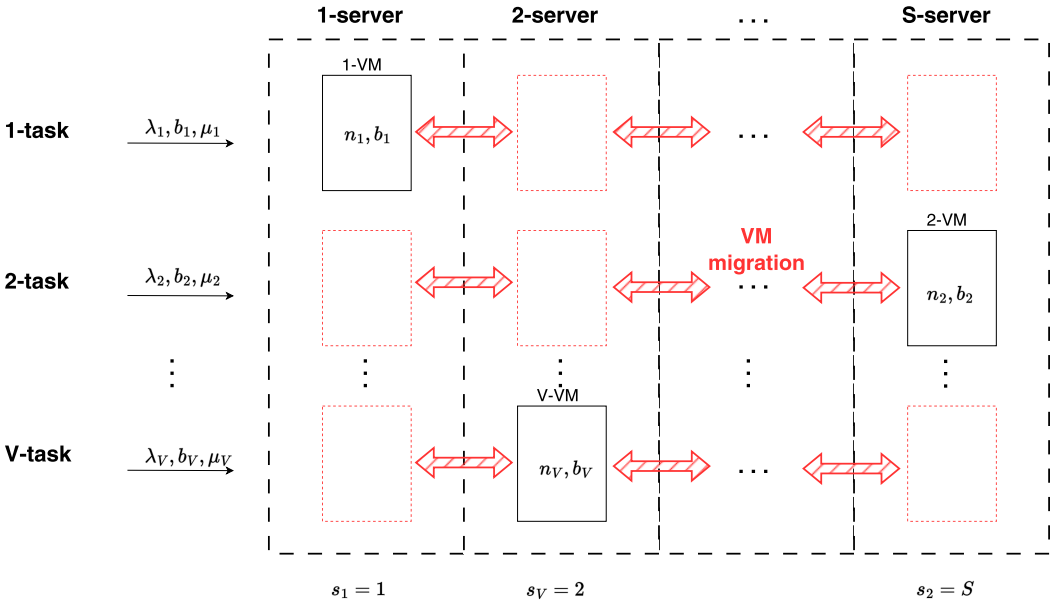


Figure 2. Scheme of the considered queuing system.

The system's stochastic evolution is captured through a CTMC $\mathbf{X}(t)$, where the state vector $\mathbf{x} = (\mathbf{n}, \mathbf{s})$ encodes both workload distribution and infrastructure configuration. Here, $\mathbf{n} = (n_1, \ldots, n_V)$ represents the number of active tasks being processed by each VM, while $\mathbf{s} = (s_1, \ldots, s_V)$ records the current server assignment for every VM. Transitions between states occur due to three fundamental events: task arrivals following Poisson processes with rates λ_v, task completions governed by exponential service rates μ_v, and VM migrations triggered by capacity constraints.

The state space $\mathcal{X}$ contains all feasible system configurations constrained by server capacities. Formally, a state $\mathbf{x} = (\mathbf{n}, \mathbf{s})$ belongs to $\mathcal{X}$ if the total bandwidth demand on each server s satisfies

$$c_s(\mathbf{x}) = \sum_{v \in \mathcal{V}} n_v b_v \cdot \mathbf{1}(s_v = s), \quad s \in \mathcal{S}, \tag{1}$$

where $\mathbf{1}(\bullet)$ denotes the indicator function. This constraint ensures no server exceeds its bandwidth capacity during normal operation.

$$\begin{aligned}\mathcal{X} = \{\mathbf{x} = (\mathbf{n}, \mathbf{s}) = (n_1, \ldots, n_V, s_1, \ldots, s_V) : \\ (n_v > 0,\ \ s_v \in \mathcal{S}) \ \ \vee \ \ (n_v = 0,\ \ s_v = 0),\ \ v \in \mathcal{V}, \\ c_s(\mathbf{x}) \leq C_s,\ \ s \in \mathcal{S}\}.\end{aligned} \tag{2}$$

A critical subset of states $\mathcal{X}_v \in \mathcal{X}$ identifies configurations where accepting a new task for VM v would violate its host server's capacity:

$$\mathcal{X}_v = \{\mathbf{x} \in \mathcal{X} :\ \ c_{s_v}(\mathbf{x}) + b_v > C_{s_v}\}, \quad v \in \mathcal{V}. \tag{3}$$

These states trigger the system's migration logic—either rejecting incoming tasks or relocating VMs to maintain service continuity, as detailed in subsequent sections.

4.2. Migration Policy: Future Utilization Minimization

The first migration ($i = 1$) policy proactively optimizes resource allocation by anticipating the impact of incoming tasks on server utilization. This approach activates when a new task arrival to VM v would exceed its current host server's capacity (s_v), triggering a migration event. The core objective is to select a target server that minimizes the projected bandwidth utilization after accommodating both the VM's existing tasks and the incoming request.

The decision process unfolds in three stages. (1) Compute the hypothetical utilization for each server $s \neq s_v$ if it were to host VM v with the additional task. Filter servers where this projection respects capacity limits. (2) From feasible candidates, choose servers with minimal projected utilization:

$$\begin{aligned}&\mathcal{S}_v^1(\mathbf{x}) \\ &\quad = \left\{ s \in \mathcal{S} :\ \ s = \arg \min_{s' \in \mathcal{S} \setminus \{s_v\}} [c_{s'}(\mathbf{x}) + (n_v + 1) b_v] \cdot \mathbf{1}(c_{s'}(\mathbf{x}) + (n_v + 1) b_v \leq C_s) \right\}, \\ &\mathbf{x} \in \mathcal{X}_v,\ \ v \in \mathcal{V}.\end{aligned} \tag{4}$$

(3) When multiple servers yield identical minimal utilization, select the smallest-indexed server for deterministic behavior:

$$s_v^{1*}(\mathbf{x}) = \min_{s \in \mathcal{S}_v^1(\mathbf{x})} s, \ \ \mathbf{x} \in \mathcal{X}_v, \ \ v \in \mathcal{V}. \tag{5}$$

This policy prioritizes long-term stability by preventing fragmented resource allocation. For example, consider a system with three servers ($C_1 = 5$, $C_2 = 6$, $C_3 = 7$ Gbps) and a VM v ($b_v = 1$ Gbps, $n_v = 4$ tasks). The projected utilization for each server would be

Server 1: Current load 3 Gbps $\rightarrow 3 + (4 + 1) \times 1 = 8$ Gbps (exceeds C_1)
Server 2: Current load 2 Gbps $\rightarrow 2 + 5 = 7$ Gbps (feasible)
Server 3: Current load 4 Gbps $\rightarrow 4 + 5 = 9$ Gbps (exceeds C_3)

Thus, Server 2 becomes the migration target despite having higher current utilization than Server 1, demonstrating the policy's forward-looking nature.

4.3. Migration Policy: Current Utilization Minimization

The second migration policy ($i = 2$) adopts a reactive strategy focused on immediate resource availability rather than future projections. Designed for systems where migration overheads are significant, this approach minimizes disruption by selecting servers with the lowest current utilization, then verifies their capacity to absorb the migrating VM.

The workflow comprises four phases. (1) Identify servers with minimal existing bandwidth consumption:

$$\overline{\mathcal{S}}_v^2(\mathbf{x}) = \left\{ s \in \mathcal{S} : \ s = \arg \min_{s' \in \mathcal{S} \setminus \{s_v\}} c_{s'}(\mathbf{x}) \right\}, \quad \mathbf{x} \in \mathcal{X}_v, \ v \in \mathcal{V}. \tag{6}$$

(2) Choose the smallest-indexed server from the lowest-utilization group:

$$s_v^{2*}(\mathbf{x}) = \min_{s \in \overline{\mathcal{S}}_v^1(\mathbf{x})} s, \ \mathbf{x} \in \mathcal{X}_v, \ v \in \mathcal{V}. \tag{7}$$

(3) Validate whether the selected server can accommodate the VM's total demand:

$$\mathcal{S}_v^2(\mathbf{x}) = \begin{cases} \overline{\mathcal{S}}_v^2(\mathbf{x}), & c_{s_v^{2*}(\mathbf{x})}(\mathbf{x}) + (n_v + 1)b_v \le C_{s_v^{2*}(\mathbf{x})}, \\ \varnothing, & c_{s_v^{2*}(\mathbf{x})}(\mathbf{x}) + (n_v + 1)b_v > C_{s_v^{2*}(\mathbf{x})}, \end{cases} \\ \mathbf{x} \in \mathcal{X}_v, \ v \in \mathcal{V}. \tag{8}$$

(4) If validation fails, migration aborts and the incoming task is rejected.

Consider the same three-server scenario with VM v ($n_v = 4$, $b_v = 1$ Gbps):

Server 1: Current load 3 Gbps (lowest)
Server 2: Current load 4 Gbps
Server 3: Current load 5 Gbps

The policy first selects Server 1. However, adding the VM's projected load ($3 + 5 = 8$ Gbps) exceeds $C_1 = 5$ Gbps, causing migration failure. This illustrates the policy's limitation in capacity-constrained environments, where myopic optimization may overlook future constraints.

Policy 1 excels in stable environments with predictable workloads, preventing capacity violations through anticipatory decisions. However, its computational complexity grows with server/VM counts due to projection calculations. Policy 2 offers lower runtime overhead by leveraging current metrics, making it suitable for highly dynamic systems. However, it risks frequent migration failures when target servers lack reserve capacity. Both policies enable QoS preservation but require careful tuning based on workload characteristics and system constraints. Thus, predictive resource allocation is that since a fixed migration policy is built, it is possible to calculate the future behavior of the system based on the current state of the system. And one of the policies under consideration assumes that after its application and the VM is moved, further migration of the VM will be minimized.

4.4. Transition Rates

The CTMC's evolution is governed by three types of state transitions, each corresponding to fundamental system events: task arrivals (with and without migration) and task completions. Let $\mathbf{e}_v$ denote the canonical basis vector with 1 at the v-th position, representing an incremental change in the task count for VM v.

$$\mathbf{Q}[(\mathbf{n}, \mathbf{s}), (\mathbf{n} + \mathbf{e}_v, \mathbf{s})] = \lambda_v, \quad c_{s_v}(\mathbf{n}, \mathbf{s}) + b_v \le C_{s_v}, \ v \in \mathcal{V}, \tag{9}$$

$$\mathbf{Q}[(\mathbf{n}, \mathbf{s}), (\mathbf{n} + \mathbf{e}_v, \mathbf{s}')] = \lambda_v, \quad c_{s_v}(\mathbf{n}, \mathbf{s}) + b_v > C_{s_v}, \\ \mathcal{S}_v(\mathbf{n}, \mathbf{s}) \neq \varnothing, \ s_v' = s_v^*(\mathbf{n}, \mathbf{s}), \ v \in \mathcal{V}, \tag{10}$$

$$\mathbf{Q}[(\mathbf{n}, \mathbf{s}), (\mathbf{n} - \mathbf{e}_v, \mathbf{s})] = n_v \mu_v, \quad n_v > 0 \ \ v \in \mathcal{V}. \tag{11}$$

When a new task arrives at VM v and its current host server s_v has sufficient residual capacity, the task is accepted without migration. This transition preserves the server assignment vector $\mathbf{s}$ while incrementing n_v.

When a new task would overload VM v's current server ($c_{s_v} + b_v > C_{s_v}$), migration to a target server s^* occurs if feasible. Here, $\mathbf{s}'$ differs from $\mathbf{s}$ only in the v-th component, which updates to the migration target s_v^* from Policy i.

Active tasks complete service at rates proportional to their count. This linear dependence on n_v reflects the memoryless property of exponential service times.

The infinitesimal generator matrix $\mathbf{Q}$ exhibits each state, which connects to at most $2V$ neighbors (arrival/completion per VM). Migration probabilities depend on current load distribution.

4.5. Performance Metrics

The stationary probability distribution $\pi(\mathbf{n}, \mathbf{s})$, obtained by solving the global balance equations, enables a quantitative evaluation of system performance. First of all, we note that QoS indicators are considered at the user session level, not at the packet level, so, in particular, one of the investigated indicators is the probability of blocking tasks entering the system, but let us consider two types of metrics: metrics that are not affected by the migration process and metrics that migration affects.

The first type includes metrics not related to the migration procedure. Blocking probability of a task for VM v:

$$B_v = \sum_{(\mathbf{n},\mathbf{s})\in\mathcal{B}_v} \pi(\mathbf{n}, \mathbf{s}), \quad \mathcal{B}_v = \{(\mathbf{n}, \mathbf{s}) \in \mathcal{X}_v : \ \mathcal{S}_v(\mathbf{n}, \mathbf{s}) = \oslash\}, \quad v \in \mathcal{V}. \tag{12}$$

Average bandwidth utilization of server s:

$$\bar{c}_s = \sum_{(\mathbf{n},\mathbf{s})\in\mathcal{X}} c_s(\mathbf{n}, \mathbf{s}) \cdot \pi(\mathbf{n}, \mathbf{s}), \quad s \in \mathcal{S}. \tag{13}$$

Average bandwidth utilization of VM v:

$$\bar{b}_v = \sum_{(\mathbf{n},\mathbf{s})\in\mathcal{X}} n_v b_v \cdot \pi(\mathbf{n}, \mathbf{s}), \quad v \in \mathcal{V}. \tag{14}$$

Average number of active servers:

$$\bar{s} = \sum_{(\mathbf{n},\mathbf{s})\in\mathcal{X}} \sum_{s\in\mathcal{S}} \mathbf{1}\{c_s(\mathbf{n}, \mathbf{s}) > 0\} \cdot \pi(\mathbf{n}, \mathbf{s}). \tag{15}$$

To define the migration-related metrics, we introduce the following auxiliary notation. The set of states triggering migration of VM v after a new task arrival:

$$\begin{aligned} \mathcal{M}_v &= \{\mathbf{x} \in \mathcal{X} : \ c_{s_v}(\mathbf{x}) + b_v > C_{s_v}, \ \mathcal{S}_v(\mathbf{n}, \mathbf{s}) \neq \oslash\} \\ &= \{\mathbf{x} \in \mathcal{X}_s : \ \mathcal{S}_v(\mathbf{n}, \mathbf{s}) \neq \oslash\}, \quad v \in \mathcal{V}. \end{aligned} \tag{16}$$

Probability of migrating VM v in state $\mathbf{x} = (\mathbf{n}, \mathbf{s})$:

$$P_v^{\mathrm{mg}}(\mathbf{n}, \mathbf{s}) = \frac{\lambda_v}{\sum\limits_{v'\in\mathcal{V}} (\lambda_{v'} + n_{v'}\mu_{v'})} \cdot \mathbf{1}\{(\mathbf{n}, \mathbf{s}) \in \mathcal{M}_v\}, \quad v \in \mathcal{V}, \ \ (\mathbf{n}, \mathbf{s}) \in \mathcal{X}. \tag{17}$$

Probability of migrating any VM in state $\mathbf{x} = (\mathbf{n}, \mathbf{s})$:

$$P^{\mathrm{mg}}(\mathbf{n}, \mathbf{s}) = \sum_{v\in\mathcal{V}} P_v^{\mathrm{mg}}(\mathbf{n}, \mathbf{s}), \quad (\mathbf{n}, \mathbf{s}) \in \mathcal{X}. \tag{18}$$

Then, migration metrics are as follows. Probability of migrating VM v:

$$P_v^{\text{mg}} = \sum_{(\mathbf{n},\mathbf{s})\in\mathcal{X}} P_v^{\text{mg}}(\mathbf{n},\mathbf{s}) \cdot \pi(\mathbf{n},\mathbf{s}), \quad v \in \mathcal{V}. \tag{19}$$

Probability of migrating any VM:

$$P^{\text{mg}} = \sum_{(\mathbf{n},\mathbf{s})\in\mathcal{X}} P^{\text{mg}}(\mathbf{n},\mathbf{s}) \cdot \pi(\mathbf{n},\mathbf{s}). \tag{20}$$

Probability of migrating VM v to server s:

$$P_{vs}^{\text{mg2sr}} = \sum_{(\mathbf{n},\mathbf{s})\in\mathcal{X}} \mathbf{1}\{s = s_v^*(\mathbf{n},\mathbf{s})\} \cdot P_v^{\text{mg}}(\mathbf{n},\mathbf{s}) \cdot \pi(\mathbf{n},\mathbf{s}), \quad v \in \mathcal{V}, \ \ s \in \mathcal{S}. \tag{21}$$

Probability of migrating any VM to server s:

$$P_s^{\text{mg2sr}} = \sum_{v\in\mathcal{V}} P_{vs}^{\text{mg2sr}}, \quad s \in \mathcal{S}. \tag{22}$$

Average bandwidth utilization of VM v when migrated:

$$\bar{b}_v^{\text{mg}} = \sum_{(\mathbf{n},\mathbf{s})\in\mathcal{X}} n_v b_v \cdot P_v^{\text{mg}}(\mathbf{n},\mathbf{s}) \cdot \pi(\mathbf{n},\mathbf{s}), \quad v \in \mathcal{V}. \tag{23}$$

Average bandwidth utilization of any VM when migrated:

$$\bar{b}^{\text{mg}} = \frac{1}{V} \sum_{v\in\mathcal{V}} \bar{b}_v^{\text{mg}}. \tag{24}$$

5. Numerical Results

In this section, we evaluate the performance of our VM migration framework. We analyze two critical scenarios to assess system behavior under varying conditions: (1) the impact of task arrival rates on resource utilization and blocking probabilities, and (2) the effects of task processing times on server workloads and migration efficiency. We present key performance metrics, including blocking probabilities, and server and VM bandwidth utilization.

5.1. Considered Scenario

The transition from 5G to 6G networks represents a paradigm shift in telecommunications, with 6G offering data rates 50 times faster and latency 10 times lower than its predecessor. These advancements enable the deployment of highly demanding applications such as extended reality (XR), holography, and digital twins, which require ultra-reliable low-latency communication (URLLC) and massive machine-type communication (mMTC). Such services demand not only higher bandwidth but also more efficient resource management to ensure seamless user experiences. In this context, VM migration becomes a critical mechanism for maintaining service continuity and optimizing resource utilization in cloud and edge computing infrastructures. This section evaluates the performance of our proposed migration framework, focusing on key metrics such as task blocking probability, server utilization, and migration efficiency.

We consider a system comprising three heterogeneous servers ($S = 3$) hosting two latency-sensitive services: XR and holography ($V = 2$) (see Table 2). The XR service, characterized by bursty workloads, has an arrival rate of $\lambda_1 = 2$ tasks/s, a processing time of $\mu_1^{-1} = 0.8$ s, and a bandwidth requirement of $b_1 = 700$ Mbps. In contrast, the holography service requires sustained bandwidth, with $\lambda_2 = 1$ task/s, $\mu_2^{-1} = 1$ s, and $b_2 = 1200$ Mbps. The servers have capacities of $C_1 = 4$ Gbps, $C_2 = 7$ Gbps, and $C_3 = 10$ Gbps, reflecting typ-

ical edge-cloud infrastructure configurations. These parameters are derived from industry benchmarks and research studies [32,33].

Table 2. Default system parameters.

Service	Parameter	Value	Units
Extended reality (XR)	λ_1	2	1/s
	μ_1	0.8	s
	b_1	700	Mbps
Holography	λ_2	1	1/s
	μ_2	1	s
	b_2	1200	Mbps
Servers	C_1	4	Gbps
	C_2	7	Gbps
	C_3	10	Gbps

Two scenarios are analyzed to assess the system's performance under varying conditions. The first scenario examines the impact of increasing task arrival rates (λ_1 ranging from 1 to 3 tasks/s), simulating periods of high demand such as live events or peak usage hours. The second scenario evaluates the effects of accelerating task processing times (μ_1 ranging from 0.5 to 1.5 tasks/s). We compare Policy 1 (future utilization minimization) and Policy 2 (current utilization minimization) across these scenarios.

The main analyzed metrics are the blocking probability of a task, the average bandwidth utilization of servers, and the average bandwidth utilization of VMs. These metrics are the most sensitive to migration. Blocking probability is a key QoS metric, as a high blocking probability can indicate a violation of a QoS agreement. It is also an indicator of migration effectiveness because, if the blocking probability increases or does not change with the selected migration policy, then a new policy should be considered. Analyzing the average utilization of servers allows us to evaluate energy consumption because low utilization results in inefficient energy consumption, which in turn will have a negative impact on the finances of the company providing the server. Average VM utilization, as well as blocking probability, have a direct impact on QoS performance and reflect whether the VMs have sufficient bandwidth to provide the services in question.

5.2. Arrival Rate Impact

The impact of task arrival rates on system performance is first analyzed through blocking probability, a critical metric in cloud infrastructure. As shown in Figures 3 and 4, the blocking probability increases gradually with higher task arrival rates, reflecting the system's growing load. For VM 1 (Figure 3), both policies exhibit similar trends, with blocking probabilities rising steadily as λ_1 increases. However, VM 2 (Figure 4) shows significant differences between policies. Policy 1 maintains lower blocking probabilities due to its proactive migration strategy, while Policy 2 experiences sharper increases, particularly around $\lambda_1 \approx 1.37$, where migration opportunities diminish. This analysis highlights Policy 1's superior ability to manage high arrival rates while minimizing service disruptions.

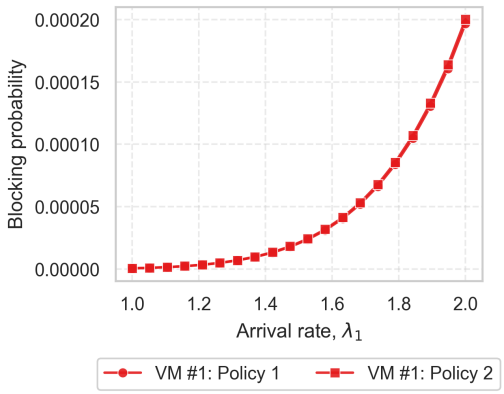


Figure 3. Blocking probability of a task for VM 1 vs. arrival rate λ_1.

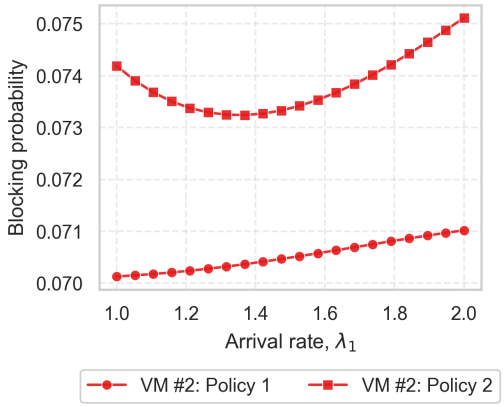


Figure 4. Blocking probability of a task for VM 2 vs. arrival rate λ_1.

Next, we examine server bandwidth utilization under varying arrival rates. Figures 5 and 6 illustrate the average utilization for Policy 1 and Policy 2, respectively. Both policies achieve high utilization across all servers, reflecting the resource-intensive nature of XR and holography services. However, Policy 1 demonstrates better load balancing, particularly for Server 2 and Server 3, where utilization remains stable even as λ_1 approaches 2 tasks/s. In contrast, Policy 2 shows more pronounced fluctuations, indicating less efficient resource allocation. The average number of active servers ranges from 1.706 to 1.91, which is 11.96%, suggesting that both policies effectively utilize available resources without overloading the system.

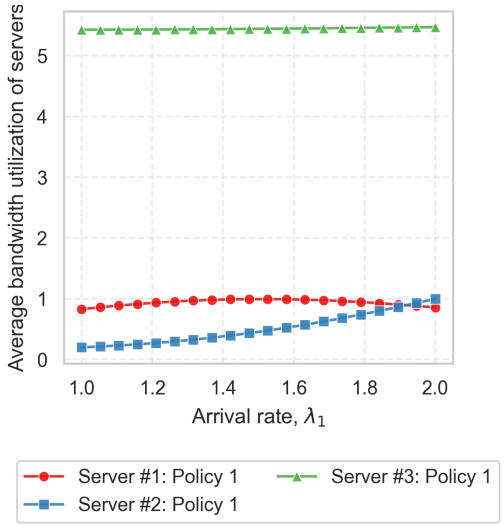


Figure 5. Average bandwidth utilization of servers for Policy 1 vs. arrival rate λ_1.

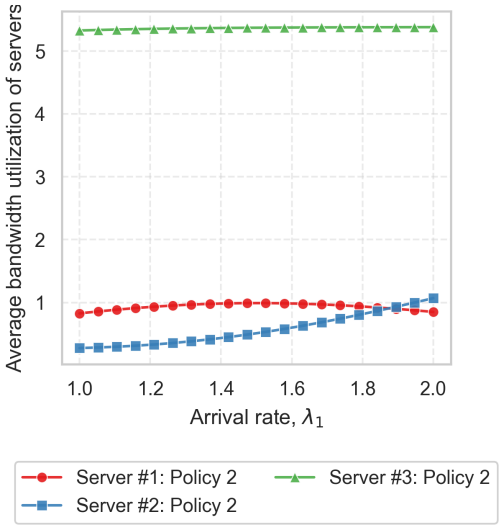


Figure 6. Average bandwidth utilization of servers for Policy 2 vs. arrival rate λ_1.

Figures 7 and 8 present the average bandwidth utilization for VM 1 and VM 2 under both policies. The results indicate similar utilization patterns for both VMs, with Policy 1 maintaining a slightly more consistent performance. This consistency is particularly evident for VM 2, which handles the more demanding holography service. Policy 1's ability to balance loads across VMs contributes to its overall effectiveness in managing high arrival rates.

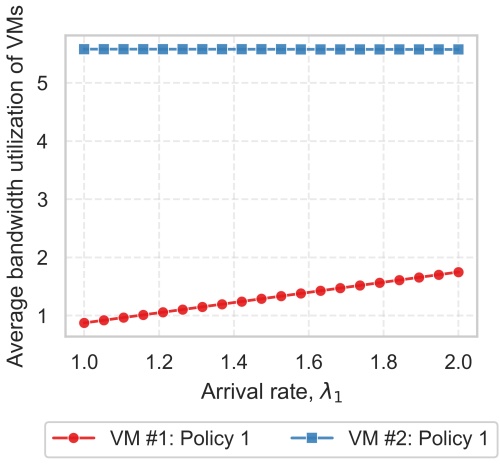


Figure 7. Average bandwidth utilization of VMs for Policy 1 vs. arrival rate λ_1.

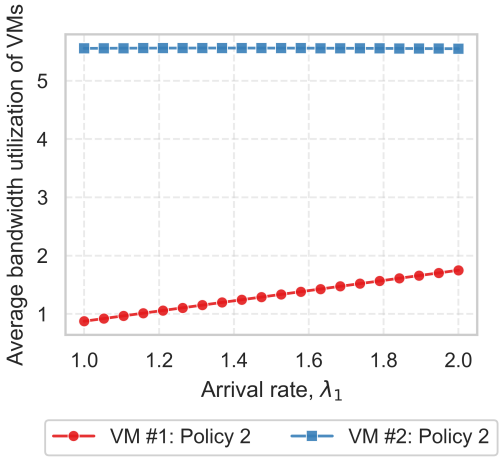


Figure 8. Average bandwidth utilization of VMs for Policy 2 vs. arrival rate λ_1.

Finally, Figures 9 and 10 provide a difference between the two policies in server and VM utilization. The results confirm that Policy 1 achieves more balanced resource distribution, with smaller deviations in server and VM utilization. Specifically, Server 2 and Server 3 show significant differences under Policy 2, while Policy 1 maintains consistent performance across all servers. These findings underscore Policy 1's advantages in minimizing blocking probabilities and optimizing resource utilization under varying arrival rates.

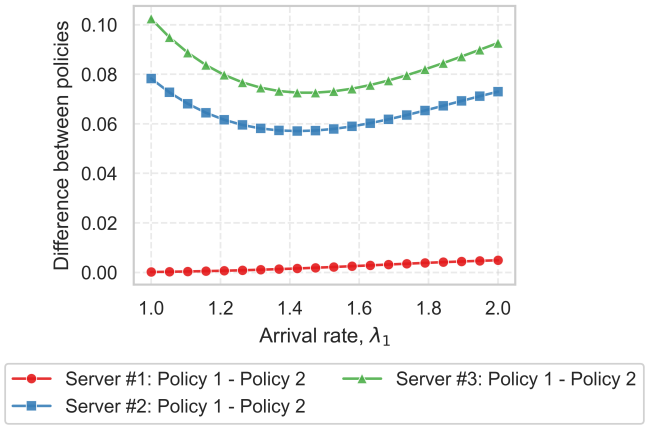


Figure 9. Difference between policies for average bandwidth utilization of servers vs. arrival rate λ_1.

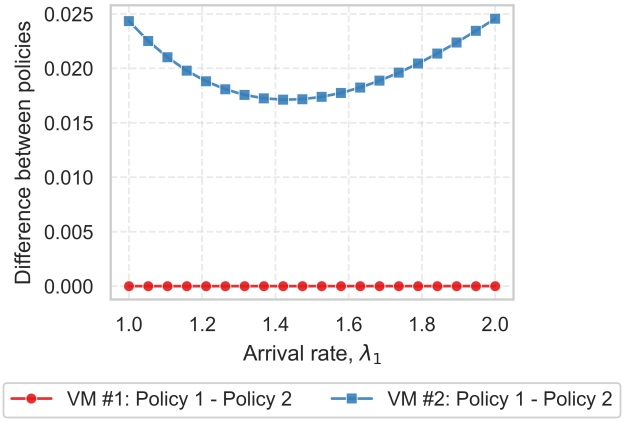


Figure 10. Difference between policies for average bandwidth utilization of VMs vs. arrival rate λ_1.

5.3. Task Processing Time Impact

The impact of task processing times on system performance is analyzed by varying the service rate μ_1 while keeping the arrival rate constant. As shown in Figures 11 and 12, accelerating task processing reduces blocking probabilities, but the extent of improvement varies between policies. For VM 1 (Figure 11), Policy 1 demonstrates a clear reduction in blocking probability as μ_1 increases, reflecting its ability to handle faster task processing efficiently. In contrast, Policy 2 (Figure 12) shows a less pronounced improvement, with blocking probabilities increasing slightly at higher μ_1 values. This behavior highlights Policy 1's advantage in managing systems with accelerated task processing.

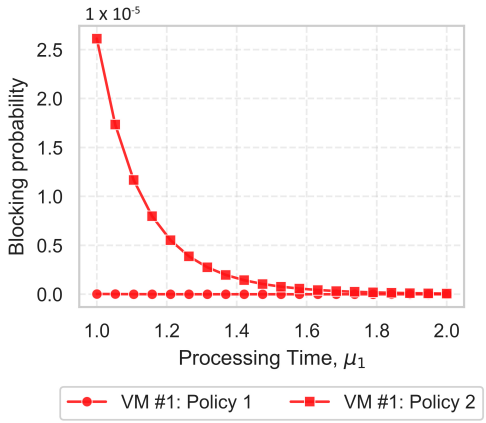


Figure 11. Blocking probability of a task for VM 1 vs. service rate μ_1.

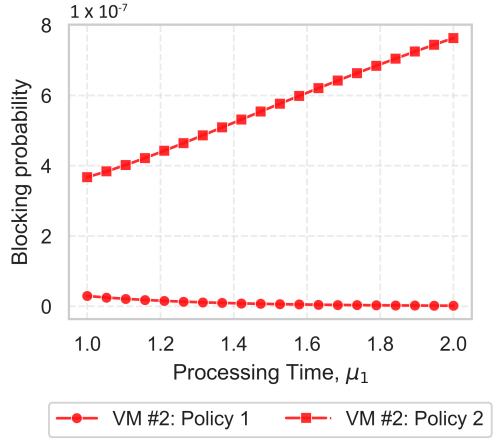


Figure 12. Blocking probability of a task for VM 2 vs. service rate μ_1.

Figures 13 and 14 illustrate the average bandwidth utilization of servers under varying processing times. As tasks are processed more quickly, the workload on servers decreases, particularly for Server 3, which remains underutilized in both policies. This underutilization is more pronounced in Policy 2, where Server 3's utilization drops significantly as μ_1 increases. This raises concerns about energy efficiency, as Server 3, being the most powerful, consumes considerable power even when idle. The average number of active servers ranges from 1.04 to 0.81, indicating that both policies consolidate workloads effectively but with differing efficiency.

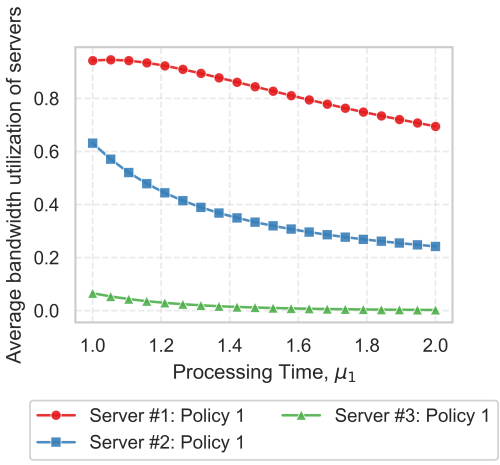


Figure 13. Average bandwidth utilization of servers for Policy 1 vs. service rate μ_1.

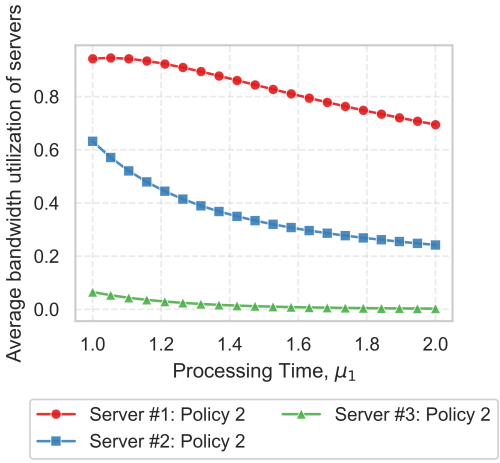


Figure 14. Average bandwidth utilization of servers for Policy 2 vs. service rate μ_1.

The impact of faster processing times on VM utilization is shown in Figures 15 and 16. Both policies exhibit similar trends, with VM 2 initially demanding more resources due to its higher bandwidth requirements. However, as μ_1 increases, the utilization of both VMs decreases, reflecting the reduced task backlog. Policy 1 maintains slightly more consistent utilization levels, particularly for VM 2, which handles the more resource-intensive holography service. This consistency underscores Policy 1's ability to adapt to varying processing speeds without significant performance degradation.

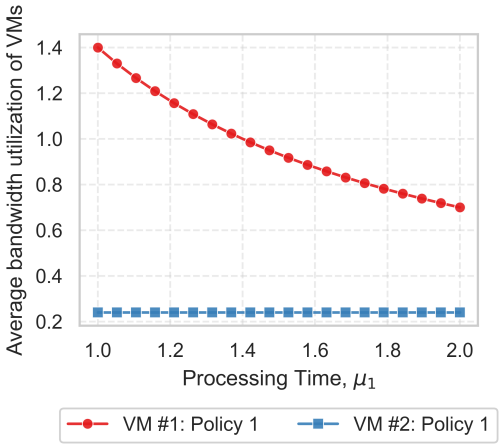


Figure 15. Average bandwidth utilization of VMs for Policy 1 vs. service rate μ_1.

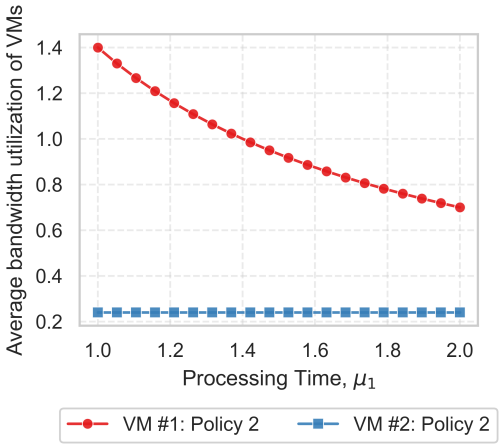


Figure 16. Average bandwidth utilization of VMs for Policy 2 vs. service rate μ_1.

Finally, Figures 17 and 18 present a difference between the two policies in server and VM utilization. The results show minimal differences, suggesting that both policies perform similarly under varying processing times. However, Policy 1's superior handling of blocking probabilities and more balanced server utilization make it the preferred choice for systems with dynamic task processing requirements. These findings highlight the importance of selecting migration policies that align with specific workload characteristics to optimize performance and resource efficiency.

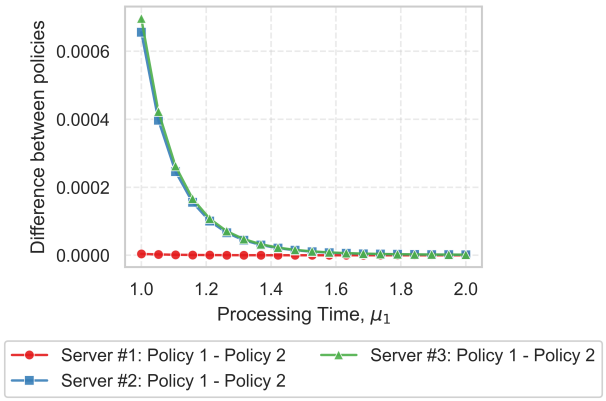


Figure 17. Difference between policies for average bandwidth utilization of servers vs. service rate μ_1.

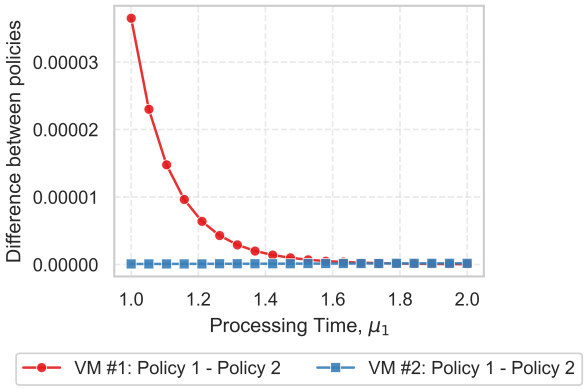


Figure 18. Difference between policies for average bandwidth utilization of VMs vs. service rate μ_1.

5.4. Discussion and Limitations

Numerical results demonstrate that the effectiveness of migration policies depends on system state variables, particularly the ratio between task arrival rates and processing times. Policy 1, which minimizes projected post-migration server utilization, consistently outperforms Policy 2 in reducing blocking probabilities. For VM 1, the average value for Policy 1 is 2.1% lower than Policy 2, while for VM 2, the same measure is 4.7% lower. One of the main purposes of numerical analysis was to analyze and compare two migration policies with each other, so the percentage results were also compared between policies. The comparison of the two policies is shown in Table 3. These findings validate the model's ability to describe and optimize cloud infrastructure behavior under varying workloads. Furthermore, the server utilization metrics Figures 5, 6, 13 and 14 exhibit no excessive congestion, implying adherence to network constraints. As we can see from the Table 3, if we need to have a lower probability of blocking, we should choose Policy 1; however, there is a heavy bandwidth utilization on both the VMs and Servers 1 and 2 with this policy.

Table 3. Policy 1 vs. Policy 2.

Metrics	Policy 1 Compared to Policy 2
Metrics vs. Arrival rate	
Blocking probability for VM 1	<2.1%
Blocking probability for VM 2	<4.7%
Average bandwidth utilization of Server 1	>0.2%
Average bandwidth utilization of Server 2	<12.6%
Average bandwidth utilization of Server 3	>1.5%
Average bandwidth utilization of VM 1	>0.00012%
Average bandwidth utilization of VM 2	>0.36%
Metrics vs. Processing time	
Blocking probability for VM 1	<2.1%
Blocking probability for VM 2	<4.7%
Average bandwidth utilization of Server 1	>0.2%
Average bandwidth utilization of Server 2	<12.6%
Average bandwidth utilization of Server 2	>1.5%
Average bandwidth utilization of VM 1	>0.00012%
Average bandwidth utilization of VM 2	>0.36%

The results of this paper can be divided into two components: construction of an algorithm for finding the VM migration policy and calculation of key performance metrics of the model. The implementation of the algorithm is based primarily on the current state of the system, so its realization occurs in real time, while the calculation of metrics is possible only when it is necessary and is not performed in real time. The main goal of the algorithm is to find the minimum value in a discrete set. It is known that the complexity of such an operation is estimated in $O(n)$, where n is the number of values in the set, in the case of a linear search, and in $O(1)$ in the case where the set of values is sorted.

Among the main limitations of the model are the exponential distribution and Poisson process, the use of which could limit the applicability of the model to systems in which task processing time and time between task arrivals have a more complex distribution. In addition, the dynamic variability of VM and server parameters in real systems is not taken into account, since, for example, the values of server capacity C_s and VM bandwidth b_v are fixed. The time delays in VM migration due to, e.g., data transfers or state synchronization are not directly taken into account either. It is also assumed that the system is stationary, i.e., sudden spikes or load surges are not considered. Thus, these limitations create a wide field for further analysis of the system.

6. Conclusions

The rapid development of cloud computing technologies has positioned VM migration as a critical mechanism for optimizing resource management, enhancing fault tolerance, and ensuring service continuity. This paper addresses the complexities of VM migration by focusing on three key decision factors: (1) determining the optimal timing for migration, (2) selecting the appropriate VM and source server, and (3) identifying the target server for migration. Our proposed model leverages queuing theory and CTMC to evaluate these decisions under dynamic workload conditions, based on two policies for selecting the target server.

In the future, we plan to expand the model by introducing cost parameters for migration (for example, bandwidth consumption for transferring the VM state). Since this study was conducted as a quantitative analysis of the mathematical model, the simulation model also needs to be further investigated to obtain more detailed results. In addition,

future research will extend this framework to MEC environments, which are integral to 6G network architectures. MEC's ability to process data at the network edge addresses key challenges in cloud computing, such as latency reduction and enhanced reliability [34]. By integrating MEC capabilities, we aim to develop migration strategies that further improve performance and scalability in next-generation networks. This work lays the foundation for intelligent, adaptive resource management systems capable of meeting the evolving demands of cloud and edge computing infrastructures.

Author Contributions: Conceptualization, I.K.; methodology, I.K.; software, A.K. (Anna Kushchazli) and K.L.; validation, A.K. (Anna Kushchazli) and K.L.; formal analysis, A.K. (Anna Kushchazli) and I.K.; investigation, A.K. (Anna Kushchazli) and I.K.; resources, A.K. (Abdukodir Khakimov); writing—original draft preparation, A.K. (Anna Kushchazli) and K.L.; writing—review and editing, I.K.; visualization, A.K. (Anna Kushchazli) and K.L.; supervision, I.K.; project administration, A.K. (Abdukodir Khakimov); funding acquisition, A.K. (Abdukodir Khakimov). All authors have read and agreed to the published version of the manuscript.

Funding: This publication was supported by the RUDN University Scientific Projects Grant System, project No. 025322-2-000.

Data Availability Statement: Data are contained within the article.

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

Abbreviations

The following abbreviations are used in this manuscript:

5G	Fifth generation
6G	Sixth generation
AI	Artificial intelligence
CPU	Central processing unit
CTMC	Continuous-time Markov chain
IaaS	Infrastructure as a service
PaaS	Platform as a service
QoS	Quality of service
RAM	Random access memory
RR	Round-Robin
SaaS	Software as a service
SLA	Service-level agreement
VM	Virtual machine
WRR	Weighted Round-Robin

References

1. Younis, R.; Iqbal, M.; Munir, K.; Javed, M.A.; Haris, M.; Alahmari, S. A Comprehensive Analysis of Cloud Service Models: IaaS, PaaS, and SaaS in the Context of Emerging Technologies and Trend. In Proceedings of the 2024 International Conference on Electrical, Communication and Computer Engineering (ICECCE), Kuala Lumpur, Malaysia, 30–31 October 2024; pp. 1–6. [CrossRef]
2. Le, T. A survey of live Virtual Machine migration techniques. *Comput. Sci. Rev.* **2020**, *38*, 100304. [CrossRef]
3. Ateya, A.A.; Alhussan, A.A.; Abdallah, H.A.; Khakimov, A.; Muthanna, A. Edge Computing Platform with Efficient Migration Scheme for 5G/6G Networks. *Comput. Syst. Sci. Eng.* **2023**, *45*, 1775–1787. [CrossRef]
4. 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]
5. Hieu, N.T.; Francesco, M.D.; Ylä-Jääski, A. Virtual Machine Consolidation with Multiple Usage Prediction for Energy-Efficient Cloud Data Centers. *IEEE Trans. Serv. Comput.* **2020**, *13*, 186–199. [CrossRef]

6. Zhou, J.; Kumar Lilhore, D.; Manoharan, P.; Tao, H.; Simaiya, S.; Jawawi, D.; Alsekait, D.; Ahuja, S.; Biamba, C.; Hamdi, M. Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing. *J. Cloud Comput.* **2023**, *12*, 85. [CrossRef]
7. Wang, W.; Casale, G. Evaluating Weighted Round Robin Load Balancing for Cloud Web Services. In Proceedings of the 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 22–25 September 2014; pp. 393–400. [CrossRef]
8. Kumar Lilhore, D.; Simaiya, S.; Guleria, K.; Prasad, D. An Efficient Load Balancing Method by Using Machine Learning-Based VM Distribution and Dynamic Resource Mapping. *J. Comput. Theor. Nanosci.* **2020**, *17*, 8928. [CrossRef]
9. Joshi, V. Load Balancing Algorithms in Cloud Computing. *Int. J. Res. Eng. Innov.* **2019**, *3*, 530–532.
10. Sarma, S. Metaheuristic based auto-scaling for microservices in cloud environment: A new container-aware application scheduling. *Int. J. Pervasive Comput. Commun.* 2021, *in press*. [CrossRef]
11. Arunagiri, R.; Kandasamy, V. Workflow scheduling in cloud environment using a novel metaheuristic optimization algorithm. *Int. J. Commun. Syst.* **2021**, *34*, e4746. [CrossRef]
12. Altahat, M.A.; Daradkeh, T.; Agarwal, A. Virtual machine scheduling and migration management across multi-cloud data centers: Blockchain-based versus centralized frameworks. *J. Cloud Comput.* **2025**, *14*, 1. [CrossRef]
13. Al-Ansi, A.; Al-Ansi, A.M.; Muthanna, A.; Koucheryavy, A. Blockchain technology integration in service migration to 6g communication networks: A comprehensive review. *Indones. J. Electr. Eng. Comput. Sci.* **2024**, *34*, 1654–1664. [CrossRef]
14. Elsaid, M.E.; Abbas, H.M.; Meinel, C. Virtual machines pre-copy live migration cost modeling and prediction: A survey. *Distrib. Parallel Databases* **2022**, *40*, 441–474. [CrossRef]
15. Li, C.; Feng, D.; Hua, Y.; Qin, L. Efficient live virtual machine migration for memory write-intensive workloads. *Future Gener. Comput. Syst.* **2019**, *95*, 126–139. [CrossRef]
16. Raghunath, B.R.; Annappa, B. Virtual Machine Migration Triggering using Application Workload Prediction. *Procedia Comput. Sci.* **2015**, *54*, 167–176.
17. Xiao, Z.; Song, W.; Chen, Q. Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *24*, 1107–1117. [CrossRef]
18. Singh, J.; Walia, N.K. Virtual Machine Selection and Migration: Challenges and Future Directions. In Proceedings of the 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 17–19 May 2023; pp. 1126–1131. [CrossRef]
19. Kamran, M.; Nazir, B. QoS-aware VM placement and migration for hybrid cloud infrastructure. *J. Supercomput.* **2018**, *74*, 4623–4646. [CrossRef]
20. Khan, M.A.; Paplinski, A.P.; Khan, A.M.; Murshed, M.; Buyya, R. Exploiting user provided information in dynamic consolidation of virtual machines to minimize energy consumption of cloud data centers. In Proceedings of the 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), Barcelona, Spain, 23–26 April 2018; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
21. Li, P.; Cao, J. A Virtual Machine Consolidation Algorithm Based on Dynamic Load Mean and Multi-Objective Optimization in Cloud Computing. *Sensors* **2022**, *22*, 9154. [CrossRef]
22. S, K.; Nair, M.K. Bin packing algorithms for virtual machine placement in cloud computing: A review. *Int. J. Electr. Comput. Eng. (IJECE)* **2019**, *9*, 512. [CrossRef]
23. Madhumala, R.B.; Tiwari, H.; Devaraj, V.C. Virtual Machine Placement Using Energy Efficient Particle Swarm Optimization in Cloud Datacenter. *Cybern. Inf. Technol.* **2021**, *21*, 62–72. [CrossRef]
24. Sheetal, A.P.; Ravindranath, K. High Efficient Virtual Machine Migration Using Glow Worm Swarm Optimization Method for Cloud Computing. *Ing. Des Syst. D'Inf.* **2021**, *26*, 591–597. [CrossRef]
25. Habib, A.; Khan, M. Reinforcement Learning based Autonomic Virtual Machine Management in Clouds. In Proceedings of the 2016 5th International Conference on Informatics, Electronics and Vision (ICIEV), Dhaka, Bangladesh, 13–14 May 2016. [CrossRef]
26. Caviglione, L.; Gaggero, M.; Paolucci, M.; Ronco, R. Deep reinforcement learning for multi-objective placement of virtual machines in cloud datacenters. *Soft Comput.* **2021**, *25*, 1–20. [CrossRef]
27. Kumar, Y.; Kaul, S.; Hu, Y.C. Machine learning for energy-resource allocation, workflow scheduling and live migration in cloud computing: State-of-the-art survey. *Sustain. Comput. Inform. Syst.* **2022**, *36*, 100780. [CrossRef]
28. Sayadnavard, M.H.; Haghighat, A.T.; Rahmani, A.M. A multi-objective approach for energy-efficient and reliable dynamic VM consolidation in cloud data centers. *Eng. Sci. Technol. Int. J.* **2022**, *26*, 100995. [CrossRef]
29. Ma, X.; He, W.; Gao, Y. Virtual machine migration strategy based on markov decision and greedy algorithm in edge computing environment. *Wirel. Commun. Mob. Comput.* **2023**, *2023*, 6441791. [CrossRef]
30. Wu, Q.; Ishikawa, F.; Zhu, Q.; Xia, Y. Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters. *IEEE Trans. Serv. Comput.* **2016**, *12*, 550–563. [CrossRef]

31. Yang, C.T.; Wan, T.Y. Implementation of an energy saving cloud infrastructure with virtual machine power usage monitoring and live migration on OpenStack. *Computing* **2020**, *102*, 1547–1566. [CrossRef]
32. Samsung. 6G. The Next Hyper Connected Experience for All. 2020. Available online: https://cdn.codeground.org/nsr/downloads/researchareas/6G%20Vision.pdf (accessed on 1 March 2025).
33. Carrascosa, M.; Bellalta, B. Cloud-gaming: Analysis of Google Stadia traffic. *Comput. Commun.* **2022**, *188*, 99–116. [CrossRef]
34. Zhang, Y. Mobile Edge Computing for Beyond 5G/6G. In *Mobile Edge Computing. Simula SpringerBriefs on Computing*; Springer: Cham, Switzerland, 2022; pp. 37–45. [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.