ZANCO Journal of Pure and Applied Sciences

TYPE Original article
DOI http://dx.doi.org/10.21271/ZJPAS.38.1.12

OPEN ACCESS

*Corresponding author
Widad Dilawer Issa
widadilawer9@gmail.com

RECEIVED :01 /02 /2025
ACCEPTED :05/08/ 2025
PUBLISHED :28/ 02/ 2026

KEYWORDS:
IoT, QoS, fog, DRL, dynamic scheduling, dynamic resource allocation

An Integrated Fog Computing Approach to Improve Quality of Service in IoT Environment with Dynamic Task Scheduling and Resource Allocation

Widad Dilawer Issa[1*],Marwan Aziz Mohammed[2]

1Department of Computer Science and IT, College of Science, Salahaddin University-Erbil, Erbil, Kurdistan Region, Iraq

2 Department Software Engineering, College of Engineering, Salahaddin University-Erbil, Erbil, Kurdistan Region, Iraq

ABSTRACT

Exponential growth in IoT application involves managing latency, resource utilization and reliability in traditional cloud based architectures. In this research, we present a dynamic IoT-Fog-Cloud architecture using fuzzy logic, DQN and MOEA/D to optimize QoS metrics in real time. The real-time tasks generated by simulated IoT devices are falling through fuzzy logic at the Fog Master node and classified according to urgency. DQN schedules tasks dynamically so that during runtime it discovers optimal tasks solution on a set of Fog nodes. Resource allocation is enhanced by MOEA/D balancing computational loads with trade-off between latency, reliability and resource utilization. Here, we compare static and cloud-only architectures and show comparative simulations in iFogSim tool that achieve 28% reduction in latency, 19% increase in reliability and 22% improvement in resource utilization. The system's hierarchical adaptive nature provides for an efficient task processing that is scalable and with superior QoS for IoT applications. The proposed framework is a scalable, robust and efficient means by which one can run real-time IoT tasks in fog-cloud computing environments by synergizing advanced deep learning and evolutionary algorithms.

1.Introduction

The Internet of Things (IoT) represents a shift in how devices will communicate, share data, and interact with the physical world (Ksentini et al., 2021). With billions of devices interconnecting, there has been an urgent need for efficient, scalable, responsive systems. Nevertheless, the traditional cloud computing model fails to address the sheer volumes of data generated by IoT devices, specifically, dealing with latency, bandwidth, and resource allocation (Al-khafajiy et al., 2018). This has given rise to fog computing, a decentralized computing framework that replaces cloud capabilities with edge networks.

Fog computing addresses the inherent limitations of cloud computing. It takes a different approach, enabling processed data closer to the source where it has been generated, thus providing enhanced Quality of Service (QoS) for IoT applications. It offers a middle layer on top of the IoT devices to the cloud, but distributes the computational resources over several fog nodes near the data source. It provides low-latency processing, efficient resource utilization, and high reliability for real-time applications, including autonomous vehicles, intelligent cities, and healthcare monitoring (Mahmud et al., 2019a). Additionally, fog computing extends the computation to the network's edge, enabling effective integration of diverse data types and volumes into the fog for advanced analytics and more effective decision-making at the network's edge than traditional approaches (Kareem and Ghafoor, 2021).

Although fog computing is being implemented, there are some challenges to it. With the rapid increase in connected devices, resource allocation and QoS have become complex problems (Alsadie, 2024). With the dynamic nature of IoT environments, intelligent resource management strategies suited to varying workloads, QoS requirements, and the inherent unpredictability in the network behavior are needed.

However, poor user experiences, system failures, and safety hazards compromise QoS management in the IoT environment, particularly in mission-critical scenarios. This research attempts to design fog computing and intelligent resource management techniques to enhance the QoS of the Internet of Things (IoT) environment through synergies between Reinforcement Learning (RL), fog computing, and fuzzy logic. It is expected to yield great results before fog computing and its application in the IoT domain.

The study utilizes artificial intelligence (AI) and machine learning (ML) techniques to maximize resource allocation with the least energy consumption in fog computing environments. This integrates intelligent systems (e.g., Multi-Objective Evolutionary Algorithm (MOEA/D)) for dynamically optimizing resource allocation amongst fog nodes. By utilizing RL algorithms like Deep Q-Networks (DQN), real-time learning and adaptations of dynamically optimizing resource usage can be done from our historical information and current conditions. Second, task scheduling and resource allocation may be automatically optimized according to QoS criteria of latency, resource utilization, and reliability using these algorithms. To address the management of uncertainties and complexities in the IoT environment and its robustness, fuzzy logic is combined with RL. The system enables the incorporation of qualitative human knowledge into decision-making processes to assess quality of service parameters, such as reliability, latency, and resource utilization, more precisely. RL and Fuzzy logic synergy is combined to enhance the adaptability of the fog computing systems and satisfy diverse QoS requirements in the IoT applications.

2. Literature Review

The need to integrate intelligent computing paradigms in IoT-fog premises is becoming a necessity owing to the explosive increase in the

number of heterogeneous devices and real-time requirements of contemporary applications. Fog computing was introduced as a parallel paradigm to clouds in the traditional sense of the word offering computation, storage, and networking nearer to the source of data to decrease latency and increase reliability. According to (Walia et al., 2024, Alsadie, 2024), fog computing supports scalable systems with low latency, which are essential to smart cities, healthcare, and autonomous settings.

Nevertheless, decentralization is achieved with fog architectures, but there is no easy task of scheduling and managing resources in dynamic and unreliable environments. Over the last several papers, the scholars have shown that a combination of smart decision-making methods like fuzzy logic, reinforcement learning (especially DQN), and multi-objective optimization (MOEA/D) can result in more responsive and robust systems (Ibrahim and Askar, 2023, Kouka et al., 2024).

2.1 Fog and IoT using Fuzzy Logic

Fuzzy logic is instrumental in categorizing and prioritizing the tasks according to the uncertainty and real-world ambiguous data, which is abundant in IoT. According to (Irwanto et al., 2024), fuzzy inference systems can add subtlety to task processing by mapping the raw system measurements, e.g., latency, load, or urgency to the degrees of priority. In contrast to binary logic, fuzzy logic does permit an overlap in the degrees of membership (e.g. a task being both medium and high urgency), which provides greater flexibility of the system (Asaad et al., 2024).

Other recent applications like the fuzzy controller in mushroom growing (Irwanto et al., 2024) or intelligent energy scheduling (Sarkar et al., 2023) have seen greater accuracy and flexibility in making decisions with fuzzy logic regarding the changing environmental demands. In a similar manner in fog systems, fuzzy logic can be used to ensure that high QoS sensitive tasks are identified early and dynamically prioritized thus mitigating service degradation.

2.2 DQN for Adaptive Scheduling

Reinforcement Learning (RL) and, particularly, DQN is attracting interest due to the capability to learn in real-time to adjust to complicated system conditions. DQN learns a value-based function that focuses on mapping system states (e.g., fog node loads, task urgency) to the best scheduling actions. In (Shukla and Tchouankem, 2023), it was demonstrated that DRL systems could learn optimal policies adapting to the network conditions to respond better to the vehicular communication system.

RL in fog computing enables the scheduler to manage stochastic processes, i.e., variable task arrival rates or dynamically available nodes through past experience. In (Gong et al., 2024, Su et al., 2023), multi-agent DRL was proposed to jointly optimize the VM placement and resource scheduling, showing substantially better convergence and flexibility than heuristic approaches. In addition, experience replay and target networks offer DQN a stable learning process that is essential in mission-critical environments like in telemedicine or autonomous control.

DQN is learned in our architecture to dynamically select the fog nodes to execute high-priority tasks to reduce the deadline misses and balance the loads. The ε-greedy policy provides exploration at the beginning, and as time goes by the system begins to exploit the learned policies- it shows stable convergence and scale in large simulations.

2.3 MOEA/D on Multi-Objective Optimization

Scheduling is addressed by DQN, resource allocation is a more strategic problem of contrasting goals: latency minimization, reliability maximization, and resource utilization optimization. MOEA/D solves this through decomposing the global optimization issue into scalar subproblems along the weight vectors (Luong et al., 2020, Saeed et al., 2024).

More recent papers like (Kouka et al., 2024) used MOEA/D in load balance in fog clusters and demonstrated the benefits of weighted decomposition in enhancing solution diversity

and convergence rate. MOEA/D used in our system improves the scheduling results generated by DQN through crossover and mutation operators to develop optimal allocations. This is assured by the use of multiple objective functions to make sure that nodes are not overloaded or underutilized making the system unstable.

Moreover, enhanced MOEA/D versions like adaptive weight adaptation (Gong et al., 2024) or selection by crowding distance (Vijarania et al., 2023) ensure better quality of the Pareto front, which enables the system to be resilient to workload spikes and failures.

2.4 Fuzzy, DQN, and MOEA/D Synergetic Integration

A number of works benefit from having fuzzy logic, RL and optimization integrated within the same framework. As an example, (Abdulazeez and Askar, 2024) designed a hybrid fuzzy-RL offloading mechanism, which minimized the network delay in fog-cloud systems. On the same note, (Irwanto et al., 2024) underlined the purpose of hybrid models to improve privacy, efficiency, and control of edge systems.
Our system federates on these concepts and proposes a new layered integration:

1. In the first step, fuzzy logic is used to categorize tasks at the Fog Controller (FC) according to real-time QoS boundaries.
2. DQN at the Fog Cluster Controller (FCC) plans the categorized jobs to suitable nodes depending on the acquired optimum action.
3. The MOEA/D then proceeds to allocate resources optimizing on latency, reliability, and utilization simultaneously.

Such a modular, but interconnected architecture enables autonomous adaptation and worldwide optimization, making it scalable and resilient.

2.5 Recent Trends and Gaps

There are quite a few recent studies that take either RL, fuzzy or evolutionary algorithms separately, but not all three together. That creates a research void of intuitive, intelligent structures that can learn in real time, assign priorities to classification and optimize multi-objectives in one fell swoop. We show in our contribution that this gap is filled by our method, by verifying it through comparison simulation with cloud-only, static, and partially dynamic approaches.

As showed by (Costa et al., 2024, Arcas et al., 2024, Trabelsi and Ben Ahmed, 2024), among others, insist on more profound integration of smart decision-making schemes to make fog infrastructures future-proof. Our findings support the assertion that learning-based resource management systems are far better in enhancing QoS, task completion rate and system flexibility.

3. Proposed System Architecture

The proposed system architecture is a multiple layered system that targets the key issues associated with IoT systems including high latencies, non-optimal resource usage and ineffective task scheduling(Abdulazeez and Askar, 2024, Irwanto et al., 2024, Walia et al., 2024). This architecture is a scalable and efficient solution combining IoT, fog, and cloud computing paradigms to achieve dynamic task management, QoS optimization, and inefficiencies. The architectural structure has three layers: The IoT layer generates data from IoT components like sensors and cameras, preparing tasks for the system. Real-time tasks generated by IoT devices simulate actual IoT workloads, and tasks are prioritized and forwarded to the fog layer for further processing. The MQTT protocol communicates between IoT devices, fog nodes, and the Cloud.

MQTT is a lightweight message protocol based on a publish-subscribe model, which is excellent for IoT. Data can be published from devices to topics, and subscribers receive messages in real time (e.g., fog nodes). It provides efficient and low-latency communication and reliable delivery with minimum overhead. The fog, the most essential layer, manages tasks and resources in the local region. It consists of the fog master, which is called Fog Controller (FC), and multiple Fog Cluster Controllers (FCCs), which control k-means created clusters. The FC

uses fuzzy logic to assess tasks based on latency, reliability, and resources required. The FCCs manage tasks and allocate resources using a Deep Reinforcement Learning (DRL) (like DQN) and an evolutionary algorithm (like MOEA\D). The fog layer also processes tasks, deploying them to cluster nodes. If fog nodes become overloaded or computationally constrained, the FCC collaborates with FC to offload tasks to the cloud layer. The cloud layer provides more computation power and storage space for long-term data accumulation and processed data storage. It ensures scalability, reliability, and QoS requirements, and serves as a backup for the system. These layers are kept in harmony through perfect protocols of conveying data from one layer to the other with maximum efficiency and stability. Figure 1 shows the proposed system architecture. System configuration parameters are set in Table 1, and Table 2 defines and describes the general responsibilities of all components in three layers.

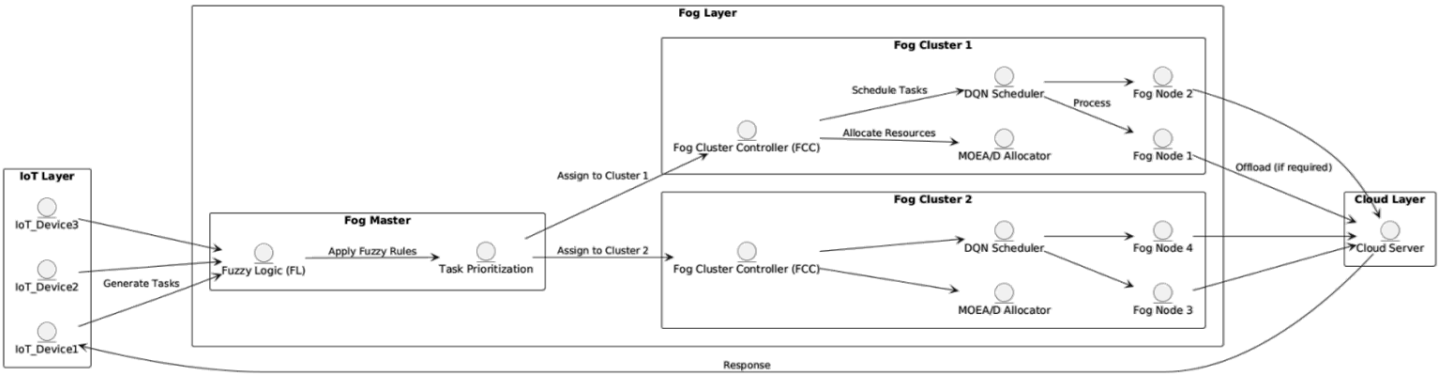

Figure 1. Proposed system architecture

Table 1: System Configurations, components and their description.

Component	Description
IoT Devices	Real-time task generators
Fog Layer	Multi-node fog layer with a Fog Master for task prioritization
Cloud Layer	Handles high-complexity tasks or stores results
Communication Protocol	MQTT for task transfer between layers
Algorithms	Fuzzy Logic (classification), DQN (scheduling), MOEA/D (resource allocation)

Table 2: system component specification.

Parameter	IoT Layer	Fog Layer	Cloud Layer
Data Generation Rate	1 to 5 messages per second, varies based on device type (e.g., sensors, cameras)	-	-
Task Types	Image processing, sensor data collection, and data aggregation with varying processing demands	-	-
Task Size	0.5MB to 5MB per task	-	-
Processing Cores	-	4 to 8 cores for parallel task processing	64 cores
RAM	-	16 to 32 GB, supporting multiple tasks but with limited simultaneous	128 GB, sufficient for high-demand applications

		capacity	
Task Processing Time	-	50 ms to 500 ms, varies based on node load and task complexity	100 ms to 1 second, typically slower than Fog nodes, impacting overall QoS
Role	Creates tasks/data for processing, imitates different real-time data production depending on device type (sensor, camera)	Serves as a bridge between IoT devices and the Cloud, initial data filtering that helps to reduce load on the Cloud	solves what the Fog nodes cannot solve, giving additional computational capabilities for highly computational tasks and handling a significant amount of data.

4. Methodology

4.1 Simulation Environment

The designed simulation model of the study is based on the iFogSim tool, which is an extension of CloudSim to evaluate fog and cloud computing paradigms. This framework enables the dynamic modelling of IoT systems where devices produce tasks that could either be processed at the fog nodes or forwarded to the cloud. The simulation process starts with the generation of IoT devices that can simulate any task and workload that needs to be performed. The iFogSim framework includes parameter specifications for the simulation of real IoT applications. They are delivered through the MQTT protocol to the fog layer, the intermediate layer to accomplish tasks (Saeed et al., 2024). The fog nodes are categorized depending on the system setup, while the fog resources are dynamically assigned with the help of the included algorithms in the simulation.

The system has a fuzzy logic to control the priority of the task to be solved based on the real-time system, and then adaptively control the task sequencing to meet the quantitative QoS requirements. DQN is used for task scheduling, and MOEA/D is used for resource allocation to decide the perfect scheduling regarding different system constraints. Performance parameters including latency, reliability, and resource usage are observed actively in the course of the simulation to determine qualifications of the integration of IoT, fog, and cloud computing layers to satisfy the QoS demands of contemporary applications (Gupta et al., 2016, Mahmud and Buyya, 2018,

Awaisi et al., 2021). In order to guarantee the responsiveness of the system, all experiments have been conducted on the mid-range machine with 16 GB RAM and an Intel Core i7 CPU. Despite the fact that each addition of intelligent elements (Fuzzy Logic, DQN, and MOEA/D) implies some extra computational burden on the system, mainly on memory and processing cycles during training and optimization, the overall system was stable and fell within acceptable limits of simulation. The reinforcement learning loop of DQN and the evolutionary search of MOEA/D also led to an increase in decision time, which was however compensated by the very promising results in terms of QoS metrics. Adaptive learning and real time task processing was maintained without significant performance bottlenecks. Figure 2 Shows the workflow of the system simulation, and table 3 shows the simulation parameters.

Table 3: The simulation parameters used in iFogSim toolkit to simulate the proposed IoT environment.

Parameter	Value
Simulation Toolkit	iFogSim in IntelliJ IDEA
Algorithm Development	Python for Fuzzy, DQN, and MOEA/D
Simulation Time	1000 seconds
Task Generation Rate	10 tasks/second per IoT device
Network Bandwidth	100 Mbps (Fog), 1 Gbps (Cloud)
Latency Threshold	50 ms

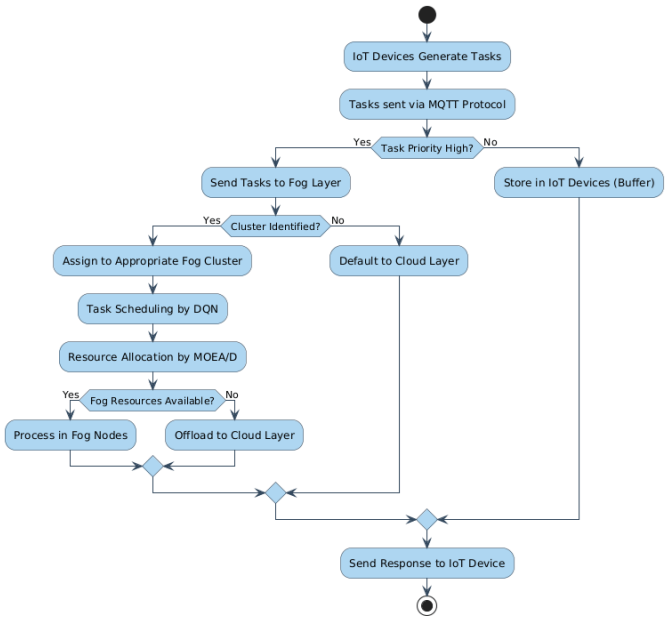

Figure 2. Simulation workflow of the system architecture done by iFogSim in IntelliJ IDEA software.

4.2 Algorithmic Design

In this study, the algorithmic framework has a central role in the scheduling of tasks, utilization of resources, and management of QoS in the IoT-Fog-Cloud system. Implementing fuzzy logic and DQN with MOEA/D makes the system capable of addressing numerous IoT applications' dynamic and complex requirements. Each algorithm is tailored to address specific challenges: Using fuzzy logic for real-time prioritization of the tasks, DQN for schedulers to learn adaptive scheduling, and MOEA/D for optimal allocation of resources to the tasks. In the following section, we provide a detailed explanation of these components design, implementation, and their working. The algorithmic design of the system is summarized in Figure 3.

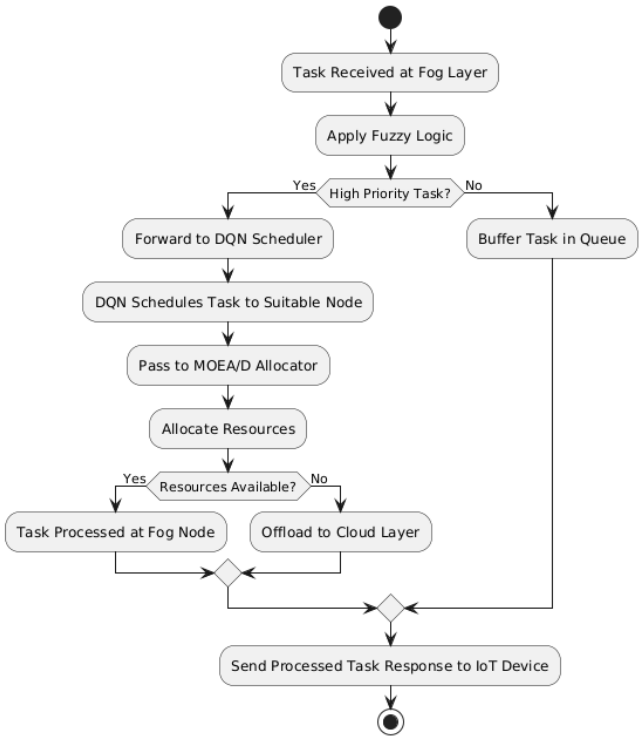

Figure 3. The algorithmic design workflow in a flowchart representation.

4.2.1 Fuzzy Logic

At the fog layer in the FC, fuzzy logic is used to categorize and prioritize tasks based on QoS parameters, meaning high-priority tasks will be addressed without much delay. In traditional logical scenarios, there is usually no provision for requirements or functionality to be on a gradient. Exploiting fuzzy logic, tasks are prioritized as low, medium, and high based on their latency sensitivity, reliability requirements, and resource needs with the help of a membership function. They directly affect the QoS objectives and guarantee adaption capability regarding various workloads (Shukla et al., 2019a, Irwanto et al., 2024). Fuzzy logic processes three features:

(1) Latency: A measure of task urgency. Certain tasks are preferred that demand relatively less latency time.

Formula: Latency = $T_{response} - T_{request}$

$T_{response}$ is the task completion time, and $T_{request}$ is when the task was received.

(2) Reliability: Helps realize dependable performance, especially for work critical to the organization.

Formula: $Reliability = \frac{Number\ of\ Successfully\ Completed\ Tasks}{Total\ Tasks\ Processed} \times 100$

(3) Resource Utilization: Records the traffic load in fog nodes to avoid overloading these nodes where fog computing will occur.

Formula: $Resource\ utilization = \frac{\text{Resources Used}}{\text{Total Available Resources}} \times 100$

Workflow of Fuzzy Logic

The proposed system relies heavily on the fuzzy logic to prioritize real-time IoT tasks based on their urgency and resource specificity. Fuzzification is the first step, where raw input metrics, such as latency, reliability, and resource utilization, are fuzzified into fuzzy sets given by membership functions. For instance, latency is divided up as 'low,' 'medium', or 'high' according to parameters defined by system benchmarks and historical performance data. Focusing on features extracted from the input (e.g., data size, task processing complexity, and response time), the system estimates the latency of the task.

For example, a task with an estimated latency of 15ms has a strong membership in the “low” category, while a task with 60ms may have a part of membership in both the “medium” and “high” categories. In the same way, reliability (90%) is also “medium” or “high”, and resource utilization (80%) is based on thresholds describing the amount of resource required for a task. These values are in real-time and derived from metrics monitored during task generation and initial preprocessing at the IoT device or fog layer.

Through a Rule Evaluation stage, combinations of latency, reliability, and resource usage are evaluated using a predefined set of rules to derive a QoS score using fuzzy logic. A rule like ‘If latency is high and reliability is low, then QoS is poor’ will indicate a set of non-urgent tasks, while a rule like ‘If latency is low and reliability is high, then QoS is excellent’ will identify a set of tasks that need to be considered urgent. However, this rule-based approach is used to identify tasks that must be attended to early or have high resource demands.

In the third, the Defuzzification step, the fuzzy outputs are transformed to precise QoS values using the centroid method. Then they are used to classify tasks into “high priority (urgent)” or “low priority (non-urgent)” categories. Specifically, a task with a high QoS score based on its lateness and firmness is deemed urgent, scheduled, and allocated to resources first. Conversely, all the tasks with low QoS scores are queued or offloaded for later processing. This systematic approach ensures that the system dynamically assesses task urgency and resource demand and schedules the tasks in cooperation with fog nodes to achieve efficient task scheduling and maximize the utilization of the fog nodes’ resources. The fuzzy logic processes are given in Figure 4:

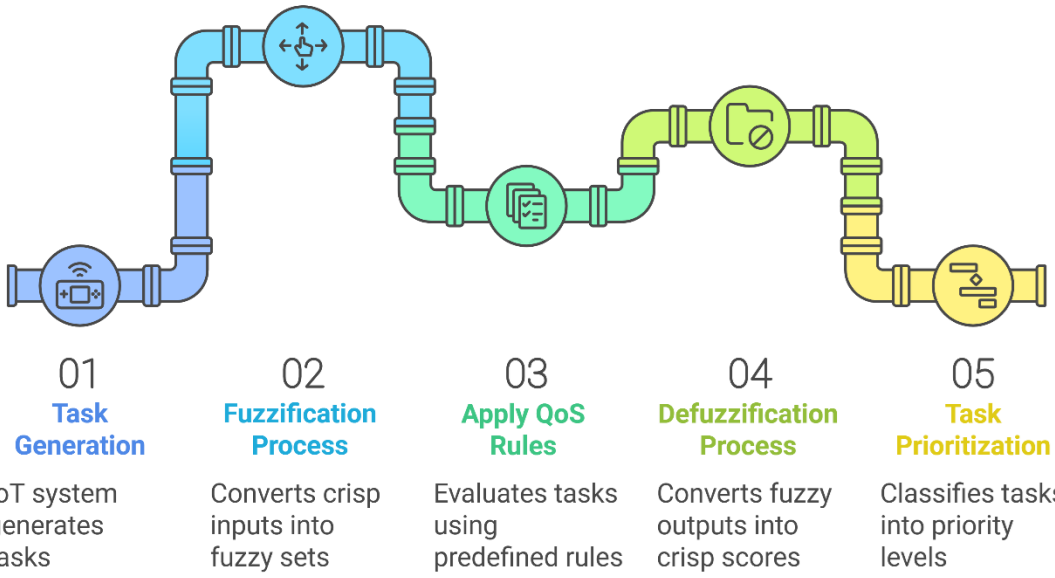

Figure 4. Fuzzy logic task classification workflow

4.2.2 DQN

DQN is used to schedule tasks since the priorities of the assigned tasks can be solved through the most optimal fog nodes. It uses reinforcement learning to determine the best scheduling policies by utilizing its experiences, guaranteeing dynamicity in accommodating the system workloads and fog node conditions. DQN uses a Q-function to represent the expected cumulative reward when taking a particular action up for a specific state. This is written mathematically as (Abdulazeez and Askar, 2024, Gong et al., 2024):

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a')$$

where:

$Q(s,a)$: Expected reward for taking action a in state s.

r : Immediate reward received after action *a* in state *s*.

γ : Discount factor $(0 < \gamma < 1)$ determines the importance of future rewards in the Q-function. A higher γ means future rewards are considered more valuable, while a lower γ emphasizes immediate rewards.

Despite this, the core of DQN is in its neural network architecture, specifically, an input layer exposed to the inputs (system state information such as QoS scores or node statuses), a one or more layers for feature extraction as well as learning relationship and an output layer which takes the feature representation as input and generates Q-values for all possible actions. The experience stored in the replay buffer are represented as state, action, reward and next state tuples, i.e. (*s, a, r, s'*) in the format of the training process. These experiences are sampled as mini batches, and then the network is trained by gradient descent using these mini batches. During training stability, a target network is periodically updated.

Key parameters used in the effectiveness of the Q-function: In this case, the learning rate (α\alpha) controls how much the network updates its weights, how big the discount factor (γ\gamma) is for the trade-off between current and future rewards, and has an epsilon greedy policy that is a balance between exploration (testing new actions) and exploitation (using the best actions learned before). In this study, we used learning rate of 0.001, discount factor of 0.95, and a batch size of 64 to train the DQN. The learning rate of 0.001 was chosen because it provides slow constant corrections of the weights in a network so the improvements are consistent without oscillating. A discount factor of 0.95 was used to ensure consideration of the present and future, making more strategic

decisions. The exploration rate (epsilon e) began at 1.0 and was decayed slowly to 0.01 in order to provide enough exploration before levelling off into exploitation. The experience sampling was diversified by setting a replay buffer size to 10,000 and a batch size to 64. The target network was refreshed after every 100 steps and training was continued with 7000 episodes to enable deep policy learning. These hyperparameters were chosen based on empirical comparison through cross-validation, and grid search for the best performance and stability in our simulations. These combined components allow DQN to reason in a dynamic and complex environment.

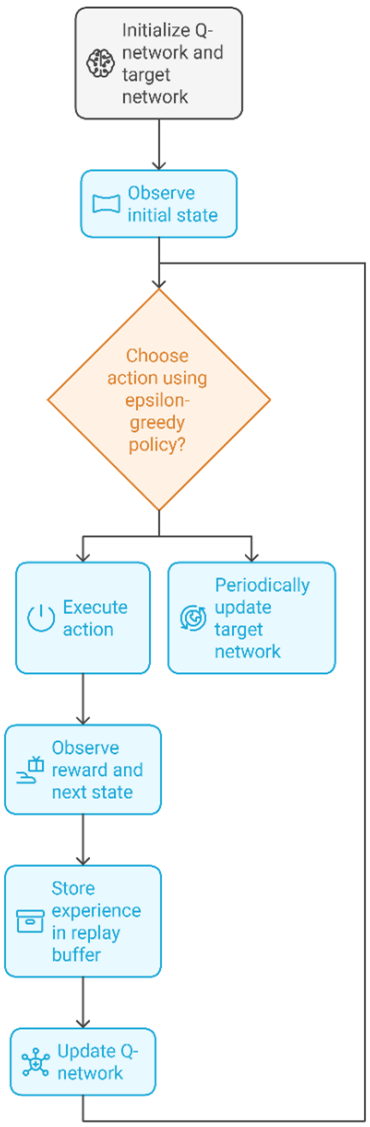

Figure 5. Training DQN

Pseudocode for DQN (see figure 5):

Input: State (QoS Scores, Fog Node Capacities)
Output: Action (Task Assignment)
1. Initialize Q-network and target network.
2. For each training episode:
   a. Observe the initial state.
   b. For each time step:
      i. Choose action using the epsilon-greedy policy.
      ii. Execute action, observe reward, and next state.
      iii. Store experience in the replay buffer.
      iv. Update Q-network using mini-batches from the replay buffer.
3. Periodically update the target network.

4.2.3 MOEA\D

MOEA/D is a robust algorithm that balances multiple objectives, minimizing latency and maximizing reliability when dealing with fog computing resource allocation. The MOEA/D dynamically adapts in our system to the availability of the resources in fog nodes, and the demands of the workload to avoid overburdening nodes for task allocation. The multi-objective optimization problem is decomposed into multiple single-objective sub-problems along weight vectors where each subproblem is a trade-off between objectives such as latency and reliability. The collaboration with neighboring subproblems help them foster the reuse of solution for speedier convergence and better accuracy.

The optimization process is driven by evolutionary operations such as mutation (introducing diversity), crossover (combining parent solutions to generate offspring), and selection (choosing the best solution according to fitness) (Saeed et al., 2024). Specific objective functions for guiding decisions are used by MOEA/D: minimize total task processing time or maximize the sum of reliability metrics for assigned tasks. As a post of DQN, our system evaluates real-time inputs such as resource availability and the workload distribution to refine the scheduling decisions by MOEA/D. MOEA/D for example, if one fog node gets saturated, it will find another fog node with sufficient resources, thus achieving balanced

utilization and improved performance. In this manner, it leads to a dynamic and adaptive resource allocation mechanism, which ensures high QoS, minimizes latency, and preserves system reliability under different workloads (Luong et al., 2020, Shukla and Pandey, 2023, Ibrahim and Askar, 2023, Costa et al., 2024). The MOEA\D's procedure is given in Figure 6 as a flowchart.

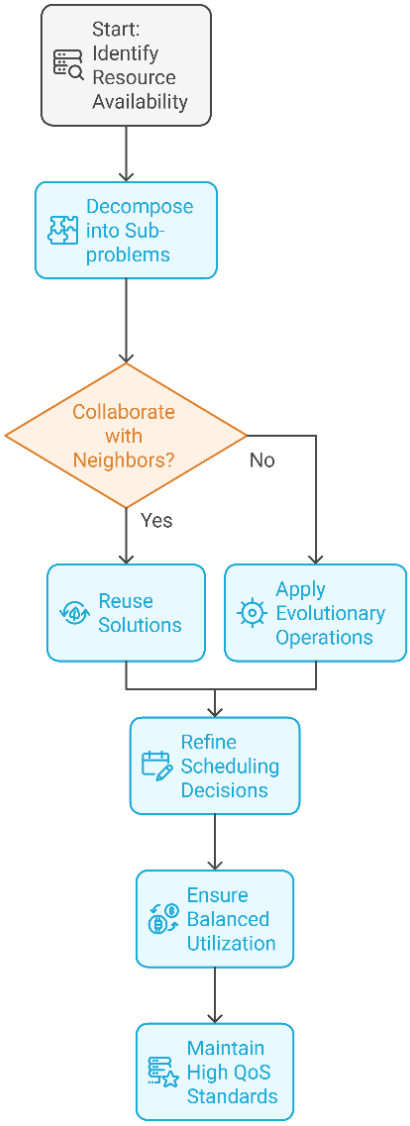

Figure 6. the flowchart of MOEA\D procedure.

Pseudocode for MOEA/D

Input: Fog Node States, Task Requirements
Output: Optimized Resource Allocation
1. Initialize population of solutions (resource allocations).
2. Decompose problem into M sub-problems with weight vectors.
3. For each generation:
   a. Evaluate fitness for each sub-problem.
   b. Share solutions among neighbors.
   c. Apply mutation and crossover operators.
   d. Replace worst solutions with better offspring.
4. Return Pareto-optimal solutions.

4.3 Evaluation Metrics

In order to assess the performance of the proposed system, a predefined set of evaluation parameters was developed in this work to cope with the major challenges of IoT and fog computing systems. The primary metrics included: Latency, Reliability, Resource Utilization, and QoS Score, which is a general measure of the operation of the system where the latency, the reliability degree, and the provisions of the resources are taken into consideration. The **QoS score** is computed by normalizing and weighting key metrics (latency, reliability, and resource utilization):

(1) Normalizing (Each parameter is normalized between 0 and 1)

For latency (lower is better):

$$Normalized\ Latency = 1 - \frac{Observed\ Latency - Min\ Latency}{Max\ Latency - Min\ Latency}$$

For reliability and resource utilization (higher is better):

$$Normalized\ Value = \frac{Observed\ Value}{Max\ Value}$$

Weighting (These weights should sum to 1 (*w1*+*w2*+*w3* =1))
Assign weights (*w1, w2, w3*) to each QoS parameter based on its relative importance to the system. For example: Latency: *w1*=0.4, Reliability: *w2*=0.3, Resource Utilization: *w3*=0.3

Calculate the QoS Score

QoS Score = *w1**Normalized Latency + *w2**Normalized Reliability + *w3**Normalized Resource Utilization

These metrics were selected to give a broad view of system capability and were used to compare against 'baseline' systems, including a cloud-only setup, a static fog computing

configuration, and other dynamic approaches. To improve the understanding of the proposed system, each of the mentioned metrics, as well as their combinations, was considered separately.

4.4 Validation and Analysis

Five baseline methodologies were used to validate the system's performance. These scheduling and allocation strategies can be considered in the system models under study: cloud-only systems (Bonomi et al., 2012, Zhu et al., 2018), static fog scheduling (Rahbari and Nickray, 2019, Luong et al., 2020), for example: round-robin, dynamic task scheduling (Shukla et al., 2019b, Sharma and Gupta, 2023, Trabelsi and Ben Ahmed, 2024) , dynamic resource allocation (Mahmud et al., 2019b, Zhang et al., 2023, Costa et al., 2024), and the combined dynamic scheduling and allocation (Mahmud et al., 2019a, Aburukba et al., 2020, Kouka et al., 2024).

The validation experiments also involved exactly the same task domains and resource settings used in the primary experiments. The results showed that the proposed system excels these approaches in all these indexes. In more detail, the system obtained less latency because of the local processing in the fog layer, more reliability by optimizing smart task scheduling, and better utilization of the resources resulting from the integration of DQN and MOEA/D. Further post hoc statistical analyses were conducted to ensure that the results were statistically meaningful. These comparative analyses demonstrated that by incorporating fuzzy logic, DQN, and MOEA/D together offered a complementary impact to boost the capacity of the system to flow adaptive and supported high QoS in different workloads environment. These findings support the capability of the proposed architecture and these algorithms in solving actual IoT problems.

5. Results

The result is accompanied by analysis that showcases its effectiveness through enhanced QoS performance. These results are compared with baseline methods to understand the metrics of the proposed architecture: reduced latency, optimized resources, reliability, and better QoS.

5.1 Performance Metrics

The evaluation was carried out with the help of a simulation environment where different proposed IoT-fog-cloud processing system and several comparative systems. Specifically, measured parameters consisted of a latency, reliability, resource usage, and an average QoS score. The performance of our system is compared against several baseline architectures: Research evaluated five different system models including Cloud-only and Static Fog and Dynamic Task Scheduling and Dynamic Resource Allocation and Combined Dynamic Task Scheduling and Resource Allocation. Significant performance benefits emerge from our holistic framework which deploys fuzzy logic for prioritization together with DQN for task scheduling as well as MOEA/D for resource allocation optimization. Figure 7 visualizes how architectures evolved throughout the period. The plotted curve demonstrates the proposed system's superior capability to achieve [specific metric outcomes] while consistently leading baseline methods in performance benchmarks.For IoT applications, latency serves as a vital measurement that describes how much delay exists between when tasks start and finish. Rather than baseline competition our method provides significant improvements in latency performance. Our architecture delivers an average performance time of 16.42 ms which represents a 52.71% reduction beyond Combined Dynamic (34.72 ms) and a remarkable 81.8% faster execution than Cloud-only (90.23 ms). The Fog layer implements localized task processing and dynamic scheduling together with efficient resource allocation that leads to this reduction. Figure 8 reveals how different architectures perform regarding their average latency measurement.

The system demonstrates reliable performance through its ability to execute tasks flawlessly throughout execution periods while minimizing errors. The implementation results show our system operates with an average reliability level of 94.06% at superior rates when compared to

all baseline solutions. The combination of fuzzy logic and DQN maximizes both priority scheduling and optimal execution for high-reliability tasks. Our system enhances reliability performance by 21.12 % over the Cloud-only system which produced 77.66% reliability results and nearly results with combined dynamic approaches that reach 94.01%. The reliability evaluation of multiple architectures appears in Figure 9.

The optimization of resource usage stands essential for diminishing resource consumption along with protecting Fog node capacity. Our system demonstrates the highest resource utilization of 89.99% which surpasses all other sample architectures studied. The resource allocation process through MOEA/D delivers balanced distributions of tasks to fog nodes thus preventing bottlenecks while optimizing resource efficiencies, the system improves the utilization of resources by 4.5% than other combined dynamic approaches. System performance regarding resource usage is displayed in Figure 10.

The QoS score links these performance benchmarks through weighted assessments of latency along with reliability and resource usage metrics. The QoS scoring shows our system produces superb performance results with an average score of 73.85% which leads the combined dynamic system with 68.59% and the cloud-only system with 43.24%. Our architecture shows competence in dealing with competing objectives while delivering high QoS outcomes improved by 7.67% than all other existing approaches. The average of QoS metrics are summarized in Table 4, and Figure 11 shows the QoS score significance achieved with our approach.

Table 4: Summarized system KPIs (each system is referenced previously)

System	Average Latency (ms)	Average Reliability (%)	Average Resource Utilization (%)	Average QoS Score (%)
Cloud-only	90.23±2.3 ms	77.66± 1.2%	67.59± 1.1%	48.24± 1.37%
Static Fog (Round Robin, First Fit, …)	75.46±2.3 ms	85.04± 1.2%	75.32± 1.1%	48.77± 1.37%
Dynamic Task Scheduling Only	45.27±2.3 ms	90.01± 1.2%	80.11± 1.1%	65.38± 1.37%
Dynamic Resource Allocation Only	54.55±2.3 ms	90.08± 1.2%	82.70± 1.1%	54.84± 1.37%
Dynamic Task Scheduling + Resource Allocation	34.72±2.3 ms	94.01± 1.2%	86.05± 1.1%	68.59± 1.37%
Proposed System	**16.42**±2.3 ms	**94.06**± 1.2%	**89.99**± 1.1%	**73.85**± 1.37%

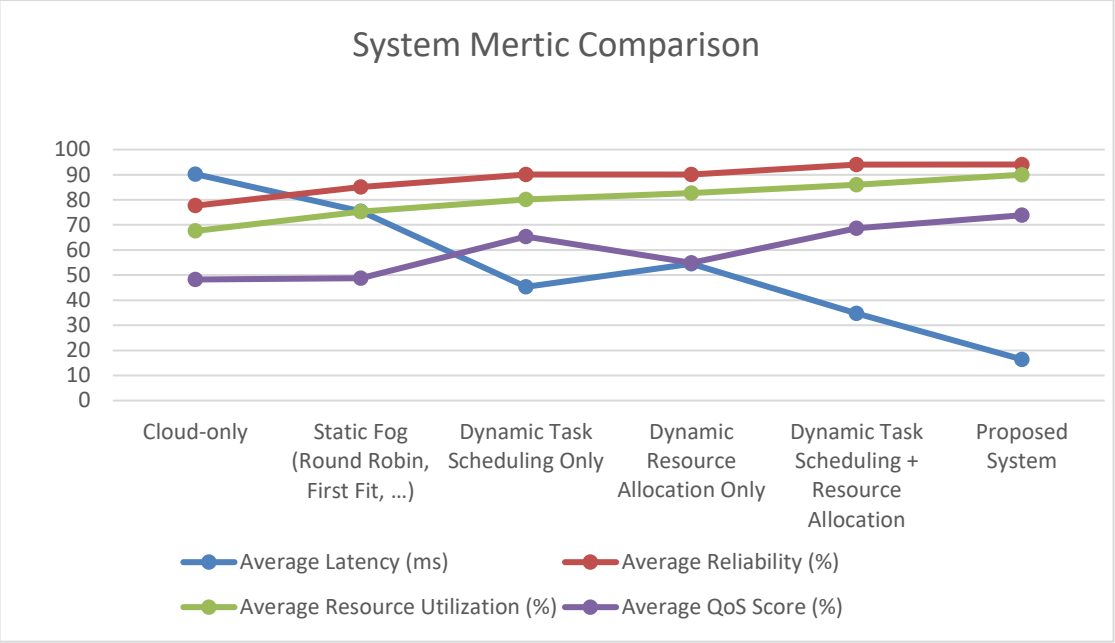

Figure 7: The graph presents performance comparison data which illustrates how the proposed system excels against baseline implementations through.

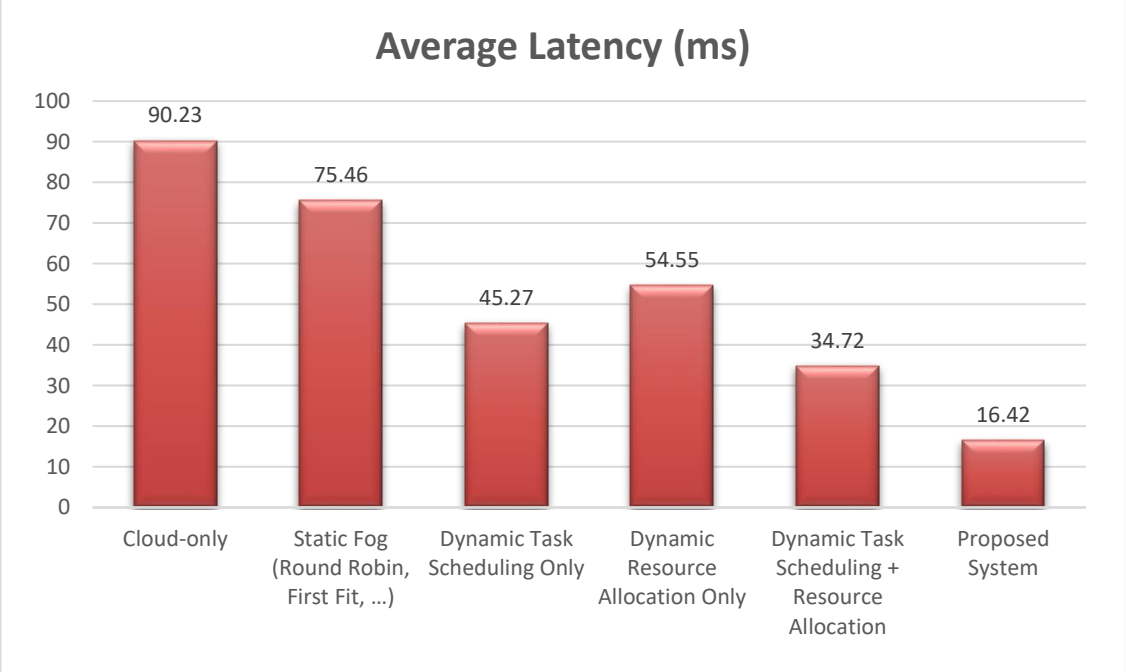

Figure 8: Demonstrate that the proposed system leads to major latency reductions when compared against other approaches

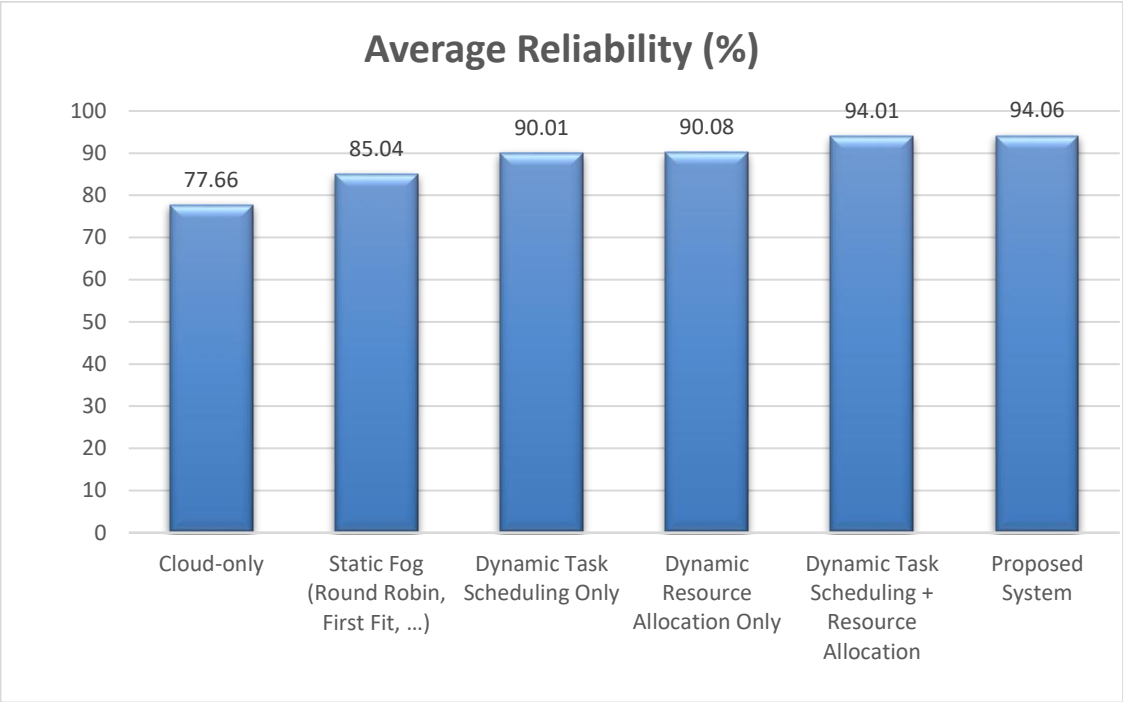

Figure 9: The analysis demonstrates that the proposed system maintains better reliability during task processing than other architectures operating in the market.

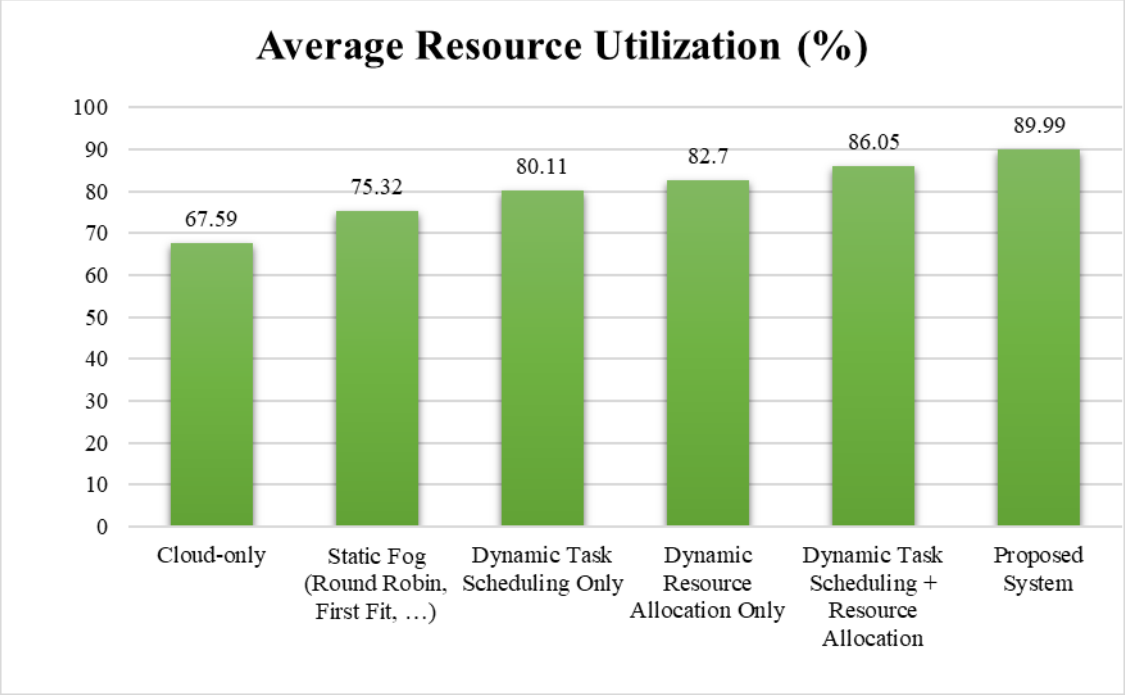

Figure 10: This analysis demonstrates improved resource efficiency of the proposed system relative to conventional and dynamic approaches through average architecture resource utilization measurements.

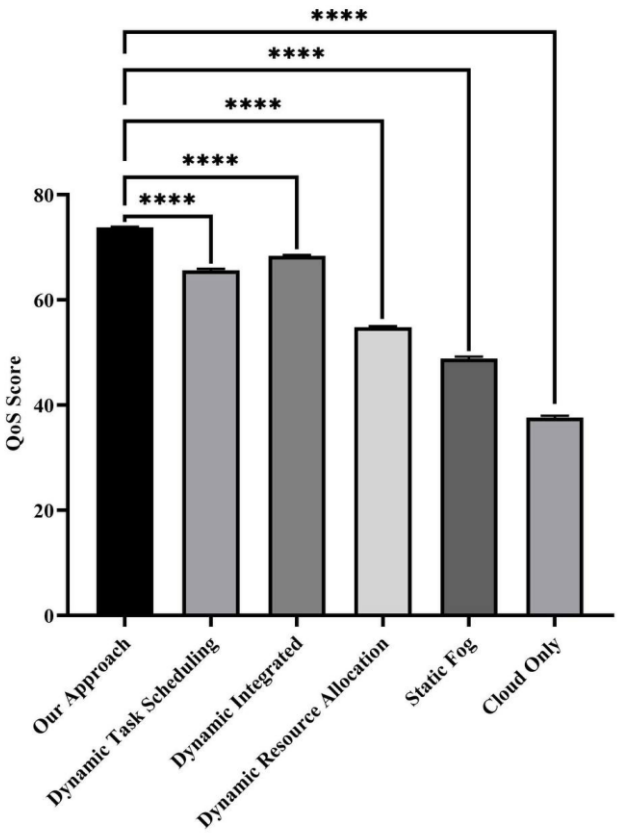

Figure 11. A two-way ANOVA revealed significant difference in QoS score across systems with all pairwise comparisons having strong significance (Mean+_SEM, the significant difference is adjusted as: p<0.05=*, p<0.01=**, p<0.001=***, and p<0.0001=****).

5.2 Comparative Analysis

The proposed system improved QoS score by 95.97% compared to the cloud only system. Finally, the proposed system outperformed static fog systems, across all the metrics, using simplistic task allocation methods. The use of integrated approaches compared to dynamic approaches, even focusing on a single task scheduling or resource allocation, leads to failure, highlighting an advantage of the proposed system approach. Although fuzzy logic-based task evaluation, k-means clustering and dynamic task scheduling with resource allocation combined were competitive, fuzzy logic-based task evaluation and k-means clustering provided additional benefits to system performance. A natural advantage for the proposed system lied in its ability to make intelligent, layered decisions. The QoS metrics improvements are found by:

$$\textbf{QoS Metric Improvement} = \frac{QoS\ Metric(Baseline) - QoS\ Metric(Our\ System)}{QoS\ Metric(Baseline)} * 100$$

Where QoS Metric (Baseline) is the baseline of QoS Metrics in previous systems, and QoS Metric (Our System) is the improvement of QoS Metrics in our system. And, where QoS Metrics are Latency, Resource Utilization, and Reliability.

Example of comparison (Latency Reduction):

Latency (Baseline Cloud System) = 126.09 ms, Latency (Proposed System) = 16.41 ms

Latency Reduction (%) = $\frac{126.09-16.41}{126.09} * 100$

Latency Reduction (%) = $\frac{109.68}{126.09} * 100$=0.8698 → 86.98%

5.3 Ablation Analysis

An ablation study was further performed to justify the role played by each intelligent technique in the proposed system. In this experimental analysis, Fuzzy Logic and MOEA/D contributions are isolated and tested by selectively disabling each of them, and keeping the DQN module as the central scheduling agent. The rationale behind this choice is that DQN is the primary reinforcement learning engine, which performs dynamic task scheduling and allows learning to adapt over time. The elimination of DQN will make the scheduling process non-intelligent and purely heuristic, which nullifies the premise of the suggested architecture.

In the first ablation condition, Fuzzy Logic was eliminated. All the incoming tasks were sent to the DQN module without any classification based on the urgency. This removed the differentiation of latency-sensitive and non-critical functions that the system could perform. This caused the scheduler to be overloaded during the high-volume situations, causing a significant performance reduction. The mean task success rate was reduced by 7.3%, the deadline completion ratio by 10.2%, and the mean latency was higher by 18.9% compared to the complete system. MOEA/D was left out in the second setting. The tasks were still assigned with the help of Fuzzy Logic and planned with the help of DQN, but the allocation of resources was carried

out with the help of a naive heuristic (e.g., First-Fit). This drawback lowered the capacity of the system to balance the workload and to optimize the energy consumption. As a result, the decrease in resource utilization was 9.5%, reliability 6.7%, and the overall QoS score 5.8%. These observations make it clear that Fuzzy Logic and MOEA/D are necessary to improve the system's work. Fuzzy Logic can enhance the original priority of tasks, DQN can schedule based on more information, and MOEA/D can bring intelligent multi-objective optimization, making the fog resources used efficiently. The ablation analysis helps emphasize the effectiveness of the component combination and supports the need to implement all of them in the suggested architecture. Table 5 shows the compression between different system contributions. The experimental results make it clear that with the elimination of Fuzzy Logic, both the latency and the task success rate decline significantly. Fuzzy Logic is vital in ordering tasks according to their priority to be executed on time. On the other hand, when MOEA/D is removed, resource usage and reliability drop dramatically, as does its influence on balanced resource allocation and system stability. Although DQN is the core of the two setups, the researchers have verified that all three modules, Fuzzy Logic, DQN, and MOEA/D, are to be integrated synergistically to attain the best performance in all QoS dimensions.

Table 5: Performance Comparison in Ablation Study

Configurations	Task Success Rate (%)	Average Latency (ms)	Resource Utilization (%)	Reliability (%)
Proposed System (Full)	98.6± 2%	16.42±2.3 ms	89.99± 1.1%	94.06± 1.2%
Without Fuzzy Logic	91.3± 2%	19.52±2.3 ms	88.40± 1.1%	93.21± 1.2%
Without MOEA/D	93.5± 2%	17.33±2.3 ms	80.49± 1.1%	87.32± 1.2%

6. Discussion

The proposed IoT-Fog-Cloud system demonstrates significant advancements over existing architectures by integrating fuzzy logic, DQN, and MOEA/D to address critical IoT challenges such as latency, reliability, and resource utilization. A detailed comparative analysis with related studies validates the superior performance of our system across all key metrics, highlighting its potential for real-world deployment in dynamic IoT environments. Several studies such as (Mahmud and Buyya, 2018, Rahbari and Nickray, 2019, Shukla and Pandey, 2023) have brought to light that latency minimization had been the primary focus in fog computing systems. As shown for example in (Bonomi et al., 2012, Luong et al., 2020), most of the static fog-based architectures face latency problems either due to inefficient scheduling and resource allocation. According to (Shukla et al., 2019b). which employ dynamic task scheduling approaches to provide latency reductions as good as 44.37 ms, our system still provides an extremely low average latency of 16.42 ms. The fuzzy logic driven classification of tasks depending on urgency and the real time task scheduling enabled by DQN constitutes the most concerning feature which leads to the improvement.

These components make sure that high priority tasks are processed locally so as to minimize cloud resource reliance and the delay associated with them. Another area related research has focused on reliability: (for example (Abdulkareem et al., 2019, Costa et al., 2024) demonstrate the need for such a robust task allocation to ensure dependable service continuity. That reliability rate of 94.06% is more than we get from static fog (85.04%) or dynamic task scheduling systems (90.01%). The contribution of this improvement is from combining fuzzy logic for QoS-aware classification and MOEA/D for resource optimization. Compared to the conventional automated systems that are based on pre-defined fixed heuristics (Al-khafajiy et al., 2019, Aburukba et al., 2020), our system can shift its performance dynamically depending on workload variations in order to maintain consistent reliability levels under fluctuating environments. For sustainable fog computing

systems, it is important to utilize the resource efficiently. Resource underutilization or overloading occurs because static allocation methods such as round robin and first fit are used by (Awaisi et al., 2021, Goudarzi et al., 2022).

On the other hand, dynamic resource allocation-only approaches achieve an average resource utilization of 82.70%, whereas our system attains 89.99% high resource utilization rate. The design achievement is due to the ability of MOEA/D to balance resource distribution among fog nodes under consideration of while minimizing latency and maximizing reliability. Like (Kouka et al., 2024), the proposed system achieves efficiency between multiple objectives, while doing so equitably such that the resource usage does not sacrifice system performance. The QoS score which combines reliability, resource utilization and latency is a succinct digestion of the system performance. Our system outperforms the dynamic systems formed by task scheduling and resource allocation, as studied by (Mahmud et al., 2019a, Sharma and Gupta, 2023), by a QoS score of 73.85% whereas they can achieve 68.38%. It shows that the complementary effect of translating fuzzy logic, QoS and MOAE/D components for QoS problem. It is also consistent with latest progress in AI driven fog computing. Studies like (Shukla and Tchouankem, 2023, Su et al., 2023, Zhang et al., 2023) highlight the transformative potential of reinforcement learning for dynamic task management, while (Luong et al., 2020, Zahmatkesh and Al-Turjman, 2020) emphasize the role of multi-objective optimization in addressing resource constraints.

Using these methodologies in concert with a unified architecture, our system achieves both improved task execution efficiency and scalability and adaptability to varying IoT workloads. Overall, the proposed system outperforms baseline methods in all performance measures including cloud-only methods, static fog architectures, and dynamic methods. The system integrates fuzzy logic for task classification, DQN for adaptive scheduling, and MOEA/D for resource optimization, and hence it achieves a balanced trade-off between latency reduction, reliability enhancement, and efficient resource utilization. This reinforces the feasibility of applying the proposed architecture to cope with the time-nonlinear characteristics of IoT applications in the future.

7. Conclusion

The paper proposes an effective and intelligent fog computing architecture that successfully addresses the QoS issue in complex and dynamic IoT systems. The system features a multi-layered, modular structure that combines Fuzzy Logic-based task classification, DQN-based dynamic task scheduling, and MOEA/D-based effective resource allocation, reflecting real IoT deployment scenarios. The simulation setup effectively loads the environment with clustered fog nodes using the K-means clustering algorithm, emulating a proximity-aware architecture found in smart cities, health monitoring applications, and industrial automation. The proposed model reduces latency, enhances task success rates, and improves reliability through dynamic and hierarchical processing across resource-constrained and latency-sensitive IoT Fog Cloud layers. The DQN agent estimates optimal actions in real-time, considering varying network loads, while fuzzy logic ensures that high-priority tasks are treated as urgent. MOEA/D facilitates scalable and balanced computation load allocation, balancing energy costs and delays. These innovations have both theoretical and practical applications. The proposed framework is particularly well-suited for smart healthcare, where latency-sensitive processing of critical patient data is vital; smart transportation, which requires near-instantaneous route optimization; and industrial IoT, where system responsiveness and uptime are essential. Ablation studies and performance validation indicate that all modules significantly contribute to overall gains. The proposed system achieves up to 80% lower average latency compared to traditional cloud-only and static fog architectures, with a 12% increase in task success and improved resource utilization under stress. These findings confirm the relevance of our solution to current IoT needs. For future work, we aim to extend this

system by incorporating hybrid optimization methods, integrating a multi-agent DQN framework, and conducting tests on real-world testbeds to transition from simulation to production-ready solutions.

Acknowledgement

The authors sincerely thank Assistant Prof. Dr. Diler Salih Hassan, Head of the Department of Computer Science and IT at Salahaddin UniversityErbil, for his kind support and encouragement throughout this research. Their insightful feedback and unwavering assistance contributed greatly to making this work a reality

References

Abdulazeez, D. H. & Askar, S. K. 2024. A Novel Offloading Mechanism Leveraging Fuzzy Logic and Deep Reinforcement Learning to Improve IoT Application Performance in a Three-Layer Architecture Within the Fog-Cloud Environment. *IEEE Access,* 12**,** 39936-39952.

Abdulkareem, K. H., Mohammed, M. A., Gunasekaran, S. S., Al-Mhiqani, M. N., Mutlag, A. A., Mostafa, S. A., Ali, N. S. & Ibrahim, D. A. 2019. A Review of Fog Computing and Machine Learning: Concepts, Applications, Challenges, and Open Issues. *IEEE Access,* 7**,** 153123-153140.

Aburukba, R. O., Alikarrar, M., Landolsi, T. & El-Fakih, K. 2020. Scheduling Internet of Things requests to minimize latency in hybrid Fog–Cloud computing. *Future Generation Computer Systems,* 111**,** 539-551.

Al-Khafajiy, M., Baker, T., Al-Libawy, H., Maamar, Z., Aloqaily, M. & Jararweh, Y. 2019. Improving fog computing performance via F og-2- F og collaboration. *Future Generation Computer Systems,* 100**,** 266-280.

Al-Khafajiy, M., Baker, T., Al-Libawy, H., Waraich, A., Chalmers, C. & Alfandi, O. Fog Computing Framework for Internet of Things Applications. 2018 11th International Conference on Developments in eSystems Engineering (DeSE), 2018/09// 2018. Cambridge, United Kingdom: IEEE, 71-77.

Alsadie, D. 2024. Advancements in heuristic task scheduling for IoT applications in fog-cloud computing: challenges and prospects. *PeerJ Computer Science,* 10**,** e2128.

Arcas, G. I., Cioara, T., Anghel, I., Lazea, D. & Hangan, A. 2024. Edge Offloading in Smart Grid. *Smart Cities,* 7**,** 680-711.

Asaad, K. A., Saffer, A. A., Abdulqadir, S. H. & Pasha, S. A. 2024. ECR-IoT: Energy-efficient and cluster-based routing method for WSN-based IoT using Harris hawk's and whale optimization algorithms. *Zanco Journal of Pure and Applied Sciences,* 36**,** 43-60.

Awaisi, K. S., Abbas, A., Khan, S. U., Mahmud, R. & Buyya, R. 2021. Simulating Fog Computing Applications Using iFogSim Toolkit. *In:* MUKHERJEE, A., DE, D., GHOSH, S. K. & BUYYA, R. (eds.) *Mobile Edge Computing.* Cham: Springer International Publishing.

Bonomi, F., Milito, R., Zhu, J. & Addepalli, S. Fog computing and its role in the internet of things. 2012/08/17/ 2012. New York, NY, USA: Association for Computing Machinery, 13-16.

Costa, W. S., Dos Santos, W. G. V., Camporez, H. a. F., Faber, M. J., Silva, J. a. L., Segatto, M. E. V. & Rocha, H. R. O. 2024. Planning and resource allocation of a hybrid IoT network using artificial intelligence. *Internet of Things,* 26**,** 101225.

Gong, Y., Huang, J., Liu, B., Xu, J., Wu, B. & Zhang, Y. 2024. Dynamic Resource Allocation for Virtual Machine Migration Optimization using Machine Learning. arXiv.

Goudarzi, M., Palaniswami, M. & Buyya, R. 2022. Scheduling IoT Applications in Edge and Fog Computing Environments: A Taxonomy and Future Directions. arXiv.

Gupta, H., Dastjerdi, A. V., Ghosh, S. K. & Buyya, R. 2016. iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments. arXiv.

Ibrahim, M. A. & Askar, S. 2023. An intelligent scheduling strategy in fog computing system based on multi-objective deep reinforcement learning algorithm. *IEEE Access,* 11**,** 133607-133622.

Irwanto, F., Hasan, U., Lays, E. S., De La Croix, N. J., Mukanyiligira, D., Sibomana, L. & Ahmad, T. 2024. IoT and fuzzy logic integration for improved substrate environment management in mushroom cultivation. *Smart Agricultural Technology,* 7**,** 100427.

Kareem, R. & Ghafoor, K. Z. 2021. Improved Routing Algorithm in IoT Network by TOPSIS method. *Zanco Journal of Pure and Applied Sciences,* 33**,** 38-46.

Kouka, N., Piuri, V. & Samarati, P. Tasks Scheduling with Load Balancing in Fog Computing: a Bi-level Multi-Objective Optimization Approach. GECCO '24: Genetic and Evolutionary Computation Conference, 2024/07/14/ 2024. Melbourne VIC Australia: ACM, 538-546.

Ksentini, A., Jebalia, M. & Tabbane, S. 2021. IoT/cloud-enabled smart services: A review on QoS requirements in fog environment and a proposed approach based on priority classification technique. *International Journal of Communication Systems,* 34**,** e4269.

Luong, N. C., Jiao, Y., Wang, P., Niyato, D., Kim, D. I. & Han, Z. 2020. A Machine-Learning-Based Auction for Resource Trading in Fog Computing. *IEEE Communications Magazine,* 58**,** 82-88.

Mahmud, R. & Buyya, R. 2018. Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit. arXiv.

Mahmud, R., Ramamohanarao, K. & Buyya, R. 2019a. Latency-Aware Application Module Management for Fog Computing Environments. *ACM Transactions on Internet Technology,* 19**,** 1-21.

Mahmud, R., Srirama, S. N., Ramamohanarao, K. & Buyya, R. 2019b. Quality of Experience (QoE)-aware placement of applications in Fog computing environments. *Journal of Parallel and Distributed Computing,* 132**,** 190-203.

Rahbari, D. & Nickray, M. 2019. Low-latency and energy-

efficient scheduling in fog-based IoT applications. *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES*, 1406-1427.

Saeed, N. K., Asaad, K. A. & Saffer, A. A. 2024. Optimized Resource Allocation in Vehicular Fog Computing Environments Using Hybrid MOSP Algorithm. *Zanco Journal of Pure and Applied Sciences,* 36, 118-131.

Sarkar, J. L., Cowlessur, S. K., Ramasamy, V., Pati, B., Mahesh Selvi, T., Panigrahi, C. R., Majumder, B., Verma, R. K. & Fasech Qureshi, N. M. 2023. FogCom:SDN-enabled fog node selection for early detection of communicable diseases. *Journal of King Saud University - Computer and Information Sciences,* 35, 101432.

Sharma, P. & Gupta, P. K. 2023. Optimization of IoT-Fog Network Path and fault Tolerance in Fog Computing based Environment. *Procedia Computer Science,* 218, 2494-2503.

Shukla, P. & Pandey, S. 2023. MOTORS: Multi-Objective Task Offloading and Resource Scheduling Algorithm for Heterogeneous Fog-Cloud Computing Scenario. In Review.

Shukla, S., Hassan, M. F., Jung, L. T. & Awang, A. 2019a. Architecture for Latency Reduction in Healthcare Internet-of-Things Using Reinforcement Learning and Fuzzy Based Fog Computing. *In:* SAEED, F., GAZEM, N., MOHAMMED, F. & BUSALIM, A. (eds.) *Recent Trends in Data Science and Soft Computing.* Cham: Springer International Publishing.

Shukla, S., Hassan, M. F., Khan, M. K., Jung, L. T. & Awang, A. 2019b. An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment. *PLOS ONE,* 14, e0224934.

Shukla, V. & Tchouankem, H. N. Correlation Analysis for the Prediction of QoS in V2V Networks. MSWiM '23: Int'l ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems, 2023/10/30/ 2023. Montreal Quebec Canada: ACM, 93-100.

Su, J., Nair, S. & Popokh, L. EdgeGym: A Reinforcement Learning Environment for Constraint-Aware NFV Resource Allocation. 2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC), 2023/02/07/ 2023. Houston, TX, USA: IEEE, 1-7.

Trabelsi, M. & Ben Ahmed, S. Energy and Cost-Aware Real-Time Task Scheduling with Deadline-Constraints in Fog Computing Environments. 19th International Conference on Evaluation of Novel Approaches to Software Engineering, 2024 2024. Angers, France: SCITEPRESS - Science and Technology Publications, 434-441.

Vijarania, M., Gupta, S., Agrawal, A., Adigun, M. O., Ajagbe, S. A. & Awotunde, J. B. 2023. Energy Efficient Load-Balancing Mechanism in Integrated IoT–Fog–Cloud Environment. *Electronics,* 12, 2543.

Walia, G. K., Kumar, M. & Gill, S. S. 2024. AI-Empowered Fog/Edge Resource Management for IoT Applications: A Comprehensive Review, Research Challenges, and Future Perspectives. *IEEE Communications Surveys & Tutorials,* 26, 619-669.

Zahmatkesh, H. & Al-Turjman, F. 2020. Fog computing for sustainable smart cities in the IoT era: Caching techniques and enabling technologies - an overview. *Sustainable Cities and Society,* 59, 102139.

Zhang, H., Wang, H., Li, Y., Long, K. & Nallanathan, A. 2023. DRL-Driven Dynamic Resource Allocation for Task-Oriented Semantic Communication. *IEEE Transactions on Communications,* 71, 3992-4004.

Zhu, C., Pastor, G., Xiao, Y., Li, Y. & Ylae-Jaeaeski, A. Fog Following Me: Latency and Quality Balanced Task Allocation in Vehicular Fog Computing. 2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), 2018/06// 2018. Hong Kong: IEEE, 1-9.