IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 March 2025, accepted 16 May 2025, date of publication 21 May 2025, date of current version 2 June 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3572084

RESEARCH ARTICLE

Novel Unsupervised Cluster Reinforcement Q-Learning in Minimizing Energy Consumption of Federated Edge Cloud

GURUH FAJAR SHIDIK[1], (Member, IEEE), OKI SETIONO[2], EDI JAYA KUSUMA[2], L. BUDI HANDOKO[1], PULUNG NURTANTIO ANDONO[1], AND MOHD FAIZAL ABDOLLAH[3]

[1]Faculty of Computer Science, Universitas Dian Nuswantoro, Semarang 50131, Indonesia
[2]Faculty of Health Science, Universitas Dian Nuswantoro, Semarang 50131, Indonesia
[3]Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia Melaka, Melaka 76100, Malaysia

Corresponding author: Guruh Fajar Shidik (guruh.fajar@research.dinus.ac.id)

This work was supported by the Intelligent Distributed Surveillance and Security (IDSS) of Universitas Dian Nuswantoro, with financial support from Kementerian Pendidikan Tinggi, Sains, dan Teknologi (KEMENDIKTISAINTEK) of Indonesia.

ABSTRACT As global demand for cloud computing rises, green computing has become essential. Federated Edge Cloud (FEC) offers improved energy efficiency compared to traditional infrastructures, yet managing distributed energy consumption remains a challenge. This research introduces an Unsupervised Cluster Reinforcement Q-Learning method in FEC (UCRL-FEC), which integrates Fuzzy C-Means (FCM) or K-Means clustering to identify migratable Virtual Machines (VMs) from overloaded hosts. The method enhances energy efficiency and workload balance by incorporating a modified reward function in Q-Learning. Experimental evaluations demonstrate that UCRL-FEC reduces energy consumption (EC) up to 1.07%, supporting reductions in both operational costs and greenhouse gas emissions, which is critical for large-scale cloud environments. In terms of Service Level Agreement Time per Active Host (SLATAH), UCRL-FEC achieves an improvement up to 1.56% over the baseline method, demonstrating enhanced efficiency in managing active host resources. Additionally, system stability improves with a reduction of up to 9.68% in Performance Degradation due to Migration (SLA-PDM), effectively minimizing service disruptions and ensuring efficient workload management. Furthermore, the method reduces overall Service Level Agreement Violations (SLAV) up to 6.06%, indicating enhanced service reliability and optimized resource allocation. A Friedman test confirms statistically significant improvements in energy efficiency, workload distribution, and system stability over baseline methods. These advancements prevent resource overutilization, enhance workload management, and extend hardware lifespan, fostering sustainable cloud operations. UCRL-FEC balances energy efficiency, performance, and scalability through dynamic resource optimization, validating it as implementable strategy for intelligent VM management in modern cloud-edge infrastructures.

INDEX TERMS Clustering method, energy efficiency, federated edge cloud, Q-learning, quality of service.

I. INTRODUCTION

Cloud computing has transformed communication technologies and distributed systems, providing scalable, secure, and cost-effective solutions for organizations. However, traditional cloud infrastructures have significant limitations, such as high latency caused by centralized processing in distant data centers, which often results in delayed responses for real-time applications. Additionally, centralized architectures are prone to single points of failure, increasing the risk of service disruptions and scalability limitations in handling diverse, geographically distributed workloads. To address

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

© 2025 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

these challenges, the FEC or Federated Edge Cloud has emerged as a framework that integrates edge or fog computing and federated learning to minimize latency and enhance performance by positioning services near end users [1].

FEC strategically places computational and storage resources closer to end users by leveraging edge nodes located near data sources. This strategy substantially decreases latency, improves real-time processing efficiency, and refines overall system performance for applications with strict latency requirements. Real-world examples include autonomous vehicles, where sensors continuously produce high volumes of data that must be processed locally to avoid delays in decision-making, such as sudden braking. Similarly, in remote healthcare monitoring systems, wearable devices must facilitate real-time monitoring of key vital signs to proactively alert medical personnel or patients of potential health concerns. Federated learning in this context supports collaborative model training across distributed nodes without the need to share raw data, thereby preserving patient privacy and minimizing bandwidth usage.

Despite its numerous advantages, efficiently managing resources in FEC remains complicated by the inherent heterogeneity and variability of edge node environments. Existing energy management techniques often struggle to handle distributed and variable workloads, leading to suboptimal energy usage and degraded performance. Several studies have addressed this issue through service scheduling optimization. Jeong et al. [2] formulated an approach to reconfigure service paths, conserving computational resources while refining performance benchmarks, achieving a 21% energy efficiency gain. Similarly, Li et al. [3] developed a dynamic service scheduling approach that reduced energy consumption by 23%. These findings highlight the critical role of service scheduling in improving energy efficiency in FEC environments.

Artificial intelligence (AI) technology offers further opportunities to optimize resource management in FEC. For example, Wang and Yu [4] combined Reinforcement Learning (RL) with federated learning for resource scheduling in edge-based IoT systems, improving decision-making while reducing latency. Another approach [1] using Q-Learning-based RL minimized energy consumption in FEC, achieving a 28% improvement in energy efficiency. In a smart city scenario, for instance, intelligent streetlights equipped with environmental sensors can use RL to adjust lighting intensity based on pedestrian activity and weather conditions, saving energy without compromising user safety.

In addition to scheduling, clustering techniques such as FCM and K-Means methods have been implemented to identify overloaded hosts and redistribute workloads efficiently, resulting in significant energy savings [5], [6]. Clustering enables efficient workload distribution among local servers, reducing energy consumption while improving workload placement and overall system performance.

Addressing the energy efficiency challenges in FEC is not only about technical optimization but also about ensuring sustainability in distributed computing systems. As the demand for faster, more efficient, and environmentally friendly systems grows, innovations that integrate clustering techniques with Reinforcement Learning can redefine energy management in FEC environments. This research endeavors to make a contribution to the development of intelligent, adaptive, and sustainable edge computing systems.

This research introduces a hybrid methodology integrating clustering with Reinforcement Q-Learning for energy-efficient service scheduling in FEC environments, which is further enhanced by a modified logarithmic reward function. Unlike previous methods, this approach addresses the limitations of static resource allocation by dynamically adapting to workload changes in real time, thereby optimizing energy efficiency and QoS. Specifically, it compares the performance of FCM and K-Means in grouping VMs from overloaded hosts. These clusters are then used in a Q-Learning-based scheduling method. The main objective of this study is to achieve energy efficiency and improved QoS in FEC settings through:

- Optimizing VM selection and scheduling through a hybrid approach that integrates clustering techniques (FCM or K-Means) with Reinforcement Q-Learning. This method enhances energy efficiency by identifying migratable VM candidates from overloaded hosts while balancing workloads effectively.
- Improving decision-making in RL Q-Learning by introducing a modified reward function tailored to FEC environments, prioritizing energy efficiency, workload balance, and QoS metrics.

In summary, this research introduces the UCRL-FEC method to optimize VM selection and resource allocation, thereby reducing energy consumption while maintaining QoS in FEC environments. By integrating clustering and RL, this approach aims to mitigate the constraints of energy utilization in distributed edge processing units, offering solutions relevant to various real-world applications.

II. PREVIOUS RESEARCH

Green computing remains a critical concern as the global adoption of cloud computing services continues to rise [7]. Numerous studies have focused on developing energy-efficient cloud infrastructures. Yenugula proposed optimizing cloud servers using the Bacterial Foraging Optimization Algorithm (BFOA) by migrating virtual machines (VMs) from overloaded to underloaded hosts, which led to reduced service costs [8]. Saxena et al. introduced VM scheduling techniques to achieve green computing, utilizing a clustering-based approach for proactive workload prediction and scheduling. Specifically, they employed the K-Medoid method for forecasting VM workloads, effectively minimizing prediction errors and maximizing resource utilization [9].

Beyond load balancing and VM scheduling, architectural advancements in cloud computing have been explored to enhance energy efficiency. Federated Edge Computing (FEC)

combines edge computing with federated learning to improve network performance by enabling decentralized collaboration among edge servers. Studies by Shi et al. and Roman et al. demonstrated FEC's potential to address limitations of traditional cloud computing, such as high latency and centralized processing. By distributing workloads to edge servers, FEC reduces the energy required for data transmission to central data centers and enhances data privacy [10], [11]. Additionally, the smaller, specialized edge devices are often more energy-efficient for specific tasks compared to traditional data centers.

However, energy management in FEC remains a challenge ss a consequence of the decentralized architecture of edge nodes. Jeong et al. improved service scheduling within FEC by reconfiguring service paths to mitigate migration overhead, achieving a 21% improvement in energy efficiency and an 80% reduction in Service Violation Rate (SVR) [2]. Similarly, Li et al. introduced a dynamic resource-aware scheduling policy (DRA-MQoS) that evaluated server resource utilization and service characteristics in real time. This approach reduced energy consumption by 23% while maintaining Quality of Service (QoS) [3]. Other notable contributions, such as Dyme and Daas, applied dynamic micro-service scheduling to enhance energy efficiency in edge environments [12], [13].

Recent contributions have also explored the use of resource-agnostic microservice offloading schemes in Industrial IoT (IIoT) environments. The RAISE framework [14] leverages resource-agnostic properties to optimize microservice offloading in IIoT networks, demonstrating improved network throughput and QoS. Similarly, the mISO mechanism [15] applies a double-auction approach for microservice offloading, achieving significant reductions in latency and improved resource utilization through resilient demand estimation and incentive mechanisms.

AI-driven approaches, particularly in edge computing, have further advanced resource optimization. Recurrent Neural Networks (RNN) [16], [17], Long Short-Term Memory (LSTM) networks [18], and Reinforcement Learning (RL) techniques [19], [20], [21] have been extensively applied. Wang and Yu [4] combined RL with federated learning for IoT resource scheduling, demonstrating reduced latency, enhanced system perception, and lower computing costs. Cui et al. [22] developed a deep learning-based (DL) resource management model for FEC, achieving reduced network latency and fast adaptation to dynamic network conditions.

A particularly effective RL model, Q-Learning, has been widely adopted for service scheduling in FEC due to its adaptability to dynamic environments. Unlike traditional models, Q-Learning does not require an explicit environment model and learns through trial and error. Over time, the iterative learning process enables Q-Learning to optimize resource allocation in FEC systems effectively.

Resource management strategies for optimizing energy consumption have also incorporated clustering techniques. Yang et al. demonstrated that detecting overloaded hosts and reallocating their workloads significantly reduces energy consumption [23]. Shidik et al. evaluated workload allocation in both heterogeneous and homogeneous environments using Fuzzy C-Means (FCM) [5]. Their findings indicated that FCM reduced energy consumption in cloud data centers, particularly in heterogeneous conditions, with energy savings of up to 9.34%. Shidik et al. further compared FCM and K-Means with other methods, such as Minimum Migration Time (MMT) and Random Choice (RC) [6]. The results revealed that K-Means achieved significant energy savings, with statistical significance supported by a p-value of 0.0105 in a Friedman Test. Hayat et al. [24] emphasized the importance of accurate host selection methods in improving energy efficiency and QoS. Clustering techniques were shown to enhance system performance by grouping hosts with similar resource characteristics, thereby optimizing workload placement.

Existing research in Federated Edge Cloud (FEC) and resource allocation primarily focuses on optimizing VM migration and scheduling. However, several challenges remain unaddressed. Many approaches struggle with scalability, as they lack adaptability to highly dynamic workloads, making them inefficient in large-scale FEC environments. Additionally, energy inefficiency remains a concern, as techniques such as path reconfiguration and clustering alone do not effectively optimize energy consumption under varying resource demands. Furthermore, traditional methods often rely on static configurations or predefined heuristics, limiting their ability to adapt to unpredictable workload distributions and resulting in suboptimal performance.

Based on the existing research background and the comparative analysis presented in Table 1, the research questions in this study are:

- Can a hybrid approach integrating clustering techniques (FCM or K-Means) with Reinforcement Q-Learning effectively optimize VM selection and scheduling, enhancing energy efficiency and workload balance in FEC environments?
- Can a modified reward function in RL Q-Learning, tailored to FEC environments, improve decision-making by prioritizing energy efficiency, workload balance, and QoS metrics?

This research proposes an Unsupervised Cluster Reinforcement Learning in Federated Edge Cloud (UCRL-FEC). The proposed UCRL-FEC method effectively addresses the limitations identified in previous research by introducing a novel combination of unsupervised clustering methods and Q-Learning for adaptive scheduling. This study will compare the performance of FCM and K-Means to understand their impact on energy efficiency and QoS. A Q-Learning-based RL model will be applied to manage service scheduling within the FEC environment, through restructuring the reward function or reinforcement criteria. In the Q-Learning, the reward function is a crucial element that guides the learning agent's decision-making process. It provides feedback on the quality of each action by assigning a reward.

TABLE 1. Research gap.

Study	Method	Strengths	Limitations	Proposed Contribution
Yenugula [8]	Bacterial Foraging Optimization Algorithm	Reduced service costs by migrating VMs from overloaded to underloaded hosts.	Focused on single-objective optimization; lacks integration with AI-based dynamic decision-making.	Integrating clustering and RL for dynamic workload optimization.
Saxena et al. [9]	K-Medoid-based VM scheduling	Accurate workload prediction and resource utilization.	High computational complexity for large-scale systems.	Introducing RL to handle dynamic environments with reduced complexity.
Jeong et al. [2]	Path reconfiguration in FEC	Improved energy efficiency and reduced Service Violation Rate.	Limited scalability across highly distributed edge nodes.	Using unsupervised clustering to enhance scalability in dynamic edge environments.
Li et al. [3]	Dynamic resource-aware scheduling (DRA-MQoS)	Real-time evaluation of resources and service characteristics.	Lacks integration with advanced machine learning models for predictive analytics.	Employing RL-based predictive scheduling with modified reward functions.
Shidik et al. [5][6]	Fuzzy C-Means and K-Means clustering	Significant energy savings and effective host selection.	Focused on static environments; limited adaptability to dynamic workloads.	Enhancing clustering with RL to improve adaptability and decision-making.
Wang and Yu [4]	RL with federated learning	Reduced latency and improved system perception in IoT environments.	Limited focus on energy consumption optimization in federated edge environments.	Combining RL and clustering for energy-efficient federated edge environments.
Cui et al. [22]	Deep learning-based resource management	Reduced network latency and adaptability to changing network conditions.	High computational overhead and limited focus on SLA improvement.	Incorporating SLA-focused reward function in RL to balance latency and energy efficiency.
Samanta et al. [14] (RAISE)	Resource-agnostic offloading scheme	Optimizes microservice offloading in IIoT networks, improved QoS, and reduced delay.	Limited scalability to heterogeneous edge environments.	Adapting RAISE principles to dynamic and distributed FEC systems.
Samanta et al. [15] (mISO)	Double-auction incentive mechanism	Improved latency and resource utilization through demand estimation and incentives.	Requires high computational complexity for auction-based mechanisms.	Incorporating mISO-inspired resilient demand estimation into RL-based scheduling models.
Dyme and Daas [12][13]	Dynamic microservice scheduling	Improves energy efficiency and workload distribution in edge environments.	Limited adaptability to highly dynamic workloads and lacks integration with reinforcement learning.	Enhancing dynamic scheduling with clustering and RL to improve energy-aware workload management in FEC.

Modifying the reward function is expected to offer insights into how changing the rules governing the Q-Value impacts energy efficiency and QoS.

III. METHODOLOGY AND SYSTEM DESIGN

A. SYSTEM OVERVIEW OF FEC

FEC is a distributed computing architecture that integrates geographically dispersed edge nodes to process data and applications closer to end users. This design aims to reduce latency, enhance responsiveness, and, most importantly, minimize energy consumption through efficient workload distribution. The FEC system is composed of three main interacting layers: the edge device layer, the edge node layer, and the federation coordinator layer. The edge device layer consists of devices such as IoT sensors, mobile devices, and gateways connected to edge nodes. These devices collect data and send it to the nearest edge node for processing [1]. The edge node layer is the central component responsible for data processing, storage, and computation, while the federation coordinator layer manages multiple edge nodes within a cluster, coordinating their interactions [25].

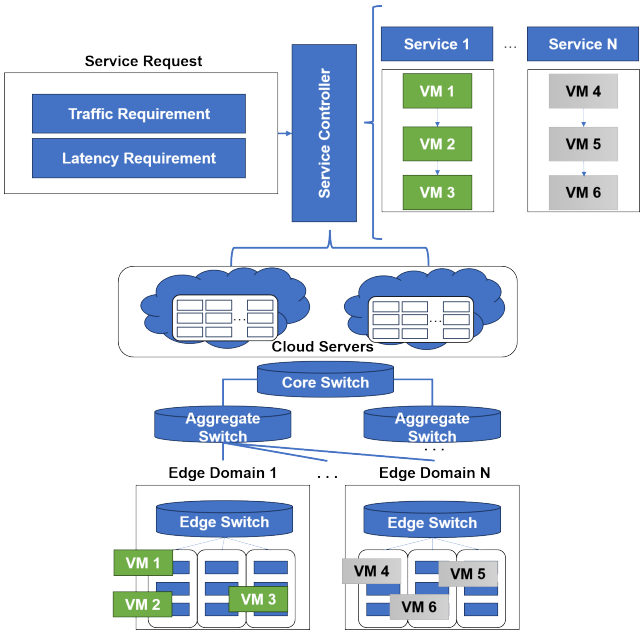

FIGURE 1. Overview of the FEC architecture.

Figure 1 shows a schematic representation of the FEC architecture, demonstrating how services are handled within this system. The process begins with a service request that outlines the latency and traffic requirements. This request is directed to the Service Controller, which allocates the necessary resources to fulfill it by determining which VMs will be used. Two services, Service 1 and Service N, are shown, each connected to several VMs, illustrating the architecture's ability to efficiently allocate resources based on user demand.

Once the VMs are identified, the Service Controller directs the request to the Edge Domain, which consists of multiple edge servers connected through Edge Switches. In this phase, VMs related to specific services are placed on the appropriate edge servers. For example, VM1, VM2, and VM3 are placed in Edge Domain 1, while VM4, VM5, and VM6 are placed in Edge Domain N.

Data from these edge servers is then forwarded through a larger network, starting from Aggregate Switches to the Core Switch, which connects to Cloud Servers in the main data center. These cloud servers handle more complex and large-scale data processing when needed. The entire process is not only guided by static rules but also dynamically optimized using RL algorithms. These algorithms continuously

learn from feedback to optimize resource allocation and VM placement based on real-time conditions. Through RL, the system is capable of adapting in real-time to fluctuations in network conditions and workloads, improving efficiency, reducing latency, and managing traffic adaptively.

This research extends the FEC architecture by integrating a modified RL algorithm for service scheduling and VM placement. The RL model operates through the RL Learning Module, providing reward-based feedback to optimize these processes dynamically. The proposed method dynamically allocates resources using the following steps:

1. The system continuously tracks CPU and RAM usage across FEC nodes.
2. When a host exceeds a predefined threshold, the proposed clustering methods (FCM or K-Means) groups VMs based on resource utilization.
3. The system selects the cluster with the lowest RAM/CPU usage to minimize resource contention.
4. The Q-Learning agent selects the optimal VM migration path, updating Q-values based on proposed modified reward function to achieve the energy efficiency and better QoS metrics.
5. The VM is migrated, and system parameters are updated to improve future decisions.

In this study, the reward function has been modified to account for RAM and CPU utilization, making it more responsive to workload changes across edge nodes.

B. REINFORCEMENT LEARNING IN FEC

In the FEC system, RL Q-Learning is employed to manage resources adaptively and responsively, as illustrated in Fig. 2. RL enables an iterative learning process, where the system learns to optimize workload distribution over time with the goal of reducing energy consumption.

The process starts by monitoring key performance parameters, such as energy consumption (EC) and system performance, which are used in the reward function. This function assesses the efficacy of the actions performed by the Q-Learning agent through the allocation of a reward value, which reflects how well a particular action improves system efficiency. The reward is then used to update the Q-Table, allowing the system to iteratively learn the optimal policy for resource allocation.

Figure 2 illustrates the Q-Learning system architecture for VM migration in Federated Edge Cloud (FEC). The VM migration process starts with system performance monitoring by Service Monitor, which collects data related to resource usage and stores it in the Status DB. When the resource usage on a host exceeds a predefined threshold, the Service Placement Manager will re-evaluate the VM placement and identify the overloaded host. Next, the Migration Manager module will apply the Unsupervised Cluster Reinforcement Learning (UCRL-FEC) method, where clustering techniques such as Fuzzy C-Means (FCM) or K-Means are used to group VMs based on their resource usage patterns. Once the clusters are formed, the system will select VMs from the cluster with

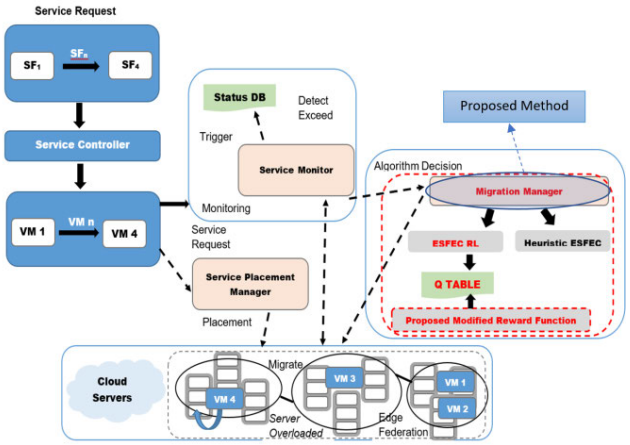

FIGURE 2. Architecture of reinforcement Q-learning for VM migration in FEC.

the lowest resource utilization rate to be migrated to reduce the load of overloaded hosts. The migration decision is made based on the Q-Table, which is dynamically updated to determine the optimal destination host. Once the decision is made, the VMs are moved to the new host, and the system updates the network state to ensure load balance and energy efficiency are maintained. With this approach, the migration process can be performed adaptively, reducing SLA violations (SLAVs) and minimizing the number of unnecessary VM migrations, thus improving the efficiency of resource management in an FEC environment [1].

The previous method, as shown in Fig. 2, highlights the role of the Migration Manager, which utilizes the ESFEC-RL module to apply RL principles. The essential construct of Q-Learning frameworks, the Q-Table, permits the system to update Q-values for each state-action pair, guiding VM migration decisions based on current system conditions.

Building on this foundation, the proposed UCRL-FEC method introduces dynamic clustering techniques, such as FCM and K-Means, which were chosen due to their widespread use in cloud environments for VM clustering and migration. These methods are among the most commonly applied in cloud computing because of their efficiency in handling large-scale data and adaptability to dynamic workloads. Previous research has also demonstrated their effectiveness in VM environments, highlighting their advantages in optimizing resource allocation and migration strategies [5], [6], [19], [29]. Moreover, K-Means is efficient and scalable, making it well-suited for large-scale FEC systems. However, its hard clustering approach may not fully capture workload variations. On the other hand, FCM, with its soft clustering capability, better handles heterogeneous workloads but is more computationally intensive. To address these limitations, this study integrates these clustering methods with Q-Learning, enabling dynamic adaptation to workload changes while optimizing energy efficiency. This study evaluates and compares these clustering methods within the FEC environment, integrating them with Q-Learning to enhance dynamic VM selection and improve overall system performance.

This dynamic clustering offers more precise migration scheduling compared to static methods. Additionally, the introduction of a logarithmic reward function enhances the system's sensitivity to performance changes, optimizing energy consumption and workload management while minimizing SLAV and reducing the number of VM migrations.

The Q-value for the selected state-action pair is updated using the Q-Learning formula (Eq. (1)):

$$Q(s, a) = Q(s, a) + \alpha \times (r + \gamma Max(Q(s', a')) - Q(s, a)) \tag{1}$$

where $Q(s, a)$ is the previous Q-value, α is the learning rate, R is the reward received, γ is the discount factor, and $Max(Q(s',a'))$ is the highest Q-value for the next state s'. This iterative process enables the agent to continuously improve its decision-making policy. As a result, the FEC system using Q-Learning can optimize service scheduling, reduce the load on overloaded hosts, enhance energy efficiency, and ensure better service quality in a distributed computing environment [1].

C. Q-TABLE

In Q-Learning, the Q-Table stores Q-values, representing the estimated reward or benefit of each state-action combination. Initially, the Q-Table is populated with default values, typically zeros or small numbers, which are updated as the agent learns from experience. The process starts with the agent observing the system's current state, which includes factors like CPU usage, memory usage, and resource availability in the FEC environment.

After observing the state, the agent selects an action, such as migrating a VM to another node, redistributing the workload, or taking no action. This selection can be based on exploitation (choosing the action with the highest Q-value) or exploration (trying a new action to discover better Q-values). After the action, the system observes the outcome, and the agent receives feedback in the form of a reward. This reward reflects how well the action achieved the desired goals, like reducing the workload on an overloaded host or improving energy efficiency. The reward function in the original ESFEC-RL framework is defined as:

$$r_t = \left(\frac{E_{SP}^{Total_{t+1}} - E_{SP}^{Total_t}}{E_{SP}^{Total_t}} \right) \tag{2}$$

Drawing from previous studies, the reward function is formulated by calculating the difference between the service path's energy usage at time t and at time $t + 1$. A negative value of reward function indicates effective action since it signifies reduced energy consumption. The smaller the reward, the more efficient the action, signaling higher energy efficiency. The Q-Learning formula is applied to update the Q-value accordingly, which considers both the received reward value and the estimated future value from the further state. The formula incorporates the learning rate (α), which controls how much the new reward affects the old Q-value, and the reduction factor (γ), which balances the importance of future versus current rewards. The updated Q-value is then stored back in the Q-Table.

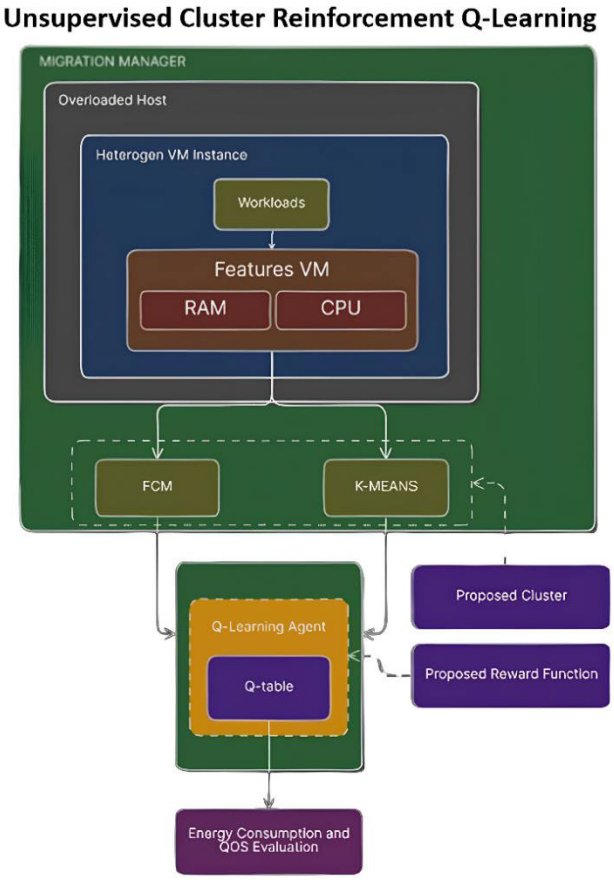

FIGURE 3. Proposed design of UCRL-FEC.

D. PROPOSED UNSUPERVISED CLUSTER REINFORCEMENT Q-LEARNING IN UCRL-FEC

This section introduces the proposed method, an improvement of the previous study, combining FCM or K-Means clustering with RL Q-Learning and adjusting the reward function through a logarithmic transformation [1]. The proposed UCRL-FEC research design is shown in Fig. 3.

From Fig. 3, UCRL-FEC manages VM migration on overloaded hosts. The Migration Manager first detects the overloaded host and maps it to various heterogeneous VMs, each handling workloads based on features like RAM and CPU. FCM and K-Means are used to cluster VMs based on these features. The clustering results are used to select the cluster with the least RAM or CPU usage, which is then passed to the Q-Learning agent.

This study also proposes modifying the reward function in the Q-Table with a logarithmic function. This modification aims to enhance sensitivity to small performance changes, enabling smoother adjustments in energy consumption while maintaining service quality. By integrating

clustering with Q-Learning, the method adapts resource allocation based on dynamic workloads. The modified Q-value update formula incorporates clustering results, optimizing resource management more effectively. This ensures continuous improvement in cluster management strategies through experience-based learning, making it a superior solution for FEC resource management. Detailed information about each stage of UCRL-FEC is provided in the following section.

E. CLUSTERING IN UCRL-FEC

To enhance resource management efficiency in FEC, UCRL-FEC applies clustering techniques when an overloaded host is detected, instead of directly applying RL as done in the previous study by Jeong et al [1]. Clustering is a key technique in computational resource management, grouping entities with similar characteristics to enable more efficient management in cloud computing environments [17], [26]. Two commonly used clustering algorithms are FCM and K-Means [27]. FCM allows entities to belong to multiple clusters, handling uncertain or overlapping data flexibly. In contrast, K-Means is simpler and faster, grouping data into exclusive clusters. In UCRL-FEC, both clustering methods are applied based on key features like RAM and CPU in a similar environment. In addition, this study both cluster methods will be evaluated with three number of clusters $C = \{3, 4, 5\}$, to achieve a balance between interpretability, computational efficiency, and meaningful data representation. Using too few clusters (e.g., 2) may oversimplify the data structure, failing to capture key distinctions, while too many clusters (e.g., more than 5) can result in over-segmentation, increasing complexity and computational cost.

1) FUZZY C-MEANS (FCM) BASED CLUSTER

Fuzzy C-Means (FCM) is a clustering algorithm that permits data points to belong to multiple clusters simultaneously. Unlike traditional hard clustering techniques, which assign each data point exclusively to one cluster, FCM computes a membership degree indicating the extent to which each data point belongs to every cluster [5]. This flexibility enables FCM to capture subtle variations in the data, making it particularly useful in dynamic environments like FEC. The core principle of FCM is to minimize the objective function, as expressed by Bezdek et al. [28]:

- Initialized:
  - Determined the intended number of clusters $C = \{3, 4, 5\}$
  - Determined the value of parameter $m (m > 1)$, which controlled the level of fuzzy membership
  - Initialized the membership matrix (U), where each element indicated the degree of membership of each data point to a specific cluster. The values of Uwere randomized with the condition that the total degree of membership of each data point to all clusters equal to 1.
- Calculated the Centroid:
  Calculated the centroid for each cluster based on the degree of membership of the data in that cluster. The centroid was calculated using the formula:

$$v_j = \frac{\sum_{I=1}^{N} u_{ij}^{m} . x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \tag{3}$$

where:

- v_j Was the centroid of the j^{th} cluster
- u_{ij} Was the membership degree of the i^{th} data point to the j^{th} cluster
- x_i Was the data of the i^{th} VM containing the features of RAM and CPU
- m Was the fuzzy parameter (typically, m is set to 2)
- N Was the total number of observed VM

- Updated the Membership Degree:
  Following the centroid calculation, the membership degree of each data point to the cluster was updated using the formula:

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{||x_i - v_j||}{||x_i - v_k||} \right)^{\frac{2}{m-1}}} \tag{4}$$

where:

- u_{ij} Was the membership degree of the i^{th} data point to the j^{th} cluster
- $| \left| x_i - v_j \right| |$ Was the distance between the i^{th} data point and the centroid of the j^{th} cluster
- C Was the number of clusters

- Updated the Membership Degree:
  The steps of computing the centroid and updating the membership matrix are iterated until the difference in membership values from one iteration to the next is less than a specified threshold, or until a maximum iteration limit is met.
- Output:
  The final output consists of the centroid for each cluster and the updated membership matrix, which shows how strongly each data point is associated with each cluster.

2) k-MEANS BASED CLUSTER

K-Means is a widely used clustering algorithm that partitions a dataset into C clusters by assigning each data point to the cluster with the nearest centroid. The algorithm begins with a random initialization of centroids and iteratively updates their positions until convergence is achieved typically when centroid movements between iterations fall below a minimal threshold or cease entirely. This process results in compact clusters that facilitate more effective data analysis.

Clustering with K-Means can be performed as follows:

a. Determine the number of clusters $C = \{3, 4, 5\}$.
b. Allocate the data randomly into the clusters.
c. Calculate the cluster centers (centroids/means) based on the data in each cluster.
d. Assign each data point to the nearest centroid/mean.

e. Return to step three if any data points are still changing clusters, if the change in centroid values exceeds the defined threshold, or if the change in the objective function value used is still above the defined threshold. [29]

The location of the centroid (central point) for each cluster, which is derived from the mean of all data values for each feature, must be recalculated. If N represents the number of data points in a cluster, i represents the i^{th} feature in the cluster, and p represents the data dimension, the centroid for the i^{th} feature is calculated using the following formula:

$$v_i = \frac{1}{M} \sum_{j=1}^{N} x_{ij} \tag{5}$$

where:

- v_i is the centroid of the i^{th} feature,
- x_{ij} is the value of the i^{th} feature for the j^{th} data point,
- N is the total number of data points in the cluster

This formula calculates the mean value of the feature i across all data points in the cluster, resulting in the new centroid position. The formula is applied across all p dimensions, so i ranges from 1 to p. To measure the distance from a data point to the cluster center, Euclidean Distance can be used. The distance measurement in Euclidean distance space is calculated using the following formula:

$$D\left(x_i, v_i\right) = | \left|x_i - v_i\right| | = \sqrt{\sum_{j=1}^{p} |x_{ij} - v_i|^2} \tag{6}$$

D is the distance between the data points x_i and centroid v_i and || represents the absolute value.

Next, the reallocation of data into each cluster in the K-Means method is based on comparing the distance between the data point and the centroid of each existing cluster. The data is strictly reassigned to the cluster whose centroid is the closest to that data point. This reallocation can be formulated as follows:

$$a_i = \begin{cases} 1, & if\ D = \min\left\{D\left(x_i, v_i\right)\right\}, \\ 0, & Otherwise. \end{cases} \tag{7}$$

a_i o is the membership value of a point x_i to the cluster center v_i, D is the shortest distance from the data point x_i to C_i the cluster after comparison, and C_i is the centroid (cluster center) i^{th}. The objective function used for K-Means is determined based on the distance and the membership value of the data points to the clusters. The membership assignment can be described as follows:

- If x_i is closest to centroid v_i, then $a_i = 1$, meaning x_i belongs to that cluster.
- If x_i is closer to another centroid, then $a_i = 0$, meaning it does not belong to that cluster.

To update the centroid v calculate the average of each data point in the cluster C_i using formula:

$$v_k = \frac{1}{\|C_k\|} \sum_{x \in C_k} x \tag{8}$$

Performed the above steps until no changes occurred in the updated centroid values v, indicated by the absence of any members moving between clusters C.

3) CLUSTER SELECTION

The cluster selection process in the proposed UCRL-FEC aims to choose the most suitable cluster based on a score calculated from CPU and RAM usage. In this process, scores for each of the three defined clusters are calculated using the equation for Score $Cluster_i$ Each cluster's score is computed by summing the total CPU and RAM usage, which is then multiplied by predetermined weights for CPU and RAM. The formula for calculating the cluster score is as follows:

$$Score_{Cluster_i} = \left(\omega_{CPU} \times CPU_{total,i}\right) + \left(\omega_{RAM} \times RAM_{total,i}\right) \tag{9}$$

$$CPU_{total} = \sum_{i=1}^{n} CPU_i \tag{10}$$

$$RAM_{total} = \sum_{i=1}^{n} RAM_i \tag{11}$$

where:

- $CPU_{total,i}$ is the total CPU usage of each member within $Cluster_i$
- $RAM_{total,i}$ is the total RAM usage of each member within $Cluster_i$
- Weight ω_{CPU} and ω_{RAM} is the weight used to determine the priority ratio of the cluster based on CPU or RAM ($\omega_{CPU} + \omega_{RAM} = 1; 0 \leq \omega_{CPU} | \omega_{RAM} \leq 1$).
- n is the number of data points within the cluster.

After calculating the score for each cluster, the function selects the cluster with the lowest score. The cluster with the lowest score is considered the most optimal, indicating more efficient resource usage or better alignment with the established criteria. The selection is made by comparing the scores of the three clusters; the cluster with the lowest score is chosen. For instance, if the score for Cluster 1 is the lowest, then Cluster 1 is selected; if Cluster 2 has the lowest score, it is selected instead; if neither, then Cluster 3 is chosen.

$$Priority\ Selection\ Cluster = Min(Score_{Cluster_i}) \tag{12}$$

$$Static\ Selection\ Cluster = (Cluster_i)\ , Cluster_i \neq NULL \tag{13}$$

F. MODIFIED REINFORCEMENT Q-LEARNING

After the cluster selection process for migratable VMs on overloaded hosts is completed, the next step in the proposed UCRL-FEC is to define the learning process in the RL mechanism. During this phase, the agent continuously updates the Q-table based on decisions it makes, optimizing future actions and improving the efficiency of VM migration in handling overloaded hosts.

Algorithm 1 focuses on the development and evaluation of the proposed Unsupervised Cluster RL Q-Learning in UCRL-FEC for managing clustered overloaded host servers in a FEC

environment using a RL approach. This algorithm builds on previous research by Jeong et al. and introduces changes, including clustering overloaded hosts as an initial input [1].

Algorithm 1 Reinforcement Learning Agent with Clustering Approach and Enhanced Reward Strategy.

Input Clustered set of high-load servers
Output Updated action-value of Q-table
1: Define Learning coefficient ($\alpha = 0.05$), where α ranges between 0 and 1.
2: Define Discount factor ($\gamma = 0.08$), where γ ranges between 0 and 1.
3: Define Reward Reduction Factor (θ), where θ ranges between 0 and 1.
4: $s_{t\leftarrow}$ The server presently subjected to excessive workload at time t
5: $a_{t\leftarrow}$ The server chosen for VM migration at the specific time t
6: r_t: Calculated reward for the action a_t
7: $Host_{dest}$: Available edge servers for VM placement
8: $Host_{status}$: Table of active host server states
9: $Num_{episodes}$: Overall training rounds
10: **Procedure** Agent using Reinforcement Learning
11: While $Num_{episodes}$ is halted **do**
12: Initiate $Host_{dest}$
13: $s_{t\leftarrow}$ **Clustered Overloaded host servers (Proposed Method)**
14: Determine and perform the action a_t with the minimum Q-value as per the Q-table
15: Update $Host_{status}$ after migration
16: **Proposed Modified r_t**
17: $Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma\ Min\ Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
18: Update Q-table with the new value
19: **end while**
20: **end procedure**

Additionally, the reward function was modified from the original Eq. (2) to several new formulas, incorporating a logarithmic reward function that focuses on RAM and CPU utilization. The modified equation is as follows:

$$r_t = \left(\frac{log(E_{SP}^{Total_{t+1}} \times CPU^{t+1}) - log(E_{SP}^{Total_t} \times CPU^t)}{log(E_{SP}^{Total_t} \times MIPS^t)} \right) \times \theta \tag{14}$$

$$r_t = \left(\frac{log(E_{SP}^{Total_{t+1}} \times RAM^{t+1}) - log(E_{SP}^{Total_t} \times RAM^t)}{log(E_{SP}^{Total_t} \times RAM^t)} \right) \times \theta \tag{15}$$

where:

r_t :The reward value at t time
$E_{sp}^{Total_t}$: Energy consumption achieved at time t
$E_{sp}^{Total_{t+1}}$: Energy consumption achieved at time $t+1$
CPU^t :The number of Instructions a particular cloud server could process per Second at t time
CPU^{t+1} :The number of Instructions a particular cloud server could process per Second at $t+1$ time
RAM^t :The percentage of temporary storage used to store the data and instruction that CPU in edge servers actively uses during program execution at t time
RAM^{t+1} :The percentage of temporary storage used to store the data and instruction that in edge servers actively uses during program execution at $t+1$ time
θ :The reduction parameter that controls the deduction toward reward calculation. The lower the value of the reduction parameter, the greater the energy reduction that will be provided in the reward

The proposed logarithmic reward function is designed to enhance the system's sensitivity to subtle changes in resource utilization (such as CPU and RAM) and energy consumption. This allows the system to adapt more quickly to dynamic workload changes, ensuring a balance between energy efficiency and QoS. By incorporating both CPU and RAM parameters into the reward function, the system can more accurately evaluate migration actions based on their impact on energy efficiency and overall performance. However, in highly heterogeneous environments, the reward function may become more complex due to the varying characteristics of resources across nodes. To address this, we introduced a reduction parameter (θ) to control the impact of small changes in resource utilization on the reward, preventing overfitting and ensuring adaptability across different scenarios.

The process starts with identifying overloaded servers, which are grouped using clustering methods such as FCM or K-Means. The RL agent then uses these clusters to determine optimal migration actions by selecting the action with the smallest Q-value from the updated Q-Table.

After a migration action is selected, the host's status is updated, and the reward is calculated based on the modified reward function, designed to reduce server load and enhance energy efficiency. This iterative process continues over multiple episodes, with the Q-table continuously updating based on new experiences. This enables the RL agent to make more efficient migration decisions, managing resources effectively while maintaining QoS in the federated cloud system.

G. DATASET

This study uses the April 20, 2011 (2011-04-20 Workload) dataset [30] to evaluate the Cluster RL Q-Learning technique's ability to minimize energy consumption in FEC. The workload includes various computationally intensive applications running on a distributed cloud infrastructure, collected every five minutes over 24 hours.

The histogram in Fig. 4 shows the distribution of CPU usage in the Federated Edge Cloud (FEC) environment of the Dataset used in this study. The dataset consists of 297,504

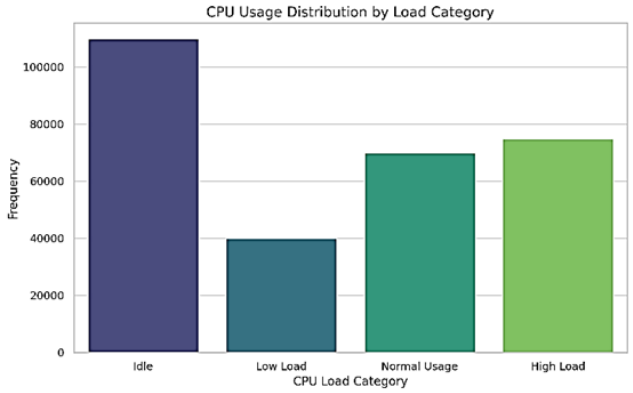

FIGURE 4. CPU usage distribution by load category.

records of CPU activity from 1,033 workloads VM where each load was collected every five minutes over 24 hours. It is categorized into four classes or clusters based on the quartile of the resource usage. The results of the workload analysis show that most of the CPU time is in the Idle state (0-2%) with the highest frequency, indicating that the system is often in an inactive or minimal workload state. The Low Load (2-4%) category has the lowest frequency, indicating that the system is rarely in a light load state and tends to immediately switch to a higher level of usage. Meanwhile, the Normal Usage (4-12%) and High Load (12-99%) categories show that when the system is active, the workload is often quite high, which emphasizes the importance of adaptation strategies in resource management. These distribution patterns provide important insights in understanding the workload patterns used in this study.

The RAM and MIPS usage parameters for the VMs were derived from the CloudSimSDN simulation environment [31], which replicates workloads based on the PlanetLab dataset. Although the original PlanetLab dataset lacks RAM usage data, this study addresses that by using CloudSimSDN to simulate the necessary RAM data. CloudSimSDN replicates complex distributed cloud environments, providing accurate modeling of key resources like CPU and RAM based on the observed workload patterns.

By incorporating CloudSimSDN, the study effectively estimates RAM usage in the clustering process. This ensures that both RAM and CPU utilization are integrated into the clustering technique, aligning with the study's goals despite the absence of explicit RAM data in the original PlanetLab dataset.

Although this dataset dates back to 2011, it has been used extensively in subsequent research [2], including recent studies [1]. This approach allows significant improvements to be shown with minimal bias. In addition, the dataset has been simulated using CloudSimSDN to estimate RAM usage, ensuring that CPU and RAM usage are accurately modeled despite there being no explicit RAM data in the original dataset.

H. EVALUATION PARAMETER

In this study, multiple evaluation metrics, listed in Table 2, are employed to assess the effectiveness of the proposed UCRL-FEC algorithm in minimizing energy consumption within FEC environments. The primary metric, Energy Consumption, quantifies the total energy utilized by all hosts throughout the simulation period. This is based on energy used that includes both static and dynamic energy consumption, allowing for comparisons between the proposed method and the baseline [32]. The second parameter, SLATAH, assesses the average time per active host that complies with the SLA. This is crucial for determining if the system can maintain required QoS while using energy-saving strategies [33].

Furthermore, SLA PDM measures performance loss caused by VM migrations, which can impact system efficiency [34]. The overall SLAV parameter calculates the percentage of cumulative SLAV during the simulation by comparing actual response times with the SLA limits. This metric reflects system reliability and QoS [35]. The Number of VM Migrations Evaluates how frequently VMs are moved between hosts during the simulation. While fewer migrations can reduce overhead and improve energy efficiency, this must be balanced with performance and SLA considerations [36]. Energy SLA Violation (ESV) in cloud computing quantifies energy efficiency by integrating both energy consumption (EC) and service level agreement violations (SLAV) across data centers, aiming to maintain sustainable operations while upholding Quality of Service (QoS) [37], [38]. Data for each parameter were collected and analyzed, with comparisons made against the ESFEC-RL method from previous research to evaluate the proposed Cluster RL Q-Learning method's impact on energy efficiency and service performance in FEC environments.

I. SIMULATION DESIGN

To validate the proposed Unsupervised Cluster RL Q-Learning (UCRL-FEC), simulations were conducted using CloudSim integrated with CloudSimSDN. CloudSimSDN, an SDN extension of CloudSim, enables simulations of networking, SDN, and Service Function Chaining (SFC) in edge and cloud data centers, making it ideal for evaluating FEC environments. This tool has been widely used and validated in previous research, including studies by Jeong et al. [1], demonstrating its effectiveness in simulating dynamic workloads and heterogeneous resource distributions.

The simulation setup in Table 3 was designed to evaluate the impact of varied CPU, memory, and storage specifications on energy efficiency in a heterogeneous environment. The use of the April 20, 2011 dataset (2011-04-20 Workload) [30] ensures consistency and facilitates direct comparison with previous studies using the same dataset. Although the dataset originates from 2011, it remains relevant for evaluating energy efficiency in FEC environments, as it has been widely adopted in prior research.

TABLE 2. Evaluation parameters for UCRL-FEC algorithm.

Parameter	Description
Energy Consumption (EC) [2]	Measures total energy usage of all hosts during simulation, including static and dynamic consumption.
SLATAH (SLA Time per Active Host) [33]	Assesses the average time per active host that meets SLA requirements, ensuring QoS compliance.
SLA-PDM (SLA Performance Degradation due to Migration) [34]	Measures the combined impact of migration-induced performance degradation on SLA compliance. A lower SLA-PDM suggests better workload balancing with minimal SLA impact.
Overall SLAV [35]	Represents the cumulative SLA violations across all hosts and simulations. It indicates the system's ability to maintain service guarantees over time.
Number of VM Migrations [36]	Tracks how frequently VMs are moved between hosts; fewer migrations improve efficiency but must balance performance.
Energy SLA Violation (ESV) [37][38]	Energy SLA Violation (ESV) is a metric in cloud computing that measures energy efficiency by considering energy consumption (EC) and service level agreement violations (SLAV). A high ESV indicates inefficiency in cloud resource management, which can impact operational costs and service quality .

TABLE 3. Simulation parameters.

Parameter	Value/Specification
Number of Edge Domains	8
Number Edge Servers per Domain	300
Total Edge Servers	2400
CPU per Server	16-core
RAM per Server	32 GB
pCPU Ratio	1:1
Max VMs per Server	16
Total VMs Deployed	1033
VM Configuration	Heterogeneous
Host Type	HP ProLiant ML110 G4
Processor	Intel Xeon 3040 @ 1860 MHz
Host RAM	4 GB

To further validate the adaptability of UCRL-FEC, this study analyzed the workload distribution in workload dataset. As shown in Fig. 4, the CPU usage in our experimental setup is categorized into four groups: Idle (0-2%), Low Load (2-4%), Normal Usage (4-12%), and High Load (12-99%). The high frequency of idle states indicates that edge nodes experience significant periods of inactivity, while the substantial occurrence of high-load conditions suggests frequent computational spikes. These variations reflect the necessity of a reinforcement learning-based approach like UCRL-FEC, which dynamically allocates resources to mitigate extreme workload fluctuations and optimize energy efficiency. The results demonstrate that our approach effectively balances resource usage and prevents excessive energy consumption during workload spikes.

The proposed UCRL-FEC method considered 12 different combinations, accounting for factors such as clustering method (FCM or K-Means), number of clusters ($C = \{3, 4, 5\}$), unsupervised cluster selection methods (Eq. (12) or Eq. (13)), and reward function methods (Eq. (14) or Eq. (15)). These parameters, which have also been utilized in previous studies, ensure a fair comparison by aligning with the settings used in related works. This approach allows for an equivalent evaluation of clustering and reinforcement learning strategies. The parameter setup for the proposed method is shown in Table 4.

Table 4 presents the parameter settings for the various proposed methods in this study, which were used in the methodology section. This research tested several variations of clustering algorithms, namely FCM and K-Means, with and without cluster priority.

The testing began with the basic methods without cluster priority (Proposed Method-1 and 2), followed by the application of cluster priorities based on RAM usage (Proposed Method-3 and 4), CPU usage (Proposed Method-5 and 6), and a combination of RAM+CPU (Proposed Method-7 and 8).

Furthermore, this study also tested the effectiveness of a modified reward function from Eq. (13) or Eq. (14) with a logarithmic approach on CPU and RAM, including $\theta = 1$ assigned to CPU or $\theta = 0.2$ assigned to RAM (Proposed Method-9 to 12). Each method was evaluated based on its effectiveness in managing workloads and optimizing resources, aiming to determine the most efficient configuration in a FEC environment. The study was designed to understand how these various parameter settings affected the overall system performance, both in terms of resource utilization and in maintaining QoS.

IV. RESULT AND DISCUSSION

The results and discussion section are organized into several phases according to the experimental findings.

TABLE 4. Parameter setup of the proposed method.

Method	Experiment Setup
Proposed Method-1	FCM (Static Selection Cluster)
Proposed Method-2	K-Means (Static Selection Cluster)
Proposed Method-3	FCM (Priority Selection Cluster based on RAM)
Proposed Method-4	K-Means (Priority Selection Cluster based on RAM)
Proposed Method-5	FCM (Priority Selection Cluster based on CPU)
Proposed Method-6	K-Means (Priority Selection Cluster based on CPU)
Proposed Method-7	FCM (Priority Selection Cluster based on RAM+CPU)
Proposed Method-8	K-Means (Priority Selection Cluster based on RAM+CPU)
Proposed Method-9	FCM (Priority Selection Cluster and Modified Reward Function Log – CPU, $\theta = 1$)
Proposed Method-10	K-Means (Priority Selection Cluster and Modified Reward Function Log – CPU, $\theta = 1$)
Proposed Method-11	FCM (Priority Selection Cluster and Modified Reward Function Log – RAM, $\theta = 0.2$)
Proposed Method-12	K-Means (Priority Selection Cluster and Modified Reward Function Log – RAM, $\theta = 0.2$)

A. ENERGY CONSUMPTION

The analysis of SLA performance degradation due to migration, illustrated in Fig. 5, reveals notable differences among various methods and cluster configurations.

Proposed Method-1, which utilizes FCM without cluster priority, demonstrates stable performance, with degradation values around 1.11% for the three-cluster configuration, 1.16% for four clusters, and 1.10% for five clusters. This indicates a consistent performance level, albeit with a slight increase as the number of clusters grows. However, the lack of cluster prioritization in this method may limit its ability to optimize workload distribution effectively.

A similar trend is observed in Proposed Method-2, which employs K-Means without cluster priority. While the degradation values are slightly lower 1.00% for both the three and five-cluster setups, and 1.08% for the four-cluster configuration—this improvement remains marginal. The results suggest that without explicit cluster prioritization, both FCM and K-Means struggle to significantly reduce performance degradation due to migration. To address this limitation, methods incorporating cluster priorities based on RAM and CPU were evaluated. Proposed Method-3 (FCM with RAM priority) and Proposed Method-5 (FCM with CPU priority) show a noticeable reduction in performance degradation, with values ranging from 0.59% to 0.68%. This improvement indicates that prioritizing resource allocation based on specific attributes, such as RAM or CPU, leads

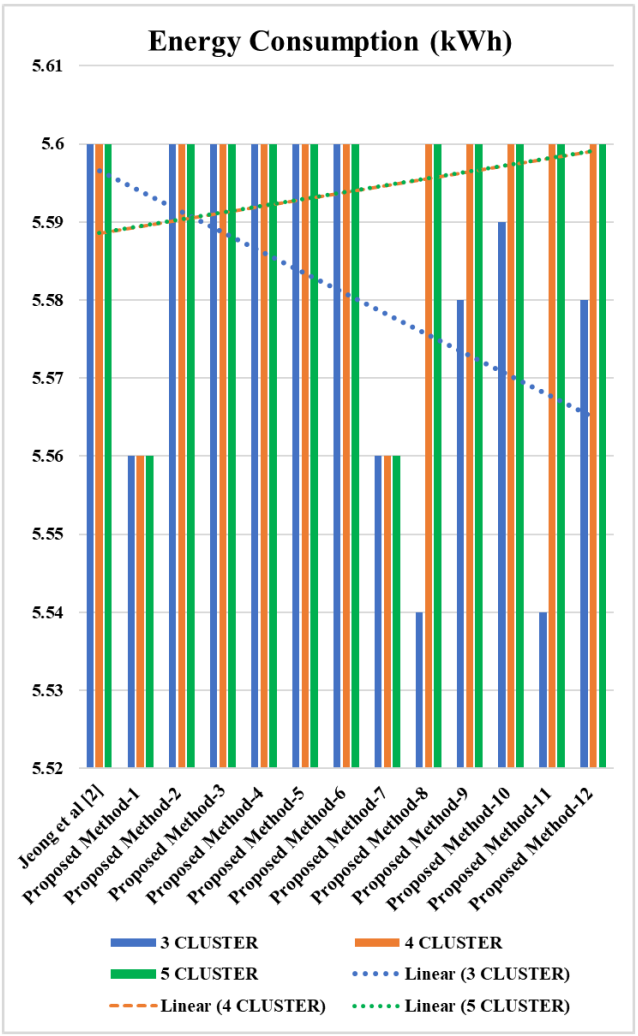

FIGURE 5. The comparison of energy consumption results.

to more efficient workload distribution. Furthermore, incorporating a modified reward function, as seen in Proposed Method-9 (FCM with Cluster Priority and Modified Reward Function Log – CPU, $\theta = 1$), provides additional enhancements. This suggests that fine-tuning the reward function can further optimize migration decisions and minimize performance degradation. In addition, the trend linear in Fig. 5 could indicate that the proposed methods are optimizing energy usage when three clusters are used in UCRL-FEC. The effectiveness of these methods may be attributed to improved resource allocation, workload distribution, or other energy-efficient mechanisms implemented in the system.

B. SLATAH

Figure 6 presents the analysis of SLATAH across various methods and cluster configurations, revealing significant variations. In methods without cluster prioritization, such as Proposed Method-1 (FCM) and Proposed Method-2 (K-Means), SLA time remains relatively stable. Proposed Method-1 records 8.21% for both the three and four-cluster

configurations, with a slight increase to 8.54% for the five-cluster setup. Similarly, Proposed Method-2 shows slightly higher SLA times, averaging 8.38% across all configurations. These results suggest that without prioritization, both methods struggle to optimize resource allocation, leading to minimal differences in SLA performance.

In contrast, when cluster priority based on RAM is introduced, a notable shift in SLA time is observed. Proposed Method-3 (FCM with RAM priority) records an increase in SLA time to 8.83% for the three-cluster configuration and 8.67% for the five-cluster setup. While this approach offers better workload distribution, the trade-off is a slight increase in SLA time due to reallocation overhead. To further enhance performance, modified reward functions were applied in methods such as Proposed Method-10 (K-Means with Cluster Priority and Modified Reward Function Log – CPU, $\theta = 1$). This approach achieves 7.60% SLA time in the three-cluster configuration, demonstrating that integrating a refined reward function can compensate for the overhead caused by cluster prioritization.

These findings highlight that while cluster prioritization improves resource management, its effectiveness can be further enhanced by incorporating adaptive reward functions that dynamically adjust migration decisions based on workload conditions.

C. SLA PDM

Next, the results in Fig. 7 show that SLA performance degradation due to migration (SLA-PDM) across various methods and cluster configurations results in several variations, reflecting the impact of cluster priority settings and reward function modifications on system performance. For Proposed Method-1 (FCM without Cluster Priority), SLA performance degradation remains stable with a value of 1.11% in the 3-cluster configuration, 1.16% in the 4-cluster configuration, and slightly decreases to 1.10% in the 5-cluster configuration. This indicates consistent performance impact without considering cluster priority.

On the other hand, Proposed Method-2 (K-MEANS without Cluster Priority) shows slightly lower SLA degradation values, with 1.00% in the 3-cluster and 5-cluster configurations, and 1.08% in the 4-cluster configuration, indicating slightly better migration performance compared to FCM without cluster priority. Methods with RAM-based cluster priority, such as Proposed Method-3 (FCM with Cluster Priority Based on RAM) and Proposed Method-4 (K-MEANS with Cluster Priority Based on RAM), demonstrate a reduction in SLA degradation, with lower values ranging from 0.59% to 0.68%.

Similarly, CPU-based and combined RAM+CPU methods, such as Proposed Method-5 (FCM with Cluster Priority Based on CPU) and Proposed Method-7 (FCM with Cluster Priority Based on RAM+CPU), also show improvements with lower SLA degradation values, reaching 0.56% in certain configurations. Methods with modified reward functions, such as Proposed Method-11 (FCM With Cluster Priority and Modified Reward Function Log – RAM, $\theta = 0.2$), also exhibit stable performance with SLA degradation values as low as 0.56%. Overall, methods with cluster priority and reward function modifications significantly reduce SLA performance degradation, with K-MEANS showing advantages in certain configurations and modified reward functions providing performance enhancements across different cluster settings.

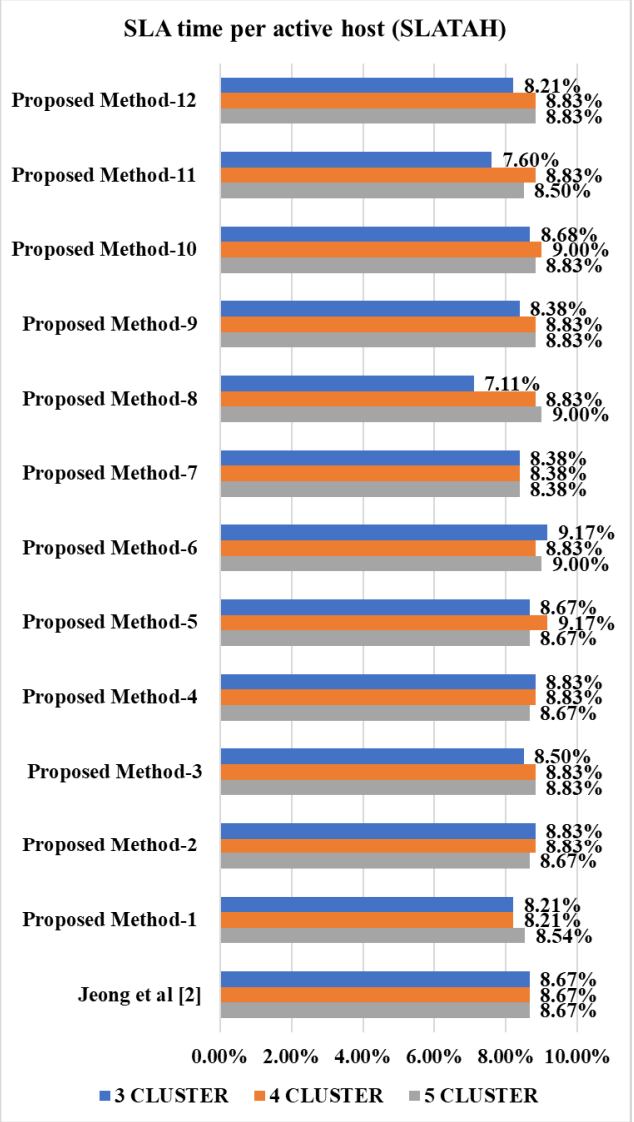

FIGURE 6. The comparison of SLATAH results.

D. OVERALL SLA VIOLATION

In Fig. 8, the analysis of overall SLAV across various methods and cluster configurations reveals significant differences in overall SLAV. For Proposed Method-1 (FCM without Cluster Priority), SLAV range around 1.11% in the 3-cluster configuration, slightly increase to 1.16% in the 4-cluster configuration, and slightly decrease to 1.14% in the 5-cluster configuration. This indicates relatively

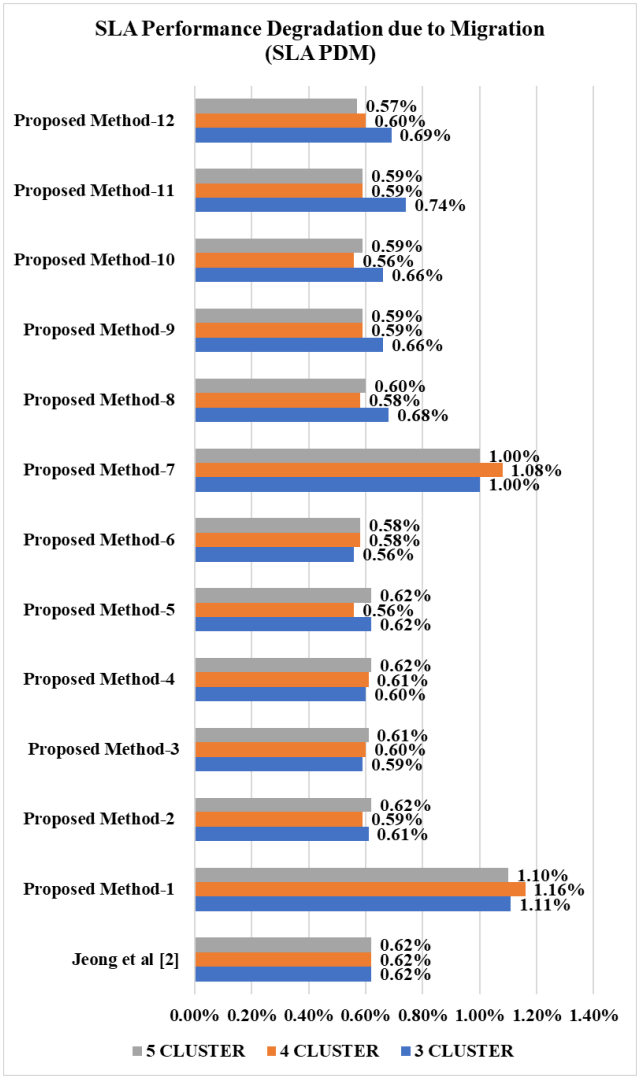

FIGURE 7. The comparison of SLA-PDM results.

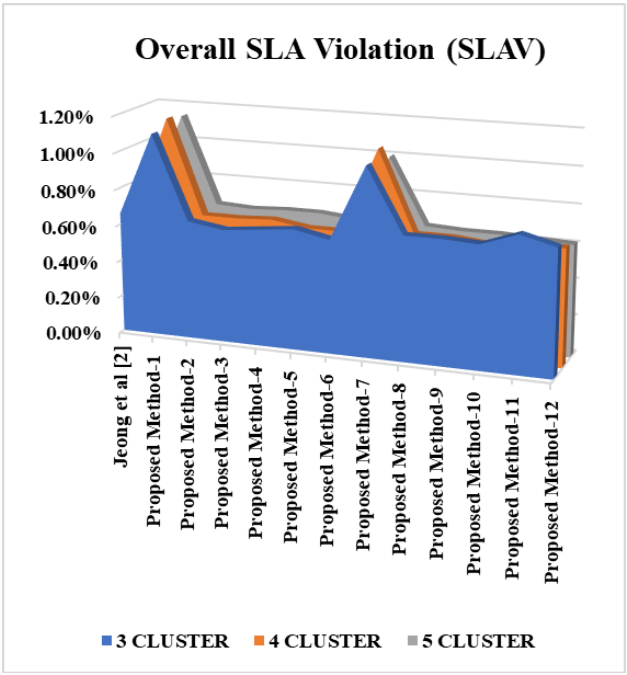

FIGURE 8. The comparison of overall SLAV results.

consistent performance for FCM, though with a slight increase in higher cluster configurations.

In contrast, Proposed Method-2 (K-MEANS without Cluster Priority) shows lower SLAV compared to FCM, with 1.03% in the 3-cluster configuration, a slight increase to 1.08% in the 4-cluster configuration, and a decrease to 1.00% in the 5-cluster configuration. Methods with RAM-based cluster priority, such as Proposed Method-3 (FCM with Cluster Priority Based on RAM), show a reduction in SLAV to 0.65% in the 3-cluster configuration, with a slight increase to 0.66% in the 5-cluster configuration. Proposed Method-4 (K-MEANS with Cluster Priority Based on RAM) shows similar results, with SLAV around 0.68% in the 3-cluster configuration and stable at 0.64% in the 4 and 5-cluster configurations. Modifications to the reward function, such as Proposed Method-10 (K-MEANS With Cluster Priority and Modified Reward Function Log – CPU, $\theta = 1$), show slightly higher SLAV, with 0.74% in the 3-cluster configuration. Overall, methods with cluster priority and reward function modifications show variations in SLAV, with K-MEANS generally outperforming FCM in some configurations.

E. NUMBER OF VM MIGRATION

The analysis of VM migration counts depicted in Fig. 9 highlights notable variations across methods and cluster configurations. For Proposed Method-1 (FCM without Cluster Priority), VM migrations number 66 under the 3-cluster setup, rise to 68 with 4 clusters, and reach 70 for 5 clusters. This trend suggests that FCM without cluster priority leads to an increased frequency of VM migrations as the cluster count grows.

In contrast, Proposed Method-2 (K-MEANS without Cluster Priority) shows a higher number of migrations compared to FCM, with 84 migrations for 3 clusters, remaining at 84 for 4 clusters, and increasing to 87 for 5 clusters. Methods that use cluster priority, such as Proposed Method-3 (FCM with Cluster Priority Based on RAM), show a significant reduction in the number of VM migrations, with 50 migrations for 3 clusters, 49 for 4 clusters, and 47 for 5 clusters. Proposed Method-4 (K-MEANS with Cluster Priority Based on RAM) also shows a reduction, with 44 migrations for 3 clusters, increasing to 46 for 4 clusters, and 48 for 5 clusters. Methods with modified reward functions, such as Proposed Method-10 (K-MEANS with Logarithmic Reward Function - CPU with Value-1), show good results with 47 migrations for 3 clusters, 47 for 4 clusters, and 46 for 5 clusters. Overall, the use of cluster priority and modified reward functions is effective in reducing the number of VM migrations, with K-MEANS

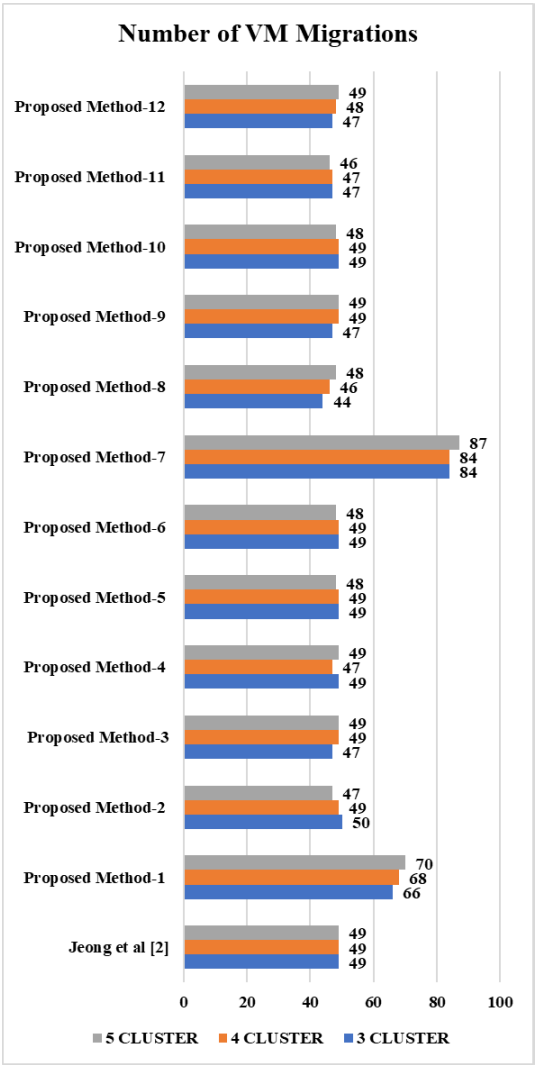

FIGURE 9. The comparison of number of VM migration results.

demonstrating better performance compared to FCM in some cluster configurations.

F. EVALUATION AND DISCUSSION

The evaluation results demonstrate that the proposed logarithmic reward function effectively balances the trade-off between energy consumption, CPU/RAM utilization, and SLA violations. In heterogeneous environments, where resource characteristics vary significantly, the reward function adapts by assigning higher rewards to actions that optimize resource utilization while minimizing energy consumption and SLA violations. In homogeneous environments, the reward function performs consistently, with smaller reward values indicating higher efficiency. This highlights the robustness of the proposed approach in both heterogeneous and homogeneous FEC systems.

This evaluation compares three primary approaches—ESFEC-RL, FCM, and K-Means—for cluster management in a distributed cloud environment comprising 300 hosts. The focus parameters for comparison include energy consumption, SLA-PDM, overall SLAV, SLATAH, and the number of VM migrations. Additionally, modifications using a logarithmic reward function based on CPU and RAM were tested to assess their impact on system efficiency.

The analysis of the data presented in Table 5 reveals that EC serves as a critical indicator for evaluating the efficiency of the various methods. The baseline ESFEC-RL method maintains a constant energy consumption of 5.60 kWh across all cluster values (C). In contrast, non-cluster-prioritized methods, such as Proposed Method-1 (FCM) and Proposed Method-2 (K-Means), exhibit a slight reduction in energy consumption to 5.56 kWh.

While this reduction suggests marginal efficiency, it is not statistically significant. Cluster-prioritized methods, spanning Proposed Methods-3 to 8, show varying energy consumption patterns, with some configurations matching the baseline while others demonstrate slight reductions. This indicates that cluster prioritization can influence energy consumption depending on the specific configurations employed.

The overall Service Level Agreement Violations (SLAV) are critical metrics indicating the frequency of overall SLAV across the evaluated methods. The baseline method records an overall SLAV rate of 0.66%, whereas the non-cluster-prioritized methods show higher violation rates ranging from 1.00% to 1.16%. This trend suggests that non-cluster-prioritized approaches are more susceptible to overall SLAV, primarily due to their inability to adapt resource allocation to the specific needs of clusters. Conversely, cluster-prioritized methods (Proposed Methods-3 to 8) exhibit improved overall SLAV rates, ranging from 0.62% to 0.68%, demonstrating that cluster prioritization can significantly enhance resource allocation accuracy and thus reduce overall SLAV.

SLATAH reflects the time efficiency in meeting SLA requirements for each active host. The baseline method records a value of 8.67%. Non-cluster-prioritized methods, including Proposed Method-1 and Proposed Method-2, present slightly higher SLATAH values, ranging from 8.21% to 8.54%, indicating reduced efficiency in meeting SLA targets. In contrast, cluster-prioritized methods yield varying results, with Proposed Method-4 (K-Means with RAM prioritization) achieving an improved value of 7.11% at $C = 3$. This outcome suggests that implementing cluster prioritization can enhance SLA time efficiency in specific configurations.

The frequency of VM migrations (PDM) serves as an indicator of system stability. The baseline ESFEC-RL method records SLA PDM value of 0.62%. In comparison, non-cluster-prioritized methods exhibit increased VM migration frequencies, ranging from 1.00% to 1.16%, reflecting a decline in stability. In contrast, cluster-prioritized methods, such as Proposed Method-4 and Proposed Method-8, report reduced VM migration frequencies between 0.56% and 0.68%. This suggests that effective cluster prioritization can lead to a decrease in VM migration frequency, thereby enhancing system stability.

TABLE 5. Results of ESFEC-RL, FCM, and K-means with 3, 4, and 5 clusters.

Method	C=3 (EC (kWh), Overall SLAV, SLATAH, SLA PDM)	C=4 (EC (kWh), Overall SLAV, SLATAH, SLA PDM)	C=5 (EC (kWh), Overall SLAV, SLATAH, SLA PDM)
Jeong et al. [2]	5.60, 0.66%, 8.67%, 0.62%	5.60, 0.66%, 8.67%, 0.62%	5.60, 0.66%, 8.67%, 0.62%
Proposed Method-1 (FCM)	5.56, 1.11%, 8.21%, 1.11%	5.56, 1.16%, 8.21%, 1.16%	5.56, 1.14%, 8.54%, 1.10%
Proposed Method-2 (K-Means)	5.56, 1.03%, 8.38%, 1.00%	5.56, 1.08%, 8.38%, 1.08%	5.56, 1.00%, 8.38%, 1.00%
Proposed Method-3 (RAM Priority, FCM)	5.60, 0.65%, 8.83%, 0.61%	5.60, 0.64%, 8.83%, 0.59%	5.60, 0.66%, 8.67%, 0.62%
Proposed Method-4 (RAM Priority, K-Means)	5.54, 0.68%, 7.11%, 0.68%	5.60, 0.64%, 8.83%, 0.58%	5.60, 0.64%, 9.00%, 0.60%
Proposed Method-5 (CPU Priority, FCM)	5.60, 0.62%, 8.50%, 0.59%	5.60, 0.64%, 8.83%, 0.60%	5.60, 0.65%, 8.83%, 0.61%
Proposed Method-6 (CPU Priority, K-Means)	5.58, 0.68%, 8.38%, 0.66%	5.60, 0.64%, 8.83%, 0.59%	5.60, 0.63%, 8.83%, 0.59%
Proposed Method-7 (RAM+CPU Priority, FCM)	5.60, 0.64%, 8.83%, 0.60%	5.60, 0.65%, 8.83%, 0.61%	5.60, 0.66%, 8.67%, 0.62%
Proposed Method-8 (RAM+CPU Priority, K-Means)	5.59, 0.67%, 8.68%, 0.66%	5.60, 0.62%, 9.00%, 0.56%	5.60, 0.63%, 8.83%, 0.59%
Proposed Method-9 (Modified Reward Log-CPU, FCM)	5.60, 0.66%, 8.67%, 0.62%	5.60, 0.62%, 9.17%, 0.56%	5.60, 0.66%, 8.67%, 0.62%
Proposed Method-10 (Modified Reward Log-CPU, K-Means)	5.54, 0.74%, 7.60%, 0.74%	5.60, 0.63%, 8.83%, 0.59%	5.60, 0.62%, 8.50%, 0.59%
Proposed Method-11 (Modified Reward Log-RAM, FCM)	5.60, 0.62%, 9.17%, 0.56%	5.60, 0.62%, 8.83%, 0.58%	5.60, 0.64%, 9.00%, 0.58%
Proposed Method-12 (Modified Reward Log-RAM, K-Means)	5.58, 0.69%, 8.21%, 0.69%	5.60, 0.64%, 8.83%, 0.60%	5.60, 0.61%, 8.83%, 0.57%

TABLE 6. Results number of VM and ESV.

Method	C=3	C=4	C=5	ESV (C=3)	ESV (C=4)	ESV (C=5)
Jeong et al [2]		49		0.037	0.037	0.037
Proposed Method-1 (FCM)	66	68	70	0.062	0.064	0.063
Proposed Method-2 (K-Means)	84	84	87	0.057	0.060	0.056
Proposed Method-3 (RAM Priority, FCM)	50	49	47	0.036	0.036	0.037
Proposed Method-4 (RAM Priority, K-Means)	44	46	48	0.038	0.036	0.036
Proposed Method-5 (CPU Priority, FCM)	47	49	49	0.035	0.036	0.036
Proposed Method-6 (CPU Priority, K-Means)	47	49	49	0.038	0.036	0.035
Proposed Method-7 (RAM+CPU Priority, FCM)	49	47	49	0.036	0.036	0.037
Proposed Method-8 (RAM+CPU Priority, K-Means)	49	49	48	0.037	0.035	0.035
Proposed Method-9 (Modified Reward Log-CPU, FCM)	49	49	48	0.037	0.035	0.037
Proposed Method-10 (Modified Reward Log-CPU, K-Means)	47	47	46	0.041	0.035	0.035
Proposed Method-11 (Modified Reward Log-RAM, FCM)	49	49	48	0.035	0.035	0.036
Proposed Method-12 (Modified Reward Log-RAM, K-Means)	47	48	49	0.039	0.036	0.034

Table 6 presents a comparison of various VM allocation methods and Energy SLA Violation (ESV) across experimental settings of C = 3, C = 4, and C = 5. The baseline method, ESFEC-RL, serves as a reference with a constant number of 49 VMs and an energy balance value of 0.037 across all settings.

The proposed methods exhibit significant variations in both the number of VMs allocated and the ESV achieved. For instance, Proposed Method-1 (FCM without Cluster Prioritization) shows an increase in the number of VMs as C rises, from 66 at C = 3 to 70 at C = 5. However, this increase comes with a rise in the ESV from 0.062 to 0.063, indicating a trade-off where more VMs are allocated at the expense of energy efficiency.

Proposed Method-2 (K-Means without Cluster Prioritization) also demonstrates a similar trend, with the number of VMs increasing from 84 at C = 3 to 87 at C = 5. Despite this, the ESV remains relatively stable with minor fluctuations, suggesting that K-Means offers slightly better energy efficiency compared to FCM without prioritization.

Methods incorporating RAM-based cluster prioritization, such as Proposed Method-3 (FCM with RAM Prioritization), show a decrease in the number of VMs as C increases, from 50 at C = 3 to 47 at C = 5. This reduction in VMs is accompanied by an improvement in energy balance from 0.036 at C = 3 to 0.037 at C = 5, highlighting that prioritizing RAM enhances energy efficiency.

Similarly, Proposed Method-4 (K-Means with RAM Prioritization) reflects a decrease in the number of VMs, from 44 at C = 3 to 48 at C = 5, while maintaining a stable ESV of 0.036. This consistency in ESV suggests that K-Means with RAM prioritization can optimize VM allocation effectively while preserving energy efficiency.

In the case of CPU prioritization, Proposed Method-5 (FCM with CPU Prioritization) and Proposed Method-6 (K-Means with CPU Prioritization) exhibit similar results, with slightly lower ESV values of 0.035. This indicates that CPU prioritization can also contribute to improved energy efficiency.

Methods that combine RAM and CPU prioritization, such as Proposed Method-7 (FCM with RAM+CPU Prioritization) and Proposed Method-8 (K-Means with RAM+CPU Prioritization), show relatively stable ESV

values. Specifically, FCM with combined prioritization maintains energy balance values ranging from 0.036 at C = 3 to 0.037 at C=5, while K-Means shows a value of 0.037 at C = 3 and 0.035 at both C = 4 and C = 5. This stability suggests that combining both RAM and CPU prioritization provides a balanced approach to energy efficiency.

Finally, methods with modified reward functions, such as Proposed Method-9 (FCM with Cluster Priority and Modified Reward Function Log – CPU, $\theta = 1$) and Proposed Method-10 (K-Means with Cluster Priority and Modified Reward Function Log – CPU, $\theta = 1$), display greater variation in ESV. FCM shows a value of 0.037 at C = 3 and 0.035 at C = 4, while K-Means varies from 0.041 at C = 3 to 0.035 at C = 4 and C = 5. This indicates that modified reward functions can influence ESV significantly.

In conclusion, methods that incorporate cluster prioritization and reward function modifications exhibit notable improvements in several key aspects, including overall SLAV, the number of VM migrations, and ESV, compared to those without cluster prioritization and the baseline approach. However, the impacts on energy consumption and SLATAH are variable, highlighting that the effectiveness of cluster prioritization strategies is contingent upon the specific parameters and settings applied.

An evaluation using the Friedman Test reveals significant differences among the tested methods concerning energy consumption, SLAV, SLATAH, and PDM across different values of C. The Friedman test's Q_F value of 20.78, which exceeds the critical chi-square value of 19.675 for 11 degrees of freedom at a significance level of $\alpha = 0.05$. The obtained p-value is smaller than 0.05, confirming statistical significance. At a significance level of 0.05, the null hypothesis is strongly rejected, indicating statistically significant differences among the evaluated methods [39]. This result supports the rejection of the null hypothesis, indicating significant differences among the methods.

The Friedman Test results confirm that the proposed method significantly improves the energy efficiency, SLA violations, and other QoS metrics compared to the previous study [2]. To ensure the robustness of these findings, future studies can include additional statistical tests and larger datasets to further validate the reproducibility of the results.

Among the tested methods, Proposed Method-4, which utilizes K-Means and a reward function that prioritizes cluster selection based on RAM, stands out as the most effective. It demonstrates superior performance in energy consumption and SLATAH while maintaining relatively low PDM values. These results suggest that Proposed Method-4 is the most efficient and effective approach for optimizing overall system performance in the context of this study.

In summary, the analysis shows that the method combining cluster priority and modified reward function has a significant impact on VM allocation and energy efficiency. Variations in the number of VMs and energy balance underline the effectiveness of this method in optimizing resource allocation and improving energy efficiency.

This modification increases the system's responsiveness to subtle performance variations, allowing for more precise energy consumption adjustments without compromising Quality of Service (QoS). By integrating CPU and RAM metrics into the reward function, the system gains a more comprehensive understanding of how each action affects both energy efficiency and overall system performance. Experimental evaluations reveal that methods utilizing the modified reward function, effectively reduce SLA violations and significantly improve energy balance compared to previous approaches. Furthermore, this integration enhances adaptability in managing the dynamic and heterogeneous characteristics of Federated Edge Computing (FEC) environments. The proposed approach successfully overcomes the limitations of prior methods, which often relied on static reward functions that were less responsive to environmental changes. Consequently, the modified reward function in UCRL-FEC not only enhances energy efficiency and resource management but also strengthens the system's ability to sustain optimal QoS in complex computing environments.

V. CONCLUSION

The discussion highlights the effectiveness of VM allocation methods in Federated Edge Cloud (FEC) systems, focusing on the integration of Unsupervised Cluster Reinforcement Learning (UCRL-FEC) and reward function modifications. Addressing the first research question, unsupervised clustering methods like Fuzzy C-Means (FCM) and K-Means effectively identify migratable VM candidates from overloaded hosts when combined with Q-Learning. Notably, Proposed Method-4 (K-Means with RAM prioritization) reduces VM numbers while maintaining energy balance, demonstrating improved efficiency in VM migration.

For the second research question, modifying the reward function within Q-Learning enhances QoS in FEC. Proposed Method-9 and Proposed Method-10, utilizing modified reward functions, show improvements in energy balance and QoS, highlighting the role of tailored reward functions in optimizing resource allocation.

The Friedman Test confirms significant differences among tested methods in energy consumption, SLA violations (SLAV), SLA time per active host (SLATAH), and performance degradation due to migration (PDM), with Q_F value of 20.78 exceeding the chi-square threshold of 19.675 for 11 degrees of freedom at a significance level of $\alpha = 0.05$, with a p-value smaller than 0.05. These results validate the effectiveness of unsupervised clustering and reward function modifications in improving energy efficiency and QoS.

Future research should explore larger cluster scales and more complex workloads, further optimizing the logarithmic reward function with CPU and RAM weights. Integrating deep reinforcement learning and testing in heterogeneous cloud environments could enhance real-world applicability.

REFERENCES

[1] Y. Jeong, E. Maria, and S. Park, "Towards energy-efficient service scheduling in federated edge clouds," *Cluster Comput.*, vol. 26, no. 5, pp. 2591–2603, Oct. 2023, doi: 10.1007/s10586-021-03338-9.

[2] Y. Jeong, K. E. Maria, and S. Park, "An energy-efficient service scheduling algorithm in federated edge cloud," in *Proc. IEEE Int. Conf. Autonomic Comput. Self-Organizing Syst. Companion (ACSOS-C)*, Aug. 2020, pp. 48–53, doi: 10.1109/ACSOS-C51401.2020.00028.

[3] Y. Li, B. Liu, E. Wu, J. Li, Z. Zhou, and W. Zhang, "DRA-MQoS: AnMQoSscheduling algorithm based on resource feature matching in federated edge cloud," *Concurrency Comput.*, vol. 35, no. 2, p. e7478, Jan. 2023, doi: 10.1002/cpe.7478.

[4] Y. Wang and J. Yu, "Resource scheduling based on reinforcement learning based on federated learning," *J. Softw.*, vol. 2021, pp. 39–45, Jan. 2021, doi: 10.17706/jsw.16.1.39-45.

[5] G. Shidik, E. Noersasongko, A. Nugraha, and O. Setiono, "Migrating virtual machine in heterogen and homogen cloud data center with efficient electricity consumption using fuzzy C-meams," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 6, pp. 199–206, Dec. 2019, doi: 10.22266/ijies2019.1231.19.

[6] G. F. Shidik, A. Nugraha, P. N. Andono, E. Noersasongko, and E. J. Kusuma, "Energy efficiency in cloud data center through unsupervised rule-based VM selection methods," *J. Eng. Sci. Technol.*, vol. 16, no. 6, pp. 5013–5026, 2021.

[7] G. M. Nyabuto, "A review of current research trends in green computing," *East Afr. J. Inf. Technol.*, vol. 7, no. 1, pp. 51–59, Jan. 2024, doi: 10.37284/eajit.7.1.1707.

[8] M. Yenugula, "Optimizing load balancing for green cloud via efficient scheduling," *Int. J. Adv. Academic Stud.*, vol. 4, no. 3, pp. 224–230, Jul. 2022, doi: 10.33545/27068919.2022.v4.i3c.1125.

[9] D. S. Saxena, D. M. Z. Khan, D. R. Singh, M. A. Agarwal, and M. P. Pramanik, "Clustering based prediction for VM workload in green computation," *Educ. Admin., Theory Pract.*, vol. 30, no. 5, pp. 9657–9666, 2024, doi: 10.53555/kuey.v30i5.2585.

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

[11] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2579198.

[12] H. Liao, X. Li, D. Guo, W. Kang, and J. Li, "Dependency-aware application assigning and scheduling in edge computing," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4451–4463, Mar. 2022, doi: 10.1109/JIOT.2021.3104015.

[13] A. Samanta and J. Tang, "Dyme: Dynamic microservice scheduling in edge computing enabled IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6164–6174, Jul. 2020, doi: 10.1109/JIOT.2020.2981958.

[14] A. Samanta, T. G. Nguyen, T. Ha, and S. Mumtaz, "Distributed resource distribution and offloading for resource-agnostic microservices in industrial IoT," *IEEE Trans. Veh. Technol.*, vol. 72, no. 1, pp. 1184–1195, Jan. 2023, doi: 10.1109/TVT.2022.3206137.

[15] A. Samanta, Q.-V. Pham, N.-N. Dao, A. Muthanna, and S. Cho, "MISO: Incentivizing demand-agnostic microservices for edge-enabled IoT networks," *IEEE Trans. Services Comput.*, vol. 16, no. 5, pp. 3523–3536, Sep. 2023, doi: 10.1109/TSC.2023.3292362.

[16] K. Kim, J.-H. Lee, H.-K. Lim, S. Oh, and Y.-H. Han, "Deep RNN-based network traffic classification scheme in edge computing system," *Comput. Sci. Inf. Syst.*, vol. 19, no. 1, pp. 165–184, 2022, doi: 10.2298/csis200424038k.

[17] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using A3C learning and residual recurrent neural networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 3, pp. 940–954, Mar. 2022, doi: 10.1109/TMC.2020.3017079.

[18] D. Wu, H. Xu, Z. Jiang, W. Yu, X. Wei, and J. Lu, "EdgeLSTM: Towards deep and sequential edge computing for IoT applications," *IEEE/ACM Trans. Netw.*, vol. 29, no. 4, pp. 1895–1908, Aug. 2021, doi: 10.1109/TNET.2021.3075468.

[19] F. Qi, L. Zhuo, and C. Xin, "Deep reinforcement learning based task scheduling in edge computing networks," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Aug. 2020, pp. 835–840, doi: 10.1109/ICCC49849.2020.9238937.

[20] T. Wang, B. Lu, W. Wang, W. Wei, X. Yuan, and J. Li, "Reinforcement learning-based optimization for mobile edge computing scheduling game," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 1, pp. 55–64, Feb. 2023, doi: 10.1109/TETCI.2022.3145694.

[21] T. Zheng, J. Wan, J. Zhang, and C. Jiang, "Deep reinforcement learning-based workload scheduling for edge computing," *J. Cloud Comput.*, vol. 11, no. 1, p. 3, Dec. 2022, doi: 10.1186/s13677-021-00276-0.

[22] T. Cui, R. Yang, C. Fang, and S. Yu, "Deep reinforcement learning-based resource allocation for content distribution in IoT-edge-cloud computing environments," *Symmetry*, vol. 15, no. 1, p. 217, Jan. 2023, doi: 10.3390/sym15010217.

[23] L. Yang, D. Yang, J. Cao, Y. Sahni, and X. Xu, "QoS guaranteed resource allocation for live virtual machine migration in edge clouds," *IEEE Access*, vol. 8, pp. 78441–78451, 2020, doi: 10.1109/ACCESS.2020.2989154.

[24] B. Hayat, K. H. Kim, and K.-I. Kim, "A study on fuzzy logic based cloud computing," *Cluster Comput.*, vol. 21, no. 1, pp. 589–603, Mar. 2018, doi: 10.1007/s10586-017-0953-x.

[25] W. Ye, K. Zheng, Y. Wang, and Y. Tang, "Federated double deep Q-learning-based computation offloading in mobility-aware vehicle clusters," *IEEE Access*, vol. 11, pp. 114475–114488, 2023, doi: 10.1109/ACCESS.2023.3324718.

[26] J. Uma, P. Vivekanandan, and S. Shankar, "Optimized intellectual resource scheduling using deep reinforcement Q-learning in cloud computing," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 5, p. e4463, May 2022, doi: 10.1002/ett.4463.

[27] Z. Salman and A. Alomary, "Performance of the K-means and fuzzy C-means algorithms in big data analytics," *Int. J. Inf. Technol.*, vol. 16, no. 1, pp. 465–470, Jan. 2024, doi: 10.1007/s41870-023-01436-y.

[28] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Comput. Geosci.*, vol. 10, nos. 2–3, pp. 191–203, Jan. 1984, doi: 10.1016/0098-3004(84)90020-7.

[29] G. F. Shidik, N. S. Sulistyowati, and M. B. W. Tirta, "Evaluation of cluster K-means as VM selection in dynamic VM consolidation," in *Proc. 22nd Asia–Pacific Conf. Commun. (APCC)*, Aug. 2016, pp. 124–128, doi: 10.1109/APCC.2016.7581475.

[30] M. H. S., S. K. T., P. Gupta, and G. McArdle, "A Harris hawk optimisation system for energy and resource efficient virtual machine placement in cloud data centers," *PLoS ONE*, vol. 18, no. 8, Aug. 2023, Art. no. e0289156, doi: 10.1371/journal.pone.0289156.

[31] U. Singh, A. Dua, and N. Kumar, "SDN based dynamic resource scheduling for large scale data centers," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, Dec. 2019, pp. 1–6, doi: 10.1109/ANTS47819.2019.9117988.

[32] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," in *Advances in Computers*. Amsterdam, The Netherlands: Elsevier B.V., 2011, ch. 3, pp. 47–111, doi: 10.1016/B978-0-12-385512-1.00003-7.

[33] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "GreenCloud: A packet-level simulator of energy-aware cloud computing data centers," in *Proc. IEEE Global Telecommun. Conf. GLOBECOM*, Dec. 2010, pp. 1–5, doi: 10.1109/GLOCOM.2010.5683561.

[34] F. Cuomo, E. Cipollone, and A. Abbagnale, "Performance analysis of IEEE 802.15.4 wireless sensor networks: An insight into the topology formation process," *Comput. Netw.*, vol. 53, no. 18, pp. 3057–3075, Dec. 2009, doi: 10.1016/j.comnet.2009.07.016.

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

[36] A. Verma, P. Ahuja, and A. Neogi, "PMapper: Power and migration cost aware application placement in virtualized systems," in *Proc. ACM/IFIP/USENIX Int. Conf. Distrib. Syst. Platforms Open Distrib. Process.*, 2008, pp. 243–264, doi: 10.1007/978-3-540-89856-6_13.

[37] S. Kumar and R. Buyya, "Green cloud computing and environmental sustainability," in *Harnessing Green It: Principles and Practices*. Hoboken, NJ, USA: Wiley, 2012, pp. 315–339, doi: 10.1002/9781118305393.ch16.

[38] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012, doi: 10.1016/j.future.2011.04.017.

[39] D. G. Pereira, A. Afonso, and F. M. Medeiros, "Overview of Friedman's test and post-hoc analysis," *Commun. Statist.-Simul. Comput.*, vol. 44, no. 10, pp. 2636–2653, Nov. 2015, doi: 10.1080/03610918.2014.931971.

GURUH FAJAR SHIDIK (Member, IEEE) received the Bachelor of Computer Science degree from Universitas Dian Nuswantoro (UDINUS), Semarang, Indonesia in, 2009, the Master of Computer Science degree from Universiti Teknikal Malaysia Melaka (UTeM), in 2011, and the Ph.D. degree from Universitas Gadjah Mada (UGM), Indonesia, in 2016. Currently, he is a Lecturer with UDINUS. His research interests include cloud computing, wireless communications, machine learning, and artificial intelligence.

OKI SETIONO received the bachelor's and master's degrees in computer science from Universitas Dian Nuswantoro, Semarang, Indonesia, in 2014 and 2017, respectively. He is currently a Lecturer and a Researcher with the Faculty of Health Sciences, Medical Records and Health Information Study Program, Universitas Dian Nuswantoro. His research interest includes cloud computing.

EDI JAYA KUSUMA received the Bachelor of Computer Science and master's degrees in computer science from Universitas Dian Nuswantoro (UDINUS), Semarang, Indonesia, in 2018 and 2020, respectively. Currently, he is a Lecturer with the Faculty of Health Sciences, UDINUS. His research interests include machine learning, artificial intelligence, and image processing.

L. BUDI HANDOKO received the bachelor's and master's degrees in computer science from Universitas Dian Nuswantoro, Semarang, Indonesia, in 2000 and 2009, respectively. He is currently a Lecturer and a Researcher with the Faculty of Computer Sciences, Information Technology Study Program, Universitas Dian Nuswantoro. His research interests include computer networks, cloud infrastructure, security, and the IoT.

PULUNG NURTANTIO ANDONO received the Bachelor of Engineering degree from Universitas Trisakti, Jakarta, in 2006, the Master of Computer Science degree from Universitas Dian Nuswantoro (UDINUS), Indonesia, in 2009, and the Doctorate (Ph.D.) degree from Institut Teknologi Sepuluh November (ITS), Surabaya, in 2016. Currently, he is a Professor and a Ph.D. Supervisor with the Faculty of Computer Science, UDINUS. His research interests include 3D image reconstruction and computer vision.

MOHD FAIZAL ABDOLLAH received the degree from Universiti Utara Malaysia, the master's degree from Universiti Kebangsaan Malaysia, and the Ph.D. degree in computer and network security from Universiti Teknikal Malaysia Melaka (UTeM). He is currently a Senior Lecturer with the Department of Computer and Communication Systems, Faculty of Information and Communication Technology, UTeM. Previously, he was a MIS Executive with EON Berhad, Selangor, and as a System Engineer with Multimedia University, Melaka, for six years. His main research interests include network and wireless technology, network management, and network and wireless security.