# Enhancing QoS for IoT Devices through Enhancing QoS for IoT Devices through Enhancing QoS for IoT Devices through Heuristics-based Computation Offloading in Heuristics-based Computation Offloading in Heuristics-based Computation Offloading in Multi-access Edge Computing Multi-access Edge Computing Multi-access Edge Computing

Marouane Myyara, Oussama Lagnfdi, Anouar Darif, Abderrazak Farchane Marouane Myyara, Oussama Lagnfdi, Anouar Darif, and Abderrazak Farchane

*Abstract*—Multi-access Edge Computing (MEC) networks, **particularly with the advent of 5G, aim to reduce latency and**  particularly with the advent of 5G, aim to reduce latency particularly with the advent of 5G, aim to reduce latency **increase speed to meet the demands of resource-intensive ap-**and increase speed to meet the demands of resource-intensive and increase speed to meet the demands of resource-intensive **plications in the Internet of Things (IoT), such as private wire-**applications in the Internet of Things (IoT), such as private wireapplications in the Internet of Things (IoT), such as private wire-**less networks, online gaming, industry, and remote healthcare.**  less networks, online gaming, industry, and remote healthcare. less networks, online gaming, industry, and remote healthcare. **These applications require guaranteed performance. However,**  These applications require guaranteed performance. However, These applications require guaranteed performance. However, **while Quality of Service (QoS) management is well established**  while Quality of Service (QoS) management is well established while Quality of Service (QoS) management is well established<br>in the Claud, immersive it nomeine a shallonge in MEC on riin the Cloud, improving it remains a challenge in MEC en-**in the Cloud, improving it remains a challenge in MEC en- vi-**in the Cloud, improving it remains a challenge in MEC en**ronments. This study addresses this challenge by proposing heu-**<br>virting aggregated by a line adoption for Li<sup>n</sup>istancian derivative heuristic computation offloading algorithms for IoT-intensive **ristic computation offloading algorithms for IoT-intensive devic-**heuristic computation offloading algorithms for IoT-intensive es in MEC networks. These algorithms aim to minimize service time while maximizing the QoS, taking into account tasks and resource characteristics to determine the optimal execution location for IoT device applications. We evaluated our approach using the EdgeCloudSim simulator, and the results demonstrate **using the EdgeCloudSim simulator, and the results demonstrate**  using the EdgeCloudSim simulator, and the results demonstrate its superiority over existing solutions. Our approach significantly **its superiority over existing solutions. Our approach significant-**its superiority over existing solutions. Our approach significantly improves QoS by reducing the service time of IoT application **ly improves QoS by reducing the service time of IoT application**  improves QoS by reducing the service time of IoT application tasks. This research fills a gap in efficient QoS improvement and **tasks. This research fills a gap in efficient QoS improvement and**  tasks. This research fills a gap in efficient QoS improvement and contributes to advances in computation offloading strategies in **contributes to advances in computation offloading strategies in**  contributes to advances in computation offloading strategies in MEC environments. It paves the way for enhanced performance **MEC environments. It paves the way for enhanced performance**  MEC environments. It paves the way for enhanced performance of IoT applications in these networks. **of IoT applications in these networks.** of IoT applications in these networks. *Abstract***—Multi-access Edge Computing (MEC) networks,**  *Abstract*—Multi-access Edge Computing (MEC) networks,

*Index Terms* - MEC, IoT, Computation Offloading, Quality of Service, Heuristic Algorithms, EdgeCloudSim. **Service, Heuristic Algorithms, EdgeCloudSim.**

## I. INTRODUCTION I. INTRODUCTION

The Internet of Things (IoT) is a rapidly expanding ecosys-The Internet of Things (IoT) is a rapidly expanding ecosystem of diverse physical objects connected through various tem of diverse physical objects connected through various networks, both wired and wireless [1]. It enhances internet networks, both wired and wireless [1]. It enhances internet utilization by linking mobile devices and sensors. However, utilization by linking mobile devices and sensors. However, IoT faces significant challenges, including high network la-IoT faces significant challenges, including high network latency, availability, and mobility, often mitigated through Cloud tency, availability, and mobility, often mitigated through Cloud Computing [2]. Resource-constrained IoT devices struggle Computing [2]. Resource-constrained IoT devices struggle with limited processing power, memory, and battery life, with limited processing power, memory, and battery life, complicating the execution of complex tasks. Additionally, the complicating the execution of complex tasks. Additionally, the surge in IoT devices can overload networks accessing Cloud surge in IoT devices can overload networks accessing Cloud servers, hindering low-latency and high-capacity applications. servers, hindering low-latency and high-capacity applications. To address these issues, edge computing paradigms, such as To address these issues, edge computing paradigms, such as Mobile/Multi-access Edge Computing (MEC), have emerged. Mobile/Multi-access Edge Computing (MEC), have emerged.

MEC, introduced by the European Telecommunications MEC, introduced by the European Telecommunications Standards Institute (ETSI) [3], enhances edge intelligence and Standards Institute (ETSI) [3], enhances edge intelligence and boosts processing and storage capabilities [4]. By bringing boosts processing and storage capabilities [4]. By bringing cloud functionalities closer to the Radio Access Network cloud functionalities closer to the Radio Access Network (RAN), it provides ultra-low latency and network context (RAN), it provides ultra-low latency and network context

Manuscript received April 19, 2005; revised August 26, 2015. Manuscript received April 19, 2005; revised August 26, 2015. Manuscript received April 19, 2005; revised August 26, 2015.

DOI: [10.36244/ICJ.2024.4.2](https://doi.org/10.36244/ICJ.2024.4.2)

awareness. ETSI identifies TOT as a key use case for MEC  $_1$ <sup>1</sup>,  $_2$ <sup>1</sup>, emphasizing the mutual benefits of their integration [2]. From  $\text{Lip}$  the MEC perspective, IoT expands MEC services to various the MEC perspective, IoT expands MEC services to various devices, while from the IoT viewpoint, MEC architecture devices, while from the IoT viewpoint, MEC architecture offers computing resources closer to users. This integration significantly aids resource-constrained IoT devices by prosignificantly and resource-constrained IoT devices by providing access to powerful computing at the network edge, enabling efficient task execution and improved service quality. enabling efficient task execution and improved service quality. According to [5], this integration provides three main benefits: recording to [5], this integration provides three main benefits:<br>reduced infrastructure traffic, lower application latency, and reduced infrastructure traffic, lower application latency, and scalable network services. The key advantage is decreased scalable network services. The key advantage is decreased latency through MEC, which shortens distances and translatency through MEC, which shortens distances and trans-mission times between resources, facilitating efficient resource mission times between resources, facilitating efficient resource<br>provision for processing IoT applications [6]. awareness. ETSI identifies IoT as a key use case for MEC [4],

provision for processing for applications [6].<br>Despite growing interest in MEC for IoT, research on Despite growing interest in MEC for IoT, research on computation offloading remains limited. Recent studies focus computation offloading remains limited. Recent studies focus on reducing latency in MEC networks [7], [8], [9], but on reducing latency in MEC networks [7], [9], [9], but offloading for resource-constrained IoT devices has received insufficient attention. Effective offloading strategies can lower Insufficient attention. Effective offloading strategies can lower latency, enhance service quality, and reduce reliance on cenlatency, enhance service quality, and reduce reliance on cen-tralized cloud systems. Optimizing MEC resources involves tralized cloud systems. Optimizing MEC resources involves managing limited server capacity to minimize execution delays managing immed server capacity to minimize execution delays and improve user experience. Fair allocation mechanisms are essential due to the diverse interests of IoT users and are essential due to the diverse interests of IoT users and edge servers. The dynamic nature of these systems highlights edge servers. The dynamic nature of these systems highlights the need for ongoing research to develop robust resource the need for ongoing research to develop robust resource allocation methods that maximize edge computing's potential<br>and factor innovation and foster innovation.

and foster innovation. This study aims to develop a heuristic-based offloading This study aims to develop a heuristic-based offloading strategy that optimizes task execution time and improves strategy that optimizes task execution time and improves QoS. This paper presents a novel heuristic-based offloading QoS. This paper presents a novel heuristic-based offloading strategy that addresses the specific challenges faced by IoT strategy that addresses the specific challenges faced by IoT<br>devices. The paper's structure includes a review of related devices. The paper's structure includes a review of related work (Section II), the system model and problem formulation work (Section II), the system model and problem formulation<br>(Section III), the proposed heuristic-based offloading strat-(Section III), the proposed heuristic-based offloading strat-egy (Section IV), performance evaluation through simulation egy (Section IV), performance evaluation through simulation results (Section V), and a conclusion with future research perspectives (Section VI). perspectives (Section VI).

# II. RELATED WORK II. RELATED WORK

The integration of IoT with MEC provides significant benefits, such as ultra-low latency, real-time data analytics, benefits, such as unta-low latency, real-time data analytics, improved resource management, increased capacity, and enimproved resource management, increased capacity, and en-hanced scalability. By localizing computing capabilities, MEC hanced scalability. By localizing computing capabilities, MEC reduces bandwidth needs and reliance on central Clouds, reduces bandwidth needs and reliance on central Clouds, minimizing data transfers to remote data centers. minimizing data transfers to remote data centers.The integration of IoT with MEC provides significant

M. Myyara, O. Lagnfui, A. Darif, and A. Farchane are arithmed with the Laboratory of Innovation in Mathematics, Applications, and Information the Laboratory of Innovation in Mathematics, Applications, and Informa-Technology, Polydisciplinary Faculty, Sultan Moulay Slimane University, Beni tion Technology, Polydisciplinary Faculty, Sultan Moulay Slimane Univerreemology, Polydisciplinary Faculty, Sultan Moulay Silinane University, Beni Mellal, 23000, Morocco. (E-mails: marouane.myyara@u[sms.ac.ma,](mailto:lagnfdi.o%40gmail.com?subject=) lagnfdi.o@ menar, 23000, Morocco. (E-mails: marouane.myyara@usms.ac.ma, hagmun.o@[gmail.com, anouar.darif@](mailto:anouar.darif%40gmail.com?subject=)gmail.com, a.farchane@gmail.com)  $\mu$  mail.com, anouar.darif  $\approx$  gmail.com, and a.farchane $\approx$  gmail.com/<br>Monuscript received April 10, 2005; revised August 26, 2015 M. Myyara, O. Lagnfdi, A. Darif, and A. Farchane are affiliated with the M. Myyara, O. Lagnfdi, A. Darif, and A. Farchane are affiliated with

Researchers have explored collaborative computation of-<br>floading and resource allocation schemes to enhance task floading and resource allocation schemes to enhance task processing efficiency in MEC systems. MEC facilitates effi-<br>cient computation, load balancing, and latency reduction by cient computation, load balancing, and latency reduction by migrating tasks to resource-rich infrastructures [10]. Recent studies include dynamic offloading algorithms that optimize<br>user experience by minimizing service time and balancing user experience by minimizing service time and balancing workloads [11], and a deadline-aware scheduling algorithm that reduces execution time for critical tasks while considering<br>task type and weight [12]. While many studies focus on task type and weight [12]. While many studies focus on optimizing task processing time, they often neglect the broader concept of service time, which encompasses both processing<br>and transmission delays. This paper advances the field by and transmission delays. This paper advances the field by introducing a novel algorithm that incorporates each task's deadline and latency tolerance to minimize service time while meeting QoS requirements.

Several research efforts have focused on optimizing computation offloading in MEC to enhance IoT device performance. One approach is the Lagrange duality resource optimization algorithm [13], which improves task offloading and resource allocation compared to traditional methods like random of-<br>floading and load balancing. This highlights the importance of floading and load balancing. This highlights the importance of efficient processing for real-time IoT applications, addressing service time and QoS requirements. A notable study [14] presents a collaborative computing framework that enables devices to partially process tasks across terminals, edge servers, and the Cloud using a pipeline-based offloading scheme. Additionally, various models have been developed to reduce latency and improve system efficiency, including an algorithm specifically designed to minimize execution latency  $[15]$ .

Effective joint resource management between MEC and the central Cloud is crucial for meeting the service demands of IoT applications, particularly given the limited capabilities of edge devices compared to Cloud infrastructures [16]. Recent<br>research has focused on MEC network workload orchestration research has focused on MEC network workload orchestration and resource allocation strategies to improve IoT application performance  $[17]$ . For instance,  $[18]$  explores computation<br>offloading and bandwidth distribution in IoT networks using offloading and bandwidth distribution in IoT networks using graph-based models for resource optimization. Heuristic meth- $\overline{\text{cos}}$ , such as the iterative heuristic mobile edge computing resource allocation algorithm [19], aim to enhance efficiency and minimize latency.

Despite advancements in MEC and IoT integration, challenges specific to resource-constrained IoT devices remain unlenges specific to resource-constrained IoT devices remain un-<br>addressed. Our primary objective is to minimize task execution<br>time, while, considering, computing, resource, constraints, and time while considering computing resource constraints and application requirements. Unlike prior studies, our research focuses on optimizing QoS for end IoT devices within an MEC framework. We propose a heuristic-based strategy for while maximizing task execution efficiency, reducing latency, and effectively utilizing computing resources. offloading and resource allocation that adheres to constraints

### $T$  mech system model, i.e. feature  $\mathbb{R}^n$ **III. SYSTEM MODEL AND PROBLEM FORMULATION**

### *A.* System Model

The MEC system model, illustrated in Figure 1, features a three-tier architecture: central cloud, MEC servers, and

wirelessly connected IoT devices. IoT devices can offload<br>resource-intensive computations to MEC servers or the cloud resource-intensive computations to MEC servers or the cloud for efficient processing. Each MEC server is linked to a wireless access point or base station, covering a specific area and serving IoT users. These servers, equipped with sufficient hardware resources, connect to the Edge Orchestrator (EO) via a backhaul link, which manages infrastructure resources,<br>server states, and capacities. Positioned close to users, the<br>servers connect to the EO through a MAN. The central cloud<br>consists of high-capacity servers provid server states, and capacities. Positioned close to users, the servers connect to the EO through a MAN. The central cloud consists of high-capacity servers provided by cloud service providers, accessible via a WAN network linking the EO to the upper layer. The offloading process begins with the device and is guided by the EO, considering workload distribution, computing resources, and network conditions.



Fig. 1: Multi-layer Multi-access Edge Computing architecture.

### **B.** Notation and Variables

Let  $M$  represent the set of MEC nodes,  $C$  denote the Central Cloud, and  $\mathcal D$  the set of IoT devices, with  $IoT_i$  indicating device *i*. Each MEC node  $m \in \mathcal{M}$  is associated with a set of Virtual Machines (VMs). The computing capacity of each VM, Virtual Machines (VMS). The computing capacity of each VM, denoted by  $\mathcal{F}$ , is measured in MIPS (Millions of Instructions denoted by  $\mathcal F$ , is ineasured in MIPS (Willions of Instructions<br>Per Second). Each IoT device i has one or more computation thas the second). Each for device  $\ell$  has one of more computation tasks, represented by  $\mathcal{T}_i$ , with each task  $\tau_{i,j}$  characterized by a length  $\mathcal{L}_{i,j}$ , indicating the data generated for the task. by a length  $\mathcal{L}_{i,j}$ , malcating the data generated for the task.<br>The computational capacity required for task j from device The computational capacity required for task  $j$  from device  $i$  is  $C_{i,j}$ . These parameters are defined in an XML file in  $\tau$  is  $C_{i,j}$ . These parameters are defined in an AML file in the EdgeCloudSim simulator [20], allowing for customization based on application characteristics. tasks, represented by  $\mathcal{T}_i$ , with each task  $\tau_{i,j}$  characterized The computational capacity required for task  $\gamma$  from device the EdgeCloudSim simulator [20], allowing for customization<br>hased on application characteristics

### C. Computation Offloading Model  $\mathbf{b}$ based on application characteristics.

In a multi-layer MEC environment, computation tasks can be executed locally on the IoT device, offloaded to MEC **C. COMPUTATION COMPUTATION** COMPUTED IN THE COMPUTATION OF LATEX SERVERS, OR Offloaded to central Cloud servers. The computation servers, or offloaded to central Cloud servers. The computation<br>task offloading involves two sets of optimization variables:

• Binary Variable A: This variable is defined as  $A = \{\alpha_{i,j} \mid$  $i \in \mathcal{D}, j \in \mathcal{T}_i$ . Here,  $\alpha_{i,j}$  takes a value of 0 if task data  $\tau_{i,j}$  is offloaded, and 1 if it is executed locally.

### Enhancing QoS for IoT Devices through Heuristics-based based of data transference of data transfer of data transfer of data transfer of da Emanong doo for to Poonsoo imager Homensies sassa<br>Computation Offloading in Multi-access Edge Computing omputation Offloading in Multi-access Edge Computing<br>.

• Binary Variable B: This variable is defined as  $B = \{\beta_{i,j}\}\$  $i \in \mathcal{D}, j \in \mathcal{T}_i$ . Here,  $\beta_{i,j}$  equals 1 if the task is offloaded to an MEC server, and  $\overrightarrow{0}$  if it is offloaded to the Cloud. to an MEC server, and 0 if it is offloaded to the Cloud. Binary variable B: This variable is defined as  $B = \{b_{i,j}\}\in \mathcal{D}$ ,  $i \in \mathcal{T}$ ). Here,  $\beta$  couple 1 if the task is offloaded  $i \in \mathcal{D}, j \in \mathcal{T}_i$ . Here,  $\beta_{i,j}$  equals 1 if the task is offloaded • Binary Variable B: This variable is defined as  $\mathcal{B} = \{\beta_{i,j}\}\$ 

# *D. Task Computation Model*

In this section, we focus on the task computation models in  $\cdot$  MEC architecture providing estimations for determining local processing times and remote processing times. *D. Task Computation*, provided this section, we rocus on the task computation models in the MEC architecture, providing estimations for determining<br>local processing times and remote processing times.

1) Local Computation: Local computation processes tasks directly on the user's IoT device, resulting in low latency directly on the user's IoT device, resulting in low latency<br>since data does not need to be transferred to a remote server. However, if the task size is large or the device's computing<br>consider in limited, offloading may be necessary. Let  $T$ capacity is limited, offloading may be necessary. Let  $\mathcal{F}_{IoT_i}$ <br>denote the computing power of the IoT device *i* in MIPS. The denote the computing power of the IoT device i in MIPS. The overall service time for local computation denoted as  $T^{IoT}$ . overall service time for local computation, denoted as  $\tau_{\tau_{i,j}}$ , can be calculated as: denote the computing power of the IoT device *i* in MIPS. The overall service time for local computation, denoted as  $T_{\tau_{i,j}}^{IoT}$ , can be calculated as: overall service time for local computation, denoted as  $T_{\tau_{i,j}}^{I \circ T}$ , can be calculated as:

$$
T_{\tau_{i,j}}^{IoT} = \frac{\mathcal{L}_{i,j}}{\mathcal{F}_{IoT_i}}
$$
\n(1)

The local computation delay depends on task size and computing power. Larger tasks or weaker computing power result in longer delays. Local computation has no transmission delays as data is processed on the user's IoT device.

2) Computation in MEC Servers: Computation in MEC servers leverages proximity to end-users to achieve low latency. Offloading tasks to nearby MEC servers typically reduces service time compared to local computation. However, task execution on MEC servers incurs delays due to wireless transmission from IoT devices. The total execution time on an MEC server includes both transmission and computation<br>delays  $I_{\text{tot}} \nabla$  denote the computing nature of the up the an MEC server includes both transmission and computation<br>delays. Let  $\mathcal{F}_{\text{MEC}_m}$  denote the computing power of the *m*-th<br>MEC server. The execution time of task  $\tau_{\text{tot}}$  on MEC server MEC server. The execution time of task  $\tau_{i,j}$  on MEC server MEC server. The execution time of task  $\tau_{i,j}$  on MEC server m can be expressed as:  $m$  can be expressed as:

$$
T^{m}_{\text{Com}(i,j)} = \frac{\mathcal{L}_{i,j}}{\mathcal{F}_{\text{MEC}_{m}}} \tag{2}
$$

The total delay, considering both transmission and compu-The total delay, considering boar dalasmission and comparately computation results from the IoT layer to the MEC server<br>through the wireless channel can be expressed as:  $\frac{1}{\pi}$  through the wireless channel can be expressed as:  $\mathbf{t}$ The total delay of the transmission and computer and computer transmission and computer and computer through the wireless channel can be expressed as:  $\frac{1}{\sqrt{2}}$ 

$$
T_{\tau_{i,j}}^{\text{MEC}_m} = T_{\text{Com}(i,j)}^m + T_{\text{Up}(i,j)}^m + T_{\text{Dw}(i,j)}^m \tag{3}
$$

where  $T^m_{\text{Com}(i,j)}$  represents the computation time for task j on MEC server m,  $T_{\text{Up}(i,j)}^m$  represents the upload data transfer time, and  $T_{\text{Dw}(i,j)}^m$  represents the download data transfer time from the MEC server back to the IoT device.  $T_{\text{conv}(i,j)}^m$  represents the computation time for task just the upload data transfer ne, and  $T^{m}_{\text{Dw}(i,j)}$  represents the download data transfer time where  $T_c^m$   $\ldots$  represents the computation time  $T_{\text{Tw}}^{m}$  is  $T_{\text{Up}(i,j)}^{m}$  represents the download data transfer time where  $T_{\alpha}^{m}$   $\ldots$  represents the computation time for task *i* where  $T_{\text{Com}(i,j)}$  represents the computation time for task from MEC server  $m$ .  $T_{\text{con}}^m$   $\omega$  a represents the upload data transfer from the server  $m$ ,  $\text{Lip}(i,j)$  represents the approximation  $\frac{3}{2}$   $\sum_{i=1}^N (i,j)$  represents the download data transfer time where  $T^m_{\text{Com}(i,j)}$  represents the computation time for task j on MEC server m,  $T_{Up(i,j)}^m$  represents the upload data transfer time, and  $T_{\text{Dw}(i,i)}^m$  represents the download da where  $T^m$  represents the computation time for task  $\phi$ where  $T_{\text{Com}(i,j)}^{\text{co}}$  represents the computation time for task j on MEC server  $m$ ,  $T_{\text{Up}(i,j)}^m$  represents the upload data transfer time<br>time, and  $T^m$  represents the download data transfer time the  $I_{\text{DW}(i,j)}$  represents the download data

*3) Computation in Cloud Servers:* Computation in Cloud *Servers*: Computation in Cloud 3) Computation in Cloud Servers: Computation in Cloud servers offers high processing power and storage capacity but<br>ineurs bigher leteney due to the distance of data transfer When incurs higher latency due to the distance of data transfer. When MEC servers cannot process offloaded tasks promptly, they are sent to the Cloud server over the wireless network. The total delay in Cloud server computation consists of transmission and processing delays.  $\beta$ ) Computation in Cloud Servers: Computation in Cloud<br>servers offers high processing power and storage capacity but servers duck to the form device.<br>
Subsequently but the capacity of the capacity of the incurs higher latency due to the distribution of distance of distance of details between servers of details between and storage capacity but servers offers ingit processing power and storage capacity out<br>incurs higher latency due to the distance of data transfer. When incurs higher latency due to the distance of data transfer. When<br>MEC servers cannot process offloaded tasks promptly, they are and the Cloud server computation consists of the consistence of the viribulation consistent to the Cloud server computation consists of the viribulation consists of the viribulation consists of the consists of the consiste delay in Cloud server computation consists of transmission<br>delay in Cloud server computation consists of transmission and proce

e computation time in the Cloud is given by: lefined as  $B = \{\beta_{i,j} \mid$  The computation time in the Cloud is given by: server in MIPS. Similar to the MIPS. Similar to the MIPS. Similar to the MIPS.

$$
T_{\text{Com}(i,j)}^{cloud} = \frac{\mathcal{L}_{i,j}}{\mathcal{F}_{\text{Cloud}}}
$$
(4)

Where  $\mathcal{F}_{\text{Cloud}}$  represents the computing power of the Cloud server in MIPS. Similar to the MEC server case, there is a transmission delay for uploading and downloading data,  $T_{Up(i,j)}$  and  $T_{DW(i,j)}$  respectively. The total density computation time and the transmission delay: denoted as  $T_{\rm N}^{cloud}$  and  $T_{\rm N}^{cloud}$  respectively. The total delay for offloading a task to the Cloud server is  $\mathcal{L}$  ,  $\mathcal{L}$  to  $\mathcal{L}$ denoted as  $T_{\text{Up}(i,j)}^{cloud}$  and  $T_{\text{DW}(i,j)}^{cloud}$  respectively. The total delay for offloading a task to the Cloud server is the sum of the  $r_{\text{Com}(i,j)}$   $\mathcal{F}_{\text{Cloud}}$ <br>Where  $\mathcal{F}_{\text{Cloud}}$  represents the computing power of the Cloud ion delay for uploading and downloading data,<br>Tcloud and Tcloud, magnestively. The total data, r offloading a task to the Cloud server is the sum of the  $T_{lln(d)}^{cloud}$  and  $T_{l(n(d))}^{cloud}$  respectively. The total delay r of those a fask to the Cloud server is the sum of the Where  $\mathcal{F}_{\text{Cloud}}$  represents the computing power of the Cloud  $T_{\text{Up}(i,j)}^{cloud}$  and  $T_{\text{DW}(i,j)}^{cloud}$  respectively. The total delay computation, MEC servers, and Cloud servers, indicating server

$$
T_{\tau_{i,j}}^{\text{Cloud}} = T_{\text{Com}(i,j)}^{\text{cloud}} + T_{\text{Up}(i,j)}^{\text{cloud}} + T_{\text{Dw}(i,j)}^{\text{cloud}} \tag{5}
$$

Figure 2 illustrates task computation models for local computation, MEC servers, and Cloud servers, indicating task offloading and allocation in the MEC architecture. It's important to note that transmission delay in the Cloud server case is typically longer due to distance and potential network congestion.  $\epsilon_{r,j}$  and  $\epsilon_{r,j}$  denote an and potential network and potential networks and potential networks and potential networks. Figure 2 illustrates task computation models for local



Fig. 2: An illustration of Computation Offloading type in MEC networks

# networks Fig. 2: An illustration of Computation Offloading type in MEC *E. Problem Formulation E. Problem Formulation E. Problem Formulation*

The objective of this study is to minimize the execution time of computing tasks in an MEC system by optimizing task of computing tasks in an MEC system by optimizing task *E. Problem Formulation* offloading and resource allocation. We consider application offloading and resource allocation. We consider application<br>constraints, available computing capacities, and resources. Our problem is formulated as a minimization problem with an objective function that incorporates task constraints, resource capacities, and offloading decision variables: problem is formulated as a minimization problem with an problem is formulated as a minimization problem with an constraints, available computing capacities, and resources. Our objective function that incorporates task constraints, resource objective function that incorporates task constraints, resource objective function that incorporates task constraints, resource capacities, and offloading decision variables: capacities, and offloading decision variables: capacities, and offloading decision variables: offloading and resource allocation. We consider application problem is formulated as a minimization problem with an

Minimize 
$$
\mathcal{P} = \sum_{i=1}^{\mathcal{D}} \sum_{j=1}^{T_i} \sum_{m=1}^{\mathcal{M}} \alpha_{i,j} T_{\tau_{i,j}}^{IoT} + (1 - \alpha_{i,j}) \times (6)
$$

$$
\left(\beta_{i,j} T_{\tau_{i,j}}^{\text{MEC}_{m}} + (1 - \beta_{i,j}) T_{\tau_{i,j}}^{\text{Cloud}}\right)
$$

Subject to:

$$
\alpha_{i,j}, \beta_{i,j} \in \{0, 1\} \quad, \forall i \in \mathcal{D}, \forall j \in \mathcal{T}_i \quad (6a)
$$

$$
\sum_{i=1}^{D} \sum_{j=1}^{I_i} (1 - \alpha_{i,j}) \times \beta_{i,j} C_{i,j} \le \sum_{m=1}^{M} \mathcal{F}_{\text{MEC}_m} \qquad (6b)
$$

$$
\sum_{i=1}^{\mathcal{D}} \sum_{j=1}^{\mathcal{T}_i} (1 - \alpha_{i,j})(1 - \beta_{i,j}) C_{i,j} \leq \mathcal{F}_{\text{Cloud}} \quad (6c)
$$

$$
\sum_{i=1}^{\mathcal{D}} \sum_{j=1}^{T_i} \alpha_{i,j} C_{i,j} \le \sum_{i=1}^{\mathcal{D}} \mathcal{F}_{I \circ T_i} \qquad (6d)
$$

 $\overline{\mathcal{O}}$ 

$$
\sum_{i=1}^{D} \sum_{j=1}^{T_i} \sum_{m=1}^{M} (1 - \alpha_{i,j}) \beta_{i,j} \mathcal{F}_{\text{MEC}_m} \le \sum_{m=1}^{M} Th_{\text{MEC}_m} \qquad (6e)
$$

The problem is subject to the following constraints: Constraint (6a) ensures that the decision variables  $(\alpha_{i,j}, \beta_{i,j})$  are binary. Constraint (6b) guarantees that the server's computing capacity is sufficient if a task is offloaded to an MEC server. constraint (6c) verifies that the capacity is adequate for tasks offloaded to the cloud server. Constraint  $(6d)$  ensures the device's computing capacity is sufficient for local execution. Finally, constraint (6e) restricts MEC server resource utilization to a specified threshold  $Th_{MEC_m}$ , typically set around 80%.

By addressing this optimization challenge, we can identify The specified threshold *The ThMECHMECH*, threshold *ThmECCM*, the sum set around 80%. that minimize total execution time while adhering to MEC effective task of the constraints. This problem can be formulated as an The construction of the processes of the MEC and the M state constraints. This problem can be formulated as an optimal solutions. This problem can be formulated as an optimal solutions. This problem can be formulated as an optimal com-Integer Linear Programming (ILP) can called inger Linear Propriation for large-scale promally computations can under impromal solutions computations for the computations of the computatio plexity, contracting and the contraction in please the complication developed, balancing solution quality with computational ef- $\frac{1}{2}$  applications. These methods enable real-time offloading decisions. developed, balancing systems and resource optimization associated with ILP, ensuring system agility in managing dynamic task arrivals. tem agility in managing dynamic task arrivals.

### IV. HEURISTIC-BASED OFFLOADING STRATEGY FOR QOS IMPROVEMENT

We propose a Heuristic-based Offloading strategy to enhance  $QoS$  (HOQoS) in MEC environments. Our goal is to mental versus (120 Qees) of the propose and the measurements. The game is to reparaments constraints, resource limitations, and application sidering time constraints, resource limitations, and application optimize of the computing decisions for computing the computing decisions of the contion space to make optimal decisions for each task, factoring in requirements. Utilizing greedy heuristics, we explore the soluthe interaction interaction in the matrix of the matrix of the matrix of the matrix of  $\alpha$  and task of  $\alpha$  of  $\beta$  and  $\beta$  and  $\alpha$  optimal decisions for each task, for each task, for each task, for each task,  $\alpha$  $\frac{p}{q}$  increasing the impute the execution of the matrix architecture layers. We propose a Heuristic-based Offloading strategy to en-

A key component of our strategy is the Offloading Decision Variable Identification (ODVI) algorithm (Algorithm 1), which determines the best offloading decision  $(\alpha_{i,j}, \beta_{i,j})$  for each incoming task  $\tau_{i,j}$ . The algorithm initializes the minimum  $\frac{d}{dx}$  and  $\frac{d}{dx}$ ,  $\frac{d}{dx}$ ,  $\frac{d}{dx}$  are algorithm minimized are infinitionally decision (and  $\frac{d}{dx}$ ) for each  $\frac{d}{dx}$ task, it calculates the local execution time  $T_{\tau_{i,j}}^{IoT}$  and estimates delays on MEC servers  $T_{\tau_{i,j}}^{\text{MEC}_{m}}$  and in the Cloud  $T_{\tau_{i,j}}^{\text{Cloud}}$ . The  $\frac{1}{100}$  in the algorithm in the minimum initializes the minimum initializes the minimum  $\frac{1}{100}$  and estimates the as integrates are continuous variables  $\tau_{i,j}$  and the decision variables as  $\tau_{i,j}$ tackles on MLC servers  $\mathcal{I}_{\tau_{i,j}}$  and in the Cloud  $\mathcal{I}_{\tau_{i,j}}$ . The task, it calculates the local execution time  $T_{\tau_{i,j}}^{IoT}$  and estimates delays on MEC servers  $T_{\text{m}}^{\text{MEC}_{m}}$  and in the Cloud  $T_{\text{m}}^{\text{Cloud}}$ . The task, it calculates the local execution time TIoT <sup>τ</sup>i,j and estimates task is assigned to the server with the shortest delay  $T_{\text{min}}$ , while respecting the maximum acceptable delay constraint. The decision variables  $(\alpha_{i,j}, \beta_{i,j})$  are then based on the chosen<br>The decision variables  $(\alpha_{i,j}, \beta_{i,j})$  are then based on the chosen The decision variables  $(\alpha_{i,j}, \beta_{i,j})$  are then based on the enosch optimal location, focusing on individual tasks rather than overall execution time.

Reference to Different Assemblance Ensure: (αi,j , βi,j ) Offloading decision variables Algorithm 1 Offloading Decision Variable Identification Algorithm 1 Offloading Decision Variable Identification (ODVI) (ODVI)

Require:  $(M, C, D, \mathcal{T}_i)$ **Ensure:**  $(\alpha_{i,j}, \beta_{i,j})$  Offloading decision variables 1:  $T_{\text{Min}(i,j)} \leftarrow \infty$ 2: Decision variables  $(\alpha_{i,j}, \beta_{i,j}) \leftarrow \emptyset$ 3: for  $i \in \mathcal{D}$  do 4: **for**  $j \in \mathcal{T}_i$  **do** 5: for  $m \in \mathcal{M}$  do 6:  $T_{\text{Min}(i,j)} \leftarrow \min(T_{\text{Min}(i,j)}, T_{\tau_{i,j}}^{IoT}, T_{\tau_{i,j}}^{\text{MEC}_m}, T_{\tau_{i,j}}^{\text{Cloud}})$  $\frac{9}{4}$  and for 8: **if**  $T_{\text{Min}(i,j)} = T_{\tau_{i,j}}^{IoT}$  then 9:  $\alpha_{i,j} \leftarrow 1$ 10: **else** 11: **if**  $T_{\text{Min}(i,j)} = T_{\tau_{i,j}}^{\text{Cloud}}$  then  $12:$ 13:  $\beta_{i,j} \leftarrow 0$ 14: **else** 15:  $\alpha_{i,j} \leftarrow 0$  $16:$  $17:$  $\frac{18}{19}$ 19: **end for** 21: return  $(\alpha_{i,j}, \beta_{i,j})$ 7: end for 7: end for 10: else 10: else 12:  $\alpha_{i,j} \leftarrow 0$ 14: else 14: else 16:  $\beta_{i,j} \leftarrow 1$ 17: end if 17: end if 18: end if 18: end if 20: end for 20: end for

The ODVI algorithm, while effective for individual task offloading, has limitations in global optimization and resource contention management. Its focus on single tasks can lead to suboptimal overall system performance and does not address resource contention among multiple tasks. To overcome these issues, we introduce the Virtual Machine Selection for Execution (VMSE) algorithm (Algorithm 2).

VMSE tackles the global optimization problem by considering overall system resource allocation when selecting a virtual machine for each incoming task  $\tau_{i,j}$ . It implicitly manages resource contention by prioritizing VMs with lower utilization. The algorithm evaluates the characteristics of incoming tasks against the current utilization of MEC servers and the Cloud, selecting the least utilized VM that meets the task's performance, resource, and time constraints. This integration significantly enhances QoS through efficient resource utilization and improved task execution efficiency.

To optimize task offloading and resource allocation, we developed the HOQoS algorithm (Algorithm 3), which integrates the features of the ODVI and VMSE algorithms. First, the ODVI algorithm evaluates available execution environments by considering MEC server performance and network latency, generating decision variable pairs based on task characteristics such as size, complexity, and time constraints. These pairs  $\frac{1}{\sqrt{2}}$ 

represent potential execution options for each task. Next, the

Enhancing QoS for IoT Devices through Heuristics-based Ennancing QoS for for Devices through Heuristics-based<br>Computation Offloading in Multi-access Edge Computing 20: end for foreign to the foreign of the second state of the second state of the second state of the second s<br>20: end for the second state of the second state of the second state of the second state of the second state o





assessing task characteristics and resource availability on MEC servers and in the Cloud. It evaluates each virtual machine's performance and chooses the one that best meets the task's requirements regarding performance, resource availability, and<br>time constraints time constraints.  $\mathcal{L}$ represent potential execution options for each task. Next, the VMSE algorithm selects the optimal execution location by The represent potential execution options for each task. Next, the VMSE algorithm selects the optimal execution location by assessing task characteristics and resource availability on MEC time constraints. time constraints.

Algorithm 3 Heuristic-based offloading for improving quality Algorithm 3 Heuristic-based offloading for improving quality of service (HOQoS) of service (HOQoS) Algorithm 3 Heuristic-based offloading for improving quality of service (HOQoS) of service (HOQoS)  $\frac{1}{2}$  Heuristics-based offloading for improving  $\frac{1}{2}$  Heuristics-based offloading  $\frac{1}{2}$ Algorithm 3 Heuristic-based offloading for improving quality  $\frac{\text{Rearries (in 205)}}{\text{Rearies (in 2, 1),M, C, D, T)}}$ 

**Require:**  $((\alpha_{i,j}, \beta_{i,j}), \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathcal{T}_{i,j})$ **Require:**  $((\alpha_{i,j}, \beta_{i,j}), \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathcal{T}_{i,j})$ <br>**Ensure:** Optimal Resource Allocation  $R_{opt}[\tau_{i,j}]$  and selected  $VM$   $VM$ <sub>selected</sub>[ $\tau_{i,j}$ ]  $N$   $N$   $N$ <sub>selected</sub>[ $\tau_{i,j}$ ] 1:  $R_{\text{opt}}[\tau_{i,j}] \leftarrow \text{null}$ <br>2. *VM*<sub>2</sub> [<sub>Ti</sub>] + null 1:  $N_{opt}[T_{i,j}] \leftarrow \text{num}$ <br>2:  $VM_{selected}[\tau_{i,j}] \leftarrow \text{null}$ <br>2:  $\mathbf{f}_{opt}[\tau_{i,j}] \leftarrow \text{null}$ 3: for  $i \in \mathcal{D}$  do 3: for  $i \in D$  do<br>4: for  $j \in \mathcal{T}_i$  do 5:  $(\alpha_{i,j}, \beta_{i,j}) \leftarrow \text{Call Algorithm 1}$ <br>6: **if**  $\alpha_{i,j} = 1$  **then**  $\triangleright$  Local allocation 6: if  $\alpha_{i,j} = 1$  then  $\beta$  Local allocation 7: Ropt[τi,j ] ← LocalNode 7: Ropt[τi,j ] ← LocalNode 8: else if βi,j = 1 then ▷ MEC allocation 6: if αi,j = 1 then ▷ Local allocation 6: if αi,j = 1 then ▷ Local allocation 8: else if βi,j = 1 then ▷ MEC allocation 8: else if βi,j1 then▷ MEC allocation 9: Ropt[τi,j ] ← MECNode 7: Ropt[τi,j ] ← LocalNode 8: else if βi,j = 1 then ▷ MEC allocation 8: else if  $\mu_{i,j} = 1$  then  $\mu_{i,j} \in \mathbb{R}$ <br>
9:  $R_{opt}[\tau_{i,j}] \leftarrow \text{MECNode}$ <br>  $\downarrow \text{Cload allocation}$ 10: **else**  $\triangleright$  Cloud allocation 11:  $R_{\text{opt}}[\tau_{i,j}] \leftarrow \text{CloudNode}$ 12: end if 12: end if 12: end if 13: *VM*<sub>selected</sub> $[\tau_{i,j}] \leftarrow$  Call Algorithm 2 14: end for 14: end for 15: end for 14: end for 14: **end for** 16: **return**  $R_{\text{opt}}$  and  $VM_{\text{selected}}[\tau_{i,j}]$ **Require:**  $((\alpha_{i,j}, \beta_{i,j}), \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathcal{T}_{i,j})$ 4: for  $j \in \mathcal{I}_i$  do<br> $\epsilon$  (example Call Algorithm 1) 6: **i**  $(\alpha_{i,j}, \beta_{i,j}) \leftarrow$  Call Algorithm 1 10: **else**<br> $\overline{P}$   $\over$ 11:  $R_{\text{opt}}[\tau_{i,j}] \leftarrow \text{CloudNode}$ 12: *end* if<br> $V^M$ 13:  $VM_{\text{sele}}$ 15: **end for**<br>16: **return** R c and *VM*  $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ **Require:**  $((\alpha_{i,j}, \beta_{i,j}), \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathcal{T}_{i,j})$  $\frac{1}{2}$  else 12: **end if** 14: *end for*  $\sum_{i=1}^{N}$ 16: *return Ropt and Selected*[*τi,j*] **Ensure:** Optimal Resource Allocation  $R_{opt}[\tau_{i,j}]$  and selected 2:  $VM_{selected}[\tau_{i,j}] \leftarrow null$ <br>2: **for**  $i \in D$ **do** 5:  $(\alpha_{i,j}, \beta_{i,j}) \leftarrow \text{Call Algorithm 1}$ 7:  $R_{\text{opt}}[\tau_{i,j}] \leftarrow \text{LocalNode}$ <br>8. **else if**  $\beta_{i,j} = 1$  then 9:  $R_{\text{opt}}[\tau_{i,j}] \leftarrow \text{MECNode}$ 10: **else**  $\rho$  Cloud allocation 11:  $R_{\text{opt}}[\tau_{i,j}] \leftarrow \text{CloudNode}$  $12:$  end if 13: *VM*<sub>selected</sub> $[\tau_{i,j}] \leftarrow$  Call Algorithm 2 15: end for 16: **return**  $R_{opt}$  and  $VM_{selected}[\tau_{i,j}]$ 6: if  $\alpha_{i,j} = 1$  then  $\triangleright$  Local allocation

The HOQoS algorithm employs a sequential task processing The HOQoS algorithm employs a sequential task processing strategy, handling tasks individually rather than in batches. strategy, handling tasks individually rather than in batches. This approach is crucial for achieving system agility and effec-This approach is crucial for achieving system agility effectively managing dynamic task arrivals. By adapting resource tively managing dynamic task arrivals. By adapting resource<br>allocation in real time, the algorithm ensures a rapid and The HOQoS algorithm employs a sequential task processing The HOQoS algorithm employs a sequential task processing strategy, handling tasks individually rather than in batches. This approach is crucial for achieving system agility and effectively managing dynamic task arrivals. By adapting resource allocation in real time, the algorithm ensures a rapid and allocation in real time, the algorithm ensures a rapid and  $\alpha$  arrival rates. Thus, sequential processing is integral to the algorithm's shifted to  $\mathcal{C}$  is in the process dependence on the projection gorithm's ability to efficiently manage dynamic environments. optimized response, even with irregular or unpredictable task gorithm's ability to efficiently manage dynamic environments.

### V. PERFORMANCE EVALUATION V. PERFORMANCE EVALUATION V. PERFORMANCE EVALUATION

V. PERFORMANCE EVALUATION<br>In the performance evaluation section, we conducted sim-In the performance evaluation section, we condited simulations using the EdgeCloudSim simulator [20]. We analyze the performance of our MEC architecture using various measurement and evaluation methods. In the performance evaluation section, we conducted sim-In the performance evaluation section, we conducted simthe performance of our MEC architecture using various mea-the performance of our MEC architecture using various mea-

### *A. Simulation Parameters A. Simulation ParametersA. Simulation Parameters A. Simulation Parameters A. Simulation Parameters*

Journal of Latex Files, Vol. 14, No. 8, August 2015 Files, Vol. 14, August 2015 5, August 2015 5, August 2015 5, August 2015 5, August 2015 7, Augu

The simulation parameters used for our evaluations are The simulation parameters used for our evaluations are summarized in Table I. The simulations are implemented in<br>Java. and the generated plots are visualized using Matlab. The Java, and the generated plots are visualized using Matlab. The experiments are conducted on a computer with 4 Intel Core i7-9600U processors clocked at 2.59 GHz and 8 GB of RAM.  $T_{\text{t}}$  is simulated for our evaluations are evaluated for our evaluations are evaluated for  $\frac{1}{2}$ I. In the simulation parameters used for our evaluations are *A. Simulation Parameters* The simulation parameters used for our evaluations are  $\mathcal{I}$ Java, and the generated plots are visualized using Matlab. The i7-9600U processors clocked at 2.59 GHz and 8 GB of RAM. i7-9600U processors clocked at 2.59 GHz and 8 GB of RAM.

IABLE I: SIMULATION PARAMETERS. TABLE I

Parameter	<b>IoT</b>	<b>MEC</b>	Cloud
Number of devices	$100 - 2.300$	14	
Number of hosts			
Number of VMs per host		8	
Number of Cores per VM		2	
VM CPU Speed (in MIPS)	4.000	10.000	100,000

EdgeCloudSim utilizes four distinct application types to<br>realistically simulate diverse scenarios as detailed in [20] realistically simulate diverse scenarios, as detailed in [20]. Table II summarizes the characteristics of each application<br>type: Heavy applications are characterized by high data transtype: Heavy applications are characterized by high data transmission, high computational intensity, and low sensitivity to<br>delays. Infotainment applications generally require modermission, ingli computational miensity, and low sensitivity to<br>delays. Infotainment applications generally require moderate data transmission, high computational intensity, and low sensitivity to delays. AR/VR applications involve high data sensitivity to delays. AN VK applications involve light data<br>transmission volumes, moderate computational intensity, and<br>high sensitivity to delays. Health applications typically feature high sensitivity to delays. Health applications typically feature moderate data transmission, moderate computational intensity, and moderate sensitivity to delays. Each application is assessed and moderate sensitivity to delays. Each application is assessed<br>based on several criteria, including usage percentage, which<br>indicates the share of each application in overall usage, and<br>Cloud selection probability which indicates the share of each application in overall usage, and marcates the share of each application in overall usage, and<br>Cloud selection probability, which demonstrates the tendency to use the Cloud for their operation. The volumes of data uploaded and downloaded, as well as the task length, emphasize  $t$  the requirements for bandwidth and processing. the requirements for bandwidth and processing. realistical control and the diverse scenarios simulation in the second in the second in  $\frac{1}{2}$ . [20]. table **H** summarizes the characteristics of each application actays. information applications generally require model-

TABLE II TABLE II<br>Application Characteristics.

<b>Characteristics</b>	Heavy	Infotainment	<b>AR/VR</b>	Health		
Task Length (GI)	45	15	Q	3		
Delay Sensitivity	0.1	0.3	0.9	0.7		
Max. Delay Req. (s)		1.5		0.5		
Data Upload (KB)	2500	25	1500	20		
Data Download (KB)	200	1000	25	1250		

In addition to the characteristics of the applications prein addition to the characteristics of the applications prein addition to the characteristics of the applications pre-<br>sented in Table II, it is important to specify the technical In addition to the characteristics of the applications presented in Table II, it is important to specify the technical sented in Table II, it is important to specify the technical  $\mathbf{E}$  the wireless local area network (WLAN) is set at 200 Mbps, HOQoS slightly increases to 1.2 seconds, while Only Cloud while the bandwidth of the wide area network (WAN) is 15<br>Mbps. The propagation delay on the WAN is 0.1 seconds. These parameters also influence system performance and These parameters also influence system performance and the simulation. parameters of the simulated environment. The bandwidth of 2.1 seconds and 1.15 seconds, respectively. For 1500 devices, should be taken into account when evaluating the results of while the bandwidth of the wide area network  $(WAN)$  is 15 should be taken into account when evaluating the results of should be take should be taken into account when evaluating the results of to evaluate the performance of different resource management *B. Simulation Results* the simulation. parameters of the simulated environment. The bandwidth of The wireless local area network (WLAN) is set at  $200$  Mpps, while the bandwidth of the wide area network (WAN) is 15 Mbps. The propagation delay on the WAN is 0.1 seconds. JOURNAL OF LATEX CLASS FILES, VOL. 14, NO. 8, AUGUST 2015 6

### *B. Simulation Results* **B.** Simulation Results

This section presents the results of simulations conducted This section presents the results of simulations conducted This section presents the results of simulations conducted to evaluate the performance of different resource management to evaluate the performance of different resource management<br>approaches and computation offloading strategies in a highdensity IoT environment. The studied approaches are: approaches and computation offloading strategies in a high-density IoT environment. The studied approaches are: to evaluate the performance of unferent resource management This section presents the results of simulations conducted<br>to evaluate the performance of different resource management evaluate the performance of unferent resource management<br>proaches and computation offloading strategies in a highto evaluate the performance of different resource management This section presents the results of simulations conducted approaches and computation offloading strategies in a high-<br>during LT environment. The station are reaches and

- Only MEC: All tasks are executed on the MEC server. This approach is straightforward but may be limited in terms of resources and processing capacity. • Only MEC: All tasks are executed on the MEC server. This approach is straightforward but may be limited in
	- Only Cloud: All tasks are executed on the Cloud server. This approach offers high processing capacity but can This approach offers high processing capacity but can<br>The distance between Figure to the distance between<br>IoT devices and the cloud. Lead to significant factory due to the distance between<br>IoT devices and the cloud.<br>• Random: Tasks are randomly distributed between the lead to significant latency due to the distance between This approach offers high processing capacity out can • Only Cloud: All tasks are executed on the Cloud server. This approach offers high processing capacity but can lead to significant latency due to the distance between
	- MEC and Cloud servers. This implementation is simple but may not be optimal in terms of performance.<br>DCOA ST: A dynamic computation Officeding approach • Random: Tasks are randomly distributed between the MEC and Cloud servers. This implementation is simple
	- DCOA-ST: A dynamic computation Offloading approach<br>based on the service time [11]. This approach considers • DCOA-ST: A dynamic computation Officiality approach based on the service time [11]. This approach considers based on the service time [11]. This approach considers based on the service time [11]. This approach considers tasks, potentially improving performance. It provides a tasks, potentially improving performance. It provides a systematic approach to task assignment but may lack systematic approach to task assignment out may fack<br>flexibility in dynamic environments.  $\frac{1}{\text{base}}$  the current system state to decide where to execute the current system state to decide where to execute systematic approach to task assignment but may lack stasks, potentiarly improving performance. It provides a the current system state to decide where to execute tasks, potentially improving performance. It provides a • DCOA-ST: A dynamic computation Offloading approach based on the service time [11]. This approach considers systematic approach to task assignment but may lack
- LCDA\*: The LCDA approach [12], which aims to min-• LCDA  $\cdot$ : The LCDA approach [12], which aims to min-<br>imize the execution time of delay-sensitive tasks while imize the execution time of delay-sensitive tasks while<br>ensuring no deadline violations. Their algorithm selects ensuring no deadline violations. Their algorithm selects servers and schedules tasks to optimize service time in a dynamic environment. While LCDA effectively adapts to changing workloads, it may not fully account for task to changing workloads, a may not fully account for tasks ersum and schedules tasks to optimize service time in a dynamic environment. While LCDA effectively adapts a dynamic chyfromnent. While ECDA enectrycry adapts<br>to changing workloads, it may not fully account for task<br>sensitivity or latency constraints. ECDA The ECDA approach [12], which all the limit<br>imize the execution time of delay-sensitive tasks while<br>ensuring no deadline violations. Their algorithm selects<br>servers and schedules tasks to optimize service time in<br>a dy • LCDA\*: The LCDA approach  $[12]$ , which aims to minimize the execution time of delay-sensitive tasks while ensuring no deadline violations. Their algorithm selects servers and schedules tasks to optimize service time in SCIISHIVILY OF TAILING CONSUMINS.

sensitivity or latency constraints.<br>e can evaluate their relative advar comparing these methodologies with our HOQoS algorithm.<br>Comparing these methodologies with our HOQoS algorithm. HOQoS integrates factors such as execution time, task sensiriogos integrates racios such as execution time, task sensi-<br>tivity to delays, latency requirements, resource utilization, and<br>server processing capabilities. Through rigorous evaluations ervis to delays, facincy requirements, resource durization, and<br>server processing capabilities. Through rigorous evaluations and performance comparisons, we illustrate the enhanced and performance comparisons, we intestrate the enhanced<br>effectiveness of HOQoS in optimizing computation offloading effectiveness of HOQoS in optimizing computation officialing<br>and resource allocation within MEC environments, ultimately improving the quality of service for IoT applications. We can evaluate their relative advantages and limitations by and resource anotation within MEC environments, ultimately We can evaluate their relative advantages and limitations by we can evaluate their relative advantages and immations by and resource allocation within MEC environments, ultimately server processing capabilities.<br>The computer their relative educations and limitations by and performance comparisons, we must alle the emianted<br>affectiveness of HOOsS in ortinizing semputation of flooding encerveness of 110Q00 in optimizing compatition of notating and resource anotation within these chynomicines, animately improving the quality of service for IoT applications. We can evaluate their relative advantages and immations by and performance comparisons, we illustrate the enhanced<br>effectiveness of HOQoS in optimizing computation offloading<br>and resonance ellectric mittin MEG environments ellimitately We can evaluate their relative advantages and limitations by HOQoS integrates factors such as execution time, task sensitivity to delays, latency requirements, resource utilization, and server processing capabilities. Through rigorous evaluations and resource allocation within MEC environments, ultimately

1) Simulation Based on Average Service Time: We evaluate the average service time, a critical factor influencing service the average service time, a critical factor influencing service<br>quality and user experience. Figure 3 illustrates the impact quality and user experience. Figure 3 illustrates the impact<br>of the number of IoT devices on service times for different<br>recourse management approaches. For 100 devices, HOOoS of the number of IoT devices on service times for different<br>resource management approaches. For 100 devices, HOQoS and LCDA\* show shallow service times of 0.8 seconds, while Only MEC and DCOA-ST display times of 0.9 seconds. At 900 devices, HOQoS maintains a service time of 1 second, while Only Cloud shows an alarming increase to 5.1 seconds, confirming its inadequacy in high-load scenarios. Other apcommitting its madequacy in high-load scenarios. Other apconfirming its inadequacy in high-load scenarios. Other approaches, such as Random and DCOA-ST, exhibit times of 900 devices, HOQoS maintains a service times of 0.9 seconds, while Only MEC and DCOA-ST display times of 0.9 seconds. At 900 devices, HOQoS maintains a service time of 1 second, confirmation in the matrix of the latter seconds. Only MEC and DCOA-ST display times of 0.9 seconds. At production, such as raindom and DOSA B4, changed three or reaches 7.1 seconds. Thang, at 2500 devices, Hogos reports<br>a service time of 1.5 seconds, whereas Only Cloud remains high at  $7.2$  seconds. 2.1 seconds and 1.15 seconds, respectively. For 1500 devices, HOQoS slightly increases to 1.2 seconds, while Only Cloud resolve  $\frac{7 \text{ 1}}{2 \text{ 200}}$  at 7.200 deviates. HOQoS remarts reaches 7.1 seconds. Finally, at 2300 devices, HOQoS reports reaches 7.1 seconds. Finally, at 2300 devices, HOQoS reports<br>a service time of 1.5 seconds, whereas Only Cloud remains a service time of 1.5 seconds, whereas Only Cloud remains  $2.1$  seconds and  $1.1$ .



*2) Simulation Based on Task Failure Rates:* In assessing

task execution success, task failure rates are key indicators and insufficient network bandwidth. Figure 4 illustrates how failure rates vary across algorithms with different numbers of IoT devices. The HOQoS approach consistently maintains low IoT devices. The HOQoS approach consistently maintains low<br>failure rates, even as device numbers increase, highlighting its robustness in high-density IoT environments. For 300 devices, HOQoS and Only MEC have low failure rates of 1%, factors, see  $\epsilon$  as the highest at  $10\%$ . At 000 devices. Only while LCDA\* has the highest at 10%. At 900 devices, Only 2) Simulation Based on Task Failure Rates: In assessing influenced by factors such as excessive virtual machine usage devices, HOQoS and Only MEC have low failure rates of 1%,



Fig. 4: Task failure rate as a function of the number of IoT dev Fig. 4: Task failure rate as a function of the number of IoT devices

Cloud shows a concerning 74% failure rate, whereas HOQoS maintains a reasonable 1.5%. With 1500 devices, Only Cloud's Fig. 4: Task factor  $\frac{1}{2}$  for  $\frac{1}{2}$  for  $\frac{1}{2}$  function of  $\frac{1}{2}$  function of  $\frac{1}{2}$  and  $\frac{1}{2}$  for  $\frac$ devices a reasonable  $\frac{1}{2}$  for  $\frac{1}{2}$ ,  $\frac{1}{2}$  and scenarios, while HOQoS remains at 2%. Finally, at 2300 failure rate rises to 91%, confirming its inadequacy in highEnhancing QoS for IoT Devices through Heuristics-based<br>
Enhancing QoS for IoT Devices through Heuristics-based Computation Offloading in Multi-access Edge Computing devices, all failure rates increase, with Only Cloud reaching a

devices, all failure rates increase, with Only Cloud reaching a critical 95% and HOQoS at 4%. devices, all failure rates increase, with Only Cloud reaching a<br>*3)*  $\frac{1}{2}$   $\frac$ 

thical 95% and HOQOS at  $+$ *n*.<br>3) Simulation Based on VM Utilization Rates: Figure 5 3) Simulation Based on VM Utilization Rates: Figure 5<br>highlights the impact of the number of IoT devices on the utilization of VMs at the MEC level. For 300 devices, HOQoS shows a utilization rate of 1%, while Only MEC reaches  $3\%$ . shows a utilization rate of 1%, while Only MEC reaches 3%.<br>Other approaches, such as Random and DCOA-ST, exhibit utilization rates of 2% and 4.5%, respectively, while LCDA\* reaches 5%. At 1100 devices, HOQoS maintains a rate of 2.5%, while Only MEC climbs to  $12%$ . The Random and 2.5%, while Only MEC climbs to 12%. The Random and<br>DCOA-ST approaches also show increases, reaching 5% and 13%, respectively. When the number of devices reaches 1700, 13%, respectively. When the number of devices reaches 1700,<br>HOQoS increases to 5%, while Only MEC rises to 23%, with LCDA\* at  $19\%$  and DCOA-ST at 23%. Finally, at 2300 devices, HOQoS presents a utilization rate of  $9.8\%$ , contrasting sharply with Only MEC, which reaches 58%. Other approaches show incre with LCDA\* at 19% and DCOA-ST at 23%. Finally, at 2300 devices, HOQoS presents a utilization rate of 9.8%, contrasting sharply with Only MEC, which reaches 58%. Other approaches show increased utilization, with Random at 12% and DCOA-ST at 27%. 60 3) Simulation Based on VM Utilization Rates: Figure 5



Fig. 5: MEC server utilization as a function of the number of IoT devices. Fig. 5: MEC server utilization as a function of the number of IoT devices.

The utilization of VM at the Cloud level is presented in servers. For 500 devices, HOQoS shows a VM utilization rate of only 0.1%, while Only Cloud reaches 1.2%. Other rate or only 0.1  $\omega$ , while Only Cloud reaches 1.2  $\omega$ . Only<br>approaches, such as Random and DCOA-ST, exhibit utilization rates of 0.8% and 1.2%, respectively, while LCDA\* reaches 0.4%. At 1300 devices, HOQoS maintains a utilization rate of 0.1%, while Only Cloud climbs to 1.1%. The Random and DCOA-ST approaches also show increases, reaching 1% and 3.7%, respectively. When the number of devices reaches 1900, HOQoS increases to 0.2%, while Only Cloud rises to<br>1.1%, with LCDA\* at 2.7% and DCOA-ST at 5.8%. Finally 1.1%, with LCDA\* at 2.7% and DCOA-ST at 5.8%. Finally, at 2300 devices, HOQoS presents a utilization rate of 0.3%, at 2500 devices, HOQOS presents a dunization rate of 0.5%,<br>contrasting sharply with Only Cloud, which reaches 1.4%. Other approaches show increasing utilization, with Random<br>at 1% and DCOA-ST at 11.9%. at  $1\%$  and DCOA-ST at 11.9%.  $\Gamma$  inc dume The unificialistic of VM at the Cloud Tovel is presented in Figure 6, exhibiting slightly lower rates compared to MEC rate of only 0.1%, while Only Cloud reaches 1.2%. Other approaches, such as Random and DCOA-ST, exhibit utilization rate of only 0.1%, while Only Cloud reaches 1.2%. Other contrasting sharply with Only Cloud, which reaches 1.4%. Other approaches show increasing utilization, with Random



 $I \circ I$  devices. Fig. 6: Could serve utilization of the number of the n Fig. 6: Cloud server utilization as a function of the number of

both MEC and cloud server resources under increasing load. In contrast, other approaches, particularly Only MEC and The collads, other approaches, particularly only MEC and<br>Only Cloud, exhibit a significant rise in VM utilization, which could lead to challenges concerning performance and energy consumption. It is important to note that Only Cloud does not utilize MEC servers, while Only MEC does not utilize cloud servers; therefore, their VM usage is not considered in this servers is the foreign the third value in this normal value is not considered in this non-The results demonstrate that HOQoS effectively manages could lead to challenges concerning performance and energy<br>consumption. It is important to note that Only Cloud does not<br>utilize MEC servers, while Only MEC does not utilize cloud  $analysis.$ servers; therefore, their VM usage is not considered in this analysis. analysis.

#### analysis. VI. CONCLUSION AND FUTURE PERSPECTIVES analysis.

This study aimed to improve service quality in Multi-access This stary aimed to improve service quality in Mari-access Edge Companing by ombianing computational roads and ano-<br>cating resources near IoT devices. The findings emphasize the earing resources hear for devices. The minings emphasize the<br>importance of offloading tasks from IoT devices with limited computing capabilities to locations with adequate resources, thereby reducing latency. The proposed heuristic algorithms for task offloading and resource allocation consider task characteristics and resource availability, resulting in decreased serstatements and resource availability, resulting in decreased service times and task failure rates. Future research will focus on developing an autonomous system that utilizes reinforcement learning and machine learning techniques to optimize task execution locations in complex MEC environments. Enhancing the algorithms' adaptive capabilities to respond dynamically to changes in network conditions and device capabilities will be a the algorithms' adaptive capabilities to respond dynamically to<br>changes in network conditions and device capabilities will be a<br>priority. Additionally, integrating advanced predictive analytics priority. Nutritionally, integrating advanced predictive analytics could enable proactive optimization of resource allocation and<br>offloading strategies by forecasting future resource needs and user demands. These advancements aim to further enhance the user demands. These advancements aim to further enhance the<br>performance and efficiency of MEC systems. offloading strategies by forecasting future resource needs and<br>wear damande. These advancements aim to further enhance the

### REFERENCES performance and the MEC systems. performance and efficiency of MEC systems.

[1] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke," Middle-ware for Internet of Things: A Survey," IEEE Internet of Things Journal, vol. 3, no. 1, pp. 70–95, Feb. 2016. *mags Journal*, vol. *5*, ho. 1, pp. 70–95, 1 co. 2010.<br>**DOI**: 10.1109/JIOT.2015.2498900. **DOI**: 10.1109/JIOT.2015.2498900. [1] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke,"

 $\frac{1}{\sqrt{2}}$ 

### Enhancing QoS for IoT Devices through Heuristics-based Computation Offloading in Multi-access Edge Computing

- [2] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on Multi-Access Edge Computing for Internet of Things Realization," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018. **doi**: [10.1109/COMST.2018.2849509](https://doi.org/10.1109/COMST.2018.2849509).
- [3] European Telecommunications Standards Institute (ETSI), "Mobileedge computing (MEC); Framework and reference architecture," ETSI GS MEC 003, 2019.
- [4] M. Patel, B. Naughton, C. Chan, N. Sprecher, S. Abeta, A. Neal, et al., "Mobile-edge computing introductory technical white paper," *White paper, mobile-edge computing (MEC) industry initiative*, vol. 29, pp. 854– 864, 2014.
- [5] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017. **doi**: [10.1109/COMST.2017.2705720](https://doi.org/10.1109/COMST.2017.2705720).
- [6] A. P. Jafari Pozveh and H. S. Shahhoseini, "IoT Integration with MEC," in *Mobile Edge Computing, Springer, Cham*, 2021, pp. 111– 144. **doi**: [10.1007/978-3-030-69893-5\\_6](https://doi.org/10.1007/978-3-030-69893-5_6).
- [7] M. Myyara, O. Lagnfdi, A. Darif, and A. Farchane, "A New Approach Based on Genetic Algorithm for Computation Offloading Optimization in Multi-Access Edge Computing Networks," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 13, no. 4, pp. 4186–4194, 2024. **doi**: [10.11591/ijai.v13.i4.pp4186-4194](https://doi.org/10.11591/ijai.v13.i4.pp4186-4194).
- [8] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-User Offloading for Edge Computing Networks: A Dependency-Aware and Latency-Optimal Approach," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 1678–1689, Mar. 2020. **doi**: [10.1109/JIOT.2019.2943373](https://doi.org/10.1109/JIOT.2019.2943373).
- [9] C. Chen, B. Liu, S. Wan, P. Qiao, and Q. Pei, "An Edge Traffic Flow Detection Scheme Based on Deep Learning in an Intelligent Transportation System," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1840–1852, Mar. 2021. **poi:** [10.1109/TITS.2020.3025687](https://doi.org/10.1109/TITS.2020.3025687).
- [10] A. Naouri, H. Wu, N. A. Nouri, S. Dhelim, and H. Ning, "A novel framework for mobile-edge computing by optimizing task offloading," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 13 065–13 076, 2021. **doi**: [10.1109/JIOT.2021.3064225](https://doi.org/10.1109/JIOT.2021.3064225).
- [11] M. Myyara, O. Lagnfdi, A. Darif, and A. Farchane, "Quality of Expe- rience Improvement and Service Time Optimization through Dynamic Computation Offloading Algorithms in Multi-access Edge Computing Networks," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 16, no. 4, pp. 1–16, 2024. **boi**: [10.5815/ijcnis.2024.04.01](https://doi.org/10.5815/ijcnis.2024.04.01).
- [12] H. Choi, H. Yu, and E. Lee, "Latency-classification-based deadlineaware task offloading algorithm in mobile edge computing environments," *Applied Sciences*, vol. 9, no. 21, p. 4696, 2019. **DOI**: [10.3390/app9214696](https://doi.org/10.3390/app9214696).
- [13] C. Eang, S. Ros, S. Kang, I. Song, P. Tam, S. Math, and S. Kim, "Offloading Decision and Resource Allocation in Mobile Edge Computing for Cost and Latency Efficiencies in Real-Time IoT," *Electronics*, vol. 13, no. 7, p. 1218, 2024. **doi**: [10.3390/electronics13071218](https://doi.org/10.3390/electronics13071218).
- [14] C. Kai, H. Zhou, Y. Yi, and W. Huang, "Collaborative cloud-edgeend task offloading in mobile-edge computing networks with limited communication capability," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 2, pp. 624–634, 2020. **doi**: [10.1109/TCCN.2020.3018159](https://doi.org/10.1109/TCCN.2020.3018159).
- [15] S. Wan, R. Gu, T. Umer, K. Salah, and X. Xu, "Toward Offloading Internet of Vehicles Applications in 5G Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4151–4159, Jul. 2021. **doi**: [10.1109/TITS.2020.3017596](https://doi.org/10.1109/TITS.2020.3017596).
- [16] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P.-V. Mekikis, A. Antonopoulos, and C. Verikoukis, "Online VNF Lifecycle Management in an MEC-Enabled 5G IoT Architecture," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4183–4194, May 2020. **doi**: [10.1109/JIOT.2019.2944695](https://doi.org/10.1109/JIOT.2019.2944695).
- [17] M. Myyara, O. Lagnfdi, A. Darif, and A. Farchane, "A Resource Allocation Strategy to Enhance User Experience for IoT Devices in Multiaccess Edge Computing," in *2024 Sixth International Conference on Intelligent Computing in Data Sciences (ICDS)*, 2024, pp. 1–7. **DOI**: [10.1109/ICDS62089.2024.10756444](https://doi.org/10.1109/ICDS62089.2024.10756444).
- [18] Z. Ding, J. Xu, O. A. Dobre, and H. V. Poor, "Joint Power and Time Allocation for NOMA–MEC Offloading," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6207–6211, Jun. 2019. **doi**: [10.1109/TVT.2019.2907253](https://doi.org/10.1109/TVT.2019.2907253).
- [19] Z. Ning, P. Dong, X. Kong, and F. Xia, "A Cooperative Partial Computation Offloading Scheme for Mobile Edge Computing Enabled Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019. **doi**: [10.1109/JIOT.2018.2868616](https://doi.org/10.1109/JIOT.2018.2868616).
- [20] C. Sonmez, A. Ozgovde, and C. Ersoy, "Fuzzy workload orchestration for edge computing," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 769–782, 2019.  **doi**: [10.1109/TNSM.2019.2901346](https://doi.org/10.1109/TNSM.2019.2901346).
- [21] P. W. Khan, K. Abbas, H. Shaiba, A. Muthanna, A. Abuarqoub, and M. Khayyat, "Energy efficient computation offloading mechanism in multi- server mobile edge computing—An integer linear optimization approach," *Electronics*, vol. 9, no. 6, p. 1010, 2020. **DOI**: [10.3390/electronics9061010](https://doi.org/10.3390/electronics9061010).



**Marouane Myyara** received his B.Sc. in Electronic and Telecommunication Engineering in 2019 and M.Sc. in Telecommunication Systems and Computer Networks in 2021 from Sultan Moulay Slimane University, Beni Mellal, Morocco. He is currently a Ph.D. candidate at the Laboratory of Innovation in Mathematics, Applications, and Information Technology (LIMATI), Polydisciplinary Faculty, Sultan Moulay Slimane University, Morocco. His current research focuses on improving the performance of Multi-access Edge Com-

puting networks (MEC), Cloud Computing, Computation Offloading, and the Internet of Things (IoT).



**Oussama Lagnfdi** received his B.Sc. in Physical Matter Science in 2020 and M.Sc. in Telecommunications Systems and Computer Networks in 2022 from Sultan Moulay Slimane University, Beni Mellal, Morocco. Currently, he is a Ph.D. candidate at the Laboratoire d'Innovation en Mathématiques et Applications et Technologies de l'Information (LIMATI), Polydisciplinary Faculty, Sultan Moulay Slimane University, Morocco. His ongoing research is focused on enhancing the performance of Internet of Things (IoT) and

Mobile Edge Computing (MEC), Artificial Intelligence, Deep Learning, and Fuzzy Logic.



**Anouar Darif** received the bachelor in IEEA (Informatique Électrotechnique, Électronique and Automatique) from Dhar El Mahraz Faculty of Sciences at Mohamed Ben Abdellah University Fez, Morocco in 2005. He received the Diplôme d'Etudes Supérieures Approfondies in Computer Sciences and Telecommunications from the Faculty of Sciences Rabat in 2007. He received the Ph.D. degree in Computer Sciences and Telecommunications from the Faculty of Sciences of Rabat in 2015. He is currently a Research and Teaching Associate in

the Multidisciplinary Faculty at the University of Sultan Moulay Slimane Beni Mellal, Morocco. His research interests include Wireless Sensor Networks (WSN), Mobile Edge Computing (MEC), Internet of Things (IoT), Cloud Computing, and Neural Networks.



**Abderrazak Farchane** received his B.Sc. in Computer Science and Engineering in June 2001 and M.Sc. in Computer Science and Telecommunication from the University of Mohammed V Agdal, Rabat, Morocco, in 2003. He obtained his Ph.D. in Computer Science and Engineering at ENSIAS, Rabat, Morocco. He is currently an Associate Professor of Computer Science in the Polydisciplinary Faculty, at Sultan Moulay Slimane University, Morocco. His areas of interest are Information Coding Theory, Cryptography, and Security.