Resource Allocation for UAV-Enabled Multi-access Edge Computing

Marcos Falcão[†], Caio Bruno Souza[†], Andson Balieiro[†], Kelvin Dias[†]

Centro de Informática (CIn), Universidade Federal de Pernambuco, Av. Jornalista Anibal Fernandes, Recife, 50740560, Pernambuco, Brasil.

*Corresponding author(s). E-mail(s): mrmfpessoal@gmail.com; amb4@cin.ufpe.br;
Contributing authors: cbbs@cin.ufpe.br; kld@cin.ufpe.br;
[†]These authors contributed equally to this work.

Abstract

In Ultra-Reliable Low Latency Communications (URLLC), balancing trade-offs between energy consumption, service availability, and strict reliability and latency requirements is a significant challenge, especially in Unmanned Aerial Vehicle (UAV)-enabled Multi-access Edge Computing (MEC) environments. The constraints imposed by the size, weight, and power limitations of UAVs further complicate this task. This study addresses optimizing resource allocation in such environments to meet URLLC demands while minimizing power consumption and maximizing service availability. We explore the virtualization layer of the Network Function Virtualization (NFV)-MEC architecture, incorporating node availability and power consumption alongside conflicting URLLC reliability and latency demands. We introduce an energy-aware model based on Continuous-time Markov Chain (CTMC) with an embedded virtual resource scaling scheme for Dynamic Resource Allocation (DRA). To solve the optimization problem related to MEC-enabled UAV node dimensioning, we propose a Genetic Algorithms (GA)-based solution. Our results demonstrate that the proposed GA-based approach achieves a superior balance, with up to a 44% reduction in power consumption compared to the First Fit with maximum resources strategy, while also improving service availability and meeting URLLC requirements. This work provides a comprehensive analysis of key virtualization parameters and their impact on critical services within a single NFV-MEC over a UAV node, offering a robust framework for future 6G network applications.

Keywords: Multi-access Edge Computing, Ultra-Reliable Low-Latency Communications, Continuous-Time Markov Chains, Network Functions Virtualization, Unmanned Aerial Vehicles

1 Introduction

The shift towards Multi-access Edge Computing (MEC) combined with Network Function Virtualization (NFV) and advancements in Unmanned Aerial Vehicles (UAVs) is considered a promising strategy for integrating the non-terrestrial component of the Sixth-Generation of Cellular Networks (6G) [1] as this integration holds potential to provide connectivity, computing resources, data, applications, and network functions closer to users, thereby extending the limits of terrestrial networks. MEC facilitates deploying core network functions and applications in close proximity to User Equipment (UE), reducing complexities associated with multiple network components [2]. NFV, on the other hand, is a carrier-driven initiative for virtualizing network functions (e.g., firewall, Domain Name System, Network Address Translation, and Access and Mobility Management Function) using virtual machines (VMs), which virtualize an entire machine down to the hardware layers, and/or containers on standard servers instead of proprietary single-purpose network devices [3].

Concurrently, UAVs offer unique advantages over regular ground nodes. UAVs can move dynamically in three-dimensional space, providing flexible and adaptive positioning to optimize coverage, reduce latency, and avoid obstacles or interference. UAVs can also be rapidly deployed or repositioned as needed, making them ideal for temporary, urgent, or evolving scenarios such as disaster recovery, large events, or changing environments. Moreover, UAVs often have a better line-of-sight, resulting in potentially higher data rates, lower latency, and improved reliability [4] which is desirable for the emerging 6G networks, which is categorized into use cases, including Ultra-reliable and Low-Latency Communication (URLLC), with stringent service requirements such as 1-millisecond (ms) system latency and 99.999% reliability [5, 6].

Typical core NFs that can be hosted on UAVs include Control plane functions e.g., Access and Mobility Function (AMF), Session Management Function (SMF), and Network Exposure Function (NEF). The UAV-MEC platform could be implemented as an Application Function (AF) and the MEC data plane acting as a particular implementation of a 5G User Plane Function (UPF) that forwards the traffic to MEC applications. For instance, AFs can include video analytics for surveillance, augmented reality (AR) applications for enhanced user experiences, and real-time data processing for IoT devices [7].

However, since UAVs face significant constraints in terms of size, weight, and power, which impact their computational and communication capabilities, the proper resource and energy management of these devices is crucial, especially when URLLC services are considered. While significant efforts have focused on MEC-enabled UAVs, particularly UAV trajectory, communication, and energy optimization, limited attention is given to the impact of the computing subsystem on MEC-UAV node performance and their services [8]. This is paramount important when virtualization is employed as it introduces practical considerations that may make unfeasible the support of services with strict latency and reliability requirements such as URLLCs. For instance, the virtualization layer is expected to handle VNF loading and faults without breaking the stringent URLLC requirements. Thus, besides under and over-provisioning issues that may respectively cause severe Service Level Agreement (SLA) violations and increased power consumption, another concern is the URLLC sensibility towards extra delays that can be caused by unexpected faults such as hardware/software failures and even resource boot and setup or reduced computational power.

In this respect, containers, which are software unities comprising source-code, libraries and dependencies and offer portable, isolated environments for running applications, prove to be cost-effective and exhibit lower startup overhead, making them suitable for supporting Virtualized Network Functions (VNFs) scaling for URLLC services [9]. However, their developmental stage compared to VMs may impact service reliability. On the other hand, VM resource instantiation, failure recovery, and computational power degradation of parallel VMs on the same physical node are overlooked factors that can break the URLLC requirements. Thus, proper UAV-MEC node dimensioning and resource allocation for supporting critical applications is challenging, considering the burden and strengths of the virtualization layer with hybrid technologies and UAVs with constraints in terms of computing resources and energy.

This paper extends our previous work [10] that addresses the resource allocation problem in MEC nodes. However, the current study proposes an energy-aware framework based on Continuous-time Markov Chain (CTMC) with an embedded virtual resource scaling scheme for Dynamic Resource Allocation (DRA). The framework analyzes how virtualization parameters impact critical services in a single MEC-enabled UAV node and proposes a Genetic Algorithms (GA)-based solution to simultaneously minimize power consumption and maximize node availability, considering reliability and latency constraints for critical applications like URLLC. The main contributions include:

- Jointly studying a hybrid VM-containerized onboard infrastructure: The integration of both VMs with strong isolation properties and the flexibility of containers, aiming to optimize the benefits of both technologies.
- Considering commonly neglected assumptions: We address often overlooked factors such as virtual resource setup delays, failures, and computational power degradation of parallel VMs on the same physical node. These play a critical role in impacting communication constraints.
- Addressing critical metrics for URLLC applications: We emphasize crucial metrics for URLLC applications, such as reliability and response times, providing a more holistic performance evaluation.
- Tracking specific to the MEC-UAV environment, including: availability and power consumption, offering a comprehensive perspective on the performance of the proposed framework in real-world scenarios.
- Proposing a Genetic Algorithms (GA)-based solution to simultaneously minimize power consumption and maximize node availability in a MEC-enabled UAV node.

Regarding the GA-based solution, various optimization methods have been applied to solve the resource allocation problem in MEC environments. Heuristic algorithms, such as GA and particle swarm optimization (PSO), provide flexible solutions but often yield approximate results with a high overhead [11]. Moreover, Convex optimization methods, such as those using Lagrangian duality, guarantee global optimality under certain conditions but may struggle with scalability in complex and dynamic environments [12]. Other options relate to Deep learning approaches, particularly deep reinforcement learning (DRL), which excel in handling high-dimensional state spaces and learning optimal policies over time but require substantial computational resources [13]. The focus of this work, however, is not on the optimization method itself but on the development of a robust mathematical model for MEC resource allocation and the formulation of the corresponding optimization problem.

The remainder of this paper is organized as follows. Section 2 discusses the contributions to the MEC-enabled UAV context, which encompasses resource allocation problems and adopted performance metrics. Moreover, Section 3 describes the system model, assumptions, and the proposed framework with its performance metrics. Model validation is in Section 4, while Section 5 outlines the UAV node dimensioning problem formulation, with numerical results in Section 6. Section 7 summarizes the contribution and discusses future directions.

2 Related Work

Existing research on UAV networks can be broadly categorized into two areas: MEC-enabled UAVs and UAV relay networks. Research on MEC-enabled UAVs primarily focuses on UAV location deployment, path planning, and power optimization. For instance, Safwat et al. [14] propose a 3D placement algorithm designed to maximize network coverage by optimizing UAV deployment, considering power and height constraints. Zhao et al. [15] address the UAV's limited energy by optimizing both transmission power and trajectory to enhance transmission efficiency. Additionally, Cai et al. [16] focus on Quality of Service (QoS) for URLLC, designing a strategy to meet URLLC requirements by maximizing transmission data rates and expanding the coverage range of the gNB.

For the Sixth Generation of Mobile Networks (6G), Artificial Intelligence (AI) and Machine Learning (ML) techniques are put forward as key enablers to provide intelligence and automation to the networks at different layers [17]. In this respect, integrating MEC-enabled UAVs with Federated Learning (FL) offers an alternative for deploying ML models that handle vast amounts of data. With MEC-UAVs, ML submodels can be trained separately, and only their parameters are transmitted to a central controller. However, due to energy and computing constraints of the MEC-UAVs, efforts have been made to extend their operational time. Solutions include adopting harvesting devices along with resource allocation and UAV placement optimization [18] or transmission power optimization via communication time and bandwidth allocation, power control, and UAV placememt [19]. Additionally, privacy challenges arise in FL regarding the model parameter transmission. For instance, authors in [20] propose covert communication to address eavesdroppers that implement inversion and differential attacks. Their solution conceals the existence of the legal link by introducing uncertainty to the received signal of eavesdroppers, although it may disrupt the reception of legitimate users. Other approaches leverage blockchain [21] or encryption techniques [22] to ensure FL security.

UAV-relay networks demonstrably enhance network throughput, as seen in recent studies. However, less attention has been given to the impact of deploying MEC servers directly on UAVs. For instance, Costanzo et al. [23] introduce a dynamic strategy for computation allocation, optimizing altitude to minimize UAV energy consumption while satisfying latency constraints. Yang et al. [24] propose a MEC-enabled network over multiple UAVs, focusing on minimizing power consumption and utilizing UAVs as backhaul and core network equipment. Bekkouche et al. explore [25] various UAV roles, concentrating on their performance as edge clouds hosting Aerial Control System (ACS) functions in the proposed testbed, considering both travel plans and computational usage to reduce energy consumption.

Another body of existing works focuses solely on MEC-related computational resource issues, which encompasses: resource placement, scheduling, node dimensioning, and DRA. Edge node dimensioning problems are usually related to the decision on the computational resource characteristics based on a given traffic load, e.g., the total number of servers, processing capacity, and storage. Emara et al. [26] propose an analytical model based on queuing theory to optimize the number of virtual resources to maximize the task execution capacity using the first fit strategy to solve it. Using a similar approach, Kherraf et al. [27] jointly solve 1) a MEC dimensioning sub-problem, 2) an application placement sub-problem, and 3) a workload assignment sub-problem.

Despite substantial focus on latency, energy consumption, and security in UAV-assisted networks, computing failure resilience, virtualization overhead, and resource availability have been relatively overlooked, despite their crucial role in resource dimensioning and allocation. To address this gap, we introduce a framework that assists service providers to optimize the dimensions of a single MEC-enabled UAV node. The goal is to maximize node availability, minimize power consumption, and simultaneously meet reliability and latency constraints. Our proposed framework complements existing solutions put forth by prior authors, offering a comprehensive approach to address these multifaceted challenges.

3 SystemModel

This section describes the analytical scaling framework and adopted performance metrics, considering a single isolated MEC-enabled UAV node, where requests originated from UEs are processed by onboard VNFs, which can be scaled up (down) to cope with intensive (mild) periods.

3.1 Computing Model

In the context of 6G networks, UAV-enabled MEC environments are envisioned to support critical applications, such as disaster recovery, remote healthcare, and intelligent transportation systems. These applications demand high reliability, low latency, and efficient resource management. For instance, in a disaster recovery scenario, UAVs equipped with MEC capabilities can provide rapid deployment of communication infrastructure, supporting real-time video streaming for situational awareness and coordination. These scenarios present key challenges including:

- **Dynamic Resource Management**: Highly dynamic environments where resource demands can vary rapidly based on application and user density.
- **Energy Efficiency**: UAVs have limited battery life, necessitating efficient power management strategies to prolong operational time.
- **URLLC**: Ensuring seamless and dependable service delivery.

To address these challenges, the proposed framework leverages the strengths of both microkernel-based VMs and containers within the MEC architecture. Each onboard VNF runs equally and independently on either a VM or a container, sharing a common physical machine [28]. VMs execute uninterruptedly while containers are scaled upon demand, providing flexibility and efficient resource utilization. Also, a centralized control unit determines the request admission, only activating containerized VNFs when all VM-hosted VNFs are busy. The containerized VNF activation comprises initializing the kernel image and launching the specified function, which is interpreted as a single transition interval (setup time), during which power and resources are consumed but no service is processed.

Fig. 1 illustrates the failure and repair model diagram for VNFs. The diagram shows the normal operational state, failure detection, and subsequent repair processes. It highlights the transition states and the time intervals associated with setup and repair, which are crucial for maintaining the system's reliability and availability.

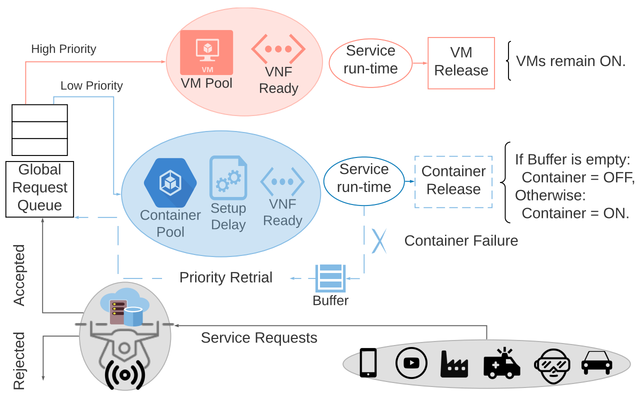


Fig. 1: Failure & Repair model diagram

To account for the VM overhead (CPU), the parallel-operating VMs influence each other, leading to a degraded computational power [29]. For the case of a single VM deployment, the task's execution rate is μ services/unit time, however, to account for the VM overhead, the task's execution rate (μ_V) was modeled considering the total number of VMs, where d is the computation degradation factor. One can observe that μ_V (given by Eq. 1) is a monotone decreasing function of d.

$$\mu_V = \frac{\mu}{(d+1)^{(n-1)}} \tag{1}$$

Containers, on the other hand, are designed to be lightweight with minimal overhead compared to traditional VMs. Their impact is generally lower due to their use of shared operating system kernels and minimal resource isolation mechanisms. Hence, in this work, we assume that the performance overhead of containers can be considered negligible [9, 30, 31].

Differently from VMs, active containerized VNFs may suffer failures during operation, which implies either a service migration to an available VM/container or a repair (triggering a new setup period), with progress being lost only in the latter case. In general, repair times will depend on the type of failure; for instance, a software component crash can be quickly fixed by the host in a few microseconds [32], while other failures may take a few milliseconds to reboot the device and VNF (e.g., 10-50ms for lightweight unikernels). Since the model exclusively deals with critical flows, only the worst-case scenario is considered. Lastly, as soon as a VNF finishes processing and there are no remaining requests, the VNF instance can either be powered down together with the host container or remain active if hosted by a VM. The shutdown delay is ignored for being significantly smaller than the setup (repair) durations [33].

3.2 Analytical Model

The system comprises a single MEC-enabled UAV with a maximum capacity of K services that are served by up to n VMs and c containers, with $K \geq n + c$, which implies a queue (q) that is limited to $K - (n + c)$ services. Service requests follow a Poisson process with rate λ (requests/ms) and server capacities of one service with an exponentially distributed service rate of μ_C for containerized VNFs, whereas for VMs, μ_V is given by Eq. 1. Control applications are likely to fit a regularly spaced packet trace (isochronous), i.e., a superposition of deterministically spaced and sporadic packet streams, where each contributes to a portion of the overall traffic, which might be modeled as a Poisson [34].

Container setup/repair times and failures are also exponentially distributed with rates α and γ, respectively. A regular first come first served queue was assumed for new requests with prioritization for retrial. We assume a standalone deployment and the system is modeled as an M/M/n+c/K queue (Fig. 2) with setup time and failure. The feasible state space is given by $\Omega = (i, j, k) \mid 0 \leq i \leq n$, $0 \leq j \leq c$, and $0 \leq k \leq K$, with $i+j \leq k$ and $i, j, n, c, k, K \in \mathbb{Z}^+$. Each state ($i$,$j$,$k$) denotes the number of services allocated to VMs (i), containers (j) and the total number of services in the system (k), respectively. Furthermore, the steady-state probabilities $\pi(i, j, k)$ are extracted from the solution of a linear system formed by the normalization condition (Eq. 2) and balance equations (Eqs. 3-16) depicted in Table 2. Please consider $(i, j, k) \in \Omega$ in all equations to follow. Table 1 summarizes the symbols adopted in the model description.

$$\sum_{i=0}^{n} \sum_{j=0}^{c} \sum_{k=i+j}^{K} \pi(i, j, k) \tag{2}$$

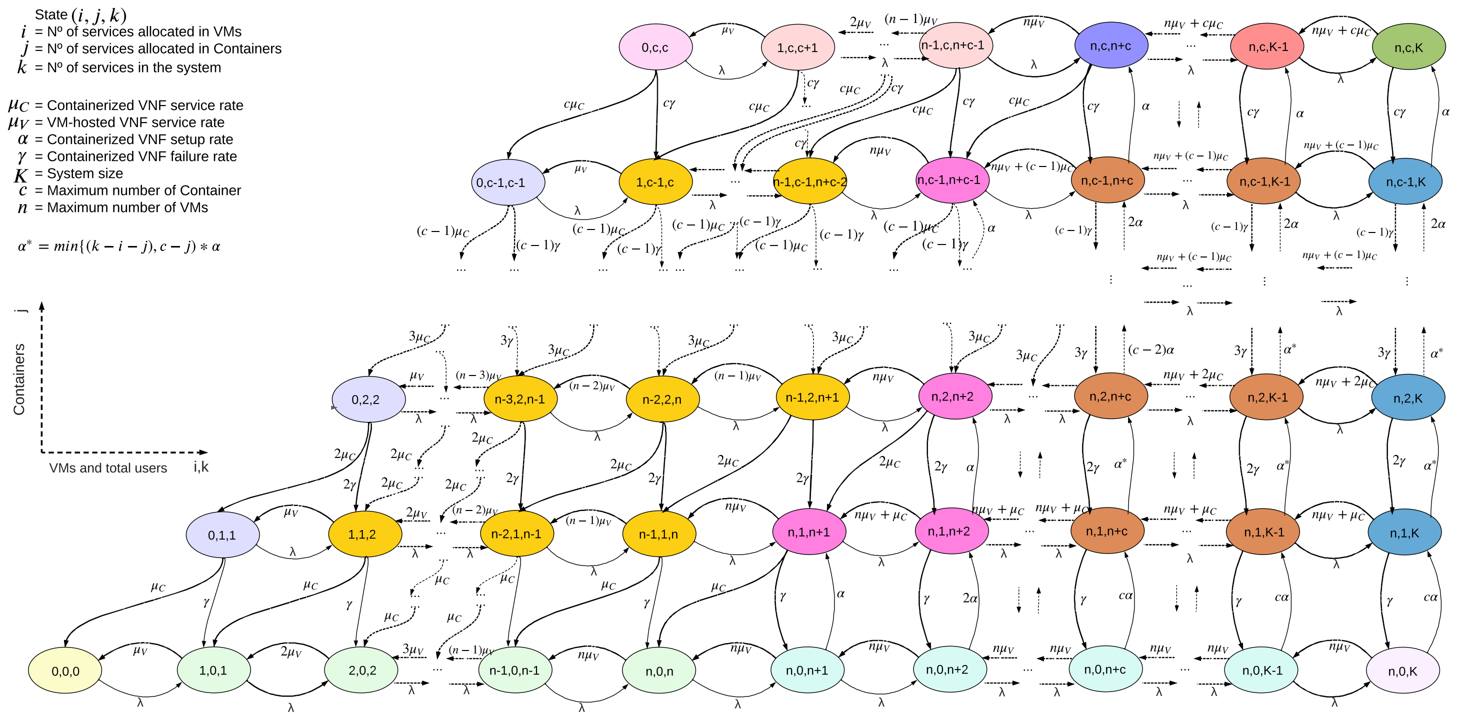

State (i, j, k)
i = Nº of services allocated in VMs
j = Nº of services allocated in Containers
k = Nº of services in the system
μ_C = Containerized VNF service rate
μ_V = VM-hosted VNF service rate
α = Containerized VNF setup rate
γ = Containerized VNF failure rate
K = System size
c = Maximum number of Container
n = Maximum number of VMs
$\alpha^* = min\{(k-i-j), c-j\} * \alpha$
Containers
j
VMs and total users
i,k
0,c,c
1,c,c+1
n-1,c,n+c-1
n,c,n+c
n,c,K-1
n,c,K
0,c-1,c-1
1,c-1,c
n-1,c-1,n+c-2
n,c-1,n+c-1
n,c-1,n+c
n,c-1,K-1
n,c-1,K
0,2,2
n-3,2,n-1
n-2,2,n
n-1,2,n+1
n,2,n+2
n,2,n+c
n,2,K-1
n,2,K
0,1,1
1,1,2
n-2,1,n-1
n-1,1,n
n,1,n+1
n,1,n+2
n,1,n+c
n,1,K-1
n,1,K
0,0,0
1,0,1
2,0,2
n-1,0,n-1
n,0,n
n,0,n+1
n,0,n+2
n,0,n+c
n,0,K-1
n,0,K

Fig. 2: Space state diagram

Table 1: Summary of symbols used in the model description

Symbol	Meaning
d	Degradation factor
μ_V	Service rate for VMs
μ_C	Service rate for containers
λ	User arrival rate
γ	Failure rate for containers
α	Setup/repair rate for containers
K	Maximum number of user in the system (system capacity)
n	Number of VMs in the system
c	Number of containers in the system
(i, j, k)	System state
i	Number of services allocated to VMs
j	Number of services allocated to containers
k	Total number of services in the system
q	Queue (buffer) size
A	Availability
R	Reliability
C	Power Consumption
T	Response Time

The single state where there are no services is represented by Eq. 3. Expression 4 covers the states where all services are running on VM-hosted VNFs, i.e., no containers are required. Next, Eqs. 5 and 6 are similar to Eq. 4, however, in the first there are services in the buffer for containers to setup while in the latter there are no containers to be turned on.

$$\lambda\pi_{(0,0,0)} = \mu_c\pi_{(0,1,1)} + \mu_c\pi_{(1,0,1)} \tag{3}$$

$$\begin{aligned}(\lambda + i\mu_v)\pi_{(i,0,i)} = \lambda\pi_{(i-1,0,i-1)} + min(i+1, n)\mu_v\pi_{(min(i+1,n),0,i+1)} \\ +\gamma\pi_{(i-1,1,i)} + \mu_c\pi_{(i,1,i+1)}\end{aligned} \tag{4}$$

$$[(\lambda + n\mu_v) + min(k-n, c)\alpha]\pi_{(n,0,k)} = \lambda\pi_{(n,0,k-1)} + n\mu_v\pi_{(n,0,k+1)} + \gamma\pi_{(n,1,k)} \tag{5}$$

$$[n\mu_v + min(K-n, c)\alpha]\pi_{(n,0,K)} = \lambda\pi_{(n,0,K-1)} + \gamma\pi_{(n,1,K)} \tag{6}$$

The leftmost diagonal states of Fig. 2 denote the system with no busy VM and some containers running services and are described by Eq. 7, while Eq. 8 refers to the system states with all VM-hosted VNFs idle and all the containers busy. In the top-left states (Eq. 9), all containers are occupied but some VMs are idle. In Eq. 10, all VMs and Containers (CTs) are busy and there are no services in the buffer. Given by Eq. 11, the top-right states are those where all VMs and CTs are Busy and some services waiting in the buffer. Similar to the previous set, Eq. 12 refers to the top-rightmost part, where all VM-hosted and containerized VNFs are busy and the system is full. The left-most states are covered by Eq. 13, where all VMs are running services and the system is full, but some containers are still scaling up. The set of states where the number of services is larger than that of both active VMs and Containers, i.e., the buffer is holding some services is described by Eq. 14. Furthermore, at the frontier, Eq. 15 refers to the states where all VMs and part of the containers are serving, but no services are buffered. Lastly, the states where both VMs and containers are partially occupied and the buffer is empty is located at the left intermediate part of Fig. 2, and its respective equation is 16. The full equation set is summarized in Table 2 and colored according to the states represented in Fig. 2.

$$[(\mu_c+\gamma)j+\lambda]\pi_{(0,j,j)} = \mu_v\pi_{(1,j,j+1)} + (j+1)\mu c\pi_{(0,j+1,j+1)} \tag{7}$$

$$[(\mu_c+\gamma)c+\lambda]\pi_{(0,c,c)} = \mu_v\pi_{(1,c,c+1)} \tag{8}$$

$$[(\mu_c+\gamma)c+i\mu_v+\lambda]\pi_{(i,c,c+i)} = \lambda\pi_{(i-1,c,c+i-1)} + (i+1)\mu_v\pi_{(i+1,c,c+i+1)} \tag{9}$$

$$[(\mu_c+\gamma)c+n\mu_v+\lambda]\pi_{(n,c,n+c)} = \lambda\pi_{(n-1,c,n+c-1)}+(n\mu_v+c\mu_c)\pi_{(n,c,n+c+1)}+\alpha\pi_{(n,c-1,n+c)} \tag{10}$$

$$[(\mu_c+\gamma)c+n\mu_v+\lambda]\pi_{(n,c,k)} = \lambda\pi_{(n,c,k-1)} + (n\mu_v+c\mu_c)\pi_{(n,c,k+1)} + \alpha\pi_{(n,c-1,k)} \tag{11}$$

$$[(\mu_c+\gamma)c+n\mu_v]\pi_{(n,c,K)} = \lambda\pi_{(n,c,K-1)} + \alpha\pi_{(n,c-1,K)} \tag{12}$$

$$\begin{aligned}[(\mu_c+\gamma)j+n\mu_v+min(K-n-j,c-j)\alpha]\pi_{(n,j,K)} &= \lambda\pi_{(n,j,K-1)} + (j+1)\gamma\pi_{(n,j+1,K)} \\ &+min(K-n-j-1,c-j-1)\alpha\pi_{(n,j-1,K)}\end{aligned} \tag{13}$$

$$\begin{aligned}[(\mu_c+\gamma)j+n\mu_v+\lambda+min(k-n-j,c-j)\alpha]\pi_{(n,j,k)} &= \lambda\pi_{(n,j,k-1)} \\ +(j+1)\gamma\pi_{(n,j+1,k)} + min(k-n-j-1,c-j-1)\alpha\pi_{(n,j-1,k)}\end{aligned} \tag{14}$$

$$\begin{aligned}[(\mu_c+\gamma)j+n\mu_v+\lambda]\pi_{(n,j,k)} &= \lambda\pi_{(n-1,j,k-1)} \\ +(j+1)\gamma\pi_{(n-1,j+1,k)} + (j+1)\mu_c\pi_{(n,j+1,k+1)} &+ (n\mu_v+j\mu_c)\pi_{(n,j,k+1)} + \alpha\pi_{(n,j-1,k)}\end{aligned} \tag{15}$$

$$\begin{aligned}[(\mu_c+\gamma)j+i\mu_v+\lambda]\pi_{(i,j,k)} &= \lambda\pi_{(i-1,j,k-1)} + (i+1)\mu_v\pi_{(i+1,j,k+1)} \\ &+(j+1)\gamma\pi_{(i-1,j+1,k)} + (j+1)\mu_c\pi_{(i,j+1,k+1)}\end{aligned} \tag{16}$$

3.3 Performance Metrics

In MEC-enabled networks, task execution at the MEC server is strongly tied to resource availability, failure resilience and response time, while power management is crucial for UAV equipment. In this section, we consider the steady-state analysis of the CTMC under study, followed by the derivation of the system's Availability (A), Reliability (R), Power Consumption (C) and Response Time (T).

Table 2: Balance Equation Summary

Eq.	State(s) (i,j,k)	Condition(s)	Description
(3)	$(0,0,0)$	n/a	Empty system.
(4)	$(i,0,i)$	$(0 \leq i \leq n)$	All services are running on VMs.
(5)	$(n,0,k)$	$(n+1 \leq k \leq K-1)$	All VMs are busy and no CT is yet active.
(6)	$(n,0,K)$	(n/a)	Similar to (4), but the system is full.
(7)	$(0,j,j)$	$(1 \leq j \leq c-1)$	No busy VM and some CTs running services.
(8)	$(0,c,c)$	(n/a)	CTs running all services.
(9)	$(i,c,c+i)$	$(1 \leq i \leq n-1)$	VMs partially busy and all CTs running services.
(10)	$(n,c,n{+}c)$	(n/a)	All CTs and VMs are serving; queue is empty.
(11)	(n,c,k)	$(n+c+1 \leq k \leq K-1)$	Similar to (9), but with waiting services.
(12)	(n,c,K)	(n/a)	All VMs and CTs are serving and the system is full.
(13)	(n,j,K)	$(1 \leq j \leq c-1)$	Full system with busy VMs and CTs either serving or scaling up.
(14)	(n,j,k)	$(1 \leq j \leq c-1)$ *and* $(n+j+1 \leq k \leq K-1)$	All VMs and CTs are serving and the queue is not empty.
(15)	$(n,j,n{+}j)$	$(1 \leq j \leq c-1)$	All VMs and part of the CTs are serving.
(16)	$(i,j,i+j)$	$(1 \leq i \leq n-1)$ *and* $(1 \leq j \leq c-1)$	VMs and CTs are partially occupied and the queue is empty.

3.3.1 Availability (A)

In our framework, Availability is the system's ability to offer the minimum amount of functional and accessible VNFs. In particular, a VNF instance is considered available if at least one of its constituents (VM-hosted or containerized) remains accessible. In brief, the MEC-enabled UAV node Availability (A) is obtained by the probability sum of all states except those representing full capacity, as shown in Eq. 17.

$$A = 1 - \sum_{i=0}^{n} \sum_{j=0}^{c} \pi_{i,j,K} \tag{17}$$

3.3.2 Reliability (R)

The designed framework also evaluates the Reliability (R) being given by Eq. 18, which combines the admitted flow, obtained by multiplying the arrival rate by the system‘s availability (A), with the effective failure rate in the entire node, i.e., it denotes the probability that a service is served without experiencing failures while being processed by MEC VNFs.

$$R = 1 - \frac{\gamma}{\lambda A} \sum_{j=1}^{c} j \left[\sum_{i=0}^{n} \sum_{k=1}^{K} \pi_{i,j,k} \right] \tag{18}$$

3.3.3 Power Consumption (C)

The computational power consumption is an important component of the operational costs and must be considered by the service provider for resource planning to address cost-performance trade-offs. In our framework, the power consumption (C) Eq. 24 describes the total power consumption by summing the power consumption for each state (idle, setup, and busy) across both VMs and containers. Each term in the equation represents the product of the mean number of resources in a particular state and the corresponding power consumption for that state.

Moreover, the mean number of active VMs and CTs in each state (idle or busy) is described in Eqs. 19-23. Eq. 19 calculates the mean number of idle VMs by considering the differences between the total number of VMs and the number of active VMs across all states. This provides an estimate of how many VMs are idle on average. The same rationale applies to the remaining equations. In addition, the notation used to denote the energy consumption of each technology and state is summarized in Table 3.

$$\overline{VM}_{idle} = \sum_{i=0}^{n} (n-i) \left[\sum_{j=0}^{c} \sum_{k=0}^{K} \pi_{i,j,k} \right] \tag{19}$$

$$\overline{VM}_{busy} = \sum_{i=0}^{n} i \left[\sum_{j=0}^{c} \sum_{k=0}^{K} \pi_{i,j,k} \right] \tag{20}$$

$$\begin{aligned} \overline{CT}_{idle} = \sum_{j=0}^{c} (c-j) \sum_{i=0}^{n} \pi_{i,j,k},\ with\ k = i + j, \\ + \sum_{j=0}^{c} \sum_{i=0}^{n} \sum_{k=i+j+1}^{K} (c - j - min(k - i - j, c - j))\pi_{i,j,k} \end{aligned} \tag{21}$$

$$\overline{CT}_{setup} = \sum_{j=0}^{c} \sum_{k=n+j+1}^{K} min(k - n - j, c - j)\pi_{n,j,k}. \tag{22}$$

$$\overline{CT}_{busy} = \sum_{j=1}^{c} j \left[\sum_{i=0}^{n} \sum_{k=j+i}^{K} \pi_{i,j,k} \right]. \tag{23}$$

$$\begin{aligned} C = P_{idle}^{VM}\, \overline{VM}_{idle} + P_{busy}^{VM}\, \overline{VM}_{busy} \\ + P_{idle}^{CT}\, \overline{CT}_{idle} + P_{setup}^{CT} \overline{CT}_{setup} + P_{busy}^{CT}\, \overline{CT}_{busy} \end{aligned} \tag{24}$$

3.3.4 Response Time (T)

We define the Response Time (T) of a VNF that processes the service as the interval between the service arrival on the edge node and its processing time, including the containerized VNF setup restart times if these events are triggered. The Mean Response Time is obtained by calculating the mean number of services in the system and the mean number of accepted services, as shown in Eq. 25.

$$T = \frac{1}{\lambda A} \sum_{k=0}^{K} k \left[\sum_{i=0}^{n} \sum_{j=0}^{c} \pi_{i,j,k} \right] \tag{25}$$

4 Validation and Analysis

The scenario of interest is composed of one semi-static MEC-enabled UAV which able to increase its coverage area. Intuitively, the higher the altitude, the larger is the coverage offered by the platform and the lower is the chance of shadowing effects, thus the service arrival rate may increase or decrease accordingly [23]. The analytical results were validated against discrete-event simulations (Figs. 3-5), where the lines denote the analytical and the markers represent simulation results. With regards to the main parameters, we have followed a subset of the 3GPP Release 16 (TR 38.824), in which the service rates μ and μ_C are 1 (1 service/ms), whereas μ_V is calculated in Eq. 1 .

The discrete-event simulator is designed to emulate the system's operation based on a sequence of events in time. Unlike real-time simulations, discrete-event simulations only update the system state at specific event times, providing simplicity and flexibility in modeling dynamic systems. This approach allows detailed tracking of resource usage by each request.

Each scenario simultaneously evaluates the impact of a pair of parameters: Fig. 3 Multiple VM Amounts (n) and Overhead Degradation Factor (d), Fig. 4 multiple container amounts (c) and failure rates (γ), and Fig. 5 multiple buffer sizes (q) and container setup rates (α), with the service arrivals ranging from 1 up to 100 requests/ms. In addition, unless otherwise stated the baseline values for failure (γ) and setup rates (α) were 0.001 and 1 unit/ms, respectively, which is in accordance with [33]. In terms of power consumption for VMs and containers for different operation states, we adopted the values from the network intensive experiment in [9], which is summarized in Table 3. The remaining parameters can be found in Table 4.

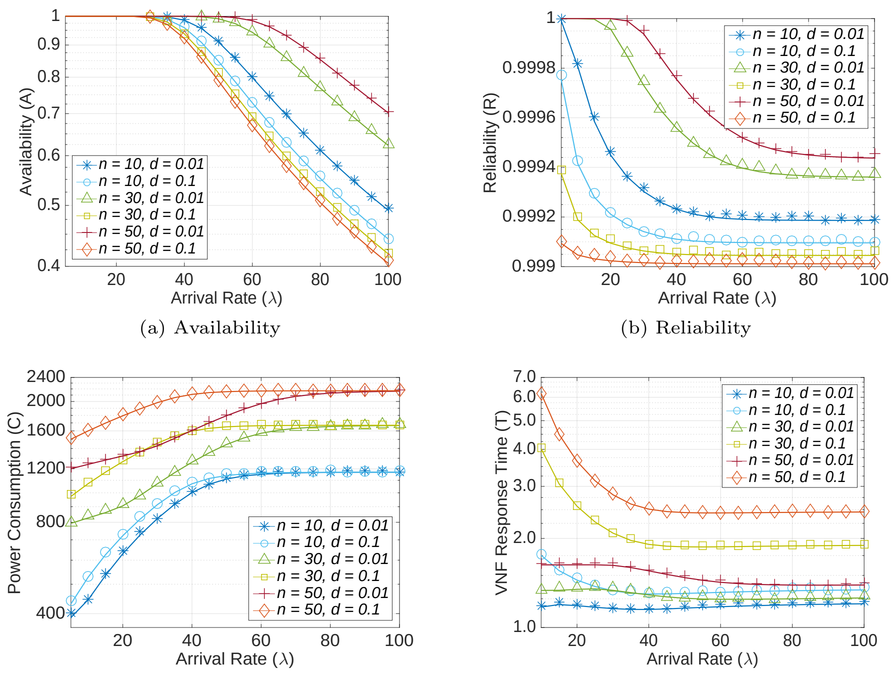
Table 3: Notation and Power Consumption Values

Virtualization	State	Status	Symbol	Value
VM-hosted	Idle	ON	P^{VM}_{idle}	20W
VM-hosted	Busy	ON	P^{VM}_{busy}	25W
Containerized	Idle	SLEEP	P^{CT}_{idle}	4W
Containerized	Setup	ON	P^{CT}_{setup}	8W
Containerized	Busy	ON	P^{CT}_{busy}	23W

Table 4: Experiment Sets

Section	Parameters	n	c	q	d	γ	α
4.1	n, d	10, 30, 50	40	10	10^{-2}, 10^{-1}	10^{-3}	1
4.2	c, γ	10	40, 60, 80	10	10^{-2}	10^{-3}, 10^{-2}	1
4.3	q, α	10	40	10, 30, 50	10^{-2}	10^{-3}	10^{-1}, 1

4.1 Multiple VMs (n) and Overhead Degradation Factor (d)



(a) Availability (b) Reliability

(c) Power Consumption (d) Response Time

Fig. 3: Multiple VMs (n) and Overhead Degradation Factor (d)

This scenario simultaneously evaluates the impact of Multiple VM Amounts (n) with 10, 30 and 50 units and the Overhead Degradation Factor (d) of 0.1 and 0.01. The remaining parameters are fixed according to the first line in Table 4.1.

4.1.1 Availability and Reliability

Figures 3a and 3b illustrate the impact of increasing the number of VMs and the overhead degradation factor on system availability and reliability, respectively. Both graphs are strictly descending, which is expected since, for all scenarios, the arrival rate ranges from 0 to 100 arrivals per unit time, rapidly surpassing the system's capacity. Given the three options for VM amounts and two for the degradation factor, this results in six curves representing different configurations.

In terms of availability (Fig. 3a), the best configuration is the top curve marked with the cross, which corresponds to the configuration with the highest number of VMs (50) and the lowest degradation factor (0.01). In this experiment, the degradation factor had a greater impact on availability than the number of VMs. This is because the degradation factor directly affects the service rate of each VM; a lower degradation factor means that the VMs can handle more requests, reducing the need for frequent container activation and thereby maintaining higher availability. This is further evidenced by the fact that the top three curves all have a degradation factor of $d = 0.01$. Additionally, the bottom three configurations, all with a degradation factor of $d = 0.1$, exhibit very similar performance, despite having three and five times the amount of VMs ($n = 30$ and $n = 50$) compared to the baseline configuration that has ten VMs ($n = 10$).

Moreover, still regarding Fig. 3a, the magnitude of the differences in availability between each curve is much larger, often differing by whole percentage points (e.g., 1% to 10%), whereas the differences in reliability (Fig. 3b) are much smaller, typically differing by only a few ten-thousandths of a percent (e.g., 0.0001%). The primary reason for the significant differences in availability is the degradation factor's impact on the VMs' service rates, which affects how often the system needs to activate containers to handle the load.

While the reliability differences are small, they indicate that the system's failure rates are low across all configurations. However, even small reductions in the effective service rate due to higher degradation can lead to significant decreases in availability because the system must rely more on containers, which are more prone to failure. Although the difference in reliability is much smaller, it remains significant for URLLC applications where high reliability is key [7].

4.1.2 Power Consumption and Response Time

In Fig. 3c, both the number of VMs (n) and the overhead degradation factor (d) significantly impact power consumption, but in different ways. Since VMs are not dynamically scalable, an increase in n naturally leads to higher overall power consumption, regardless of the network load (λ). This is because more VMs mean more active resources consuming power.

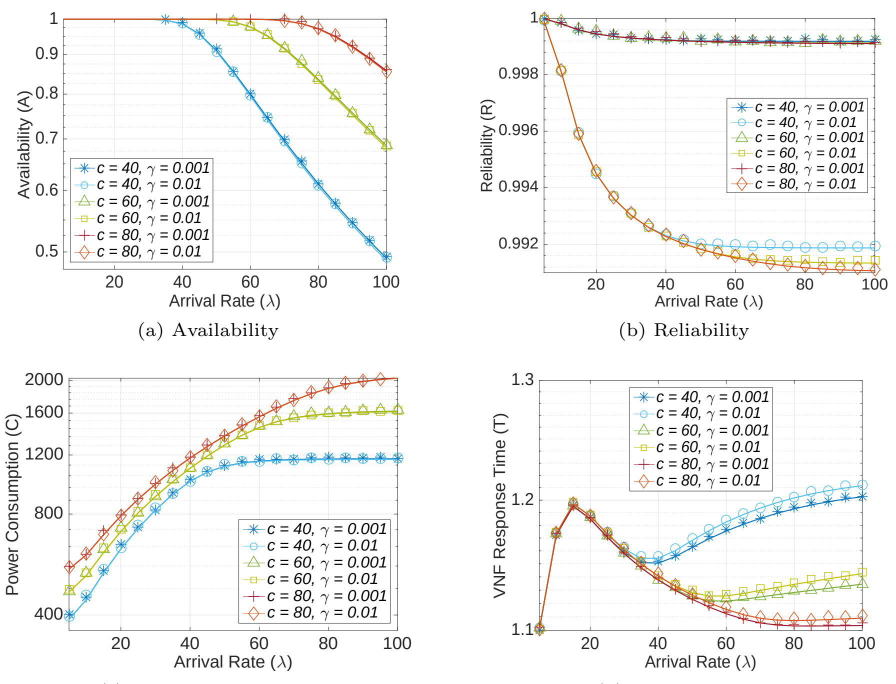
The overhead degradation factor (d), on the other hand, primarily affects non-saturated systems and particularly impacts configurations with higher VM values (30 and 50). For a network load of 100 requests per unit time ($\lambda = 100$), the curves with the same number of VMs overlap, indicating a saturated system where all resources are processing services. However, each pair of curves has different throughput due to varying d values. For instance, the power consumption difference between configurations with 30 VMs and a degradation factor of 0.01 ($n = 30$, $d = 0.01$) and 30 VMs and a degradation factor of 0.1 ($n = 30$, $d = 0.1$) shows a maximum gap of $500W$ at $\lambda = 35$. Similarly, the difference for configurations with 50 VMs and a degradation factor of 0.01 ($n = 50$, $d = 0.01$) and 50 VMs and a degradation factor of 0.1 ($n = 50$, $d = 0.1$) is approximately $600W$ at the same reference point. These differences are significant since each pair has the same number of VMs, highlighting the impact of the degradation factor on power consumption.

Another notable observation in Fig. 3c is the similar performance of the curves for configurations with 30 VMs and a degradation factor of 0.1 ($n = 30$, $d = 0.1$) and 50 VMs and a degradation factor of 0.01 ($n = 50$, $d = 0.01$) in the interval from $\lambda = 20$ to $\lambda = 50$. Despite the first having almost half of the VM resources, their power consumption is similar. This indicates that the impact of the degradation factor is amplified by the number of VMs; higher degradation values result in lower overall service rates, causing VMs to remain in Busy mode for longer periods and forcing the system to activate containers more frequently.

Unlike power consumption, changes in the number of VMs have very little impact on the response time (Fig. 3d) compared to the results of the curves with different degradation factors. This indicates that the positive effects of increasing the number of VMs can be mitigated by higher degradation factors. Generally, there are multiple nuances involving each parameter, service load, and performance metrics that might conflict, highlighting the importance of an adequate dimensioning process. For instance, comparing the two curves with 50 VMs, the response time almost doubles when the degradation factor increases from 0.01 to 0.1 ($n = 50$, $d = 0.01$ to $d = 0.1$). Additionally, considering availability and reliability, the curve with 50 VMs and a degradation factor of 0.01 ($n = 50$, $d = 0.01$) had the best performance. However, for power consumption, it was one of the worst-performing configurations.

4.2 Multiple Containers (c) and Failure Rates (γ)

A larger γ signifies smaller intervals between successive containerized VNF failures. Unlike the setup rate (α), this parameter enhances the system's performance as it decreases. Hence, γ was varied by factors of 10 and 100. Additionally, containers are not susceptible to significant overhead issues, and there is no correlation between the number of containers and their individual service rates. For this resource type, failures are isolated events. In this scenario, we reduced and fixed the total number of VMs ($n = 10$) to underscore the impact of the number of containers (c) and the their failure rate (γ). For small $\lambda < 10$, most services are processed in VMs, which does not significantly influence the results.



(a) Availability (b) Reliability (c) Power Consumption (d) Response Time

Fig. 4: Multiple Container Amounts (c) and Failure Rates (γ)

4.2.1 Availability, Power Consumption and Response Time

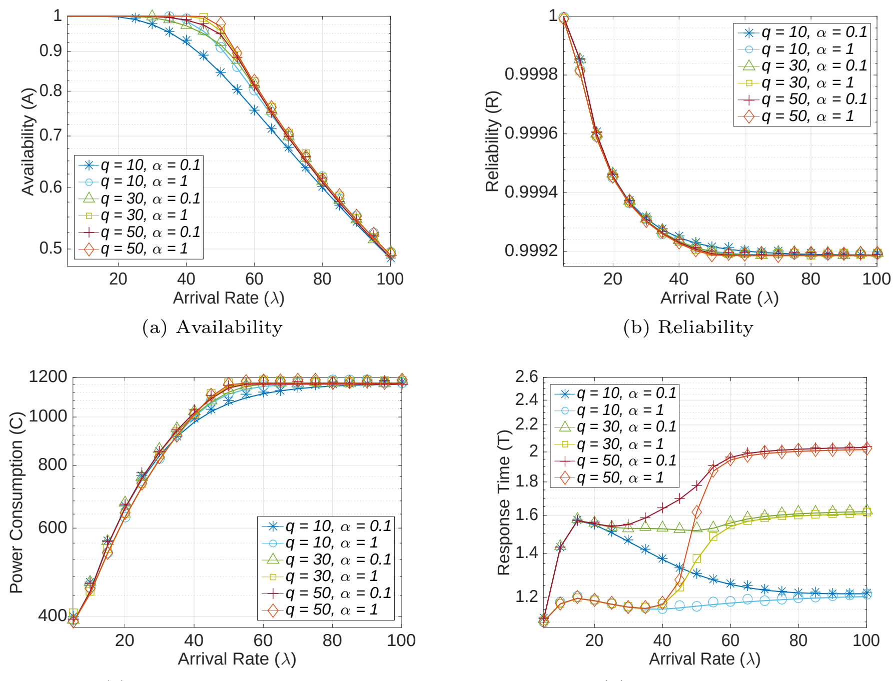
It was observed that the failure rate (γ) values exert minimal influence on the Availability (Fig. 4a), Power Consumption (Fig. 4c), and Response Time (Fig. 4d) within the considered ranges. This does not imply that γ has no impact on these metrics; rather, for a significant impact, γ must be at least of the same order of magnitude as the setup rate (α), causing containers to fail more frequently and rapidly degrading system capacity. Conversely, the number of containers (c) had a substantial impact on these metrics, especially for high arrival rates ($\lambda \to 100$), where the maximum differences in each figure were recorded.

In Fig. 4d, the Response Time shows a spike at $\lambda = 10$, corresponding to the fixed maximum number of VMs ($n = 10$). From this point on, containers are activated, explaining the sudden spike due to setup time and the subsequent drop when these resources become available to process incoming services. Finally, within each pair of curves, there is a slight increase in Response Time at different λ values, explained by resource limits and further queue activation, which do not involve a processing unit.

4.2.2 Reliability

Regarding the Reliability (Fig. 4b), the failure rate (γ) remains a critical component and significantly affects the curves. This can be observed as the three top curves share the same failure rate (γ) of 0.001, maintaining reliability above 99.9% for all arrival rate values. Conversely, the number of containers (c) has minimal influence. This is because reliability is primarily determined by the probability of service interruption due to failures, which is directly influenced by the failure rate (γ). Since each container operates independently, the addition of more containers (c) does not significantly reduce the probability of failure of individual containers. The system's reliability is thus more sensitive to the failure rate (γ) than to the number of containers available, because even with a large number of containers, frequent failures (high γ) would still cause significant interruptions.

4.3 Multiple Queue Sizes (q) and Container Setup Rates (α)



(a) Availability (b) Reliability (c) Power Consumption (d) Response Time

Fig. 5: Multiple Queue Sizes (q) and Container Setup Rates (α)

Larger setup rates (α) and queue spaces (q) mean smaller container setup delays because higher setup rates (α) indicate that containers can be set up faster, leading to more VNF instances becoming available per unit time. Similarly, greater buffer capacity (q) means that more incoming service requests can be stored, preventing immediate drops or failures when the system is under high load. The results could have been amplified for larger queue spaces (q) or for a higher ratio between setup rate (α) and failure rate (γ). However, to maintain the same total amount of resources as in previous experiments, we opted for $q = 10, 30,$ and 50. Most configurations performed similarly for the Availability (Fig. 5a), except for the curves with buffer space of ten units ($q = 10$). This is because a smaller buffer capacity means that the system is more likely to drop incoming requests when it is under load, leading to lower availability. In fact, Availability, Power Consumption (Fig. 5c), and Response Time (Fig. 5d) are likely to increase as $q \to +\infty$. This is because larger buffer capacities allow the system to handle more requests, thus increasing availability and response time. However, larger buffers also mean that more requests are waiting to be processed, which can lead to higher power consumption due to the increased workload. On the other hand, the Reliability (Fig. 5b) is barely affected by the increase in the buffer space (q). This is because reliability is solely dependent on the failure rate (γ) of the containers. Even with a larger buffer, if the containers are failing at the same rate, the overall reliability will not significantly change.

4.3.1 Availability and Reliability

Regarding variations in setup rate (α), the differences in Availability (Fig. 5a) from the curves with queue space of ten units ($q = 10$) were significant, similar to the results for different queue spaces (q). This was somewhat expected since higher setup rate (α) means more available containers per unit time, thereby increasing availability. On the other hand, the reliability results (Fig. 5b) were not significantly affected by either parameter. We believe that the ratio between setup rate (α) and failure rate (γ) contributed to this result; if failure rate (γ) were higher, different setup rate (α) values might result in significant differences in reliability, with the curves for higher setup rate (α) more likely to present lower reliability. Although the difference is small, the configuration with queue space of ten units ($q = 10$) and setup rate ($\alpha = 0.1$) was the most reliable since fewer containers are serving per unit time, resulting in fewer failures.

4.3.2 Response Time

With respect to the Response Time (Fig. 5d), two configuration groups overlap until the arrival rate (λ) reaches 20 and 40 units. Then, the groups split and form three pairs of curves, each with the same queue size (q), crossing each other for multiple arrival rate (λ) values. The reason for such behavior is related to the resource availability empowered by higher setup rate (α) and the use of queues to store service surplus. Therefore, configurations with higher setup rate (α) and lower queue size (q) are likely to respond much faster. For arrival rates ($\lambda \to +\infty$), the setup rate (α) becomes irrelevant since the containers are continuously processing services, only resetting in case of failure. On the other hand, higher queue size (q) values allow the system to store more services without actually processing them, resulting in higher response times.

4.4 Result Summary

A discrete-event simulator was used to validate the proposed model under varying conditions (light, moderate, and heavy workloads). The following parameters were used to configure the simulation scenarios:

- **Network Load (λ):** The arrival rate of service requests, ranging from light to heavy loads, to assess system performance under different traffic conditions.

- **Number of VMs (n) and Containers (c):** To evaluate the impact of varying resource quantities on system performance.
- **Overhead Degradation Factor (d):** To model the performance degradation of VMs due to overhead.
- **Failure Rates (γ):** The rate at which containers experience failures during operation.
- **Service Rates (μ_V and μ_C):** The processing rates of VMs and containers.

The validation results show the significance of considering various factors in the system's design and resource planning to achieve optimal performance. Additionally, the results emphasize the challenge of managing opposing performance metrics, as slight parameter modifications can positively impact one metric while negatively affecting others. The following lines describe a summary of our findings:

- **Availability:** Increasing the number of VMs and containers generally improves availability and reliability. However, higher overhead degradation factors can reduce system availability, necessitating more frequent container activation and higher failure rates.
- **Reliability:** The system's reliability remained consistently high across light to moderate workload conditions. However, a noticeable drop was observed under heavy workloads, suggesting that the system's ability to maintain uninterrupted service is challenged by higher demands.
- **Power Consumption:** Power consumption is closely linked to system availability. Configurations with higher availability tend to consume more power due to the increased number of active resources.
- **Response Time:** The response time increases with higher failure rates and setup delays, affecting the system's ability to meet URLLC requirements.

These findings underscore the need to identify the optimal or suboptimal set of parameters that enable the system to achieve reasonable performance in terms of URLLC while balancing the trade-offs between the performance metrics. Therefore, in the next section, we present the proposed optimization approach, which leverages Genetic Algorithms (GA) to dynamically adjust resource allocation parameters, ensuring the system maintains optimal performance under varying conditions.

5 Problem Formulation and Proposed Scheme

This section describes the MEC-enabled UAV node dimensioning as a multiobjective problem, which takes into account the objectives discussed in the previous section. Moreover, it proposes a scheme based on Genetic Algorithms (GA) to solve the problem, detailing its structure, parameters, and operation.

5.1 Optimization Problem

Different objectives must be considered for proper MEC-enabled UAV resource dimensioning, which may be formulated as an optimization problem as follows in Eq. 26. Given a service demand characterized by the arrival (λ) and service rates when running on virtual machines and containers (μ_n and μ_c, respectively) and the containerized VNF setup and failure rates (α and γ, respectively), determining the most appropriate node dimension in terms of the number of virtual machines, containers and buffer size ($x^* = [{x_1}^*, {x_2}^*, {x_3}^*] \in X$) that maximizes the system's availability (A) and minimizes power consumption (C) while also satisfying reliability (R) and response time (T) constraints (R_{min}, T_{max}). X means the feasible solutions for the problem, x_1^i, x_2^i, x_3^i denote the number of VMs, CTs and buffer size that compose the solution x^i, which are defined within $[Imin_j \ Imax_j]$, with $j = 1, 2, 3$, respectively.

Focusing on a specific objective may deteriorate others or violate constraints, which in the context of URLLC, could result in a lack of support for some applications. For instance, the main requirement for the remote surgery scenario is the communication latency. To mitigate such an effect, we propose an evolutionary scheme based on genetic algorithms (GA) that copes with the given conflicting objectives and constraints that uses an adjusted dominance concept.

$$\begin{aligned} Maximize\ A(x)\ and\ Minimize\ C(x) \\ Subject\ to\ R(x) \geq R_{min}, \\ T(x) \leq T_{max}, \\ x^i \in X, x_j^i \in \mathbb{Z}^+, \\ with\ j = 1, 2, 3\ and\ x_j^i \in [Imin_j\ \ Imax_j] \end{aligned} \tag{26}$$

GA has been widely adopted for solving wireless network optimization problems [35] and multiple solutions to reduce its convergence time have been proposed [36]. In this work, we assume that the GA is adopted for the dimensioning phase, i.e., before proper operation.

5.2 Chromosome Structure and Fitness Function

The GA relies on evolving a solution set given through the combination of several possible sets through its operators: selection, crossover and mutation. In this work, the GA handles multiple MEC-enabled UAV node configurations simultaneously in each interaction.

The individual X (chromosome) represents the maximum number of VMs, CTs and buffer size for the solution x^i, i.e., $x_1^i, x_2^i, x_3^i \in \mathbb{Z}^+$. The possible node configurations evolve as different generations and produce more appropriate solutions represented by the set of resources. In this respect, each individual from the current population x^i is compared to the entire population in terms of all four performance metrics: Availability (A), Reliability (R), Power Consumption (C), and Response Time (T).

To address this problem, a fitness function based on the concept of dominance was adopted. This function evaluates each individual by comparing it to the rest of the population using four performance metrics: Availability (A), Reliability (R), Cost (C), and Time (T). Equations 28-31 describe the number of individuals dominated for each metric. In addition to the points scored in these comparisons, an individual's fitness is increased by another constant ω_r and/or $\omega_t \in \mathbb{R}^+$ if the Reliability (R) and

Response Time (T) exceed predefined thresholds (R_{min} and T_{max}, respectively), as specified in Equations 32 and 33. For the population size considered in the subsequent experiments, we set $\omega_r = 1$ and $\omega_t = 1$. The resulting fitness function, described in Equation 27, aggregates all these components.

$$\begin{aligned} Fitness(X^i) = \sum_M D_M(X^i) + \sum_S B_S(X^i), \\ with\ M\ being\ A, R, C,\ and\ T, \\ B\ assuming\ R\ and\ T, \\ i, j \in \mathbb{Z}^+ and\ i, j \leq L, \\ where\ L\ is\ number\ of\ individuals \end{aligned} \tag{27}$$

$$D_A(X^i) = \sum_{i \neq j, j=1}^{L} \begin{cases} 1 \ , if\ A(X^i) \geq A(X^j) \\ 0 \ , otherwise \end{cases} \tag{28}$$

$$D_R(X^i) = \sum_{i \neq j, j=1}^{L} \begin{cases} 1 \ , if\ R(X^i) \geq R(X^j) \\ 0 \ , otherwise \end{cases} \tag{29}$$

$$D_C(X^i) = \sum_{i \neq j, j=1}^{L} \begin{cases} 1 \ , if\ C(X^i) \leq C(X^j) \\ 0 \ , otherwise \end{cases} \tag{30}$$

$$D_T(X^i) = \sum_{i \neq j, j=1}^{L} \begin{cases} 1 \ , if\ T(X^i) \leq T(X^j) \\ 0 \ , otherwise \end{cases} \tag{31}$$

$$B_R(X^i) = \begin{cases} \omega_r \ , if\ R(X^i) \geq R_{min} \\ 0 \ , otherwise \end{cases} \tag{32}$$

$$B_T(X^i) = \begin{cases} \omega_t \ , if\ T(X^i) \leq T_{min} \\ 0 \ , otherwise \end{cases} \tag{33}$$

5.3 GA Operators and Parameters

We adopted the classical GA operators, i.e., the roulette wheel for selection and uniform operator for the crossover, hence, the highest fitness individuals are more likely to move to the next generation while creating new individuals from the crossover process, while a bit mutation operator was also employed.

Multiple tests were conducted to define the crossover (P_c) and mutation (P_m) probabilities. Five values within the intervals [0.2, 1] and [0.01, 0.09] were tested for P_c and P_m, respectively (Table 5). The test scenario was composed of up to 50 VMs, 50 Containers and 50 Buffer positions, with the remaining parameters being described in Table 7) and λ was fixed to a mid range value of 50. For each test case, five simulation instances were performed. The highest average fitness value for the last generation's population was selected for each test case (Fig. 6). The best performance was obtained by test case 16, in which P_c and P_m assumed 0.8 and 0.01, respectively. Thus, these

Table 5: Crossover and Mutation Probabilities

Parameters	Value
Crossover probability (P_c)	0.2/0.4/0.6/0.8/1
Mutation probability (P_m)	0.01/0.03/0.05/0.07/0.09

Table 6: Selected GA Parameters

Parameters	Value
Generations (G)	100
Population size (L)	50
Crossover probability (P_c)	0.8
Mutation probability (P_m)	0.01

values were applied in the next section together with the population size (L) and number of generations (G), which were set to 50 and 100, respectively (Table 6).

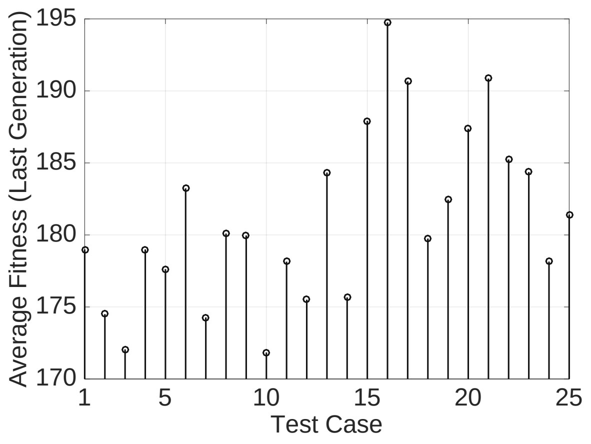


Fig. 6: Test Cases for Pc and Pm

5.4 GA Scheme Execution Flow

Given a maximum estimated service load (requests/ms) and resource limits for VMs, Containers and Buffer size, an initial set of candidate solutions is randomly generated (first population). Then, the individuals are evaluated using the dominance-based fitness function (Eq. 27), which takes four performance metrics, besides an additional score for each threshold that it overcomes. Moreover, the operators: selection, crossover and mutation are applied in this order, generating an entirely new population. In addition, to ensure that the best individual will not be lost during the selection process, we have also employed elitism, i.e., the most fit individual is guaranteed in the next generation without undergoing mutation. Finally, the process repeats until the maximum number of generations (G) is reached (stop criterion). Then, the best individual is chosen as the final solution, which represents the most suitable resource configuration (maximum number of VMs, Containers and Buffer size). The GA scheme execution flow is depicted in Fig. 7.

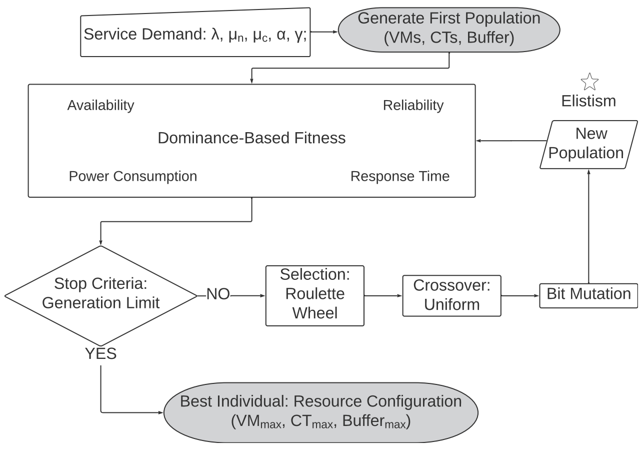

Service Demand: λ, μ_n, μ_c, α, γ;
Generate First Population (VMs, CTs, Buffer)
Availability
Reliability
Dominance-Based Fitness
Power Consumption
Response Time
Elitism
New Population
Stop Criteria: Generation Limit
NO
Selection: Roulette Wheel
Crossover: Uniform
Bit Mutation
YES
Best Individual: Resource Configuration (VM_max, CT_max, Buffer_max)

Fig. 7: GA Scheme Execution Flow

With regards to the overhead introduced by the use of GAs, this strategy is more suitable in the planning or dimensioning phases [37] [38] [39]. GAs can calculate optimal parameters in a separate computational structure or in advance, prior to the UAV-MEC system's actual operation. This allows the optimization to be conducted offline, avoiding real-time computational load on the UAVs. Additionally, pre-calculated parameter sets can be prepared for reference, enabling the system to adapt dynamically by switching to the corresponding set if operational parameters change during the mission. This approach eliminates the need for real-time GA execution, significantly reducing computational overhead while ensuring the system maintains performance. In addition, solutions to reduce GA time convergence have been proposed in the literature [36]. Thus, while GAs introduce overhead, these strategies harness optimization benefits without compromising the operational efficiency of resource-constrained UAVs.

6 Evaluation

The following lines describe the evaluation scenarios, convergence time of the proposed GA scheme and its results compared to three other approaches that are based on the first-fit strategy. The evaluation scenarios share the same parameter values such as service, failure and setup rates, as well as the same resource limits, but differ in terms of request loads, being characterized as Low, Mid and High. The goal is to assess the proposed scheme and the other strategies under different request rates. IN addition, the same power consumption values as in Table 3 were adopted and the remaining parameters are summarized in Table 7. The following subsection details the GA convergence results. For each point, 30 instances were performed and the average results are presented considering a 95 confidence level, which was obtained by the Bootstrap method, with 'resample' size and the number of (re)samplings equal to 30 and 1000, respectively. No bars were drawn due to a small difference between the upper and lower bounds.

6.1 GA Convergence

The GA convergence was examined with regard to the population's average fitness. To this end, an intermediate load fixed value $\lambda = 50$ was selected. We extended the GA's evolution process by adopting 150 generations to verify if the average fitness would significantly change after 50 generations (see Fig. 8). It was observed that the average fitness increases sharply in the first 25 generations as the GA explores the search space. In the next 75 generations (from the 25th approximately up to the 100th), the average fitness also rises but in a much longer slope, which indicates that the GA is refining already existing solutions. Lastly, from generation 100 onwards, the average fitness seems to become stable, with no significant changes taking place, denoting that the individuals from these populations have similar fitness values.

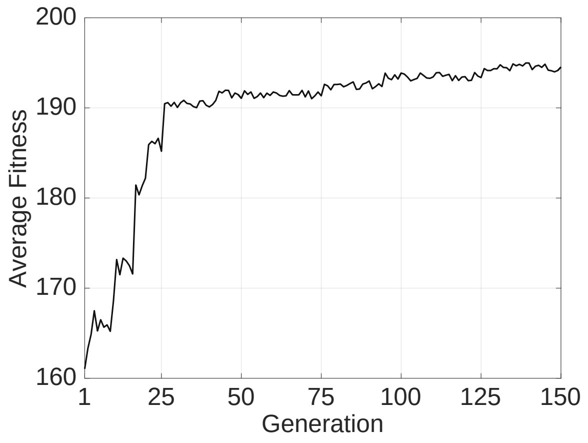


Fig. 8: GA Convergence Test

6.2 Evaluation Scenarios

The evaluation takes multiple network loads (λ) segmented in Low, Mid and High as follows. The request arrival rate of each channel was defined within the interval 10 to 30 for the low load, 40 to 60 for intermediate (mid) load and 80 to 120 for high range. For each of those load ranges, a total of 10 values of λ are drawn and thus, the

Table 7: Scheme Evaluation Parameters

Parameters	Value
VMs Range	[1,50]
CTs Range	[1,50]
Buffer Sizes	[10,20,30,40,50]
VM/CT Service rates	1
Overhead Degrad. Factor	0.01
Container Failure Rate	0.001
Container Setup Rate	0.1
λ (Low Range)	[10,30]
λ (Mid Range)	[40,60]
λ (High Range)	[80,120]
Reliability Threshold	0.9995
Response Time Threshold	5 ms

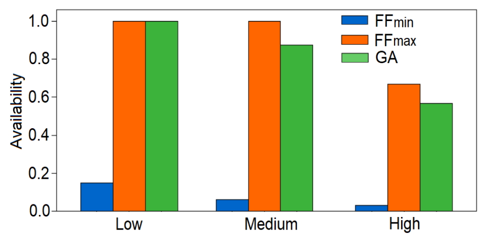
experiment is repeated 10 times. Then, the mean values for each performance metric (A,R,C and T) in each of these scenarios are calculated.

The remaining parameters are kept the same for each scenario allowing the evaluation of the system under different levels of stress conditions. The adopted parameters are as follows: container setup and failure rates α and γ are fixed 0.1 and 0.001, whereas the container service rate is fixed to 1. However, since the adoption of the degradation factor $d = 0.01$, the VM service rate is defined in Eq. 2, considering the baseline rate $\mu = 1$. Hence, its final value depends on the adopted number of VMs, which is varied between 1 and 50 with step of 1 unit. The same range and step are applied to the containers, while the buffer size varies between 10 and 50 with a step of 10. Thus, the size of the search state space if given by $5(50^2) = 12,500$ possible resource configurations. These parameters are summarized in Table 7. Please assume that the power consumption constants are the same as in Table 3.

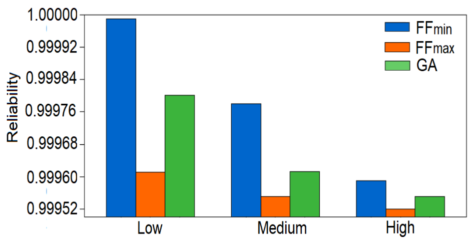
In addition to these different scenarios, we analyzed the effectiveness of our scheme by comparing it to schemes based on the first-fit strategy, similar to [26]. The First-Fit scheme (denoted as FF) tests each configuration possibility, i.e., tries out every possible configuration set from 1 VM, 1 CT and 10 buffer spaces up to 50 VMs, 50 CTs and 50 buffer spaces. The first tuple that overcomes the predefined reliability (R_{min}) and response time (T_{max}) limits is considered the final answer. Please note that no restrictions were defined for the Availability nor the Power Consumption.

The First Fit strategy was segmented in two alternatives, which differ in terms of the exploration order of the available state space. While the FF_{min} searches for the appropriate configuration departing from the minimum established values, i.e., VM = 1, CT = 1 and buffer size = 10, upgrading first the VMs, second the CTs and last the buffer dimension, the FF_{max} tries the opposite, i.e., starts the search from the maximum configuration values, i.e., VM = 50, CT = 50 and buffer size = 50, downgrading first the buffer variable, then the containers and only then the VM amounts. Although both may reach the same answer if only one matches the constraints, FF_{min} most likely results in configurations with a smaller number of resources, while FF_{max} should respond with more resources.

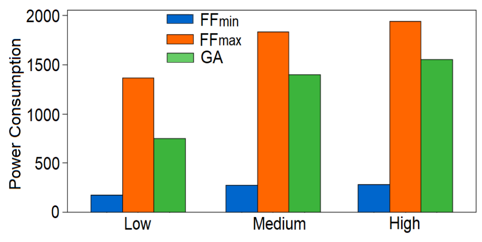
6.3 Numerical Results



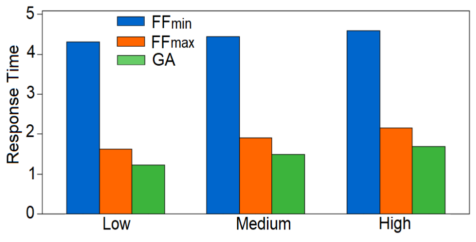
(a) Availability



(b) Reliability



(c) Power Consumption



(d) Response Time

Fig. 9: Optimization Strategy Comparison

Considering the Availability (Fig. 9a), it was noted great discrepancy among scenarios (Low, Medium and High loads) and adopted schemes (FF_{min}, FF_{max} and GA). The first reason for such a great gap between their results is the lack of the inferior bound for this metric. For instance, the FF_{min}, is free to select any given resource configuration as long as it produces at least the predefined reliability (R_{min}) and response time (T_{max}). Hence, this scheme resulted in poor availability values (under provisioning) for each scenario (14.98%, 5.96% and 2.98%, respectively). The same applies to the FF_{max}, except that it first handles large resource configurations, which will often result in over provisioned configurations, i.e, availability of 99.99%, 99.99% and 66.81%, respectively for low, medium and high loads. On the other hand, the GA scheme has a more balanced approach due to the design of its fitness function (27), achieving a similar result to the FF_{max} in the first scenario (low load), but dropping to 87.42% and 56.85% in the medium and high load scenarios, resulting in a difference of 12.58% and 9.96%, respectively. This difference was somehow expected since maximizing the Availability directly impacts conflicting metrics (e.g., power consumption), which should be minimized. Thus, the GA suggested least available configurations so as to balance off these other performance metrics.

With respect to the Reliability defined in Eq. 18, Fig. 9b shows that the FF_{min} outperforms both FF_{max} and the GA. Again, these results reflect the GA's balanced approach, which is similar to the FF_{min} in the low load scenario, whereas sits closer to the FF_{max} on the remaining metrics. The FF_{min} scheme is far superior in both low and medium scenarios since it activates the minimum container amounts, thus very few failures occur. Contrarily, the FF_{max} will usually respond with a higher number of containers, which is likely to result in more failures. Lastly, the high load scenario presents the least significant difference among the schemes, although there were great discrepancies in the Availability in the previous experiment (Fig. 9a). In other words, it means that each scheme opted for very different resource configurations, yet, since the Reliability is dependent on the amount of failure-prone containers and that the high load practically forces container activation, the failure probability among each scheme is likely to converge to the adopted inferior bound of 0.9995. However, this does not mean that schemes converge in the total number of containers since the Reliability only takes into account the accepted requests, not the overall arrival rate.

With regards to the scenario of MEC-enabled UAVs, the power consumption metric is by far the most interesting. Indeed, this evaluation (Fig. 9c) is closely related to the availability (Fig. 9a), and in most cases, the schemes will share similarities among both experiments. For instance, the least available scheme (FF_{min}) is also the one that consumes less energy, whereas FF_{max} is the opposite, being the largest consumer. However, the GA, which shares a similar availability to the FF_{max} in the low load scenario and differs 9.96% from that same scheme in the high load case, differs 44% and 19% in terms of power consumption for the same low and high scenarios, respectively. The reason for the large gap relies on the fact that FF_{max} stops at the first resource configuration that meets the Reliability and Response Time thresholds, thus, large resource amounts are suggested, while the GA tries multiple possibilities not only with respect to those limits but also minimizing power consumption.

The following lines are related to the Response Time metric (Fig. 9d). First, it was noted that the FF_{min} scheme responds with the highest delays, surpassing $4ms$ in each scenario, reaching $4.59ms$ in the high load case, which is close to the adopted superior bound of $5ms$. This denotes its design that most often will select configurations with a higher number of VMs, which in turn, is the main cause for higher Response Times due to its associated degradation factor d. On the contrary, both FF_{max} and GA respond with much faster response times, under $2ms$ in most cases. However, please note that the GA overcomes the FF_{max} in every scenario, with large differences of 24%, 42% and 21% for the low, medium and high load scenarios, respectively. Again, we believe that two factors are key: (1) the availability (Fig. 9a) and (2) the distribution of requests to each resource type. In the first case, the fact that the GA allows less requests to be processed in the medium and high load scenarios compared to the FF_{max}, with differences of 12.58% and 9.96%, respectively, allows gains in terms of response time of 42% and 21%, compared to the FF_{max}. On the latter, for the low load scenario, although both schemes share similar availability (close to 100%), The GA probably selects configurations with relatively more containers than VMs, which significantly decreases the overall response time (24%).

7 Conclusions and Future Directions

In this work, we proposed a CTMC-based framework for the virtualization layer of a MEC node, specifically designed to address the resource allocation challenges in UAV-enabled MEC environments while meeting the stringent requirements of URLLC. Our study explored a hybrid VM-containerized infrastructure, integrating the strong isolation properties of VMs with the flexibility of containers to optimize their combined benefits while aslo considering often overlooked factors such as virtual resource setup delays, failures, and computational power degradation of VMs on the same physical node, highlighting their critical impact on communication constraints. By emphasizing key metrics for URLLC applications, such as reliability and response times, we provided a holistic performance evaluation that also included availability and power consumption specific to the MEC-UAV environment, offering a comprehensive and holistic perspective on the framework's performance. Besides the framework and its analysis, we formulated a multi-objective problem related to MEC-enabled UAVs' static resource dimensioning, adopting a GA-based approach to solve it, resulting in balanced resource configurations that leverage the above-mentioned metrics. For future directions, the proposed multi-objective problem can be adapted to other bio-inspired or machine learning approaches and compared to state-of-the-art algorithms. Additionally, we plan to extend the framework by incorporating other network subparts (e.g., RAN), as well as trajectory and coverage characteristics commonly explored in UAV literature, which may impact multiple performance metrics such as the UAV mechanical consumption due to node mobility.

Acknowledgements. This work was supported by the National Council for Scientific and Technological Development (CNPq) Project No. 433142/2018-9, the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, the CNPq Research Productivity Fellowship (Grant No. 312831/2020-0), and the Pernambuco Research Foundation (FACEPE) (Grant No. IBPG-0096-1.03/16).

Declarations

- This paper is equally contributed by each author as everyone wrote a section of it. Besides, there was collaborative efforts in brainstorming the idea of this paper, proofread and formatting of this paper.
- The authors declare there is no conflict of interest.
- Ethics approval and consent to participate
- Consent for publication
- Data availability under request
- Code availability under request

References

[1] Prathyusha, Y., Sheu, T.-L.: Resource allocations for coexisting embb and urllc services in multi-uav aided communication networks for cellular offloading. IEEE Transactions on Vehicular Technology, 1–14 (2023) https://doi.org/10.1109/TVT.2023.3340654

[2] Ali, R., Zikria, Y.B., Bashir, A.K., Garg, S., Kim, H.S.: Urllc for 5g and beyond: Requirements, enabling incumbent technologies and network intelligence. IEEE Access **9**, 67064–67095 (2021) https://doi.org/10.1109/ACCESS.2021.3073806

[3] Zhao, L., Zhou, G., Zheng, G., Chih-Lin, I., You, X., Hanzo, L.H.: Open-source multi-access edge computing for 6g: Opportunities and challenges. ArXiv **abs/2111.11354** (2021)

[4] Ranjha, A., Javed, M.A., Piran, M.J., Asif, M., Hussien, M., Zeadally, S., Frnda, J.: Towards facilitating power efficient urllc systems in uav networks under jittering. IEEE Transactions on Consumer Electronics, 1–1 (2023) https://doi.org/10.1109/TCE.2023.3305550

[5] Tian, M., Li, C., Hui, Y., Cheng, N., Yue, W., Fu, Y., Han, Z.: On-demand multiplexing of embb/urllc traffic in a multi-uav relay network. IEEE Transactions on Intelligent Transportation Systems, 1–14 (2023) https://doi.org/10.1109/TITS.2023.3332022

[6] Zhang, Y., Zhao, L., Zheng, G., Chu, X., Ding, Z., Chen, K.-C.: Resource allocation for open-loop ultra-reliable and low-latency uplink communications in vehicular networks. IEEE Transactions on Vehicular Technology **70**(3), 2590–2604 (2021) https://doi.org/10.1109/TVT.2021.3061582

[7] Falcao, M., Souza, C.B., Balieiro, A., Dias, K.: An analytical framework for urllc in hybrid mec environments. The Journal of Supercomputing **78**, 2245–2264 (2021) https://doi.org/10.1007/s11227-021-03945-8

[8] Di, H., Zhu, X., Liu, Z., Tu, X.: Joint blocklength and trajectory optimizations for urllc-enabled uav relay system. IEEE Communications Letters **28**(1), 118–122 (2024) https://doi.org/10.1109/LCOMM.2023.3335655

[9] Morabito, R.: Power consumption of virtualization technologies: An empirical investigation. In: 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), pp. 522–527 (2015). https://doi.org/10.1109/UCC.2015.93

[10] Falcão, M., Souza, C., Balieiro, A., Dias, K.: Dynamic resource allocation for urllc in uav-enabled multi-access edge computing. In: 2023 Joint European Conference on Networks and Communications and 6G Summit (EuCNC/6G Summit), pp. 293–298 (2023). https://doi.org/10.1109/EuCNC/6GSummit58263.2023.10188346

[11] Xue, D., Guo, Y., Li, N., *et al.*: Cross-domain coordination of resource allocation and route planning for the edge computing-enabled multi-connected vehicles. Journal of Cloud Computing **12**(1), 33 (2023) https://doi.org/10.1186/s13677-023-00415-9

[12] Elgendy, I.A., Meshoul, S., Hammad, M.: Joint task offloading, resource allocation, and load-balancing optimization in multi-uav-aided mec systems. Applied Sciences **13**(4), 2625 (2023) https://doi.org/10.3390/app13042625

[13] Chen, J., Cao, X., Yang, P., Xiao, M., Ren, S., Zhao, Z., Wu, D.O.: Deep reinforcement learning based resource allocation in multi-uav-aided mec networks. IEEE Transactions on Communications **71**(1), 296–309 (2023) https://doi.org/10.1109/TCOMM.2022.3226193 . Online published 1 Dec 2022

[14] Safwat, N.E.-D., Hafez, I.M., Newagy, F.: 3d placement of a new tethered uav to uav relay system for coverage maximization. Electronics **11**(3) (2022)

[15] Zhao, M.-M., Shi, Q., Zhao, M.-J.: Efficiency maximization for uav-enabled mobile relaying systems with laser charging. IEEE Transactions on Wireless Communications **19**(5), 3257–3272 (2020) https://doi.org/10.1109/TWC.2020.2971987

[16] Cai, Y., Jiang, X., Liu, M., Zhao, N., Chen, Y., Wang, X.: Resource allocation for urllc-oriented two-way uav relaying. IEEE Transactions on Vehicular Technology **71**(3), 3344–3349 (2022) https://doi.org/10.1109/TVT.2022.3143174

[17] Iliev, T.B., Ivanova, E.P., Stoyanov, I.S., Mihaylov, G.Y., Beloev, I.H.: Artificial intelligence in wireless communications - evolution towards 6g mobile networks. In: 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), pp. 432–437 (2021). https://doi.org/10.23919/MIPRO52101.2021.9597147

[18] Do, Q.V., Pham, Q.-V., Hwang, W.-J.: Deep reinforcement learning for energy-efficient federated learning in uav-enabled wireless powered networks. IEEE Communications Letters **26**(1), 99–103 (2022) https://doi.org/10.1109/LCOMM.2021.3122129

[19] Pham, Q.-V., Zeng, M., Ruby, R., Huynh-The, T., Hwang, W.-J.: Uav communications for sustainable federated learning. IEEE Transactions on Vehicular Technology **70**(4), 3944–3948 (2021) https://doi.org/10.1109/TVT.2021.3065084

[20] Hou, X., Wang, J., Jiang, C., Zhang, X., Ren, Y., Debbah, M.: Uav-enabled covert federated learning. IEEE Transactions on Wireless Communications **22**(10), 6793–6809 (2023) https://doi.org/10.1109/TWC.2023.3245621

[21] Lu, Y., Huang, X., Zhang, K., Maharjan, S., Zhang, Y.: Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles. IEEE Transactions on Vehicular Technology **69**(4), 4298–4311 (2020) https://doi.org/10.1109/TVT.2020.2973651

[22] Moriai, S.: Privacy-preserving deep learning via additively homomorphic encryption. In: 2019 IEEE 26th Symposium on Computer Arithmetic (ARITH), pp.

198–198 (2019). https://doi.org/10.1109/ARITH.2019.00047

[23] Costanzo, F., Lorenzo, P.D., Barbarossa, S.: Dynamic resource optimization and altitude selection in uav-based multi-access edge computing. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4985–4989 (2020). https://doi.org/10.1109/ICASSP40776.2020.9053594

[24] Yang, Z., Pan, C., Wang, K., Shikh-Bahaei, M.: Energy efficient resource allocation in uav-enabled mobile edge computing networks. IEEE Transactions on Wireless Communications **18**(9), 4576–4589 (2019) https://doi.org/10.1109/TWC.2019.2927313

[25] Bekkouche, O., Samdanis, K., Bagaa, M., Taleb, T.: A service-based architecture for enabling uav enhanced network services. IEEE Network **34**(4), 328–335 (2020) https://doi.org/10.1109/MNET.001.1900556

[26] Emara, M., ElSawy, H., Filippou, M.C., Bauch, G.: Spatiotemporal dependable task execution services in mec-enabled wireless systems. IEEE Wireless Communications Letters **10**(2), 211–215 (2021) https://doi.org/10.1109/LWC.2020.3024749

[27] Kherraf, N., Alameddine, H.A., Sharafeddine, S., Assi, C.M., Ghrayeb, A.: Optimized provisioning of edge computing resources with heterogeneous workload in iot networks. IEEE Transactions on Network and Service Management **16**(2), 459–474 (2019) https://doi.org/10.1109/TNSM.2019.2894955

[28] Fautrel, T., George, L., Fauberteau, F., Grandpierre, T.: An hypervisor approach for mixed critical real-time uav applications. In: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 985–991 (2019). https://doi.org/10.1109/PERCOMW.2019.8730705

[29] Mavridis, I., Karatza, H.D.: Combining containers and virtual machines to enhance isolation and extend functionality on cloud computing. Future Gener. Comput. Syst. **94**, 674–696 (2019)

[30] Felter, W., Ferreira, A., Rajamony, R., Rubio, J.: An updated performance comparison of virtual machines and linux containers. In: 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 171–172. IEEE, ??? (2015)

[31] Pahl, C., Brogi, A., Soldani, J., Jamshidi, P.: Cloud container technologies: a state-of-the-art review. IEEE Transactions on Cloud Computing **7**(3), 677–692 (2017)

[32] Kulkarni, S.G., Liu, G., Ramakrishnan, K.K., Arumaithurai, M., Wood, T., Fu, X.: Reinforce: Achieving efficient failure resiliency for network function

virtualization-based services. IEEE/ACM Transactions on Networking **28**(2), 695–708 (2020) https://doi.org/10.1109/TNET.2020.2969961

[33] Kaur, K., Dhand, T., Kumar, N., Zeadally, S.: Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers. IEEE Wireless Communications **24**(3), 48–56 (2017) https://doi.org/10.1109/MWC.2017.1600427

[34] Anand, A., Veciana, G.: Resource allocation and harq optimization for urllc traffic in 5g wireless networks. IEEE Journal on Selected Areas in Communications **36**(11), 2411–2421 (2018) https://doi.org/10.1109/JSAC.2018.2874122

[35] Balieiro, A.M., Falcão, M., Dias, K.L.: An evolutionary scheme for secondary virtual networks mapping onto cognitive radio substrate. Wirel. Commun. Mob. Comput. **2019**, 1872765–1187276519 (2019) https://doi.org/10.1155/2019/1872765

[36] Chen, S., Newman, T.R., Evans, J.B., Wyglinski, A.M.: Genetic algorithm-based optimization for cognitive radio networks. In: 2010 IEEE Sarnoff Symposium, pp. 1–6 (2010). https://doi.org/10.1109/SARNOF.2010.5469780

[37] Benamer, A.R., Boussetta, K., Hadj-Alouane, N.B.: A genetic algorithm for the placement of latency-sensitive multiplayer game servers in the fog. In: 2021 IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2021). https://doi.org/10.1109/GLOBECOM46510.2021.9685952

[38] Ruiz, L., Durán, R.J., Miguel, I., Khodashenas, P.S., Pedreno-Manresa, J.-J., Merayo, N., Aguado, J.C., Pavón-Mariño, P., Siddiqui, S., Mata, J., Fernández, P., Lorenzo, R.M., Abril, E.J.: A genetic algorithm for vnf provisioning in nfv-enabled cloud/mec ran architectures. Applied Sciences (2018)

[39] A multi-objective genetic optimization for spectrum sensing in cognitive radio. Expert Systems with Applications **41**(8), 3640–3650 (2014) https://doi.org/10.1016/j.eswa.2013.12.010