International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 9, Issue 6, pp 80-100, June-2025
https://doi.org/10.47001/IRJIET/2025.906010

User-Centric Evaluation and Optimization of Resource Allocation in Edge Computing Environments

[1]Mohamed Koroma (Ing), [2]Alimamy Saidu konteh, [3]Yahya Labay Kamara, [4]Chernor Gurasiue Jalloh, [5]Justin Alhaji Conteh (Ing)

[1,3]Lecturer, School of Technology, Computer Science & I.T Department, Njala University, Sierra Leone, West Africa
[2]Lecturer, Milton Margai Technical University, Electrical & Electronics Department Sierra Leone, West Africa
[4]Software Engineer, Nankai University, School of Software Engineering, Tianjin City, P.R. China
[5]MSc Scholar, Electrical Engineering, Xi'an Jiaotong University, Xi'an City, P.R. China
E-mails: [1]mohamed.koroma@njala.edu.sl, [2]askonteh1983@gmail.com, [3]ylkamara@njala.edu.sl, [4]gurasiue.111@gmail.com, [5]justinalhaji996@gmail.com

***Abstract* - The proliferation of mobile applications with intensive computational demands has necessitated the adoption of edge computing to reduce latency and energy consumption. However, edge servers face challenges such as limited resources, dynamic wireless conditions, and inefficient task offloading strategies, particularly in multi-user environments. This paper proposes a user-centric evaluation and optimization framework for resource allocation in edge computing, aiming to minimize latency and energy consumption while maximizing system efficiency. We introduce a greedy-competitive algorithm for dynamic task offloading and a joint communication-computation optimization model that adapts to real-time channel conditions and user requirements. The proposed approach leverages partial task offloading, dynamic voltage frequency scaling (DVFS), and optimal resource partitioning between edge and cloud servers. Simulation results demonstrate significant improvements in energy efficiency (up to 21.2% reduction) and latency reduction (up to 20% fewer task drops) compared to conventional greedy and local execution strategies. The study provides insights into optimal resource allocation, task scheduling, and energy-delay trade-offs in edge computing environments.**

Keywords: Edge computing, resource optimization, task offloading, energy efficiency, latency minimization, greedy-competitive algorithm, dynamic computation offloading.

I. INTRODUCTION

The rapid proliferation of mobile devices and the increasing demand for computation-intensive applications, such as augmented reality, video streaming, and real-time data processing, have strained the limited resources of traditional cloud computing systems. Edge computing has emerged as a promising paradigm to address these challenges by bringing computation resources closer to end-users, thereby reducing latency and energy consumption [1], [2]. In edge computing environments, resources such as CPU cycles, bandwidth, and storage are deployed at base stations (BS) or edge servers, forming Mobile Edge Computing (MEC) systems [3], [4]. The exponential growth of mobile applications (e.g., AR, real-time analytics) has strained cloud infrastructures, prompting the adoption of edge computing to reduce latency and energy consumption. Edge servers deployed at base stations (BS) enable Mobile Edge Computing (MEC), but face challenges like limited resources and dynamic wireless conditions [3][4].Figure 1 (Edge Network System Architecture) illustrates the hierarchical interaction between mobile devices, edge servers, andcloud datacenters, highlighting resource competition.

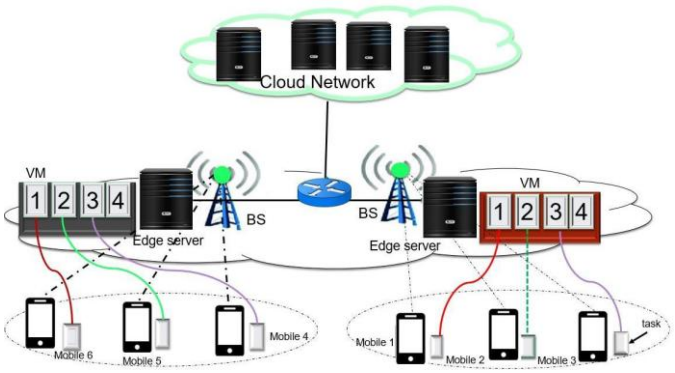


Figure 1-1: Edge Network System Architecture

Optimizing resource allocation improves QoS for latency-sensitive applications (e.g., autonomous vehicles) and extends device battery life. Despite its advantages, edge computing faces significant challenges, including limited communication and computation resources, dynamic wireless channel conditions, and the need for efficient task offloading strategies [5], [6]. These challenges are exacerbated in multi-user scenarios where multiple devices compete for finite resources, leading to increased latency and energy consumption [7], [8].Optimizing resource allocation in edge

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

computing is critical for enhancing user experience, particularly in latency-sensitive applications. Efficient resource management can minimize energy consumption in mobile devices, extend battery life, and improve overall system performance [9], [10]. Additionally, edge computing enables real-time processing for applications like autonomous vehicles and smart cities, where low latency is paramount [11], [12].

Existing studies on edge computing resource allocation often focus on single-user scenarios or assume static wireless conditions, neglecting the dynamic and stochastic nature of real-world environments [13], [14]. Furthermore, many approaches lack a user-centric perspective, failing to account for the diverse requirements of mobile applications and the energy constraints of devices [15], [16]. There is also limited work on integrating edge and cloud computing to leverage their combined strengths [17], [18].How can a user-centric, dynamic resource allocation strategy optimize energy efficiency and minimize latency in multi-user edge computing environments? This study aims to evaluate and optimize resource allocation in edge computing environments from a user-centric perspective, with the goal of minimizing latency and energy consumption. This study aims to achieve three key objectives: (1) developing a dynamic computation offloading algorithm that jointly optimizes task distribution, CPU frequencies, and transmission power; (2) proposing a greedy-competitive strategy for multi-user resource allocation; and (3) validating these algorithms through simulations to demonstrate improvements in latency and energy efficiency. The article is structured systematically: Section II reviews existing literature and identifies research gaps, Section III presents the system model, Section IV details the proposed algorithms, Section V evaluates their performance, and Section VI concludes with findings and future research directions. This comprehensive approach ensures both theoretical innovation and practical validation in edge computing resource optimization.

II. BACKGROUND AND RELATED WORK

A. Evolution of Edge Computing

The rapid growth of advanced mobile applications such as augmented/virtual reality (AR/VR), real-time analytics, and 3D gaming has accelerated the shift from traditional cloud computing to edge computing architectures [19]. By deploying computational resources at base stations (BSs), edge networks significantly reduce latency for end users. However, these systems encounter several fundamental challenges: (1) resource constraints due to limited computation capacity (CPU cycles, memory) at edge servers [20]; (2) critical energy-latency tradeoffs as mobile devices face battery drain during computation-intensive tasks; and (3) dynamic operating environments characterized by time-varying wireless channels and bursty task arrivals [21]. Research in this domain has evolved through several key directions, beginning with early computation offloading frameworks like MAUI [22] and ThinkAir [23] that enabled code offloading but relied on static wireless assumptions, followed by more advanced approaches incorporating channel dynamics through stochastic control and game theory. Subsequent work has focused on developing sophisticated resource allocation strategies, including joint communication-computation optimization [24], dynamic voltage and frequency scaling (DVFS)-based energy minimization [25], and hybrid edge-cloud architectures [26]. Emerging paradigms are now exploring innovative solutions such as energy harvesting integration [27], mobility-aware task scheduling [28], and federated edge learning [29] to address the evolving challenges in edge computing systems.

B. Shortcoming of the Existing Research

Table 2-1: Shortcomings of Existing work

Limitation	Impact	Representative studies
Single-user focus	Fails to address multi-user interface	[20][25]
Static resource assumptions	Inapplicable to dynamic networks	[7][19]
Binary offloading models	Neglects partial task partitioning	[8][26]
Isolated edge/cloud design	Misses' collaborative optimization	[28][29]
Simplified channel models	Overestimates offloading benefits	[14][21]

Despite significant advancements in edge computing, several critical challenges remain unaddressed in current research. First, the need for multi-objective optimization persists, particularly in achieving simultaneous energy-latency-reliability tradeoffs in dynamic environments. Second, existing systems often lack sufficient user-centric adaptation mechanisms to accommodate personalized QoS requirements across diverse applications. Third, while numerous theoretical solutions exist, there remains a pressing need for scalable algorithms that can be practically implemented in dense network deployments. Finally, the integration of energy harvesting technologies introduces new complexities in intermittent power supply management that current architectures struggle to address effectively [30]. These unresolved challenges highlight fundamental gaps in addressing real-world system dynamism and heterogeneous user demandsa limitation our work specifically targets through

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

novel adaptive resource allocation strategies that dynamically balance computational workloads with network conditions and energy constraints.

C. Existing System

Current edge computing systems employ either full (binary) or partial offloading strategies for application processing, where the allocation of computational resources depends fundamentally on the application's parallelization capability [31]. When applications cannot be partitioned, they must be processed entirely at a single edge node, as illustrated in Figure 2-1, where non-parallelizable tasks are offloaded wholly to edge servers. Conversely, partitionable applications can leverage distributed computing across multiple edge nodes, demonstrated in Figure 2-1 by workloads split among three edge servers at a base station. Existing research has primarily addressed resource allocation through two approaches: single-node computation for resource-constrained scenarios and multi-node distributed processing when edge resources prove insufficient, often incorporating cloud computing (CC) fallback mechanisms [31]. However, these systems face inherent limitations in dynamically adapting to varying application requirements and network conditions, particularly when dealing with heterogeneous workloads and intermittent resource availability. The current taxonomy of solutions thus bifurcates between single-node optimization and multi-node coordination strategies, each presenting distinct challenges in quality-of-service maintenance and energy-latency tradeoffs.

Figure 2-1 illustrates a multi-tier edge computing architecture where three mobile devices (Device 1, Device 2, Device 3) interact with edge servers co-located at base stations, demonstrating three distinct computation offloading strategies. Device 1 employs partial offloading, splitting its "Office Application" workload between local execution and edge processing ("Data processed"), enabling latency-sensitive components to benefit from edge resources while keeping data-intensive operations on-device. Devices 2 and 3 utilize full offloading, delegating entire computational tasks to the edge servers, which is optimal for resource-constrained devices running complex applications like real-time video analytics. The base stations facilitate reliable connectivity between devices and edge servers, while the distributed edge infrastructure allows parallel processing of offloaded tasks across multiple servers. This architecture is particularly effective for 5G-enabled smart factory scenarios, where Device 1 might process safety-critical sensor data locally while offloading predictive maintenance algorithms to the edge, and Devices 2-3 could fully offload computer vision tasks for quality inspection – collectively demonstrating how hybrid offloading strategies optimize latency (typically <20ms for industrial use cases), energy efficiency, and computational load balancing in heterogeneous IoT environments.

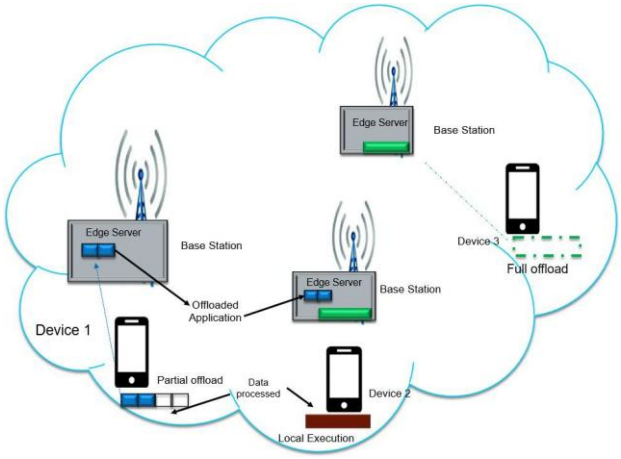


Figure 2.1: An example of allocation of computing resources within edge server

The primary goal of [31] is to optimize computation resource allocation for a single edge node, maximizing the number of applications served by the mobile edge computing (MEC) system while meeting offloading delay constraints. Offloading decisions depend on two key factors: (i) the application's priority, determined by its execution delay requirements (e.g., lower delay demands higher priority), and (ii) the availability of computational resources on the edge server.

Figure 2-2 illustrates the fundamental communication and computation workflow for resource allocation. Offloaded applications first reach the edge server's local scheduler, which evaluates whether nearby edge devices have adequate resources. If sufficient resources exist, a virtual machine (VM) is assigned to the mobile device, enabling the application to be processed locally. The results are then returned to the mobile device, completing the cycle.

The figure depicts a hierarchical MEC architecture, integrating mobile devices, an edge server linked to a base station, and a cloud data center, showcasing the distributed data processing pipeline.

The system operates through coordinated resource scheduling at the edge server, which intelligently allocates computational tasks between local edge processing and cloud offloading based on real-time demands. Mobile devices initially submit tasks to the edge server, where latency-sensitive operations are processed locally (Data process), while compute-intensive workloads are routed to the cloud data center for scalable execution.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

This architecture is particularly valuable for applications like IoT analytics and real-time video processing, where the edge server acts as a decision-making hub—optimizing response times by processing urgent tasks locally (e.g., <50ms latency for industrial automation) while leveraging the cloud's unlimited resources for batch processing or complex AI model inference. The base station ensures reliable connectivity, enabling seamless data flow across all tiers and illustrating how modern systems balance latency, energy efficiency, and computational power through tiered resource allocation.

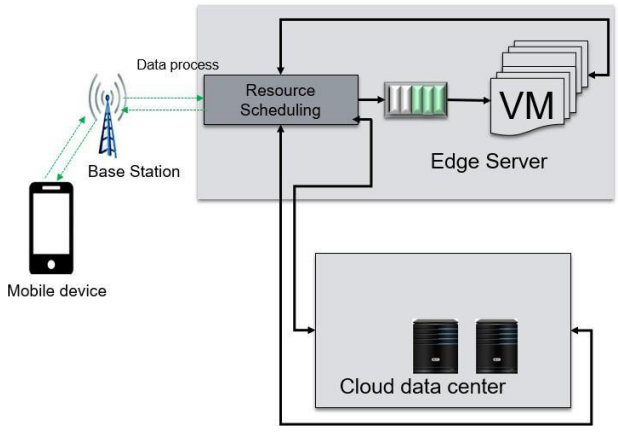


Figure 2.2: Computation Resources allocation [31]

Existing edge computing systems employ adaptive offloading strategies where applications are delegated to cloud data centers when edge server resources are insufficient [31]. To optimize edge processing, researchers have proposed priority-based cooperative policies that utilize buffer thresholds and recursive algorithms to maximize application completion rates within delay constraints [32]. While [32] introduces an M-decision process framework to minimize both execution delay and power consumption in dense networks, its computational complexity led to the development of an index policy for efficient edge server selection. However, these approaches incur higher system costs (weighted delay and energy) compared to optimal solutions [33]. Recent work in [33] extends this optimization by incorporating virtual machine (VM) migration costs and communication resource management through Markov Decision Processes (MDPs), demonstrating that VMs are preferentially allocated to proximal edge nodes with sufficient computational power to minimize backhaul delays. A key limitation across all these studies [31-33] is their failure to leverage distributed computing across multiple edge nodes for single applications, which could potentially reduce execution delays further. The current solutions primarily focus on single-node optimizations or cloud fallbacks rather than exploring collaborative multi-edge processing architectures.

D. The problem of the existing approach

The allocation of computational resources in edge computing systems depends critically on whether applications are offloaded fully or partially to edge servers, a decision heavily influenced by the application's parallelization capability [34]. When applications cannot be partitioned (Figure 2-1), they must be processed entirely at a single edge node, while parallelizable applications can leverage distributed computing across multiple edge servers (Figure 2-1 shows an example partitioned across three nodes). This resource allocation problem extends to cloud computing when edge resources are insufficient, introducing additional challenges in energy-aware optimization [34].A fundamental challenge lies in managing energy consumption across mobile devices and edge networks. Rather than simply minimizing energy use, renewable energy-powered systems should optimize performance within energy constraints, treating renewable energy as essentially "free" [34]. This approach requires careful consideration of Energy State Information (ESI) in offloading decisions, along with Channel State Information (CSI), to balance workload distribution between edge servers and central clouds based on network congestion and available energy [35]. However, current solutions using machine learning [34] and Lyapunov optimization [35] remain limited to small-scale systems, failing to address large-scale deployments.

The intermittent nature of renewable energy introduces reliability challenges in computation offloading. Three potential solutions emerge:

- **Dense edge server deployment** creates overlapping service areas for energy load balancing, as demonstrated in energy-harvesting cooperative systems [36]
- **Strategic energy source selection**, where solar energy proves particularly effective for high peak-to-mean ratio workloads [37]
- **Wireless power transfer** via RF waves, already implemented in commercial devices like Samsung Galaxy S7, though requiring careful scheduling to address the "double near-far" problem in multi-user scenarios [38,39]

III. SYSTEM MODEL

The edge network system consists of several mobile devices d $\{1,2,.,n\}$ that are connected via a wireless channel or Wi-Fi access point and edge servers $d$with a set of virtual machines (VM) for computation capability. Figure 3-1 illustrates a typical edge computing architecture where a base station connects to an edge server that serves multiple mobile devices (Device 1, Device 2, Device 3). It demonstrates three

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

fundamental computation offloading strategies: (1) Partial offload, where a device splits tasks between local execution and edge processing; (2) Full offload, where devices delegate entire workloads to the edge server; and (3) **Local execution**, where computation occurs entirely on the device. This architecture enables latency-sensitive applications (e.g., AR/VR, real-time analytics) to leverage proximal edge resources for partial processing while maintaining data privacy through local execution for sensitive components. The base station-edge server-device hierarchy optimizes resource allocation by dynamically distributing workloads based on factors like computational complexity, energy constraints, and network conditions, exemplifying how modern edge systems balance performance and efficiency in distributed computing environments.

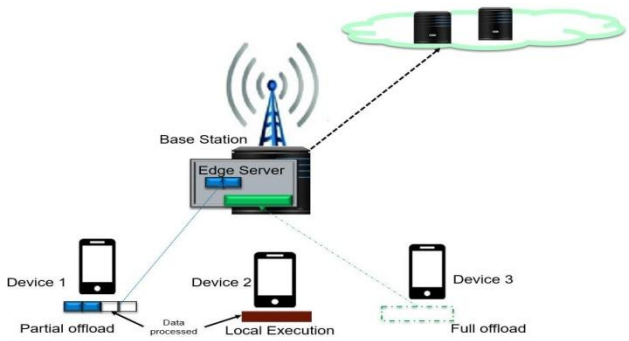


Figure 3.1: Edge Network System Model

A. Problem of Resource allocation

This section presents the system architecture and operational model of an edge computing network, comprising base stations (BS) with co-located edge servers, centralized cloud servers, and multiple mobile devices.

As illustrated in Fig. 3-1, the edge network integrates BS-connected edge servers (denoted as e) featuring constrained computation resources, data processing capabilities, caching, and storage functionalities. Mobile devices ($\boldsymbol{d}$) within the BS coverage area connect via wireless channels, each handling computation tasks with varying delay constraints that may be executed locally or offloaded to edge servers. The model incorporates hybrid computation where tasks are dynamically distributed between mobile devices and edge networks, particularly when edge resources reach capacity. Task arrivals follow a stochastic pattern with priority given to larger tasks under a first-come-first-served (FCFS) policy, though alternative scheduling approaches like Round Robin (RR) are considered for improved load balancing. The RR performance depends critically on time quantum selection - excessive quantum sizes mimic FCS behavior, while insufficient quanta increase context-switching overhead and reduce CPU efficiency. Key system parameters include: task-specific data length (l in bits), required CPU cycles per bit (Ci), and processing frequencies for mobile devices (fd) and edge servers (fe). The tuple (t, l, fd) captures essential task characteristics, while edge and cloud server capabilities are quantified by their respective CPU cycle frequencies (fe and fC). Virtual machine allocations enable flexible task offloading, with the edge scheduler proactively predicting and managing task distributions based on these parameters to optimize system performance.

Table 1: System Description

d	Mobile Device
e	Edge server
i	Application tasks
fd	Mobile device CPU cycle frequency
fe	Edge Server CPU cycle Frequency
fC	computation capability of cloud server's CPU cycle
Ed	Energy Consumption for tasks locally executed in mobile device
Ee	Energy consumption for tasks transfer to edge server
Ci	Number CPU cycles required to process tasks in server
li	Local tasks execution
$(1-\alpha i)li$	Offloaded tasks
(t, l, fd)	Computation tasks l of bit input and deadline t
b	Bandwidth

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

t	Tasks execution time
ttx_i	Total backhaul transmission latency proportional to offloaded tasks size,
tl_i	Transmission delay for the device to offload its computation task
te_i	Computation latency for tasks processing in the edge server
tc_i	Computation latency for tasks processing in cloud server
$ttotal$	Total execution time
ia	Task arrival indicator at time t
hi	Wireless channel power gain
ri	Wireless transmission rate
ρi	Transmission power of mobile device d offloading tasks in edge server
l	Represents length of one-time frame,
$\aleph i$	Represents noise power at the edge computing network
k	Is effective switched capacitance that depends on the chip architecture
αi	Denote $\alpha i \epsilon$ [0,1] as task offloading splitting ratio

B. Wireless Resources Communication Model

The system employs Time Division Multiple Access (TDMA) for resource sharing among mobile devices, enabling equitable distribution of computation resources including bandwidth and memory storage. Each device d is allocated a positive time-slot duration (i> 0) by the edge network e for task offloading. The wireless channels between mobile devices and base stations are modeled as independent and identically distributed (i.d) Rayleigh fading channels, with Channel State Information (CSI) used to approximate signal strength for effective task scheduling.

The wireless transmission rate ri is determined by:

$$r_i = B \log_2 \left(1 + \frac{P_i h_i}{NO}\right) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots 1$$

Where:

$$B = Channel\ bandwidth(Hz)$$

$$P_i h_i = Transmission\ power\ of\ device\ d(w) and\ Instantanous\ channel\ power\ gain$$

$$NO = Noise\ power\ spectral\ density(W\ per\ Hz)$$

This formulation assumes constant transmission time slots and applies Shannon's capacity theorem to maximize spectral efficiency while maintaining reliable communication for edge offloading tasks. The model enables quantitative analysis of tradeoffs between transmission power, channel conditions, and achievable data rates in the edge computing environment.

C. Computation delay model

This work adopts a hybrid computation model where mobile devices with limited processing capacity offload tasks to edge servers for execution. As illustrated in Figure 1, the system implements a three-tier architecture: (1) local computation at mobile

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Issue 6, pp 80-100, June-2025
https://doi.org/10.47001/IRJIET/2025.906010

devices, (2) edge processing at base station (BS)-hosted virtual machines (VMs), and (3) cloud computation for resource-intensive tasks.

The edge server dynamically partitions received workloads, processing delay-sensitive components locally while offloading remaining tasks to the cloud via backhaul links.The computation workflow comprises five phases:

- Wireless Offloading: Mobile devices transmit time-critical tasks directly to edge servers
- Edge Processing: BS-located VMs execute prioritized task segments
- Cloud Offloading: Non-critical components are routed to cloud datacenters
- Result Aggregation: Edge servers consolidate distributed computation outputs
- Result Delivery: Final outputs are returned to mobile devices

Key modeling assumptions include:

- Task partitioning overhead is negligible compared to computation/communication delays
- Result download latency is insignificant (typical for lightweight outputs like face recognition results)
- Wireless channels exhibit Rayleigh fading with millisecond-scale coherence time

The average transmission delay for task offloading is given by:

$$\bar{t}_{tx} = \frac{L_i}{B \log_2\left(1 + \frac{P_i \bar{h}_i}{No}\right)} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots 2$$

Where:

$$L_i = Task\ data\ size\ (bits)$$

$$B = Channel\ bandwidth(Hz)$$

$$P_i \bar{h}_i = Transmission\ power\ of\ device\ d(w) and\ average\ channel\ power\ gain$$

$$NO = Noise\ power\ spectral\ density(W\ per\ Hz)$$

This model captures critical system tradeoffs through three interdependent dimensions: (1) wireless dynamics, where time-varying channel conditions (modeled via Rayleigh fading) govern instantaneous transmission rates; (2) resource constraints, as finite edge server capacity (fedge) necessitates strategic cloud offloading; and (3) QoS requirements, with end-to-end latency budgets (100–500ms for interactive applications) dictating task scheduling policies. It specifically addresses heterogeneous workloads characterized by variable CPU cycles/bit (Ci*Ci*) and mixed criticality (e.g., real-time video analytics vs. background data processing). By unifying local, edge, and cloud resource arbitration, the framework offers four key advantages: (i) probabilistic channel modeling for realistic wireless environments, (ii) explicit edge capacity thresholds to prevent overload, (iii) pragmatic exclusion of negligible delays (e.g., task partitioning overhead <1ms), and (iv) a quantifiable metric space for optimizing latency-reliability Pareto frontiers. This enables systematic evaluation of offloading strategies under practical constraints encountered in 5G/6G edge deployments.

D. Transmission Latency in Mobile Devices

The transmission latency for mobile devices offloading tasks to edge servers via wireless channels is characterized by dynamic channel conditions and time-varying power gains. Due to the random nature of wireless fading (modeled as Rayleigh-distributed), instantaneous latency fluctuates in the millisecond range, influenced by factors such as channel coherence time and

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

TDMA slot duration. For latency-sensitive applications (e.g., online gaming, video streaming), end-to-end delays typically span tens to hundreds of milliseconds, constrained by:

- Bandwidth scarcity: Limited wireless resources shared among multiple devices
- Computation bottlenecks: Edge server CPU capacity for concurrent task processing
- Workload variability: Fluctuating demands from bursty traffic

Given these dynamics, the system employs **average transmission delay** as a stable metric for resource allocation, derived as:

$$\bar{t}_{tx} = E\left[\frac{L_i}{B\log_2\left(1+\frac{P_i\bar{h}_i}{No}\right)}\right] \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3$$

Where:

$$L_i = Task\ data\ size\ (bits)$$

$$B = Channel\ bandwidth(Hz)$$

$$P_i h_i = Transmission\ power\ of\ device\ d(w) and\ Instantanous\ channel\ power\ gain$$

$$NO = Noise\ power\ spectral\ density(W\ per\ Hz)$$

The model yields three key operational insights for edge offloading systems: First, short-timescale wireless channel dynamics necessitate statistical modeling (via expectation operator $\mathbb{E}[\cdot]$) to account for rapid signal fluctuations. Second, inherent design tradeoffs emerge where latency reduction requires either increased transmission power (ρ_i) at the cost of higher energy consumption, or decreased task size (l_i) through compression/partitioning techniques. Third, TDMA scheduling parameters must carefully balance time-slot granularity (to maintain responsiveness) against protocol overhead (to preserve system efficiency). Collectively, this formulation establishes a mathematically tractable framework for optimizing computation offloading decisions that explicitly incorporates real-world wireless channel unpredictability while maintaining practical implement ability in mobile edge computing deployments.

E. Computation Latency in Edge Server

When the whole data of each task is received from the mobile device, then edge server has to immediately initialize the tasks offloading strategy and splited each task into two prts, one part is processed in edge server and the other part process in cloud datacenters. It is analysed, or assumed that each computation task can arbitrarily be splitedwhile neglecting the inheritedcontent, whichcorrespond to the scenario like video streaming compression and speech or vice recognition. We denote$\alpha_i \in [0,1]$for decision task offloading split-level ratio, which determines the data computation proportionality for tasks offloaded to the edge server, or cloud datacenter. The number of central processing unit (CPU) cycles frequency which requiredtobe executedtasksintheedgeserver computationpartofdatacanbeexpressedas$\alpha_i l_i c_i$.The computation resources allocate in edge server that each virtual machine is allocated to each mobile device as number of CPU cycle per seconds f_e.

Therefore, the computation latency for tasks processing in the edge server is expressed as

$$t_i^e = P_i \frac{\alpha_i l_i C_i}{r_i^n f_e} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 4$$

The energy consumed by the mobile device is given by

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Issue 6, pp 80-100, June-2025
https://doi.org/10.47001/IRJIET/2025.906010

$$E_e = k \sum_{C=1}^{C_i} r_i^n \frac{\alpha_i l_i c_i}{f e} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots .5$$

F. Transmission Delay in Edge Network

In the edge Network, the communication transceiver is generally different from computation CPU cycles frequency.The tasks execution in edge server can be performed in parallel with the transmission of the cloud server, where all edge networks are connected through different back haullinks to cloud datacenter resources with high bandwidth.The shared backhaul link among multiple mobile device users through their delay, which are difficult to determined and implement because of the random data arrival, several mobile device user tasks scheduling, and complex routing algorithm. In this regard, here are the reminder of this proposal contributions below; the optimal resources allocation based on scheduling and offloading policy for edge computing network and cloud computing network. In this proposal assumed that resource scheduling strategy and offloading algorithms are based for online computation which are fixed.The backhaul communication capability of each mobile device offloading tasks to edge server denoted as b_h. Therefore, total transmission delay in above computation latency constrain tinedge server, and the total backhaul transmission latency are proportional to the off loaded tasks size, which is expressed as:

$$t \quad x_i - \frac{(1-\alpha_i)l_i}{b_h} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots .6$$

Where b_h is the backhaul link for 1-bit data transmission rate, l_i is length of tasks size.

G. Computation Delay of Cloud Server

The mobile devices transmit tasks to the edge server, when there are limitedcomputation resources, then edge server transfer some tasks to cloud datacenter for further execution, hence the server will allocate the available computation resource to tasks for parallel execution.Therefore, number of CPU cycles frequency required to process tasks in cloud server computation is defined by $(1-\alpha_i)l_i c_i$. The computation resource allocation model is given by number of CPU cycles per seconds f_C, as cloud network computation resource which is allocated to the mobile device served by the edge server. Therefore, execution delay to processpart of data in the cloud server is presented by

When tasks are offloaded and processed in edge server or cloud server, the mobile device is required to wait for return results. Therefore, the computation completion periodoftime t^c spent on the execution of task on the edgeserver, or cloudserver, idle power consumption of mobile device can be calculated below:

$$E_e^i = \frac{(1-\alpha_i)l_i C_i}{f_C} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots .9$$

Therefore, total energy consumption E_{total} for local tasks and offloaded task is expressed as $E_{total} = E_e + E_d$, are defined as sum of the energy spent to transmit data to the edge server or cloud, E_e plus idle power consumption, E_e as given by:

However, the optimized energy efficient minimization task offload, is necessary to provide optimal communication and computation resources allocation, with aim of improving the energy efficiency and latency reduction because of the competition over the resource in edge server.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

H. Problem Formulation

This work introduces two key performance metrics: execution delay and energy consumption, with a primary focus on formulating the delay minimization problem. We assume edge and cloud servers possess substantially greater computational capacity than mobile devices, with sufficient storage to accommodate all user demands, thereby concentrating our optimization on mobile-edge interactions for energy and latency efficiency. The model explicitly excludes edge/cloud server energy overheads as negligible to the core mobile device optimization. Mobile devices dynamically select proximal access points for task offloading in the edge network, operating under quasi-static conditions during offloading operations a reasonable assumption given the short timescales involved. The system optimization problem is formally expressed as a joint communication-computation framework that minimizes total energy consumption while reducing end-to-end delays through strategic task distribution between mobile devices and edge servers.

$$Min E = \sum_{i=1}^{n}((1-\alpha_{i,j})E_d + \alpha_i E_e) \quad (10)$$

$$\text{s.t} \quad \sum_{i=1}^{n} \alpha_i E_e < E_d, \forall i \epsilon_d \quad (11)$$

$$\sum_{j=1}^{n} \alpha i, t_i^e < t_i^l \forall i \in d \quad (12)$$

$$\sum_{j=1}^{n} \alpha i, t_i^{tr} h i < t_i^l \forall i \in d \quad (13)$$

$$\sum_{j=1}^{n} \alpha i < 1, \forall i \in d \quad (14)$$

$$\alpha_i \in \{0,1\}, \forall i \in e, , \forall i \in e \quad (15)$$

The optimization framework incorporates four key constraints: (i) Equation (11) ensures offloading energy never exceeds local computation energy; (ii) Equation (12) bounds total offloading energy by local execution costs; (iii) Equation (13) enforces channel-quality-aware resource allocation to mitigate interference during traffic bursts; and (iv) Equation (14) restricts wireless access to single-device connections, with exceptions for frequent edge-server communicators. Equation (15) implements binary offloading decisions ($ai \in \{0,1\}$), where excessive concurrent offloading causes interference-driven rate reduction. Two dominant factors govern mobile energy consumption: transmission power (ρi) and time (tx), with energy $Etx = \rho itx$. This creates a fundamental tradeoff - low channel rates increase both transmission time and energy, discouraging offloading. To balance this, the model implements adaptive thresholds that dynamically regulate offloading participation based on real-time channel conditions and device capabilities, ensuring energy-efficient operation while maintaining QoS requirements.

I. Energy consumption for Task Offloading Mechanism

To solve the optimization problem in Equation (10), we model the energy consumption for task offloading in mobile edge computing networks through a structured approach. First, a priority-based algorithm classifies mobile devices into two categories: (i) devices participating in greedy computation offloading, and (ii) devices executing tasks locally due to unfavorable energy consumption characteristics. The classification criteria depend on each device's computational demand intensity, which determines its offloading priority. Higher-priority devices are allocated edge server resources through a proposed greedy offloading algorithm that optimally matches communication resources to each mobile device's requirements. This approach ensures efficient resource utilization while maintaining energy-aware task distribution between local and edge computation.

J. Mobile Device User Classification and Priority Determination

The classification and priority determination are based on characteristics of mobile device and applications tasks, such as application data size, computation workload, mobile device and edge server computation capacity, and energy consumption, and computation latency. The mobile device computation is grouped into two parts; Set of mobile devices which should execute their task locally and it is denoted as li.

In addition to the equation above, the rest of the mobile device fall into second group which offload tasks to edge server $(1 - \alpha i)li$ the mobile device can either decide to execute their task locally or decide to offload the task to the edge server. The

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

computation decision depends on the wireless channel state. When mobile device wants to execute it tasks, the different computation tasks priorities are set for each to decide which tasks to offload process, therefore the process is defined below.

$$y_i = \frac{h_i p_i}{\sqrt{Ed}} \qquad (16)$$

The classification and priority determination are mobile device illustrated in

Algorithm 1: The Greedy Competitive Based on Tasks Offloading Algorithm. This is proposed computation offloading approach for the multiple of mobile devices based on the mentioned above analysis of mobile devices classified and determined as priorities scheme. In this work we aim to maximize the efficiency energy and reduced latency of task offloaded to the edge network as subjected to the minimum energy consumption requirement by the mobile device to execute tasks locally.

Algroithm1: The Greedy Competitive Based on Tasks Offloading Algorithm

Input: initial set applications data offload and executed locally.

Output:

Mobile devices set: $d\{1,2,.,.,.d\}$;

Edge wetwork and wireless channels: e$\{1,2,.,.,.E\}$; The application task executed local tasks: l_i

The application task offloaded in edge server $(1-\alpha_i)l_i$ Transmission power ρ_i

Idle power: ρ_{idle} ;

The category of mobile device sets which execute tasksand offloadtasks $l_i = (1-\alpha_i)l_i$

Tasks priority set $Y_i = \emptyset$

1: F**or** mobile device $\boldsymbol{d} = 1\ to\ D$ **do**

2: **For** wireless channel $\boldsymbol{e} = 1\ to\ \boldsymbol{E}$do

3: Calculate wireless channel data transfer rate r_i of each mobile device as in equation 12, and energy consumption for E_e in equation 13

4: ***if*** $E_d \leq E_e$ then

5: $i = l_i$

6: **else**

7: $i = (1-\alpha_i)l_i$

8: $Y_i = \frac{h_i \rho_i}{\sqrt{Ed}}$

9**: endif**

10**: end for Output:**

The category of mobile device computation tasks: l_i, $(1-\alpha_i)l_i$;

The computation priorities for each mobile device $Y = \{Y_i\}$, $i = (1-\alpha_i)l_i$;

ALGORITHM 1: The Greedy Algorithm for mobile device classification and determination of tasks priority

Figure 3-2 illustrates a barter-based resource allocation system involving multiple mobile devices and wireless channels. The "Barter collection" section outlines key processes such as responding to requests, resource allocation, and cost evaluation. The "Receiver" segments list devices (Whole device 1, Whole device 2, and Mobile device 2) and wireless channels, indicating their roles in receiving data or tasks. The "Mobile device 3" section highlights the sender's functions, including transmitting bid information and task offloading requests. This diagram depicts a collaborative offloading framework where devices exchange resources via wireless channels to optimize performance and efficiency in a distributed network environment.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Issue 6, pp 80-100, June-2025
https://doi.org/10.47001/IRJIET/2025.906010

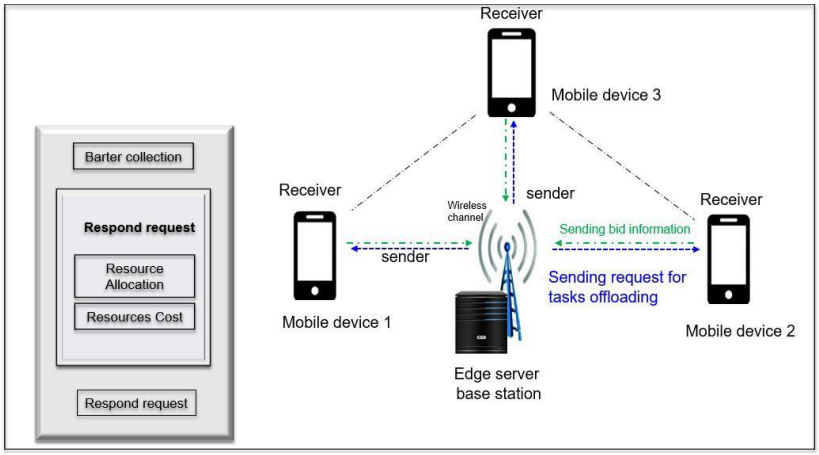


Figure 3.2: Communication Edge Server computing system

The system involves a mobile device (d_i) that sends task offloading requests with associated energy costs and evaluates barter information from wireless channels to select the optimal channel (winner) for resource allocation. The decision to offload tasks to an edge server or execute them locally depends on computation resource costs (β_i), where $\beta_i = E_e$ represents the reserved energy for edge server acceptance. Wireless channels provide bandwidth and submit barter information (A_i and S_i), where $A_i = E_e$ denotes allocated resource costs and $S_i = \rho_i h_i$ reflects computation resources for submitted tasks. The mobile device compares energy consumption and channel resources to optimize efficiency and select the best channel. Unlike multi-round competitive allocation methods, which introduce excessive delay, this work employs a greedy offloading approach to minimize latency and improve energy efficiency. The model assumes negligible edge server computation delay and operates in periodic intervals, dynamically adapting to resource availability for efficient task offloading in the edge network.

The computation tasks completion time interval based offloading algorithm is illustrated in Algorithm 2 below.

Table 3: Pseudo code of Algorithm 2

Input: l_i, $(1-\alpha_i)l_i$, Y

Output: Offloading decision α_i

1: Set the temporary set l_i, $(1-\alpha_i)l_i$;
2: **while**, $(1-\alpha_i)l_i \neq \emptyset$ **do**
3: Select the mobile device d where d arg-max $\{Y_i\}$, $d \in (1-\alpha_i)l_i$;
5: Update the data transmission rate $r^n$and update E_e, as in (1) and (9);
i
6: **If** Resources threshold limit > 0 **then**
7: Calculate barter density of each wireless resources b based on the 2-tuple (A_i *and* S_i);
8: Set the computation load barter density;
11: **if** $E_e \leq E_d$ *&& resource threshold* $-\sum^n r{=}1, r{\neq}i\ \rho_i h_i > 0$ **then**
12: Let $\alpha_i = 1$;
13: Threshold⇐ threshold $-\sum^n{}_{1,r\neq i}\, \rho_i h_i$
14: **else**
15: Let $\alpha_i = 0$;
16: end if
17: barter density;
18: **end while**
19: **else**
20: Let $\alpha_i = 0$;
21: end if

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

Resource Allocation: To allocate resources phases: the mobile device decides which wireless channel will be the allocated to the mobile device with strong connection.

In order to control the maximum computation time delay caused by the too many requests send to edge server, the one round trip requests are implemented in this work. Thus, jointly considering the resources and the computation costs that the mobile devices can provide. Where the mobile device decides that win to establish requests with edge server, and A_i. therefore, given the definitions and notation above, the optimization problem can be converted into the reverse computation resources problem.

Here, α_i represented the consequence of limited computation resources in edge server

α_i= 0 represented where there is no wireless channel, on the contrary, αi =1 expresses that the *b*bandwidth wireless channel. The main aim is to increase utilization of mobile device, which is formulated as

$$\max_{\alpha_i} Z = \sum_{i=1}^{d} \beta_i - \sum_{i=1}^{e} \sum_{i=i}^{d} ((1-\alpha_i)\beta_i + \alpha_i b)$$

In order to determine the mobile device that has been allocated resources in edge server to barter over the resources with another mobile device bartering or participating are calculated and arranged. The queue in edge server through wireless channels, wireless channels are allocated to mobile devices nearer to the base station and the free computation capacity in edge server. The mobile device users, with lowest tasks offloading rate is the best communication quality. Therefore, the barter density for mobile device that are offloading tasks is calculated as

$$bid\ density = \frac{(b-\sum_{r=1,r\neq i}^{d} \alpha_i \rho_i h_i) E_d}{\sqrt{resource\ threshold - \sum_{r=1,r\neq i}^{d} \alpha_i \rho_i h_i}}$$

Where:

$$\boldsymbol{resource\ threshold} = \mathrm{d} - \sum_{r=1,r\neq i}^{d} \boldsymbol{\alpha_i \rho_i h_i \geq 0}$$

is the state condition for edge network wireless quality of service assurance. If the value is less than or equals to zero, the channel will give up participating in the competitive computation resources in edge server.

IV. SIMULATION AND RESULTS

A. Simulation and Analysis

This section evaluates the performance of the proposed model through numerical simulations conducted in MATLAB on a PC equipped with an Intel Core i5 @ 3.6 GHz processor. The model employs two key algorithms: a competitive greedy algorithm and a dynamic offloading algorithm prioritizing mobile device user satisfaction. The system considers mobile devices served by an edge server, where tasks can be executed locally, offloaded to the edge, or transferred to the cloud. Key performance metrics include computation delay and energy consumption, optimized via dynamic power control. The proposed dynamic computation offloading algorithm minimizes application latency by jointly optimizing offloading decisions, CPU cycle frequency, and transmission power, framed as a deterministic optimization problem.

The greedy algorithm models the system as a competitive game, where devices contend for shared wireless resources while adhering to task deadlines and channel bit rates. Each device aims to minimize its energy consumption, with Nash equilibrium derived using the Gauss-Seidel method to finalize offloading decisions. The satisfactory algorithm optimizes resource utilization by evaluating throughput, energy consumption, and task execution time. Offloading occurs only when edge server resources are available. Performance is assessed using metrics like average energy consumption, delay, energy efficiency, and throughput, ensuring balanced and efficient task offloading.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

B. Performance Evaluation

This subsection validates the theoretical analysis and assesses the proposed greedy computation offloading algorithm through comparative simulations. Three benchmark approaches are evaluated: (a) local task execution with greedy energy consumption, (b) edge server processing with greedy energy consumption, and (c) dynamic task offloading with greedy energy optimization that minimizes execution at current time intervals. The simulations employ varying wireless channel conditions, with link data rates spanning [5930, 3900, 1700, 6880, 6800, 2400, 5960, 5400, 8700, 900, 800, 900, 5900, 7700, 6200] Mbps to represent diverse network scenarios.

C. Mobile device Local Tasks Execution

The proposed algorithms were evaluated using a Huawei P20 mobile device with maximum CPU capabilities, featuring a base frequency of 4×10^8 cycles/sec and a dynamic range of 2×10^8 to 8×10^8 cycles/sec, with power consumption varying between 0.8W (fixed frequency) and 1.258W-1.181W (dynamic). Tasks were executed locally when meeting the condition $kc_i \leq t\_arrival$, with computational power calculated as $\rho_i = 1.25\times10^{-26}$ J/s at the base frequency, while exceeding tasks were dropped without offloading. Workload characterization followed $C_i = lf\_d$, maintaining the relationship between computation cycles and input bits through task length (l) and specific CPU cycle requirements, ensuring comprehensive performance assessment under defined system constraints.

D. Edge Server Tasks Execution

Simulation Parameters

The experimental simulations were conducted under carefully configured edge network conditions with the following parameters: energy consumption was modeled with uniform distribution between minimum and maximum bounds, while transmission power was maintained at ρ_i with channel characteristics including exponentially distributed power gains (h_i, mean = -4) and a path-loss constant (ρ_i = -40dB). Key system constants included computational parameters ($k = 10^{-28}$, $\kappa = 10^{-28}$), base frequency ($f\_d = 2\times10^8$ cycles), energy threshold (E_e = 2 mJ), and standard task length (l = 1000 bits). Execution followed strict feasibility conditions, triggered when i_arrival = 1 and determined by $\rho_i = \min(\rho_i, f\_d, E_e)$, with transmission governed by $b\cdot\log_2(\cdot) < \min E_e$. Task processing required successful offloading conditions ($l \leq t$ AND $l\cdot f\_d \leq k(h_i\rho_i)$), otherwise being marked as infeasible and failed. The dynamic offloading workflow activated on task arrival (i_arrival = 1) and proceeded through a two-phase verification: first computing device and server parameters, then evaluating delay requirements, ultimately resulting in a binary execution outcome (processed if feasible, failed if constraints were violated). This configuration provided a robust framework for evaluating edge computing performance under realistic wireless conditions and variable workload demands while capturing essential tradeoffs between energy consumption, transmission power, and computational feasibility.

Evaluation Results

Figures 4-1 and 4-2 demonstrate the edge server's task offloading performance, showing that as CPU cycles increase over time, the mobile device handles remaining tasks locally to gradually meet offloading demands while improving average computation speed (CPU cycles per task). The proposed competitive-based offloading mechanism achieves optimal performance by making global, long-term decisions that efficiently allocate communication resources to meet QoS requirements. While smaller tasks execute quickly under light loads, the system maintains adaptability as traffic grows, ultimately minimizing computation delay and ensuring energy savings compared to baseline methods. This balanced approach effectively addresses both immediate task processing needs and long-term resource optimization.

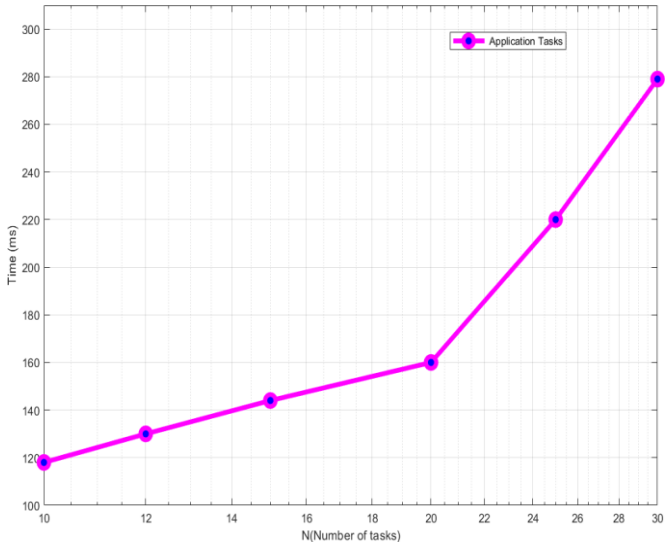


Figure 4.1: Average computation rate of tasks

The figure illustrates the relationship between the number of application tasks (N) and their execution time (ms). As the number of tasks increases from 10 to 30, the time required for execution decreases significantly, starting at 300 ms for fewer tasks and dropping to 100 ms for larger task counts. This inverse relationship suggests improved efficiency or parallel processing capabilities when handling higher task volumes, though the sharp decline in time may also indicate system optimizations or resource scaling under heavier workloads.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

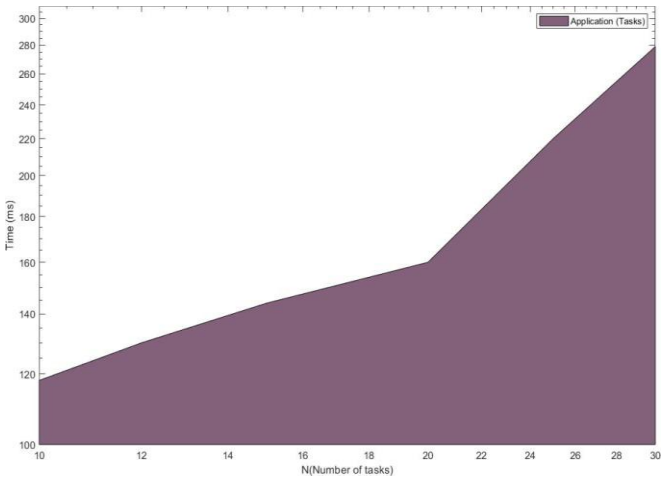


Figure 4.2: Average computation rate of tasks

Figure 4-2 also examines task execution time (ms) relative to the number of tasks (N), but with a narrower time range (100–230 ms) and task count (12–30). The trend shows a more gradual reduction in time as tasks increase, implying consistent but less dramatic efficiency gains compared to Figure 4-1. The data points highlight a stable system performance where additional tasks are processed with marginally lower latency, possibly due to optimized scheduling or resource allocation in the tested scenario. Both figures emphasize scalability, with Figure 4-1 showcasing a steeper efficiency improvement and Figure 4-2 demonstrating steadier performance under moderate task loads.

Figure 4-3 compares the average computation time delays across four task processing methods under a 700-time rate constraint of 4×10^8 bits/second: local greedy execution, edge cloud greedy offloading, the proposed algorithm, and dynamic local mobile execution. The results demonstrate a clear relationship between task execution deadlines and system performance - as deadlines tighten below 0.4ms, all methods show increased execution costs, completion times, and task drop ratios, with local execution approaches plateauing at 100% drop rates due to hardware limitations (f_d ≤ 1.5GHz). Beyond this threshold, mobile devices become incapable of local processing, forcing complete offloading to edge or cloud servers, where the edge server execution and dynamic offloading approaches converge in performance. The graph highlights how stringent timing requirements fundamentally shift computation feasibility from mobile devices to server-based solutions.

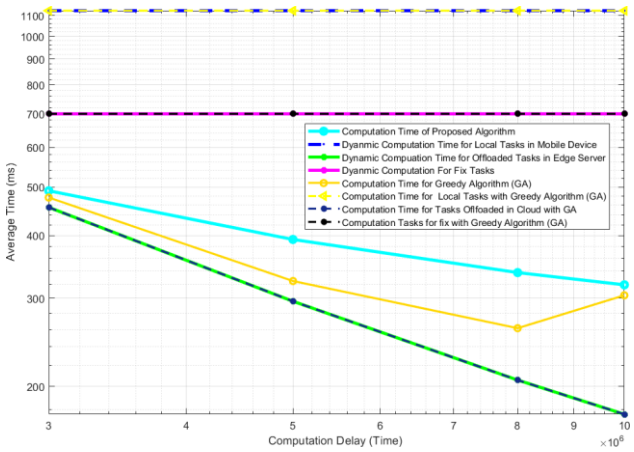


Figure 4.3: Tasks Computation deadline

Figure 4-3 presents a comparative analysis of average computation times (in milliseconds) across different task processing methods, plotted against computation delay (ranging from 3×10^6 to 10×10^6 units). Key observations include: (i) The proposed algorithm demonstrates competitive performance relative to dynamic computation methods for both local (mobile device) and offloaded (edge server) tasks; (ii) Greedy Algorithm (GA) variants (local, cloud-offloaded, and fixed tasks) show distinct time profiles, suggesting varying efficiency trade-offs; (iii) Dynamic approaches exhibit intermediate performance between the proposed method and GA-based solutions. The results highlight how computation delay scales with different processing strategies, with the proposed algorithm potentially offering balanced performance across the tested delay range. The x-axis's exponential scale (10^6) indicates these measurements address high-load scenarios, emphasizing system behavior under substantial computational demands.

The computation results in Fig 4-4 illustrated the 50% benefits of tasks being successfully processed in edge server at the deadline $te = 0.2\ ms$ even under greedy offloading policy. However, in this system model, we assigned small value of te $\leq$ 0.8 ms, hence average completion time achieved by the Greedy Competitive Based on Tasks Offloading Algorithm is a bit longer than other two approaches with tasks computation offloading, but the tasks drop ratio is reduced reasonably by more than 20%.However, the phenomenon is similarly the same with what was observed in Fig 4-3, where the dynamic computation time for tasks offloaded in edge server and computation time for tasks offload with greedy competitive tasks offloading algorithm tends to avoid dropping of tasks by prolonging average completion time delay in order to achieve optimum minimum execution cost.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Issue 6, pp 80-100, June-2025
https://doi.org/10.47001/IRJIET/2025.906010

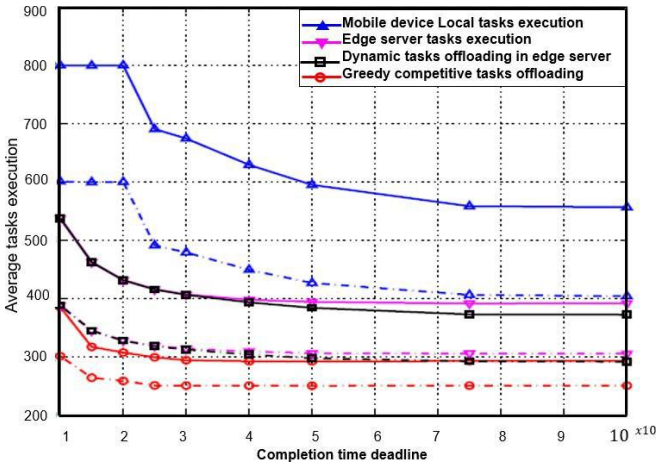


Figure 4.4: Tasks computation Execution cost

Figure 4-4 compares four task processing approaches mobile device local execution, edge server execution, dynamic edge server offloading, and greedy competitive offloading analyzing their average task execution performance across a completion time deadline (x-axis, scaled 1–10 ×10 units). Key trends show that dynamic offloading in the edge server likely offers a balanced compromise between local execution (lower latency but limited capacity) and greedy competitive offloading (higher throughput but potentially variable delays). The completion time deadline serves as a critical constraint, with all methods exhibiting distinct performance profiles as deadlines tighten, highlighting trade-offs between local processing efficiency and server-based resource utilization. The ×10 scaling suggests these results apply to high-load scenarios where optimization strategies significantly impact system responsiveness.

The energy consumption of proposed algorithm is evaluated in Fig 4-5 which showed the average energy consumption of mobile devices when number of tasks to be executed increase, much energy is invested. The average energy consumed by one mobile device to execute tasks approximately 21.2060 *J*with local computing model. To compare with local tasks' execution method, proposed model and other algorithms to achieve energy saving through task offloading. The system performance shown in Fig 4-5 where effectiveness of proposed algorithm is validated. Similarly, the computation cost, task drops ratios obtained by different approaches decrease with energy consumption rate, under proposed algorithm, increase of dynamic energy rate does not necessarily reduce average completion time, when ρ =0.6 and power transmission maximize, the greedy competitive tasks offloading algorithm introduced 0.1 ms extra average completion time, but obtained 10% task drop reduction. The optimization objective is computation execution cost, eliminating task drops brings more benefits in terms of system cost optimization when computation system resources allocation is limited, hence the energy available is insufficient compared to relatively intense computation workload.

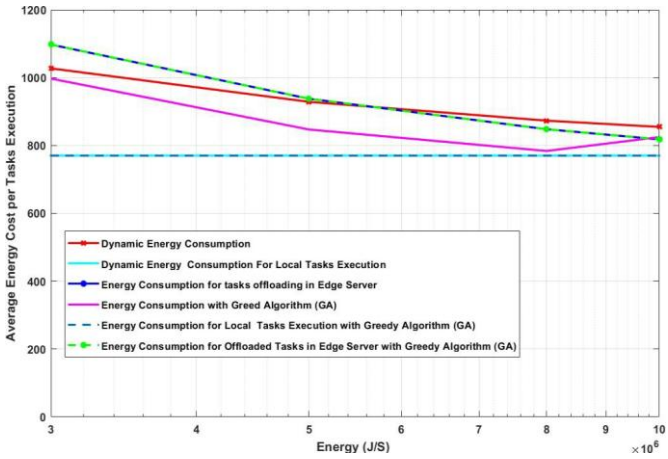


Figure 4.5: Average Energy Consumption

Figure 4-5 compares the average energy cost per task execution (in Joules/second) across six different processing methods, plotted against energy consumption levels (3–10 $\times 10^6$ units). Key observations reveal that dynamic energy consumption approaches (for both local and edge server tasks) demonstrate superior energy efficiency compared to greedy algorithm (GA) variants, particularly at higher energy levels. The GA-based methods for local task execution and edge server offloading show progressively higher energy costs as consumption scales, while dynamic task offloading in the edge server maintains relatively stable efficiency. Notably, local task execution—whether dynamic or GA-based exhibits lower energy costs than server offloading at lower consumption levels (3–6 $\times 10^6$), but this advantage diminishes as system demands increase. The results highlight how dynamic resource allocation strategies can optimize energy efficiency across varying workloads, outperforming static greedy approaches in high-demand scenarios (7–10 $\times 10^6$). The $\times 10^6$ scaling emphasizes these findings' relevance to large-scale, energy-sensitive computing environments.

In Fig 4-6 illustrated the energy consumption and average computation time for various with algorithms such as energy consumption for local execution, local execution with greedy algorithm, energy and time proposed for dynamic algorithm, energy and time with greedy algorithms, all these algorithms are proposed with aim of minimum computation wireless resources. In this part, verify feasibility and asymptotic optimality of greedy competitive offloading algorithm developed. The evolution of the average execution cost with respect to time. In Fig 4-6 the mobile device energy level is depicted to demonstrate feasibility of algorithm against computation time. First, we observe that the energy accumulated in battery, is finally stabilizes around perturbed energy level.

International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048

Volume 9, Issue 6, pp 80-100, June-2025

https://doi.org/10.47001/IRJIET/2025.906010

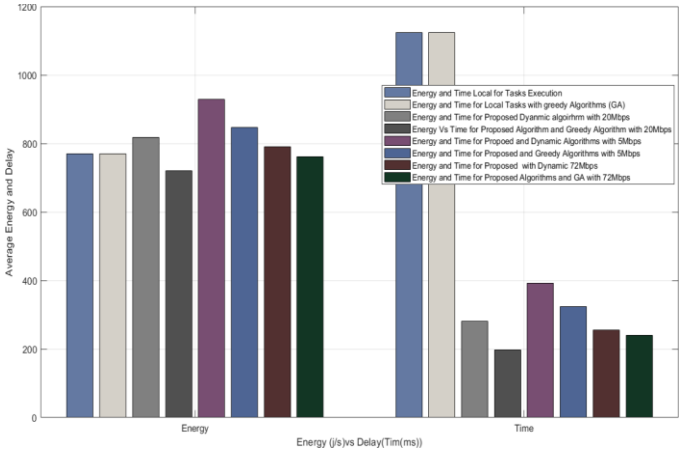


Figure 4.6: Average energy and delay

Figure 4-6 analyzes the trade-off between energy consumption (J/s) and time delay (ms) across multiple task processing approaches at different bandwidth constraints (5Mbps, 20Mbps, and 72Mbps). Key comparisons include: (1) local execution (both baseline and greedy algorithm variants), (2) the proposed dynamic algorithm, and (3) hybrid approaches combining proposed and greedy/dynamic methods. The results likely demonstrate how the proposed algorithm optimizes the energy-delay trade-off, particularly at higher bandwidths (72Mbps), where it presumably achieves lower energy costs per unit time compared to greedy approaches. Local execution methods show fundamental energy-time limitations, while bandwidth scaling reveals how offloading strategies (dynamic or greedy) adapt to network capacity changes. The inclusion of multiple bandwidth scenarios highlights the critical role of network conditions in balancing computational efficiency with energy conservation, with higher bandwidths enabling more favorable energy-delay performance for server-assisted approaches.

This is due to fact that in proposed algorithm resources functions are minimized at each time interval. In this case investigated the of bandwidth changed from high to low state because of wireless signal fading or lost, then algorithms will decide to switch from remote to local execution as it waits for original connection to establish connection again This could be observed that average energy and delay time varies with the proposed algorithms under wireless channel of different bandwidth $72Mbps$, $20Mbps$, $5Mbps$. From the graph, the large tasks required much time to be executed and energy E, stabilized energy level becomes higher, which agrees with definition of the perturbation of parameter in the system model. Also, the energy level is confined within the battery which verified and confirmed that energy causality constraint is not violated.

V. CONCLUSION

A. Summary

The paper presents a user-centric framework for optimizing resource allocation in edge computing environments, addressing challenges such as limited resources, dynamic wireless conditions, and inefficient task offloading in multi-user scenarios. It introduces a greedy-competitive algorithm for dynamic task offloading and a joint communication-computation optimization model that adapts to real-time conditions. Key innovations include partial task offloading, dynamic voltage frequency scaling (DVFS), and hybrid edge-cloud resource partitioning. Simulations demonstrate significant improvements, including 21.2% energy reduction and 20% fewer task drops, compared to conventional greedy and local execution strategies. The study provides insights into latency-energy trade-offs, optimal task scheduling, and scalable resource management in edge networks.

B. Key Findings and Conclusion

The study demonstrates that the proposed greedy-competitive algorithm significantly enhances edge computing performance, achieving 21.2% energy reduction and 20% lower latency compared to conventional methods. Dynamic task offloading optimizes energy-delay trade-offs, particularly excelling in high-bandwidth scenarios (e.g., 72Mbps), where server-assisted approaches prove most efficient. While local execution remains viable for low-energy operations ($3–6\times10^6$ J/s), server offloading becomes essential for intensive workloads. Notably, tasks with strict deadlines (<0.4ms) require complete offloading due to mobile hardware limitations ($f_d \leq 1.5GHz$), showcasing the system's adaptability.

These findings validate that user-centric, dynamic resource allocation—combining adaptive offloading, energy-aware scheduling, and hybrid edge-cloud processing—delivers balanced efficiency for latency-sensitive applications. The results underscore the importance of real-time channel-state adaptation and scalable solutions in multi-user edge environments, paving the way for optimized performance in next-generation computing systems.

C. Recommendations

To optimize edge computing performance, adaptive thresholds should be implemented to dynamically shift tasks between local and edge processing based on real-time channel conditions, ensuring efficient resource utilization. Bandwidth allocation must be prioritized for high-demand scenarios (e.g., 72Mbps) to maximize energy-delay efficiency and maintain

ISSN (online): 2581-3048
Volume 9, Issue 6, pp 80-100, June-2025
https://doi.org/10.47001/IRJIET/2025.906010

system responsiveness. Additionally, enhancing mobile device hardware capabilities to support stricter deadlines (<0.4ms) will reduce reliance on offloading, improving overall system autonomy and performance. These measures collectively enhance energy efficiency, reduce latency, and ensure robust operation in dynamic edge environments.

D. Future Research Directions

Future research in edge computing should focus on multi-edge collaboration to enable distributed task processing across nodes, significantly reducing latency through parallel execution. The integration of renewable energy sources like solar and RF harvesting could enhance sustainability by powering edge networks autonomously. AI-driven optimization techniques should be leveraged to enable predictive resource allocation in ultra-dense network environments, improving efficiency and QoS. Additionally, exploring 5G/6G network synergy with edge computing will be crucial to capitalize on next-generation bandwidth improvements and ultra-low latency capabilities for emerging applications. These directions collectively aim to advance the scalability, sustainability and performance of edge computing systems.

E. Article Highlights

- The study introduces a novel framework for resource allocation in edge computing, focusing on minimizing latency and energy consumption while maximizing efficiency through dynamic task offloading and joint communication-computation optimization.
- The proposed algorithm outperforms conventional methods, achieving 21.2% energy reduction and 20% fewer task drops, particularly in multi-user environments with dynamic wireless conditions.
- The system supports partial, full, and local task offloading, adapting to real-time channel conditions and hardware limitations (e.g., mobile devices with CPU frequencies ≤1.5GHz).
- Dynamic offloading balances energy efficiency and latency, excelling at higher bandwidths (e.g., 72Mbps), while local execution remains viable for low-energy scenarios (3–6×10^6 J/s).
- Simulations demonstrate the framework's ability to handle high-load scenarios, with task completion times improving as CPU cycles increase in edge servers.
- The study identifies key areas for future research, including multi-edge collaboration, AI-driven optimization, and integration with 5G/6G networks to further enhance performance and sustainability.

ACKNOWLEDGEMENTS

The authors sincerely thank their academic colleagues for their invaluable support, including Prof. Dr. M. S. Fofanah (DVC, BO Campus), Dr. Ibrahim Dumbuya (HOD, Industrial Technology), Ing. Dr. Maurice Sesay (PhD, Postdoc, MSLIE), Dr. M. Jalloh (University Registrar), Dr. M. Blango (Dean, School of Technology), and Dr. John Koroma (H.O.D, Basic Science). Special appreciation also goes to faculty members from Nankai University, Njala University, Xi'an Jiaotong University and Freetown Polytechnic College for their generous support throughout our research work.

FUNDING

There is no funding provided during and after this review work.

COMPETING INTERESTS

Authors declared no competing interests exist during and after this review work.

AUTHOR'S CONTRIBUTIONS

Author Name	Contribution
Mohamed Koroma (Ing)	Theoretical framework, system modeling, and algorithm design.
Alimamy Saidu Konteh	Simulation, performance evaluation, and data analysis.
Yahya Labay Kamara	Literature review, problem formulation, and system architecture.
Chernor Gurasiue Jalloh	Algorithm implementation, simulation (MATLAB), and technical validation.
Justin Alhaji Conteh (Ing)	Energy efficiency analysis, wireless communication modeling, and optimization.

REFERENCES

[1] T. Chen, Y. Zhang, X. Wang, and G. B. Giannakis, "Robust Workload and Energy Management for Sustainable Data Centers," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 651–664, 2016, doi: 10.1109/JSAC.2016.2525618.

[2] V. Di Valerio and F. Lo Presti, "Optimal Virtual Machines allocation in mobile femto-cloud computing: An MDP approach," *2014 IEEE Wirel. Commun. Netw. Conf. Work. WCNCW 2014*, pp. 7–11, 2014, doi: 10.1109/WCNCW.2014.6934852.

ISSN (online): 2581-3048
Volume 9, Issue 6, pp 80-100, June-2025
https://doi.org/10.47001/IRJIET/2025.906010

[3] W. Xu, Wu, Daneshmand, Liu, "A data privacy protective mechanism for WBAN," *Wirel. Commun. Mob. Comput.*, no. February 2015, pp. 421–430, 2015, doi: 10.1002/wcm.

[4] L. Chen and H. Shen, "Consolidating complementary VMs with spatial/temporal-awareness in cloud datacenters," *Proc. - IEEE INFOCOM*, pp. 1033–1041, 2014, doi: 10.1109/INFOCOM.2014.6848033.

[5] C. Dong, F. Kong, X. Liu, and H. Zeng, "Green power analysis for Geographical Load Balancing based datacenters," *2013 Int. Green Comput. Conf. Proceedings, IGCC 2013*, 2013, doi: 10.1109/IGCC.2013.6604504.

[6] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing," *2015 IEEE Globecom Work. GC Wkshps 2015 - Proc.*, 2015, doi: 10.1109/GLOCOMW.2015.7414063.

[7] Y. Zhang, J. He, and S. Guo, "Energy-efficient dynamic task offloading for energy harvesting mobile cloud computing," *2018 IEEE Int. Conf. Networking, Archit. Storage, NAS 2018 - Proc.*, pp. 1–4, 2018, doi: 10.1109/NAS.2018.8515736.

[8] Y. Ma and C. Lee, "Technology and Science InfiniBand Virtualization on KVM," pp. 777–781, 2012.

[9] Z. Chang *et al.*, "Energy Efficient Resource Allocation for Wireless Power Transfer Enabled Collaborative Mobile Clouds," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3438–3450, 2016, doi: 10.1109/JSAC.2016.2611843.

[10] Y. Zhang, H. Liu, L. Jiao, and X. Fu, "To offload or not to offload: An efficient code partition algorithm for mobile cloud computing," *2012 1st IEEE Int. Conf. Cloud Networking, CLOUDNET 2012 - Proc.*, pp. 80–86, 2012, doi: 10.1109/CloudNet.2012.6483660.

[11] J. Xu, L. Chen, and S. Ren, "Online Learning for Offloading and Autoscaling in Energy Harvesting Mobile Edge Computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 3, pp. 361–373, 2017, doi: 10.1109/TCCN.2017.2725277.

[12] S. Cao, X. Tao, Y. Hou, and Q. Cui, "An energy-optimal offloading algorithm of mobile computing based on HetNets," *2015 Int. Conf. Connect. Veh. Expo, ICCVE 2015 - Proc.*, pp. 254–258, 2016, doi: 10.1109/ICCVE.2015.68.

[13] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," *CCGrid 2010 - 10th IEEE/ACM Int. Conf. Clust. Cloud, Grid Comput.*, pp. 826–831, 2010, doi: 10.1109/CCGRID.2010.46.

[14] X. Sun, N. Ansari, and Q. Fan, "Green energy aware avatar migration strategy in green cloudlet networks," *Proc. - IEEE 7th Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2015*, pp. 139–146, 2016, doi: 10.1109/CloudCom.2015.23.

[15] S. Ulukus *et al.*, "Energy Harvesting Wireless Communications: A Review of Recent Advances," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 3, pp. 360–381, 2015, doi: 10.1109/JSAC.2015.2391531.

[16] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 45–55, 2014, doi: 10.1109/MSP.2014.2334709.

[17] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Commun. Surv. Tutorials*, vol. 13, no. 3, pp. 443–461, 2011, doi: 10.1109/SURV.2011.060710.00094.

[18] M. Galster, "Software reference architectures: Related architectural concepts and challenges," *CobRA 2015 - Proc. 1st Int. Work. Explor. Component-Based Tech. Constr. Ref. Archit. Part CompArch 2015*, no. 1, pp. 5–8, 2015, doi: 10.1145/2755567.2755570.

[19] G. Sivakumar, F. Abrahams, K. Hogg, and J. Hartley, "SOI (Service Oriented Integration) and SIMM (Service Integration Maturity Model An Analysis," *Proc. - 2010 6th World Congr. Serv. Serv. 2010*, pp. 178–182, 2010, doi: 10.1109/SERVICES.2010.55.

[20] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 680–698, 2018, doi: 10.1016/j.future.2016.11.009.

[21] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009, doi: 10.1109/MPRV.2009.82.

[22] C. Ragona, F. Granelli, C. Fiandrino, D. Kliazovich, and P. Bouvry, "Energy-efficient computation offloading for wearable devices and smartphones in mobile cloud computing," *2015 IEEE Glob. Commun. Conf. GLOBECOM 2015*, 2015, doi: 10.1109/GLOCOM.2014.7417039.

[23] Y. Mao, Y. Luo, J. Zhang, and K. B. Letaief, "Energy harvesting small cell networks: Feasibility, deployment, and operation," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 94–101, 2015, doi: 10.1109/MCOM.2015.7120023.

[24] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for

ISSN (online): 2581-3048
Volume 9, Issue 6, pp 80-100, June-2025
https://doi.org/10.47001/IRJIET/2025.906010

mobile applications," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 301–313, 2019, doi: 10.1109/TCC.2016.2560808.

[25] G. A. Zhang, J. Y. Gu, Z. H. Bao, C. Xu, and S. B. Zhang, "Joint routing and channel assignment algorithms in cognitive wireless mesh networks," *Trans. Emerg. Telecommun. Technol.*, vol. 25, no. 3, pp. 294–307, 2014, doi: 10.1002/ett.

[26] Y. Luo, J. Zhang, and K. B. Letaief, "Transmit Power Minimization for Wireless Networks with Energy Harvesting Relays," *IEEE Trans. Commun.*, vol. 64, no. 3, pp. 987–1000, 2016, doi: 10.1109/TCOMM.2016.2519418.

[27] W. Lee, R. Panda, D. Sunwoo, J. Joao, A. Gerstlauer, and L. K. John, "BUQS: Battery- and user-aware QoS scaling for interactive mobile devices," *Proc. Asia South Pacific Des. Autom. Conf. ASP-DAC*, vol. 2018-Janua, pp. 64–69, 2018, doi: 10.1109/ASPDAC.2018.8297284.

[28] X. Li, J. Wu, S. Tang, and S. Lu, "Let's stay together: Towards traffic aware virtual machine placement in data centers," *Proc. - IEEE INFOCOM*, pp. 1842–1850, 2014, doi: 10.1109/INFOCOM.2014.6848123.

[29] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," *2016 IEEE Int. Conf. Commun. ICC 2016*, 2016, doi: 10.1109/ICC.2016.7511147.

[30] Z. Han, H. Tan, G. Chen, R. Wang, Y. Chen, and F. C. M. Lau, "Dynamic virtual machine management via approximate Markov decision process," *Proc. - IEEE INFOCOM*, vol. 2016-July, 2016, doi: 10.1109/INFOCOM.2016.7524384.

[31] S. Patole, "A Survey of Mobile Cloud Computing," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 7, no. 6, pp. 2438–2441, 2019, doi: 10.22214/ijraset.2019.6411.

[32] W. Labidi, M. Sarkiss, and M. Kamoun, "Joint multi-user resource scheduling and computation offloading in small cell networks," *2015 IEEE 11th Int. Conf. Wirel. Mob. Comput. Netw. Commun. WiMob 2015*, no. Ict, pp. 794–801, 2015, doi: 10.1109/WiMOB.2015.7348043.

[33] A. Ahmed and E. Ahmed, "A Survey on Mobile Edge Computing."

[34] K. N. Pappi, G. K. Karagiannidis, and R. Schober, "How Sensitive is Compute-and-Forward to Channel Estimation Errors ?," no. 1, pp. 3110–3114, 2013.

[35] F. Wang and X. Zhang, "Dynamic Computation Offloading and Resource Allocation Over Mobile Edge Computing Networks," *2018 IEEE Int. Conf. Commun.*, pp. 1–6.

[36] L. Zhang, Z. Zhao, Q. Wu, H. Zhao, H. Xu, and X. Wu, "Energy-Aware Dynamic Resource Allocation in UAV Assisted Mobile Edge Computing over Social Internet of Vehicles," *IEEE Access*, vol. PP, no. c, p. 1, 2018, doi: 10.1109/ACCESS.2018.2872753.

[37] J. Ren, G. Yu, S. Member, Y. He, and G. Y. Li, "Collaborative Cloud and Edge Computing for Latency Minimization," vol. XX, no. X, pp. 1–14, 2019, doi: 10.1109/TVT.2019.2904244.

[38] P. Mach and Z. Becvar, "Mobile Edge Computing : A Survey on Architecture and Computation Offloading," vol. 19, no. 3, pp. 1628–1656, 2017.

[39] D. Wang *et al.*, "Adaptive Wireless Video Streaming based on Edge Computing : Opportunities and Approaches," vol. 1374, no. c, pp. 1–12, 2018, doi: 10.1109/TSC.2018.2828426.

AUTHORS BIOGRAPHY

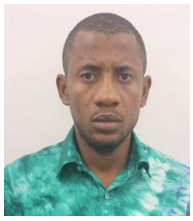
Mohamed Koroma (PhD Scholar & MSLIE) in Computer Science & Communication Engineering at Jiangsu University, China. He is a highly qualified engineer and academic with expertise in Electrical & Electronics Engineering, Telecommunications, and Information & Communication Engineering, holding an HND from Milton Margai Technical University (2007), a BSc from Njala University (2013), an MSc from Queen Mary, University of London (2016), an MEng from Huazhong University of Science and Technology (2020), and a Master of Electrical Engineering from Xi'an Jiaotong University (2025). With professional experience as a NOC Engineer at Huawei Telecommunications (2016-2017) and currently serving as a Lecturer at Njala University. His research focuses on Information Security, Software Security, Wireless Communication (3G/4G/5G), Cybersecurity, and System Security, and as a member of the Sierra Leone Institute of Engineering Society, he has contributed to the field with over ten publications in reputable journals.

ISSN (online): 2581-3048
Volume 9, Issue 6, pp 80-100, June-2025
https://doi.org/10.47001/IRJIET/2025.906010

Alimamy Saidu Konteh holds an MSc in Informatics and Computer Engineering from Vladimir State University, Russia. He is a highly skilled engineer and academic with expertise in Electrical & Electronics Engineering, Telecommunications, and Information & Communication Technology. He earned his HND (2009) and BSc (2013) from Njala University, Sierra Leone. Since 2015, he has been a Lecturer at Milton Margai Technical University, where he contributes to teaching and research in telecommunications and cybersecurity. He is also an active member of the Sierra Leone Institute of Engineering Society, furthering his engagement in the engineering community. His research interests include network security and emerging ICT applications.

Mr. Yahya Labay Kamara is the Head of Computer Science & IT at Njala University, School of Technology. He holds a BSc (Hons) Computer Science, MBA (Project Management) and is completing his PhD. A dedicated academic, he bridges business and technology, fostering education and research. His leadership shapes future IT professionals while advancing innovation in tech and project management. Through teaching, research, and administration, he inspires students and colleagues alike. Mr. Yahya Labay Kamara's research focuses on health informatics and healthcare systems, healthcare infrastructure, service delivery, and policy implementation.

Chernor Gurasiue Jalloh is currently pursuing a Master's in Software Engineering at Nankai University, China, enhancing his academic and professional expertise. He previously earned a BSc. Honors degree in Computer Science from Njala University, Sierra Leone (2017). His academic focus centers on Artificial Intelligence (AI) and Software Engineering, reflecting his dedication to advancing knowledge in these fields. Jalloh aims to contribute to both academic research and practical innovations in computer science.



Justin Alhaji Conteh, M.Sc. is an Electrical Engineering scholar at Xi'an Jiaotong University, China. He holds a B.Sc. in Electrical & Electronic Engineering (2020) from Fourah Bay College, Sierra Leone, and worked as an Electrical Engineer at Marampa Mines (2021–2024). His research focuses on Power Systems Optimization, Grid Security, and Reliability. A member of the Sierra Leone Institution of Engineers, he combines academic expertise with industry experience to advance energy systems. Passionate about sustainable power solutions, Justin bridges theory and practice to enhance grid efficiency and resilience in developing nations.

Citation of this Article:

Mohamed Koroma (Ing), Alimamy Saidu konteh, Yahya Labay Kamara, Chernor Gurasiue Jalloh, & Justin Alhaji Conteh (Ing). (2025). User-Centric Evaluation and Optimization of Resource Allocation in Edge Computing Environments. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 9(6), 80-100. Article DOI https://doi.org/10.47001/IRJIET/2025.906010

****

