International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2025/22 Volume 12, Issue 3, May – June (2025)

RESEARCH ARTICLE

Optimizing Fog Computing Performance Using Q-Learning: A Simulation-Based Study

Nada M. Sallam
Faculty of Computer Studies, Arab Open University, Riyadh, Saudi Arabia.
✉ n.sallam@arabou.edu.sa

Samah Osman
Faculty of Computer Studies, Arab Open University, Riyadh, Saudi Arabia.
s.osman@arabou.edu.sa

Received: 02 March 2025 / Revised: 13 May 2025 / Accepted: 19 June 2025 / Published: 30 June 2025

Abstract – Fog computing complements cloud computing by locating computational resources near end-users, sharply lowering latency and enhancing data processing efficacy for mission-critical applications. With the rapid proliferation of Internet of Things (IoT) devices and data generated from them, it has become crucial for efficient and dynamic management of resources in fog environments. Classic resource provisioning approaches tend to depend on fixed algorithms or heuristics, which are weak in coping with the extremely dynamic and heterogeneous nature of contemporary fog networks based on changing workloads, user mobility, and diverse network conditions. We propose and test a Q-learning-based optimization approach in an endeavor to improve performance in a simulated fog computing network. OMNeT++ is used for simulation, a sophisticated network simulation platform. The suggested solution employs a hierarchical fog network topology composed of master and slave fog nodes that collaborate to carry out user requests and allocate resources. Employing Q-learning, an algorithm for reinforcement leaning, the system learns in terms of real-time network status, allowing it to dynamically maximize resource allocation and task scheduling choices to enhance main performance metrics. The performance of the system, that is, throughput, response time, and drop rate, is assessed for different user loading scenarios (5, 10, 15, 20, and 21 users). The experiment is conducted in two different configurations: one being the baseline without the suggested Q-learning mechanism ("without Q-learning") and another with the Q-learning mechanism enabled ("with Q-learning"). The baseline case has deteriorated performance metrics as the user base grows, reflected in higher response times, reduced throughput, and increased drop rate, which indicate congestion and poor resource management. Conversely, the situation employing Q-learning shows drastic improvements in all the metrics tested. Outcomes indicate a higher throughput, an enormous reduction in response time, and a lower rate of drops, particularly under heavy user loads. The Q-learning algorithm efficiently alleviates network congestion and optimizes resource allocation by learning to choose optimal actions from a dynamically updated reward system based on network state information. The simulation setup details, such as particular configurations of mobility patterns (circular and linear), radio wave propagation, network elements (compute brokers, base brokers, routers, access points), and MQTT app parameters, are introduced. The state space, action space, and reward function of the reinforcement learning model are specified, including Chapman probability to potentially serve to improve state transition modeling and decision-making precision within dynamic contexts. The findings show that the Q-learning-based approach significantly surpasses the conventional approach (embodied by the baseline) in sustaining network effectiveness, controlling congestion, and maximizing resource allocation under different loads and mobility. This research confirms the viability of reinforcement learning methods for resource optimization in complex fog computing environments, with a view to future investigation of more sophisticated machine learning approaches. Subsequent work would involve investigating hybrid learning models, testbed implementation in real-world environments, and thorough scalability analysis to enhance system responsiveness and resilience further.

Index Terms – Fog Computing, Q-Learning, OMNeT++, Reinforcement Learning, Resource Management, Throughput, Latency, Task Scheduling, IoT.

1. INTRODUCTION

Fog computing, as an extension of cloud computing, is placed strategically nearer to end-users and Internet of Things (IoT) devices. This allows for the delivery of computational, storage, and networking facilities at the network edge, largely minimizing the use of remote cloud data centers [1, 2]. The main aim of fog computing is to reduce latency, contain bandwidth usage in the core network, and improve the efficiency of time-constrained and spatially dispersed applications, especially those generated due to the widespread proliferation of IoT devices.[3, 4, 5]

The fast pace of development and ubiquitous deployment of IoT devices across diverse fields, such as smart cities, industrial automation, healthcare, and transportation, have

RESEARCH ARTICLE

resulted in an enormous amount and speed of data generation. Conventional centralized models of cloud computing usually struggle to cope with the low-latency feedback and high-bandwidth processing requirements of such applications [6] Fog computing resolves this by providing decentralized data processing and resource-sensitive utilization features in the middle area between edge devices and the cloud [7, 8]. A key problem in developing effective fog computing systems is the efficient handling and allocation of heterogeneous resources across multiple kinds of user requests and dynamic workloads. [9, 10]

It is further complicated by issues such as device mobility, network heterogeneity, diverse application requirements, and ensuring Quality of Service (QoS) [2, 11]. Traditional resource allocation methods, which are based on static algorithms or naive heuristics, are often not capable of adapting very well to the ever-changing conditions within a distributed fog network [4**,** 12].

This static paradigm may incur suboptimal performance in terms of throughput, response time, resource usage, and energy efficiency, particularly under high load or dynamic workloads [13]. To break through the shortcomings of static methods, smart and adaptive solutions must be employed [6**,** 14]. Reinforcement learning (RL), more specifically Q-learning algorithms, provides a viable paradigm towards realizing adaptive resource control and smart task scheduling in dynamic fog networks [15]

RL agents can acquire optimal decision-making policies by interacting with the environment, adjusting their policy based on measurable performance indicators and rewards. Such capability renders Q-learning an ideal solution to optimize resource allocation and task offloading choices in real-time uncertain fog environments [8**,** 16]. This study is driven by the timely need for intelligent and adaptive resource control mechanisms in fog computing networks in order to accommodate modern IoT applications' rising complexity and magnitude.

The main aim of this study is to deploy and test the efficacy of a Q-learning based dynamic resource management technique in a simulated fog computing network developed with the help of the OMNeT++ network simulator. Particularly, we seek to show how a reinforcement learning solution can improve on the throughput, response time, and drop rate of key performance indicators over a baseline with no such adaptive smarts. [17]

The architecture of the simulated system in this paper is hierarchical with master and slave fog nodes. Slave nodes are responsible for performing local data processing operations, whereas master nodes perform more sophisticated requests, task allocation, and coordination within their clusters. We also investigate the integration of Chapman probability functions to possibly increase the learning efficiency and decision accuracy of the Q-learning agent in dynamic networks [9]. The main contributions of the paper are as follows:

- Execution and evaluation of a Q-learning policy-based dynamic resource management scheme in an extensive simulation environment for fog computing built with the help of OMNeT++.
- In master-slave hierarchical architecture design and simulation of fog nodes are customized to manage scalable and resource-conscious workloads.
- Integration and initial experimentation of Chapman probability functions to maybe enhance the Q-learning agent's learning performance and decision accuracy in resource allocation problems.
- Extensive experimentation and analysis of system performance under different user loads (5, 10, 15, 20, 21 users) and mobility patterns, determining the effect of the Q-learning method on the main performance measures such as throughput, response time, and drop rate in comparison to a baseline non-adaptive one.

The rest of this paper has the following arrangement: Section 2 explains the methodology, the simulation configuration, network model, and Q-learning algorithm implementation. Section 3 discusses the results from the simulations. Section 4 offers an extensive discussion of the results, explaining the outcomes and comparing the system performance with and without the application of Q-learning. Section 5 discusses the limitations of the study. Section 6 outlines future work directions. Section 7 concludes the paper by outlining the primary findings and implications.

2. METHODOLOGY

This work applies to a simulation-based approach to comprehensively assess the performance of a Q-learning-based resource optimization approach in dynamic fog computing environments. The simulation is conducted based on the OMNeT++ network simulator, with configurations established for scenarios simulating realistic user mobility, varying application workloads, and hierarchical fog network architecture. [11]

2.1. FogNet Computing Model and Simulation Setup

The fog computing system, which is simulated, consists of two distinct clusters with a total area of 1500m × 1500m. One of the salient aspects of the proposed model lies in the introduction of a hierarchical structure of the nodes within the two clusters, which includes master and slave fog nodes. The slave nodes are primarily responsible for executing local jobs and processing local requests. Master nodes function in a coordination role, managing inter-node communication, mapping intricate tasks to nodes, and load balancing for the

RESEARCH ARTICLE

cluster. Hierarchical organization allows for efficient resource management and scalability. The simulation also incorporates a combination of static user totals (5, 10, 15, 20, and 21) to try out the performance of the system under different loads. Along with this, to imitate the dynamicity of edge environments, two kinds of user mobility models – linear and circular mobility – are employed. [18]

The novelty of the proposed method lies in the integration of the Q-learning algorithm, a model-free reinforcement learning approach, which aims to facilitate intelligent and adaptive decision-making for task offloading and resource allocation in the fog network [19]. In order to evaluate the effectiveness of this method, simulation and evaluation are performed under two scenarios:

Scenario 1: Without Q-learning (Conventional Setup):

This is a baseline since it's a more traditional or non-adaptive resource allocation approach where task allocation is based on pre-established rules without real-time learning.

Scenario 2: With Q-learning (Intelligent, Adaptive Setup):

In this scenario, the Q-learning algorithm is pretty much at play. The algorithm learns optimal actions from what it observes in the system (e.g., node load, delay, available CPU/bandwidth) and adapts its policy through rewards obtained. The reward system is designed such that it promotes the action which gives lesser response time, improved throughput, and reduced drop rates. [20]

Also, as presented in this study, applying the Chapman probability functions in the Q-learning algorithm is possible. This integration is aimed at potentially increasing the accuracy in predicting upcoming state changes, hence speeding up learning con-vergence and making more informed decisions, particularly in probabilistic and dynamic network environments [9].

The performance of the system in both scenarios is quantitatively determined by key parameters: throughput, response time, and drop rate [13]. The use of energy was considered but not fully investigated in this work because of scope or data available and thus will be addressed in future work. The simulated output is carefully examined to demonstrate the potential of employing reinforcement learning techniques in optimizing performance in fog computing systems [17]. Figure 1 illustrates the overall simulation process phases, conceptually similar to research workflows found in literature [7] and Table 1 provides a summary outlining the different phases of the fog computing network simulation study, including the tasks performed and their expected outcomes.

Table 1 Phases of Fog Computing Network Simulation

Phase	Task	Expected Outcome
Literature Review	Study existing literature on fog computing and Q-learning	Comprehensive understanding of current methods and identification of gaps in research
	Identify key performance metrics (throughput, response time, drop rate, energy efficiency, computational overhead)	Expanded list of relevant metrics for evaluating fog computing performance
Simulation initialization	Defining network topology with 2 clusters (1800m x 1800m) and multiple hierarchical levels	A comprehensive simulation network model that includes nodes, connection, and wireless properties
	Determine Master and Slave fog nodes with adaptive task offloading strategies	Functional fog computing environment with hierarchical structure and intelligent resource distribution
	Create user requests using random mobility models and different user numbers (5, 10, 15, 20, 21).	Realistic workload and mobility simulation for stress testing the fog network
Q-learning Implementation	Define state space, action space, and reward function, incorporating QoS constraints	Q-learning model tailored to fog computing with dynamic resource allocation strategies

RESEARCH ARTICLE

	Initialize Q-table and parameters (α, γ, ε) based on empirical studies and hyperparameter tuning	A refined reinforcement learning model prepared for simulation and adaptive training
	Use Q-learning with real-time learning updates in a simulation setting.	Adaptive resource management mechanism improving fogNet efficiency
Data Collection	Run simulations with varying traffic loads and static user counts (5, 10, 15, 20, 21).	Collected dataset including throughput, response time, drop rate, and power consumption
	Record performance metrics across various load conditions, including energy consumption	Comprehensive dataset for performance evaluation and scalability testing
Performance Evaluation	Analyze throughput, response time, drop rate, and energy efficiency	Quantitative assessment of Q-learning's impact on fog network optimization
	Plot and display performance indicators for various scenarios.	Results are shown graphically for better comprehension and presentation.
Comparison with Other Researchs	Obtain performance numbers from recent studies on optimizing fog computing.	Comparative data for benchmarking the Q-learning approach against existing techniques
	Compare Q-learning results with traditional methods (e.g., static, heuristic-based, deep learning approaches)	Insights into the strengths and potential limitations of Q-learning in fog networks
Analysis & Conclusion	Interpret results, discuss findings, and highlight practical implications	Comprehensive discussion on the feasibility and real-world applicability of Q-learning in fog computing
	Provide recommendations for future research, including hybrid learning models and real-world deployments	Future research directions to enhance fog computing optimization strategies

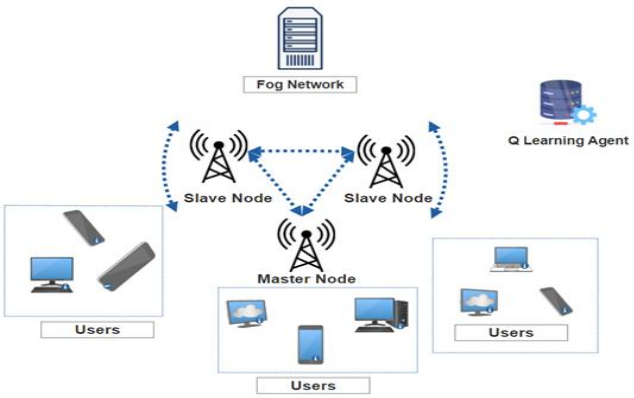

Figure 1 Phases of the Research Simulation Process

RESEARCH ARTICLE

Figure 1 illustrates the Key Phases Involved in the research study, from literature review and simulation initialization to data collection, performance evaluation, and analysis.

2.2. FogNet Simulation Parameters

FogNet simulator is a simulated environment set to emulate an identical wireless fog computing network with wireless communication and mobility of the users as the primary parameters. These parameters are selected to develop a testbed that is manageable yet representative in assessing the complexities of the fog computing system. The simulation is carried out with OMNeT++, which is a modular C++ discrete event network simulator well known for having a very high level of flexibility in modeling a wide range of network protocols and network structures [11]. The fundamental Q-learning parameters and simulation parameters are shown in Table 2.

Table 2 Simulation Parameters

Parameter	Value
Number of Users	5, 10, 15, 20, 21
Compute Broker	Standard Compute
Base Broker	Standard Compute
Router	IPv4 Network Configurator
Radio Medium	Ieee80211ScalarRadioMedium
Q-Learning Parameters	
Exploration Rate (ε)	0.1 (after initial decay, initially 1.0)
Learning Rate (α)	0.5
Discount Factor (γ)	0.9
Number of States	5
Number of Actions	3

Q-learning algorithm is employed in the simulation to support adaptive control of resources based on changing network conditions. Empirical values for the most important Q-learning parameters: ε (exploration rate), α (learning rate), and γ (discount factor) that were selected aim at a compromise between the necessity to explore new potential actions and exploiting the best known ones at a given moment [6, 21]. State space is calculated dynamically from real-time observation of decision-making-critical network conditions (e.g., queue lengths, node loading, possible delays). Action space is the limited number of resource management actions available for execution by the agent. The key goal of setting up and running the simulation using these parameters is to examine the performance of Q-learning in governing computing and communications resources in the fog setting. In particular, the simulation hopes to accomplish the following objectives:

- Compare the effect of various user mobility patterns (linear movement and circular movement) on key performance metrics like throughput, response time, and drop rate under the Q-learning environment.
- Examine system performance at different user loads ranging from 5 to 21 users to investigate the response and scalability of Q-learning-based resource management.
- Contrast system performance during execution using Q-learning versus not using Q-learning, focusing on the impact of the reinforcement learning technique in enhancing decision-making processes and overall network performance [13, 14].

2.3. Network Topology and Components

FogNet simulation environment consists of a set of essential components, and each of them has a different role in the entire system of the simulated fog computing environment. They provide a desired set of features bundled together in an attempt to simulate a good fog system with adaptive workload distribution capacity and mobile user service delivery capacity [11]. OMNeT++ has been utilized as the simulation environment [16], which is a C++-based component-based simulation environment for communications network simulation and analysis [22].

1. Compute-Broker

Compute-brokers are processing nodes of the fog layer. They are tasked with running computational tasks and controlling system resources. Compute-brokers in this simulation are strategically positioned to run computationally demanding tasks and process them. They are also tasked with making decisions about task offloading, especially in coordination with the Q-learning agent for dynamic resource management.

2. Base-Broker

Base-brokers are distribution and coordination points in the network. They are primarily data routing management and low-latency communication facilitation between end-user devices and compute-brokers. Base-brokers are an intermediate step, distributing the data load and facilitating low-latency, efficient traffic in their zones [23].

3. Wireless User Devices

These are the client devices, which are wirelessly linked to the fog infrastructure through access points. They are emulating real clients with varied traffic patterns and requiring varied services. Mobility of devices introduces network dynamic topology and load fluctuation, and this is a significant factor to prove the responsiveness and resilience of the system under various circumstances.

RESEARCH ARTICLE

4. Router

Routers are used in the center to forward data packets between different segments of the network. Routers in FogNet build end-to-end communication between user terminals, base-brokers, and compute nodes. IPv4 network configurators are used during simulation to route and manipulate addresses.

5. Access Point (AP)

Access points provide wireless access for user mobile devices to access the fog computing infrastructure. APs simulate physical wireless interfaces ubiquitously provided in enterprise or urban environments and provide end-users with low-latency mobility and service access [24].

2.3.1. FogNet Simulation Scenario

As conceptually illustrated in Figure 2 and Figure 3, FogNet is a systematic fog computing framework made up of wireless clients, access points, routers, compute-brokers, and base-brokers. They are all interconnected to simulate real fog configurations with distributed processing and decision-making [25]. Simulation is specifically designed to simulate the dynamic of user mobility, task allocation, and impact of dynamic resource allocation policies with various traffic and mobility patterns [26].

We opted for OMNeT++ in this case, as it is very extensible and can be applied to simulate tailored network protocols. OMNeT++ allows for a good implementation of network layer protocols and straightforward integration of, for example, machine learning algorithms like Q-learning to analyze their impact on the performance of fog computing in a reproducible and controlled environment [11].

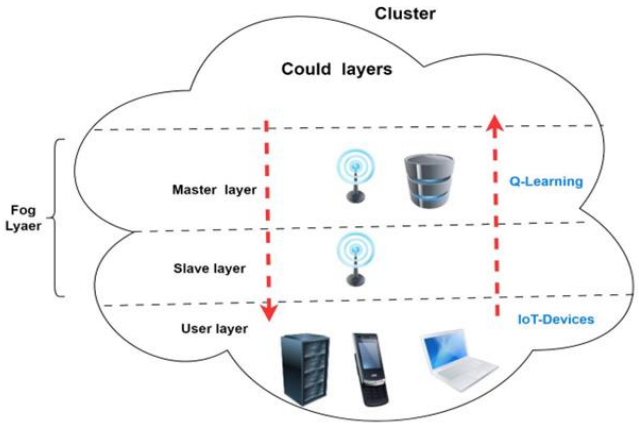

Figure 2 A Conceptual Diagram

Figure 2 Shows the main components of the FogNet simulation environment including wireless users, access points, routers, and fog nodes (compute/base brokers).

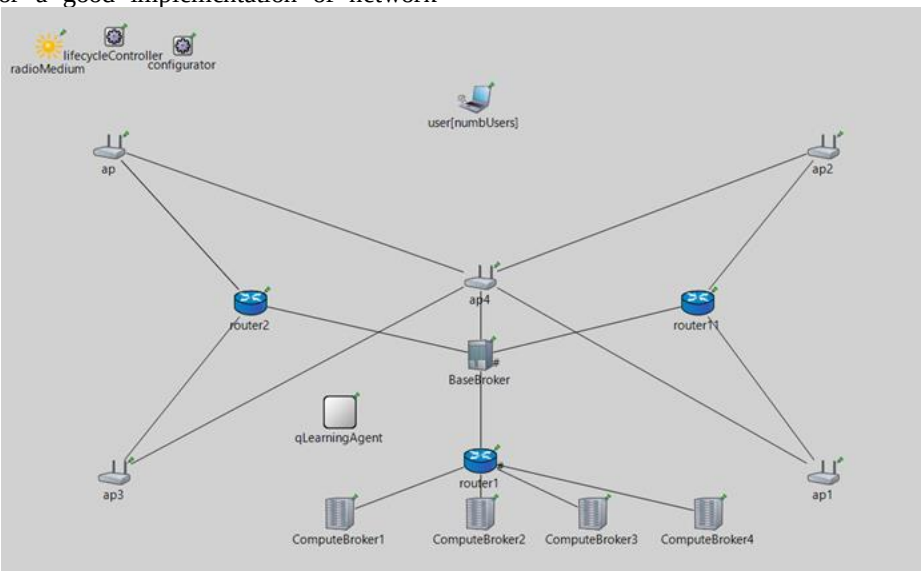

Figure 3 Illustrative Layout of a FogNet Simulation Scenario Within the Defined Cluster Area, Highlighting the Interconnection and Placement of Key Components Like Wireless Clients, Access Points, Routers, Compute Brokers, and Base Brokers

2.4. Mobility and Radio Configurations

To obtain best simulate realism and other dynamic aspects of actual fog comput-ing environments, simulation parameters are blended alongside tailored settings of user mobility, radio transmission, WLAN configuration, queuing behavior, and application-layer traffic. They are emulated using the OMNeT++ simulator to simulate dynamic wireless user behavior and intercommunication between different fog com-

RESEARCH ARTICLE

puting nodes as realistically as possible. The parameters are set for a scalable and realistic model to predict the behavior of the network in different and hostile environments.

2.4.1. Mobility Models

Mobility is the most ubiquitous attribute in fog computing networks where end devices are primarily mobile, affecting network load, required resources, and connectivity. The two specific mobility models are used in different mobility patterns:

1. Circular Mobility

This model encourage user to move in a circle with certain parameters: center location (cx = 400m, cy = 400m), radius r = 350m, speed = 50 m/s, and initial direction = 180°. This model simulates constant travel in a region, e.g., what happens with city hotspots or UAV patrol [27].

2. Linear Mobility

The linear model emulates linear motion, typical of a user moving down a road or corridor. It is called at 40 m/s in velocity, 0° in direction, zero acceleration, and at an update time of 80 ms [24]. Such mobility models are used to offer experimental control with varied levels of user dynamism and are essential to the verification of mobility's impact on performance criteria such as throughput, delay, and packet loss ratio [26].

2.4.2. Radio Transmission Setup

Transmitter Power: 2 mW

The chosen transmission power is consistent with typical real-world settings for wireless communication in short to medium range. This ensures sufficient signal strength for localized communication within fog node coverage areas, which is essential for maintaining low-latency services and conserving energy in fog networks [10].

2.4.3. WLAN Configuration

The WLAN settings are tuned based on the IEEE 802.11g standard to represent realistic Wi-Fi conditions commonly encountered in fog network deployments. Key settings include:

- Operating Mode (OpMode): "g" (IEEE 802.11g)
- Bitrate: 50 Mbps * Minimum Bitrate: 10 Mbps
- Enhanced Distributed Channel Access (EDCA): True
- Frame Capacity: 20
- Maximum Queue Size: 14
- RTS Threshold: 3000 Bytes
- Retry Limit: 7
- CW Minimum (Data): 30
- CW Minimum (Broadcast): 30

These parameters emulate a realistic wireless communication system supporting basic Quality of Service (QoS) capabilities, such as prioritized traffic handling through EDCA. EDCA's ability to provide differentiated access to the wireless medium is important for fog networks supporting heterogeneous traffic classes with varying delay sensitivities [28].

2.4.4. Queue Settings

Efficient queue management is simulated to model network load and congestion behavior accurately:

- Queue Type: Drop Tail Queue
- Frame Capacity: 40
- Routing Table Forwarding: true

The drop Tail queuing is a simple method in which incoming packets are dropped when the queue reaches its maximum capacity. This behavior realistically replicates conditions under heavy network traffic load. Simulating this functionality helps in determining how fog nodes handle overload situations and maintain service quality when resources become limited [23, 29].

2.4.5. Message Queuing Telemetry Transport (MQTT) Applications

The Message Queuing Telemetry Transport (MQTT) protocol is utilized in the simulation to model lightweight messaging between IoT devices and fog nodes. MQTT's publish/subscribe model is well-suited for scalable message exchange patterns typical of fog environments. The simulation employs the following MQTT settings for application traffic generation:

- Message Length: 1MB (Note: This seems unusually large for typical MQTT messages. It might refer to the *task data* carried over MQTT)
- Local Port: 127.0.0.1 (Loopback address, likely representing local processing or communication within a node/cluster)
- Dest Port: 127.0.0.1 (Same as local port)
- Send Interval: Various (0.5s, 4s) - Simulating different traffic rates.
- Starting Time: 0.0s
- Stop Time: 200s or 500s or 1000s - Varying simulation durations.
- Publish To Topics: "Fog Message 1 (MQTT)"

RESEARCH ARTICLE

- Task Size: 1024 Bytes (1KB) - This seems more consistent with a typical task payload size compared to the 1MB message length. Assuming this is the actual size of the data/task being processed.

These settings allow for studying the network's behavior under various application workloads and message rates, which directly impact network performance measures such as latency, resource utilization, and packet loss [20, 21, 5].

2.5. Base Applications and Radio Medium

The intricate simulation configuration is designed to produce a realistic representation of a fog computing environment within the OMNeT++ simulation framework. The configuration supports an extensive examination of performance parameters like efficiency in resource utilization, end-to-end latency, and data rate under various network settings and mobility scenarios [30]. By including fundamental fog elements like compute and base brokers, a radio medium, and realistic application-level variables, the simulation closely represents real-world instances of distributed edge computing.

2.5.1. Base Broker Application Configuration

- Application Module: BrokerBaseApp3
- Local Port: 1001
- MIPS: 0 (since not a computational node)

The Base Broker, as a master node or cluster coordinator in some definitions, is deployed to handle routing of data and distribution within the fog network. Although it does not engage in direct heavy computational loads (as characterized by the MIPS of 0), it is a point of node that will facilitate data forwarding and control flow between wireless clients and compute brokers [31]. It facilitates task delegation, packet forwarding, and traffic balancing through forwarding incoming messages to available computing resources strategically. Its presence simulates a coordination layer that typically exists in actual fog architectures to facilitate decentralized decision-making in a coordinated fashion [20, 21].

2.5.2. Compute Broker Application Configuration

- Application Module: ComputeBrokerApp3
- Local Port: 1001
- MIPS Range: 1000–4000
- Message Length: 100 Bytes (This may be the control message size, distinct from Task Size/Message Length above)
- Send Interval: 1 second

Compute Broker nodes are provisioned to perform computationally expensive tasks. Each node is assigned a variable amount of MIPS (Million Instructions Per Second) in a specified range (1000–4000 MIPS) to simulate the computational heterogeneity found in many different edge devices. The MIPS range supports experimentation with resource scheduling algorithms under underloaded and overloaded loads. The setup in terms of message size and send interval creates a stream of task requests and enables the analysis of task queuing, processing delay, and throughput under various workload conditions [21, 5]. This setup simulates the loads under which edge devices handle IoT traffic data, e.g., smart city or industrial IoT applications.

2.5.3. Radio Medium Configuration

- Module: Radio Medium (standard OMNeT++ module)

Radio Medium module shows the wireless communication environment. It simulates the physical layer that is responsible for transmitting and propagating signals among wireless users, access points, and fog nodes. This module is the key to realistically simulate realistic wireless transmission conditions like signal attenuation, interference, and propagation delay [12]. Its inclusion ensures that, aside from computational efficiency, the simulation also accounts for communication realism, which is also paramount to establish overall performance in fog networks, especially with the variance and degradation sensitive nature of wireless links.

2.6. Free Space Model

The Free Space Model is applied in signal propagation, where signal strength reduces with the inverse square of the distance from receiver to transmitter in a free line-of-sight path. The model provides a standard in describing performance in perfect wireless communication with least environmental interference [12, 4].

2.7. Master Fog Node Forwarding Function (Baseline Logic)

In the baseline scenario (without Q-learning), the Master Fog Node employs a forwarding algorithm to decide whether an incoming job should be processed locally or transmitted to a neighboring node or cluster. This decision logic typically relies on predefined thresholds and checks the current load or status of potential processing nodes. As described in the original text, this approach might involve comparing current load against a fixed threshold and potentially using probabilities (though the link to Chapman probability here is less clear than in the Q-learning context). The parameter M—the maximum number of steps or hops allowed for forwarding is also relevant here, influencing the decision scope. As a benchmark for this non-adaptive approach, a parameter M = 5 is mentioned, noted for potentially providing a low drop rate and low reaction time in this specific predefined logic [14].

RESEARCH ARTICLE

This fixed logic contrasts with the learning-based decision-making provided by Q-learning.

2.8. Q-Learning Algorithm Implementation Details

Q-learning is an off-policy model-free reinforcement learning technique in which an agent learns an optimal policy by learning through exploration of an environment to maximize cumulative future reward.

The algorithm maintains and updates a Q-table of the quality (Q-value) of taking an action in a state. Update rule is derived from reward from environment and maximum expected future reward from next state [1, 6]. Critical parameters employed for our Q-learning agent in this simulation are:

1. Epsilon (ε)

Exploration rate refers to how frequently the agent chooses an action at random instead of choosing the action with the most Q-value (exploitation). Epsilon is initialized to 1.0 in order to encourage exploration of the states and actions on a large scale. The value is then reduced throughout the range of the episode or period to an optimal amount, in this study 0.1, in an effort to influence the agent to exploit acquired information [6]. The issue is that there is policy maximization compromise.

2. Alpha (α)

The learning rate indicates the proportion of new information that supplants previous information during Q-value update. Here, an alpha of 0.5 is employed so that recent experience affects the agent heavily without eliminating present knowledge completely.

3. Gamma (γ)

The gamma-value describes the relative importance assigned to rewards yet to come versus immediate rewards. A gamma of 0.9 is employed, i.e., high value to future rewards. It instructs the agent to learn policies that maximize long-term performance, and not the present action by itself.

4. Number of States

The states number *S* defines the number of significant conditions or settings of the FogNet world which can be perceived by the Q-learning agent. States within the paper are conceived to be built dynamically from real-time observation of network performance metrics and node health. These might be aggregation metrics of node loads, queue depth, mean response times, or interconnectivity in a cluster. Representative state space must be defined prior to learning by the agent.

5. Actions Size

Action space *A* specifies the range of discrete actions from which the Q-learning agent may choose in any state. Such activities within this simulation are known as different strategies to accomplish an impending task or request under the hierarchical fog model. These activities are:

- Execute the task on the local current (slave) node.
- Send/offload the task to the master node of the cluster.
- Repartition the work on a different node in the cluster.

For every state-action pair (s, a), a Q-value Q (s, a) is kept. They are updated progressively in cycles utilizing the Q-learning update equation with respect to the received reward. The reward function is such that reward is positive for increased throughput, reduction of response time, and decrease in drop rates, and negative for all other cases [23, 20].

2.8.1. Utilization of Chapman Probability in Q-learning

Chapman probability, traditionally applied in scientific studies of sto-chastic processes like Markov chains, is applied in the Q-learning algorithm in this study with the intention of strengthening state change modeling in the dynamic setting of the fog network. The application of Chapman probability can ideally predict more accurately what occurs to the network state when the agent acts a particular way [10].

1. State Transition Modeling

Using Chapman-Kolmogorov principles, we attempt modeling the transition probability of the current network state ‘s’ to the future state ‘s’’ via action ‘a’. This makes the agent extremely aware of the unpredictability of change in the network via its action and outside activities [22].

2. Reward Function Design

The reward function is designed to allow the agent to be incentivized to reach good network states. Under probabilistic state transitions insight, the reward function can hopefully be tuned so that it matches the long-run value of reaching some good network states (e.g., low congestion states) [4].

3. Q-learning Policy Optimization

Merging the probabilistic state transition information, which was designed on Chapman probability theory, with the Q-learning algorithm allows the agent to learn an improved policy that takes into consideration the natural stochasticity and dynamicity of the fog network.

This may contribute to more stable and efficient task scheduling and resource allocation decisions [28, 10].

2.9. Network Conditions Simulated in FogNet

For estimation of the Q-learning mechanism in different operating conditions, simulation consists of cases for different network conditions:

RESEARCH ARTICLE

1. Normal Operation

Devices and users function with ordinary traffic loads and patterns in ordinary network circumstances.

2. High Traffic Load

High traffic load operation emulates overload scenarios through the representation of high amounts of users and data packet generation rate, imposing stresses on network resources [26].

3. Mobility Models

The users possess advanced connectivity and data transfer patterns via the linear or circular mobility models witnessed, bringing in resource handling complexity.

2.10. Various Channel Types Simulated in FogNet

To evaluate the flexibility and robustness of the FogNet platform, particularly when the Q-learning agent is in charge, the paper suggests scenarios simulating various wireless channel conditions. The changes are introduced to simulate real-world problems that deteriorate the network performance and need to be resolved by smart management policies [30].

2.10.1. Channel Congestion

Here, the simulation emulates conditions of increased congestion of the communication channel that is often caused by heavy user activity, transmission data bursts, or low channel bandwidth [27]. Channel congestion results in a variety of effects:

- Very high collision rates for data to be retransmitted in packets and decreased effective bandwidth.
- Lower aggregate throughput due to many nodes fighting overuse of the channel.
- Communication latency was enhanced, which influenced timeliness of data exchange and task offloading.

Dynamic routing and task-creation decisions are modified using the Q-learning algorithm to mitigate the effects of congestion. The agent is learned to identify congested nodes or channels and attempts to favor less congested channels or task offloading to compute brokers with lower current load levels to maximize network efficiency as a whole.

2.10.2. Channel Attack Simulation (DoS Example)

A rudimentary test of the system's strength under adverse circumstances is carried out by merely simulating a simple case of malevolent action, i.e., a Denial-of-Service (DoS) attack on portions of the network (e.g., entry points or compute-brokers). The attacks are simulated by artificially causing high packet frequencies with targeted goals and thereby obtaining:

- Artificially caused starvation of resources at attacked nodes.
- Potential network segmentation or service degradation of legitimate nodes attempting to reach the attacked resources.
- Heavy packet loss and overall quality of service degradation.

The objective here is to observe how the Q-learning agent reacts in such a scenario. The learning adjustment of the agent with respect to real-time performance metrics can enable it to recognize poor node/channel performance and route tasks to other unaffected regions in the fog environment [24, 28]. It therefore has some form of availability of service under threat.

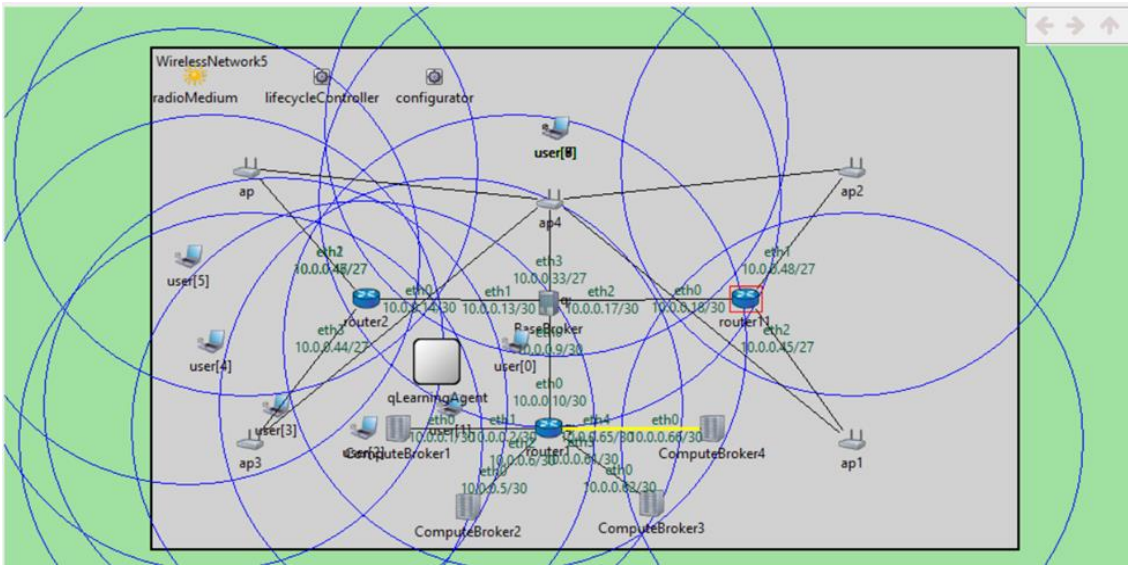

Figure 4 Impact of Channel Conditions on Network Performance

RESEARCH ARTICLE

Figure 4 showing central Q learning agent makes intelligent decision, likely for offloading computational tasks from wireless users to the most suitable compute broker. The goal is to optimize network resource allocation and reduce latency by learning the most efficient routing algorithms. Such channel variation situations are referred to for understanding real-world relevance of reinforcement learning techniques to uncertainty management and quality of service provisioning in fog-based systems under possibly varying wireless environments.

2.11. Working Mechanism of Simulation

The working mechanism of the simulation that is used in the FogNet system can specifically simulate the fog computing network operations in real-time, specifically those involving smart decision-making and adaptive resource allocation using reinforcement learning [27]. It operates in a sequence of crucial steps: initialization phase, then dynamic clustering operations, and ongoing interaction with the Q-learning agent for improved long-term performance.

2.11.1. Initialization Phase

During the beginning of simulation, the topology of the entire fog network is set up in the OMNeT++ framework. The whole network entities like base brokers, compute brokers, routers, access points, and wireless users are instantiated and created and initialized with their respective parameters (i.e., transmission power, MIPS capacity, queue capacities, and initial links). The above setup initializes the fundamental simulated fog infrastructure. In the meantime, the Q-learning agent is also set up and initialized in preparation for decision-making during the simulation [24]. The most important steps in initializing the agent are:

1. Q-table Initialization

The Q-table, where the Q-values for every conceivable state-action pair are stored, is initialized, often with zeros. The zero-filled table is the agent's starting knowledge base, and it allows it to start estimating the possible rewards (and undiscovered penalties) of different actions in given states by learning by trial and error over the course of the simulation process.

2. Epsilon (ε) Setup

The exploration rate ε is seeded (usually to a value such as 1.0 initially and then annealed to 0.1 according to Table 2) to regulate the probability that the agent selects a random action. Large initial ε provides complete exploration of the state-action space.

3. Alpha (α) Setup

The α learning rate (set at 0.5) is determined to control the pace at which the agent updates its Q-values using experience a new. An optimal alpha should optimize learning new information and preserving learned information.

4. Gamma (γ) Setup

The discount factor γ (which is set at 0.9) is designed to regulate the relative magnitude of rewards at future times. An almost-1 setting is designed to emphasize long-term improvement in performance. The settings outlined above allow the Q-learning agent to start to learn about the simulated world and learn optimally an adaptive policy for resource scheduling and task offloading decisions.

2.11.2. Clustering Operation in FogNet

Clustering is applied in FogNet as a method to enhance scalability, efficiency, and administrability of the distributed fog computing resources. The nodes in the fog are logically divided into clusters, and roles are allocated dynamically in such clusters:

1. Master Fog Node

A node plays the role of a master node in a cluster. The master node runs resources in its cluster, manages inter-cluster communication, and is generally tasked with performing heavyweight or complex operations that may not be efficiently performed by slave nodes [13]. Based on ongoing performance metrics and recommendations from the Q-learning agent (if available), the master node decides on load balance, possible task migration (either within a cluster or across clusters), and orchestration of services.

2. Slave Fog Node

In a cluster, there are slave nodes. They mainly carry out normal or localized processing duties. Slave nodes may delegate work to the master node or other appropriate nodes if heavily loaded or having limited resources. Such hierarchical, tiered setup aids in lessening the task management burden and resource allocation in an immense number of fog nodes. The dynamic nature of clustering is that nodes will have the ability to change roles or redefine cluster membership based on parameters such as dynamic load, availability of resources, and user mobility status [24].

2.11.3. Clustering Overhead and Q-Learning Optimization

Although clustering enhances manageability, it is achieved at the expense of overhead in inter-node coordination, synchronization, and control for trying to form and maintain clusters. Ineffective management of this overhead hurts network performance. The Q-learning algorithm is incorporated into the simulation to make optimal resource management decisions *within* the clustering framework. The Q-learning agent attempts to learn dynamic decision-making policies for situations like:

RESEARCH ARTICLE

- Which node to execute a task on (local slave, master, or another cluster).
- Potentially affects task distribution strategies to prevent overloading certain nodes or clusters.
- Making resource allocation choices that optimally reduce total overhead and maximize performance measures.

Since it learns through (reward/penalty for its actions), the Q-learning agent tries to make optimal choices in reducing the adverse effect of clustering overhead and maximizing peak performance. The agent tries to keep administrative complexity in equilibrium relative to computational simplicity so that the fog network is dynamic and responsive to changing conditions [12, 22]. Therefore, the simulation offers a ground for testing the influence of resource scheduling algorithms based on intelligent learning over such key performance parameters such as latency, throughput, resource utilization, and task completion ratio in a realistic fog computing scenario with hierarchical clustering and dynamic workloads.

2.12. Simulation Execution and Flowchart

The core intention of the FogNet simulation, and most importantly with the application of the Q-learning-based intelligent decision-making model, is to ensure optimal utilization of resources, offload task management, and reduce latency in an ever-changing scenario of fog computing. All this becomes achievable using the OMNeT++ platform that allows modular simulation of key factors such as user mobility, wireless communication, and processing at the fog nodes [8, 9]. The simulation procedure is an observation, learning (for when Q-learning is active), and decision-making loop as depicted in the flowchart in Figure 5. The flowchart depicts the general process flow from user action initiation and task generation to the smart management of resources by fog nodes (slave/master) and optimization of decisions by the principles of reinforcement learning.

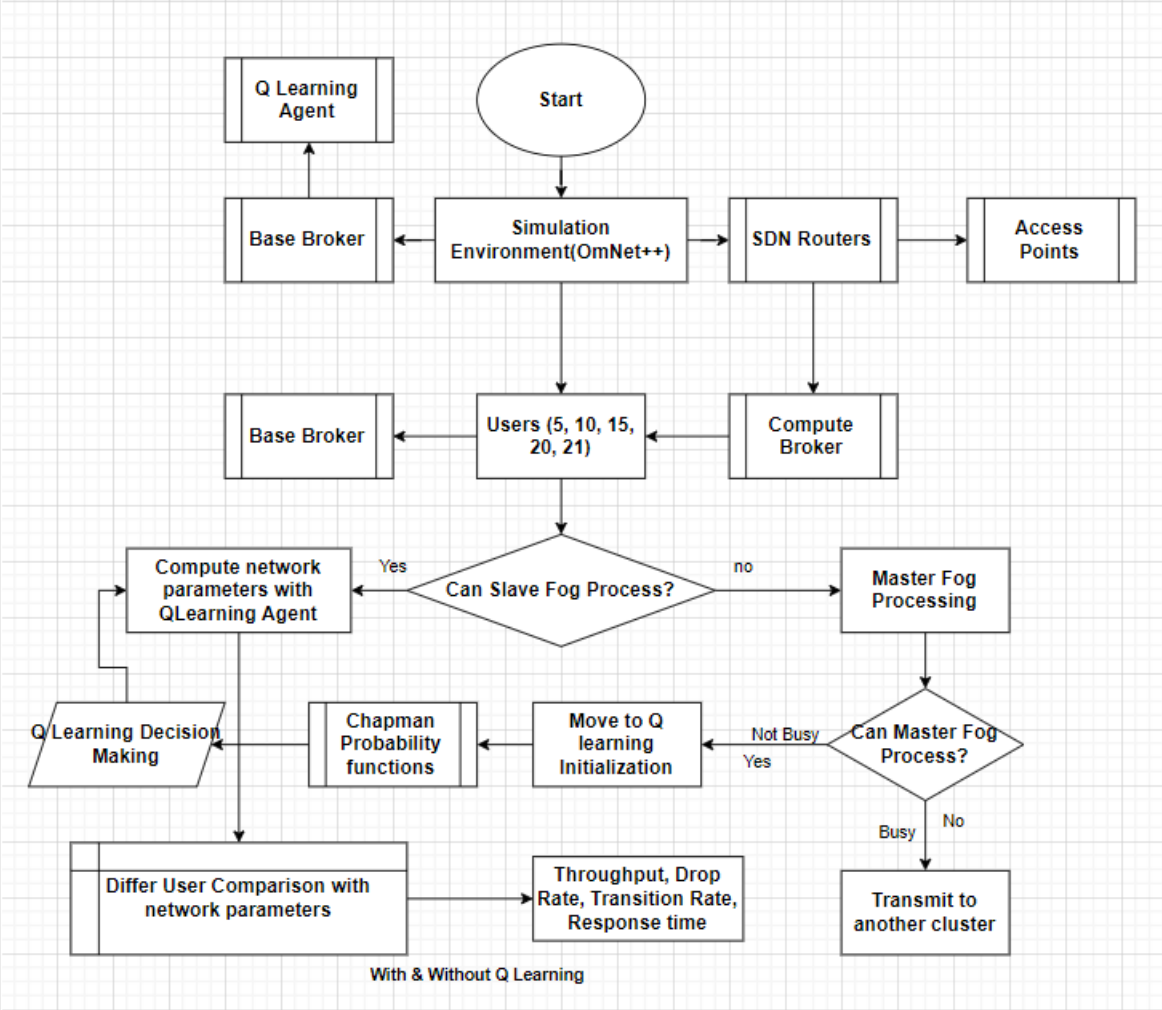

Figure 5 FogNet Simulation Flowchart

RESEARCH ARTICLE

1. Simulation Initialization

Simulation starts with the initialization of the OMNeT++ environment and the deployment of all entities of the network (base brokers, compute brokers, routers, access points, wireless users) along with their initial setting parameters.

2. Generation of tasks

Wireless users (node 5, 10, 15, 20, 21) generate service requests or tasks according to the given application parameters and mobility models [10].

3. Initial Routing

The tasks are initially routed, usually by routers, to the fog layer (compute brokers) for possible processing or forward routing.

4. Slave Fog Node Evaluation

The task being received is first forwarded to a local (slave) fog node. It is determined whether this slave node is adequate (adequate resources, low load) to complete the task optimally.

5. Decision Integration via Q-Learning (if running)

If the slave node determines that it can do the job, or needs to decide (in the Q-learning case), the Q-learning agent evaluates the present state of the network ‘s’ [22].

- From its learned policy (from the Q-table) and current state ‘s’, the agent determines an action ‘a’ (e.g., execute locally, offload to master, offload to other cluster).
- The selected action ‘a’ is taken in the simulation environment.
- The environment enters a new state ‘a’’, and the agent is rewarded ‘R’ for performing action ‘a’ in state ‘s’.
- The reward is computed from resulting performance metrics (e.g., throughput change, response time, drop rate).
- The Q-value for the state-action pair (s, a) is updated using Q-learning formula as shown in the equation (1).

$$Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \max Q(s', a') - Q(s, a)] \quad (1)$$

- We can use Chapman Probability axioms here in order to modify the state transitions or modify the reward setup, improving the learning accuracy.
- The current state ‘s’ is updated to the new state ‘s''.

6. Implicit Comparison & Decision-Making

The outcome of the decision-making by the Q-learning agent (or the baseline reasoning) decides the destination of the task local processing, offloading, or dropping [10]. Performance measures (throughput, response time, drop rate) are tracked at all times on the basis of these outcomes for both the cases with as well as without Q-learning so that the performance can be compared directly.

7. Fallback/Offload to Master Fog Node

If the priority issue that the task *cannot* be carried out efficiently by the slave fog node (either case), or the Q-learning agent decides offloading, the task is forwarded to a master fog node.

- The master node checks whether it is free (states "Not Busy").
- If it is a master node, it runs the task.
- If the master node is engaged or thinks that it ought to (possibly because of some Q-learning or baseline problem), the task could be queued to another cluster to a node to run [22].

8. Continuous Learning (if needed)

Q-learning goes on continuously in the simulation. Whenever the agent encounters new and difficult situations (states) where its Q-values are uncertain or non-optimal, it keeps sampling and updating its Q-table so that it can construct its policy dynamically.

9. Log Performance Metrics

During the simulation run, important performance measures like Throughput, Drop Rate, Transition Rate (for task migration/offloading), and Response Time are logged for each run task.

10. Comparison and Analysis

Performance measures achieved when simulation halts in the scenario *with* Q-learning and *without* Q-learning are compared. Total or average response time, throughput, and drop rate are computed under several user load and mobility scenarios.

11. End of Simulation Comparison with Previous Work

Compared graphically or quantitatively (if there are available benchmark results) to other research papers on improving fog computing performance.

12. Termination of Simulation

The simulation will stop when the stop time is reached as set.

The above process ensures that the simulation can effectively simulate the operation of a fog computing network under varying conditions and provide a good foundation for research on the impact of the Q-learning-based resource allocation policy.

RESEARCH ARTICLE

2.13. Simulation Mechanisms and Protocols

The simulation uses some common networking protocols and mechanisms to effectively simulate the operation of a fog computing system:

1. Data Packet Transfer

The data packets are transferred over the simulated network based on the base-level wireless and wired communication protocols supported by OMNeT++.

2. Wireless Communication

The wireless communication model adheres to the IEEE 802.11g standard, simulating traits such as data rate adaptation, possibility of interference, and signal attenuation with propagation (using models such as the Free Space Model). WLAN fine-grained setup and radio transmitter power control are realistic simulations of Wi-Fi network behavior [4, 10].

3. Queue Management

Drop Tail queuing disciplines are used in network nodes (e.g., brokers) for simulating how data packets are buffered and served within changing traffic load and congestion conditions [20, 23].

4. MQTT Protocol

Message Queuing Telemetry Transport (MQTT) is used for simulating the lean messaging behavior of example IoT applications sending data (information) to fog nodes [20, 21, 5].

2.14. Mathematical Model

2.14.1.Chapman Probability (Context in Stochastic Processes)

Chapman probability are laws of Chapman-Kolmogorov equation that are the foundation of Markov chain and other stochastic process research.

The equation offers the likelihood of moving from a specific state to another one within an interval of time as opposed to all intervening states.

In this work, these laws are being contemplated to possibly emulate the probabilistic transition between one network state to another in the fog environment graphically [10, 22].

1. Transition Matrix (P)

When the state transitions in a network are defined as a Markov chain, it is possible to use a Transition Matrix P to denote the state-to-state probabilities of transition in one step [22].

$$P = \begin{bmatrix} P_{11} & P_{12} \dots & P_{1n} \\ P_{21} & P_{22} \dots & P_{2n} \\ P_{n1} & P_{n2} \dots & P_{nn} \end{bmatrix}$$

2. Chapman-Kolmogorov Equation

It helps in finding the transition probabilities over multiple steps, enabling the prediction of network state changes over time. As shown in equation (2), it expresses the Markov property of transition probabilities.

$$P^{(k+n)} = P^{(k)} . P^{(n)} \quad (2)$$

Integrating these probabilistic modeling concepts into the Q-learning framework (as discussed in Section 2.8.1) aims to provide the agent with a more informed understanding of how its actions influence the future state of the dynamic fog network, potentially leading to more robust decision-making[17, 26].

These probabilistic model ideas integrated with the Q-learning setup (outlined in Section 2.8.1) are likely to give the agent a clearer idea of how its action is responsible for the dynamic fog network's future state and, thus, possibly result in more informed decisions.

2.14.2. Q-Learning Model Formulation

The Q-learning algorithm relies on the following mathematical model

1. States (S)

The collection of possibly observable states of the fog network ecosystem. These are, as illustrated, dynamic values such as node loads, queue lengths, delay, and connectivity. The Q-learning agent always knows these states.

2. Actions (A)

Finite set of actions that the agent can execute in any state, corresponding to task handling choices (local processing, offloading to master, offloading to other cluster). The agent chooses an action a from action set ‘A’ when in state ‘s’ based on an exploration-exploitation policy, e.g., the ε-greedy policy

- With probability ε to explore, the agent selects a random action ‘a’ from ‘A’.
- With probability 1−ε, the agent picks action a with highest Q-value for state s

It tries to balance exploration of possibly better unknown actions.

3. Reward (R)

Upon execution of an action ‘a’ by an agent in state ‘s’, the environment rewards it with an immediate reward R (s, a) for doing so. The reward function is used to measure the desirability of the resulting outcome as against performance measures. Rewards are rewarded for better throughput, response time, and drop rate, and penalty or negative rewards are imposed for worse-than-these measures [3].

RESEARCH ARTICLE

R (s, a) = Temporary value of executing action ‘a’ in state ‘s’

(3)

As shown in equation (3), R (s,a), it represents the reward function, which gives the immediate feedback an agent receives after taking action ‘a’ in state ‘s’.

4. Q-Function (Q)

The Q-function Q (s, a) is the discounted cumulative expected future reward of executing action *a* in state *s* and using the optimal policy. The Q-table is iteratively updated by employing the Q-learning update rule in equation (4).

$$Q(s, a) = Q(s, a) + \alpha [R(s, a) + \gamma \max Q(s', a') - Q(s, a)] \quad (4)$$

As shown in equation (4) defines the Q-learning update rule, where the Q-function Q(s,a) estimates the expected cumulative reward of taking action ‘a’ in state ‘s’. Where:
α is the learning rate, controlling the size of the update step.
γ is the discount factor, controlling the importance of future rewards.
‘s'’ is the next state reached by taking action *a* in state *s*.

5. Policy (π)

The Q-function maximizes expected rewards, guiding resource allocation decisions dynamically.

$$\pi(s) = \arg\max Q(s, a) \quad (5)$$

As shown in equation (5), it defines the optimal policy π which selects the action ‘a’ that maximizes the Q-value in state ‘s’. This policy will cause the agent to select the action with maximum probability of future reward maximization and thus dynamically optimize the resource allocation decisions for the fog network.

2.15. Pseudocode for FogNet Simulation with Q-Learning

Following is the algorithm 1 for FogNet Simulation with Q-Learning.

Initialize Simulation

For each user in [5, 10, 15, 20, 21]:

Connect user to AP

For each user in users:

Send data to AP

For each AP:

Forward data packet to connected Router

For each Router:

Transmit data to Compute Broker

For each data in Brokers:

If Slave Node can process data packet:

Process data packet in Slave Node

Computing throughput, response time, drop ratio

Else:

If Master Node can process data:

Process data packet in Master Node

Computing throughput, response time, drop rate

Else:

Transmit data to another cluster

Compute total throughput, response time, drop rate

Declare Q-Learning parameters

For each episode in Q Learning function:

Declare state S

For each step in episode:

Choose action A using policy calculated from Q Algo

Take action A, observe reward R(S, A) and next state S'

Update Q(S, S) through:

Q(S, A) = Q(S, A) + alpha * [R(S, A) + gamma * max_A' Q(S', S') - Q(S, A)]

Update state S = S'

Computing throughput, response time, drop rate with Q Learning

Compare performance results with and without Q Learning

Analysis performance metrics

Compare results with other researchs paper

End of Simulation

Algorithm 1 FogNet Pseudocode with Q-Learning

3. RESULTS

This section presents the results obtained from the OMNeT++ simulations comparing the performance of the FogNet environment operating without Q-learning (baseline) and with the proposed Q-learning-based resource management strategy.

The simulations were conducted for varying numbers of static users (5, 10, 15, 20, and 21) to evaluate performance under increasing load. We analyze key performance metrics: throughput, response time, and drop rate. The results are presented graphically and in a summary table to facilitate understanding and comparison [4, 7].

RESEARCH ARTICLE

3.1. Performance Analysis Without Q-Learning (Baseline Scenario)

In the baseline scenario, task allocation and resource management follow a predefined, static logic (e.g., based on simple thresholds or the Master Fog Node forwarding function described in Section 2.7) without any adaptive learning based on real-time performance feedback.

The results below show the network performance metrics as the number of users increases. Different mobility patterns (linear and circular) were tested, but for clarity, the results aggregated or representing a typical outcome from these patterns are presented to show the general trend of the baseline performance degradation under load.

3.1.1. Throughput

Throughput measures the rate at which data is successfully transmitted and processed within the network. Higher throughput indicates more efficient data handling and better network performance. Figure 6 illustrates the observed throughput in the scenario without Q-learning as the number of users increases.

As depicted in Figure 6, the network throughput shows a noticeable decline as the number of users increases from 5 to 21. At 5 users, the throughput is relatively high, but it progressively decreases as more users contend for limited resources at the fog nodes (both slave and master) and communication channels. This decline is expected in a system using static resource management, as it struggles to efficiently distribute the increasing workload and manage congestion when demand grows beyond the initial design assumptions.

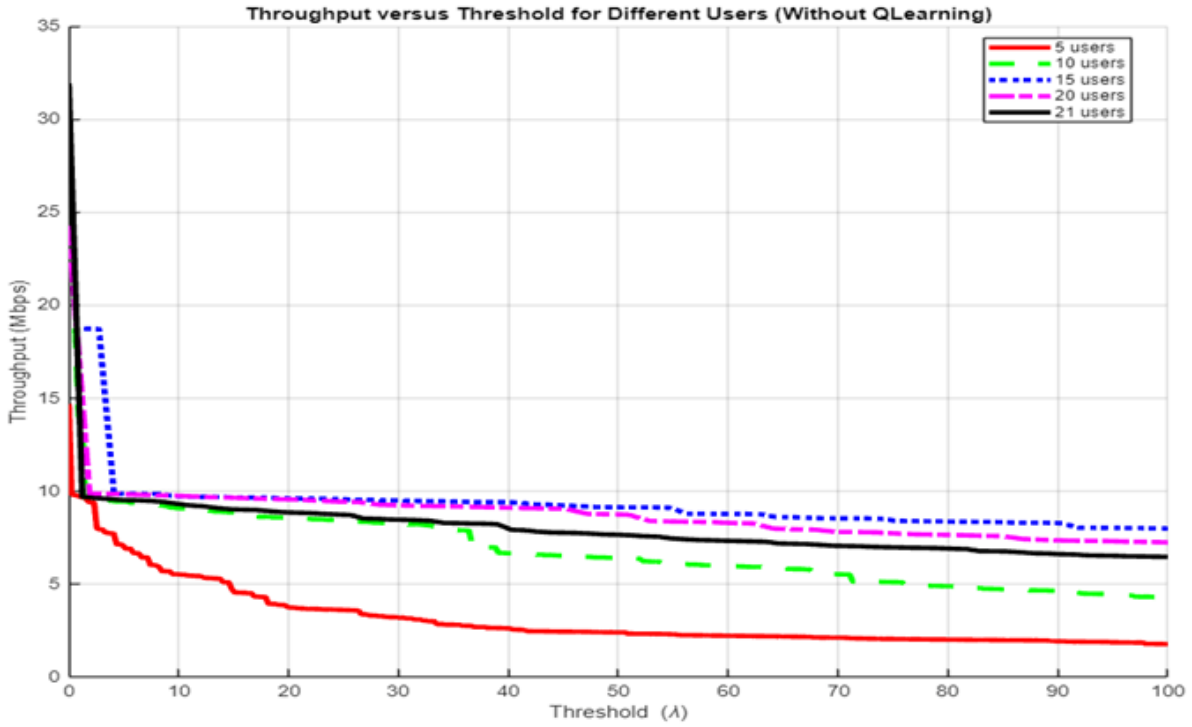

Figure 6 Throughput Without Q-Learning vs. Number of Users

3.1.2. Response Time

Response time is the duration from when a user request is initiated until a response is received. Lower response times indicate faster processing and better user experience, especially crucial for latency-sensitive IoT applications.

Figure 7 presents the average response time observed in the baseline scenario.

Figure 7 clearly shows that the average response time increases significantly as the number of users grows. With only 5 users, the response time is relatively low.

However, as the load increases to 10, 15, 20, and 21 users, tasks experience longer delays due to increased queuing at fog nodes and communication bottlenecks.

This indicates that the static resource allocation strategy is ineffective in managing workload and preventing delays under higher traffic loads.

RESEARCH ARTICLE

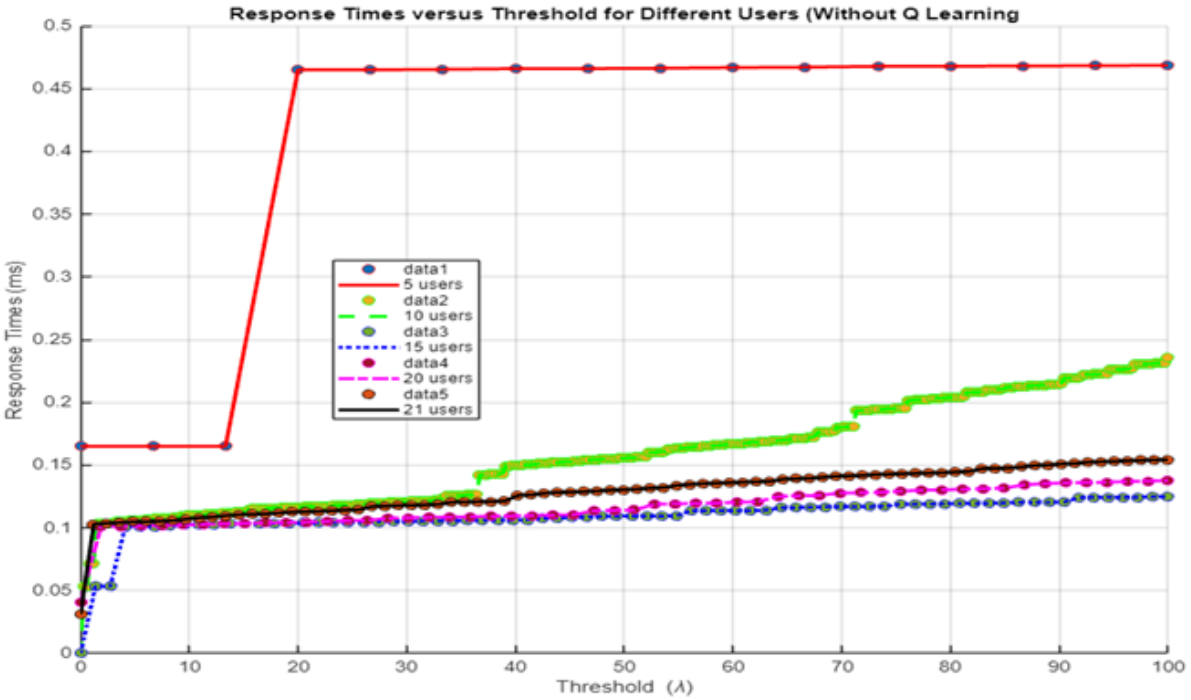

Figure 7 Response Time Without Q-Learning vs. Number of Users

3.1.3. Drop Rate

Drop rate is the percentage of data packets or tasks that are lost during transmission or processing due to congestion, queue overflows, or other network issues. A lower drop rate signifies a more reliable network with fewer data losses. Figure 8 shows the packet drop rate observed in the baseline scenario.

As seen in Figure 8, the drop rate tends to rise with the increasing number of users. While relatively low at 5 and 15 users, there are noticeable increases at 10, 20, and 21 users. This variability might depend on specific simulation instances or interaction with mobility, but the general trend indicates that under higher loads, the baseline network management struggles to prevent packet loss, likely due to queues reaching capacity at congested nodes [24, 25].

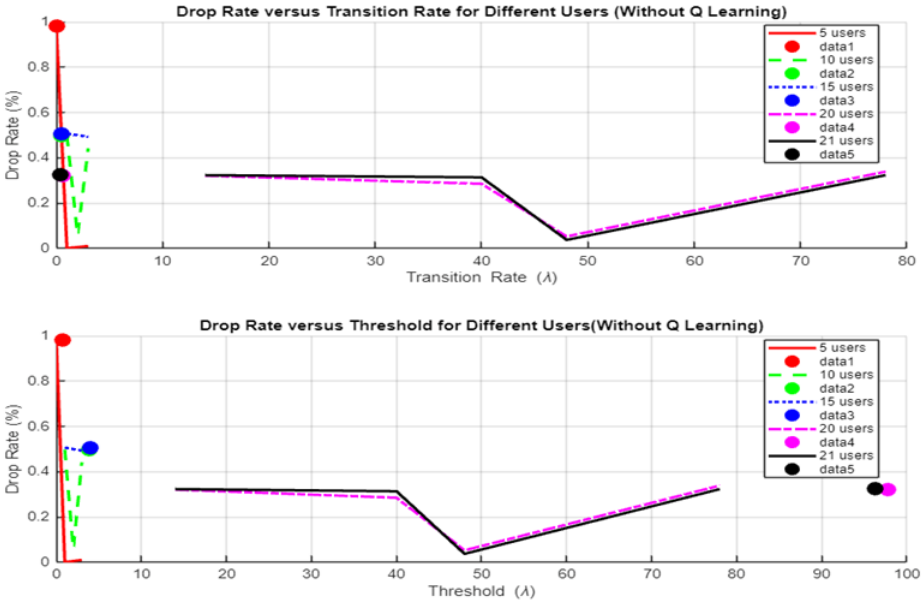

Figure 8 Drop Rate Without Q-Learning vs. Number of Users

RESEARCH ARTICLE

3.1.4. Effect of Number of Users (5, 10, 15, 20, 21) Comparisons Results

We analyze the network performance metrics with varying numbers of users to understand the scalability and capacity of the FogNet without Q-learning.

- Throughput

Expected to decrease as the number of users increases due to higher competition for resources as it is visualized in figure 9.

- Response Time

Likely to increase with more users, leading to potential delays as it is shown in figure 10.

- Drop Rate

May rise with the increasing load, causing more packet losses as it is visualized in figure 11.

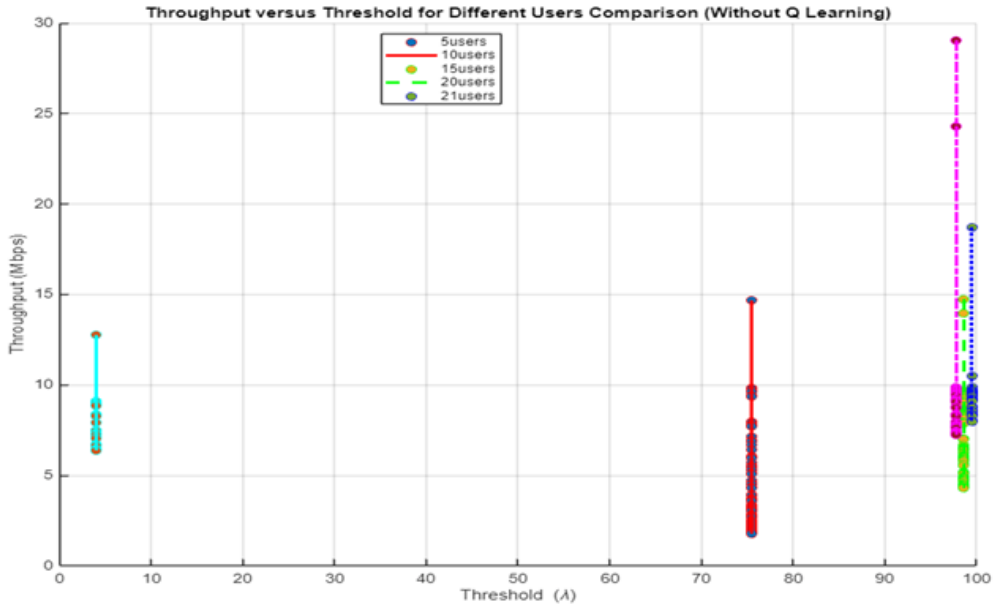

Figure 9 Throughput

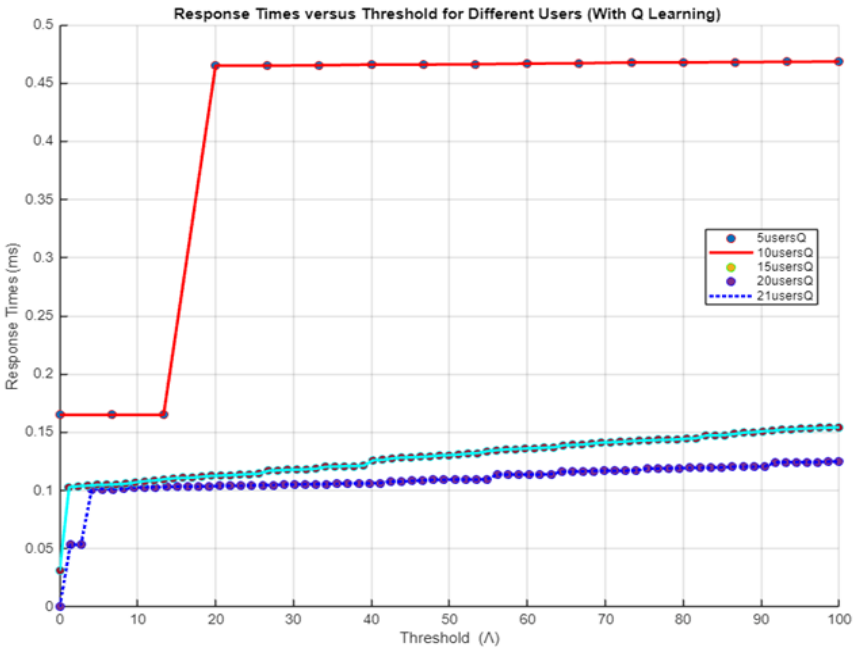

Figure 10 Response Time

RESEARCH ARTICLE

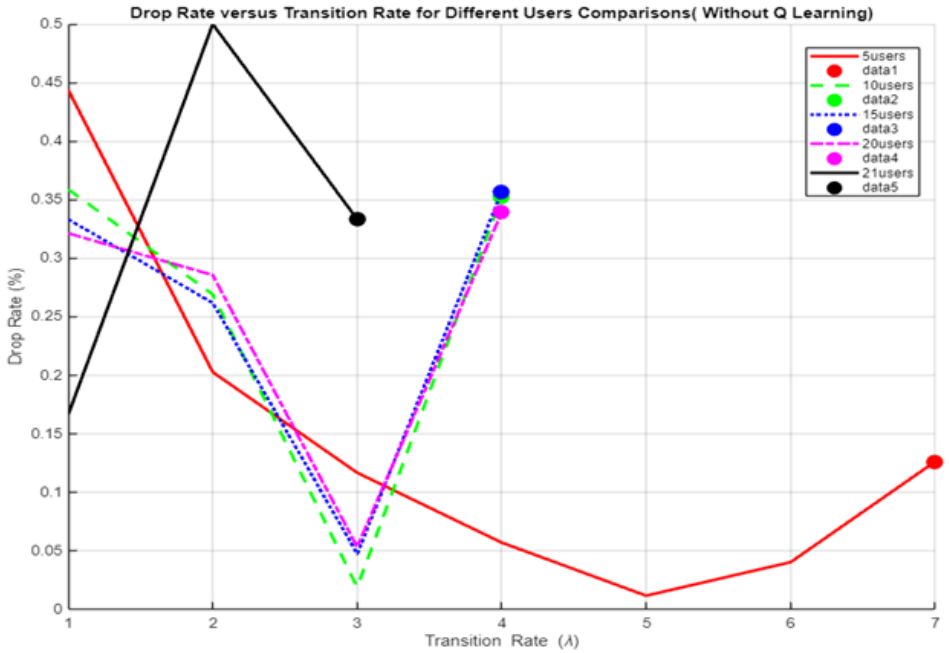

Figure 11 Drop Rate

3.1.5. Summary of Without Q-Learning Scenario

The results from Figures 6, 7, 8, 9, 10 and 11 collectively demonstrate that the FogNet simulation, when operating without an adaptive Q-learning strategy, exhibits predictable performance degradation as the user load increases. Throughput decreases, response times increase, and the drop rate rises, highlighting the limitations of static resource management in handling dynamic and scaling workloads in a fog computing environment [12, 20].

3.2. Performance Analysis With Q-Learning

In the second scenario, the FogNet simulation incorporates the Q-learning agent to dynamically manage network resources and task offloading decisions. The Q-learning agent learns from real-time network conditions (states) and performance outcomes (rewards) to adjust its decision-making policy, aiming to optimize overall system performance. The results below show the network performance metrics when Q-learning is applied.

3.2.1. Throughput

Figure 12 illustrates the network throughput observed in the scenario with Q-learning as the number of users increases. Figure 12 shows that with Q-learning enabled, the network maintains significantly higher throughput levels across all user counts compared to the baseline scenario (Figure 7). Although there is still a slight decrease in throughput as the user count rises from 5 to 21 (as expected due to finite resources), the degradation is less pronounced. At 5 users, the throughput is higher than the baseline. More importantly, under higher loads (15, 20, 21 users), the system with Q-learning sustains substantially better throughput, indicating its effectiveness in managing resources to process a larger volume of data successfully.

3.2.2. Response Time

This presents the average response time observed in the scenario with Q-learning. As Figure 13 demonstrates that the average response time in the Q-learning scenario remains considerably lower than in the baseline scenario (Figure 7), particularly as the user load increases.

While response times do increase with more users, the rate of increase is less steep, and the absolute values are lower. This indicates that the Q-learning agent's adaptive task offloading and resource allocation decisions are effective in reducing delays and improving the responsiveness of the fog network, even under higher congestion levels. The missing data points for 20 and 21 users might indicate simulation anomalies or data collection issues in those specific runs, but the trend up to 15 users is clear.

3.2.3. Drop Rate

Figure 14 shows the packet drop rate observed in the scenario with Q-learning. Figure 14 shows that the drop rate is significantly reduced when Q-learning is employed compared to the baseline scenario (Figure 7). For lower user counts (5 users), the drop rate is very low.

Although it increases slightly with higher user loads, the drop rates remain substantially lower than the baseline scenario

RESEARCH ARTICLE

across all user numbers. This result strongly suggests that the Q-learning agent's ability to adaptively route tasks and avoid congested nodes effectively minimizes packet loss, leading to a more reliable network [14, 25].

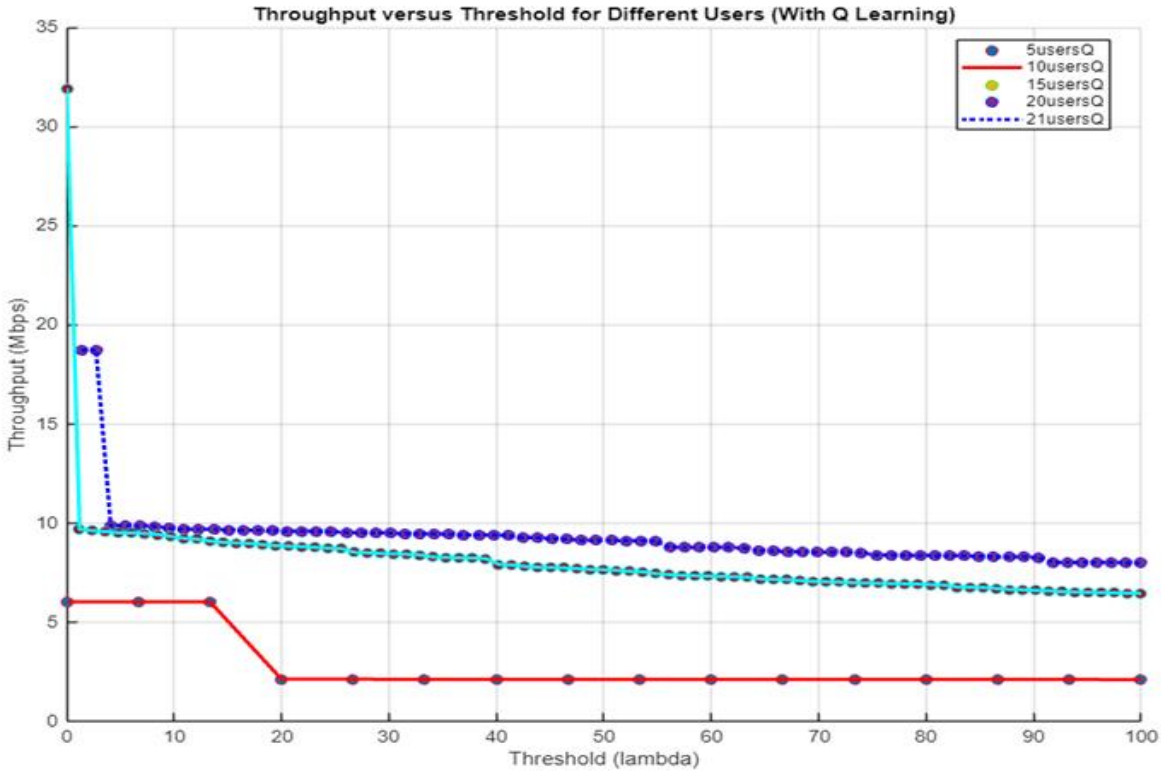

Figure 12 Throughput With Q-Learning vs. Number of Users

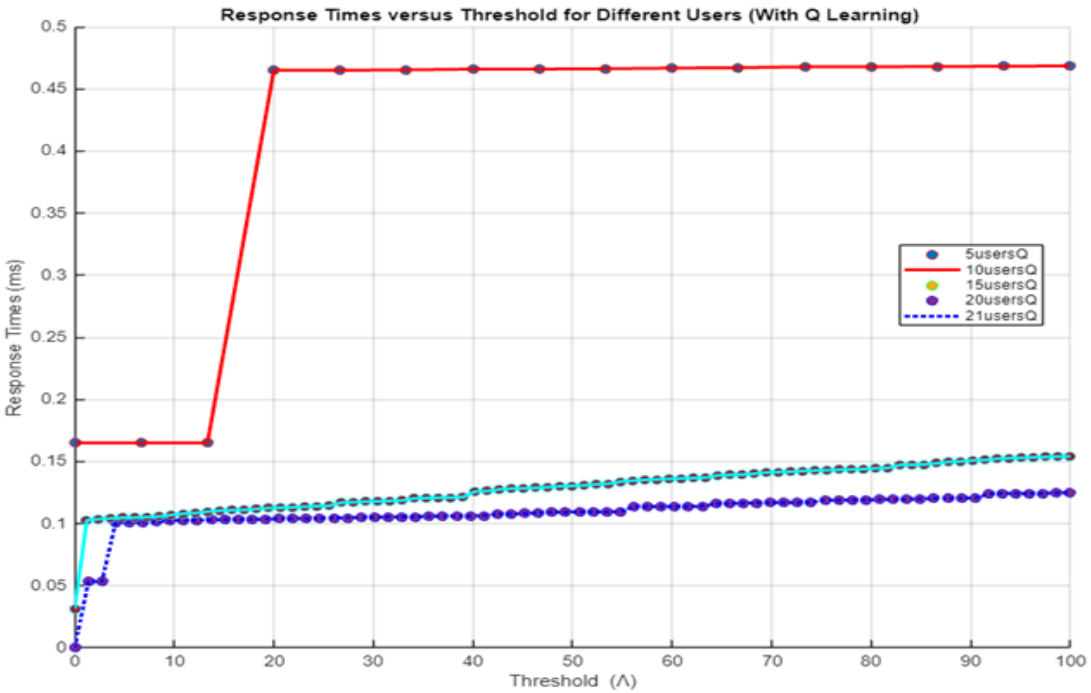

Figure 13 Response Time With Q-Learning vs. Number of Users

RESEARCH ARTICLE

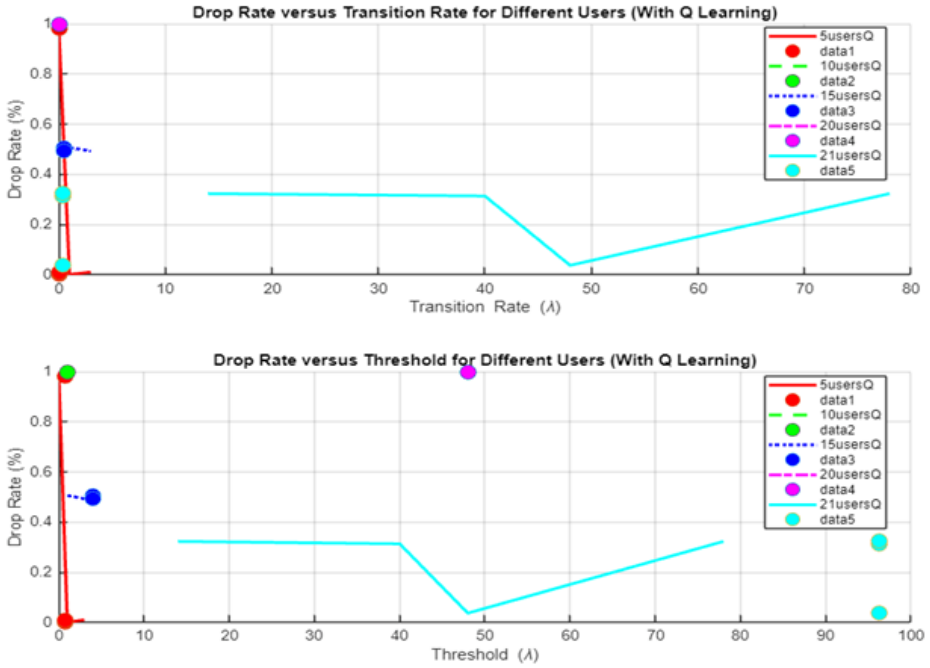

Figure 14 Drop Rate With Q-Learning vs. Number of Users

3.2.4. Effect of Number of Users (5, 10, 15, 20, 21) Comparisons Results

We analyze the network performance metrics with varying numbers of users to understand the scalability and capacity of the FogNet without Q-learning.

- Throughput

Q-learning aims to optimize resource allocation, potentially maintaining higher throughput even as the number of users increases as it is shown in figure 15.

- Response Time

As shown in figure 16, the Q-learning could help in managing resources more efficiently, thereby reducing the response time compared to the scenario without Q-learning.

- Drop Rate

The use of Q-learning is expected to minimize packet losses by adapting to the network conditions dynamically, thus lowering the drop rate as compared to the non-Q-learning scenario.

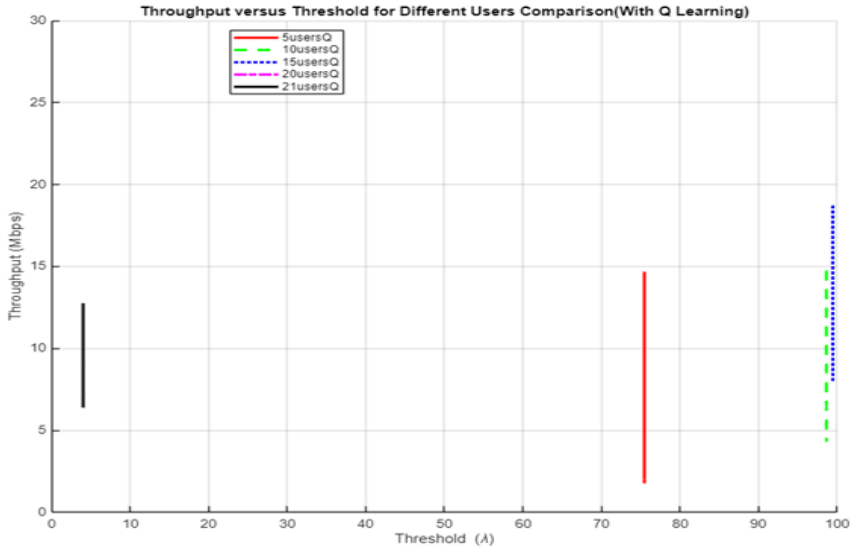

Figure 15 Throughput

RESEARCH ARTICLE

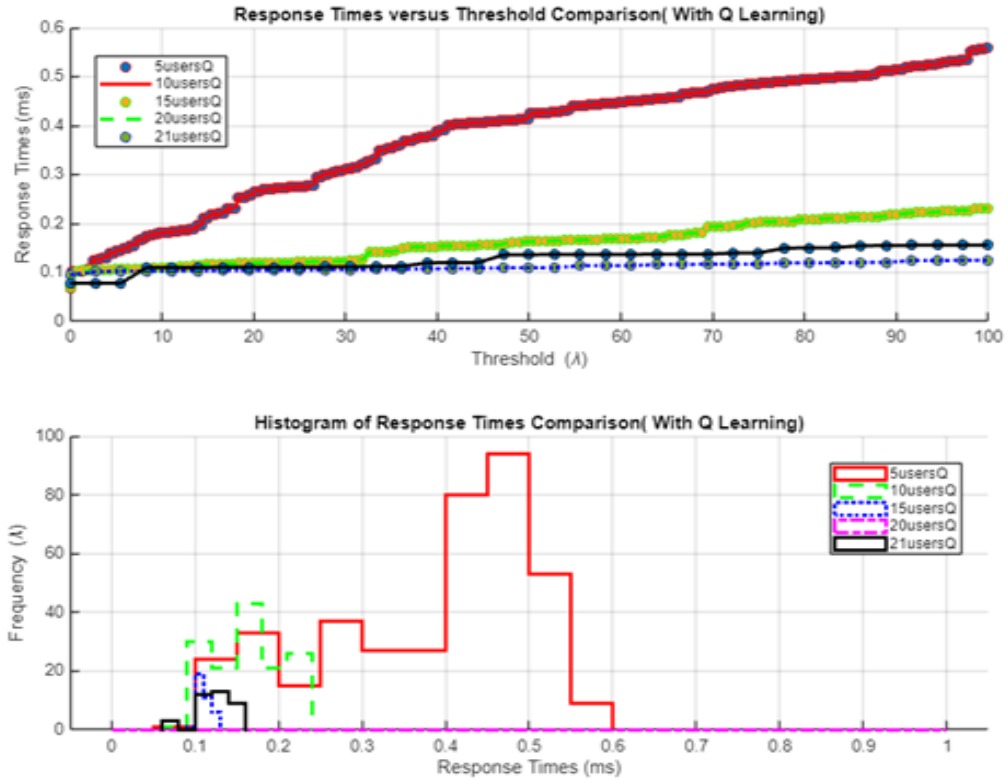

Figure 16 Response Time

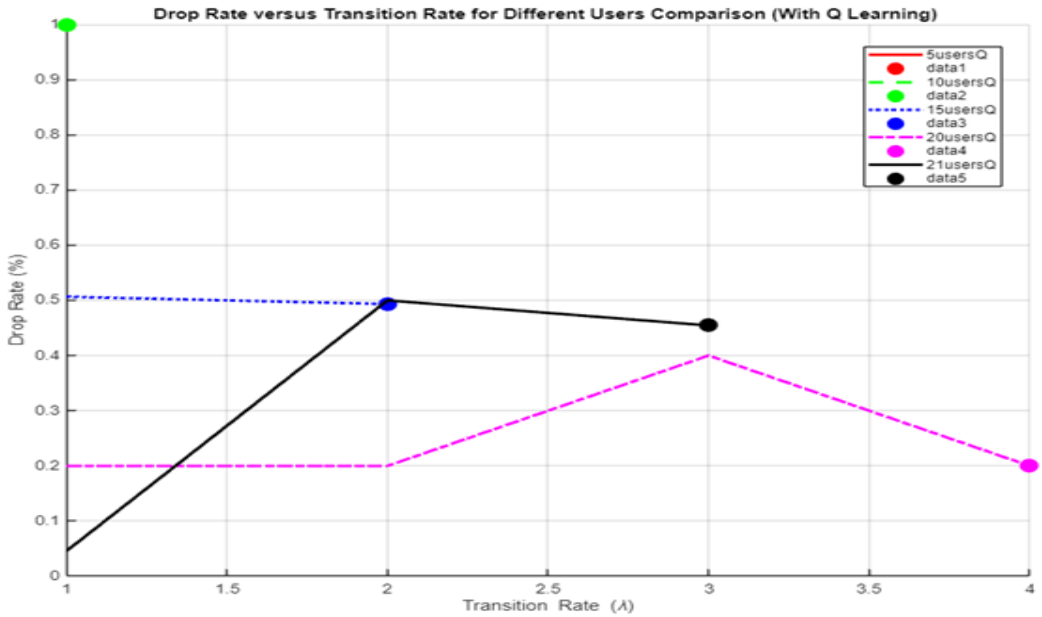

Figure 17 Drop Rate

3.2.5. Summary of with Q-Learning Scenario

The results from Figures 13, 14, 15, 16, 17 and 18 indicate that integrating Q-learning into the FogNet simulation leads to marked improvements in key performance metrics under varying user loads. Higher throughput, lower response times, and reduced drop rates are observed, demonstrating the

RESEARCH ARTICLE

effectiveness of the adaptive, learning-based approach in managing resources and handling increasing workloads in a dynamic fog computing environment.

3.3. Comparative Summary of Simulation Results

Table 3 provides a direct quantitative comparison of the average performance metrics for both scenarios (without and with Q-learning) across the different user counts.

Table 3 Comparative Performance Metrics: Without vs. With Q-Learning

Scenario	Users	Q-learning	Throughput (mbs)	Response time (ms)	Drop rate%
Without Q-learning	5	No	83.45	44.508	2.067
	10	No	79.762	49.660	3.687
	15	No	74.452	43.765	2.263
	20	No	70.013	40.769	3.523
	21	No	69.914	40.107	3.016
With Q Learning	5	Yes	124.123	39.760	0.336
	10	Yes	110.145	37.987	1.919
	15	Yes	98.127	33.660	1.840
	20	Yes	90.145	---	2.238
	21	Yes	87.675	---	2.987

The data in Table 3 quantitatively supports the trends observed in the individual plots. For instance, at 21 users, throughput with Q-learning (87.675 Mbps) is significantly higher than without (69.914 Mbps), and the drop rate with Q-learning (2.987%) is lower than without (3.016%, although the difference is smaller at this highest load point compared to others like 10 or 20 users). At 15 users, the throughput with Q-learning (98.127 Mbps) is substantially better than without (74.452 Mbps), response time is lower (33.660 ms vs 43.765 ms), and drop rate is lower (1.840% vs 2.263%). These numbers provide concrete evidence of the performance benefits achieved by implementing the Q-learning-based resource management strategy in the simulated FogNet environment, particularly under moderate to high user loads.

4. DISCUSSION

The results from the OMNeT++ simulation outputs are strong predictors that the implementation of a Q-learning-based solution for resource management significantly enhances the performance of a fog computing network compared to a non-adaptive, static equivalent. Key findings from key performance metrics – throughput, response time, and drop rate consistently demonstrate the advantages of incorporating reinforcement learning in addressing the dynamic volatility of fog environments.

Simulation results show that, without Q-learning, the performance of the FogNet worsens in a predictable manner as the user load increases. The decrease in throughput (Figure 6), the rise in response time (Figure 6), and the increase in drop rate (Figure 7) for high user counts (10, 20, 21 users particularly) indicate the inability of static resource allocation methods to scale with increased demand as well as manage congestion effectively. This is in accordance with the problems of conventional approaches observed in dynamic and heterogeneous fog networks [4, 7]. On the other hand, the application of the Q-learning algorithm leads to remarkable improvement in network performance.

As observed through Figures 12, 13, and 14, the Q-learning system has a better throughput, reduced response time, and severe drop rate reduction for different user loads compared to the baseline. For example, when there are 15 users, Q-learning provides approximately 32% more throughput (98.127 Mbps vs. 74.452 Mbps), roughly 23% lower response

RESEARCH ARTICLE

time (33.660 ms vs. 43.765 ms), and a lower dropping rate (1.840% vs. 2.263%) (Table 3). This shows that the Q-learning agent can learn to make better decisions on task offloading and resource allocation by learning the state of the network in real time. The strength of Q-learning is its ability to learn iteratively from the interactions with the environment. By offering rewards or penalties based on the outcome of its actions (e.g., how the choice to offload a task impacts response time or congestion), the agent continuously updates its Q-table to approximate the optimal policy [4, 7].

This allows the system to select actions dynamically (e.g., run locally, offload to master, offload to other cluster) that are most likely to yield better performance measurements given the current network conditions. For instance, when a slave node is overloaded, the Q-learning agent will figure out that offloading workload to a less-loaded master node or another cluster yields higher rewards (lower delay, lower drop rate) than local processing. This flexibility also provides uniform computational load distribution across fog nodes and prevents the occurrence of solitary congestion points, thereby enhancing overall system efficiency and removing bottlenecks. The master-slave hierarchical structure most likely supports the Q-learning process by providing definite decision points and resource pools. The decisions by the Q-learning agent between offloading in slave, master, and even other clusters utilize the structure to efficiently distribute tasks. The simulated mobility patterns (circular and linear) introduce dynamic connectivity and load variations in distribution, which the Q-learning agent is designed to adapt to by learning to modify its policy based on observed state transitions and their corresponding outcomes [4, 6].

While not completely proven in this specific study's results, the theoretical incorporation of Chapman probability into the Q-learning algorithm (explained in Section 2.8.1) possesses the theoretical potential to further improve the agent's probabilistic understanding of state transitions in dynamic network worlds and thereby provide more accurate long-term decision-making [9, 10]. Compared to traditional heuristic-based approaches based on predesignated rules, Q-learning allows for a more inherently adaptable approach. It does not require an explicit model of the network dynamics and can only learn effective policies by trial and error and experimentation. It is therefore very well adapted to dynamic and complex fog computing environments where the network dynamics change rapidly. Whereas this study primarily contrasts Q-learning with a simple static baseline within the simulation, the enhancements revealed are consistent with the potential set forth by other research examining machine learning and reinforcement learning for resource management within fog and edge computing [7 , 9, 17].

And even without actual energy consumption outcomes presented in this paper, the task distribution optimization and congestion alleviation that Q-learning can achieve can indirectly improve energy efficiency by reducing unnecessary retransmissions (lowered drop rate) and possibly optimizing resource utilization among nodes from being overloaded continuously by some high-power nodes [13]. The flexibility capability further enhances network resilience by allowing the algorithm to continue to modify policies in real time with real-time performance metrics to help achieve sustained efficiency even in dynamic traffic or localized degradation (as studied in part in simulated channel variations). Difficulty remains, however, in applying reinforcement learning, including Q-learning, to actual fog computing environments. Some of these include computational expense of training and running the agent, convergence time to the optimal policy for the model, and achieving scalability and real-time responsiveness for very large-scale networks with distributed agents. The state space can increase exponentially in very complicated scenarios, which can compromise convergence.

Additionally, development of an effective reward function that accurately reflects the preferred performance goals under any imaginable network situation is necessary and not straightforward [4]. Even though the simulated attack over the channel is already providing an indication of robustness, more sophisticated and distributed adversarial models would require more complex defense methods, potentially involving multi-agent Q-learning or incorporation with expert security modules [18, 19, 20, 21]. In summary, the simulation outcomes clearly confirm the superiority of the Q-learning-based resource control mechanism over a static approach in improving throughput, response time, and drop rate in a simulated hierarchical fog computing system. This demonstrates the potential of reinforcement learning as a valuable tool to improve performance and manage complexity in dynamic fog scenarios and providing avenues for further innovation and deployment in real-world systems.

5. LIMITATIONS

However, the present study offers important revelations concerning the efficiency of Q-learning in resource management within simulated FogNet environments, it is advisable to note the following limitations:

- Simulation-Only Evaluation relies on simulation with OMNeT++. Although the simulated environment was structured in order to mimic real conditions, the simulation environment cannot adequately reflect all the intricacies, uncertainties, and overheads present in re-al-world physical fog networks, e.g., varying hardware performance, intricate interference patterns, or diverse network technologies out of the range of the specified parameters.
- The Limited Mobility Models in circular and linear mobility patterns the actual user movement is much more

RESEARCH ARTICLE

dynamic, unpredictable, and subject to environmental barriers or social interactions, which might affect network connectivity and performance differently. More sophisticated mobility models such as Random Waypoint with pause times or with spatial constraints would be a more realistic approach [4, 15].

- Static Environment Assumptions about user load and mobility are dynamic, some underlying parameters such as wireless configuration (e.g., fixed 802.11g configuration) and fundamental task sizes were kept static. In reality, these parameters have extreme variations depending on the environment, application type, and underlying network infrastructure and affect the best resource manage-ment approach.
- Simplified Security Scenarios contains a straightforward channel congestion simulation and a less complex Denial-of-Service (DoS) attack scenario. This doesn't examine the entire range of advanced or distributed threat models (e.g., spoofing, man-in-the-middle, distributed denial of service (DDoS)) actual fog networks could encounter, nor does it discuss the Q-learning agent's particular defensive mechanisms beyond the implicit avoidance of deteriorated paths.
- Q-learning Implementation like Deep Q-Networks (DQN) to cope with bigger state spaces, or multi-agent Q-learning (MARL) where learning agents are several in number and learn cooperatively or competitively, may possibly provide better performance, scalability, and resilience in more complicated and distributed fog contexts.
- Uncertainty in Chapman Probability Application discussed conceptually, the tangible effect and strict implementation of Chapman probability concepts for quantitatively enhancing state transition modeling or reward shaping in the Q-learning update rule within this simulation were not measured or shown as a certain contribution. Its application remains more of a theoretical nature within the confines of the provided results.
- In Energy Consumption Analysis despite being cited as a probable metric, an in-depth analysis and report of energy consumption findings were not provided, reducing the evaluation of the effect of Q-learning on energy efficiency.

6. FUTURE WORK

Based on the results and recommendations of this study, the following directions are proposed for future study in order to further develop the field of smart resource management in fog computing:

1. Real-World Testbed Testing and Deployment

Deployment testing of the Q-learning model as suggested by implementing it in a real-world testbed or combining it with a small-scale cloud-fog-edge system will be crucial in an attempt to mimic under realistic conditions, e.g., actual hardware constraints, random network jitter, and operation overheads.

2. More Realistic and Varying Mobility Scenarios

Having more realistic and varying mobility scenarios, for example, models such as Random Waypoint with realistic pause time, human mobility models, or obstacle-sensing movement with realistic geography topographies (e.g., city maps), would add much to the accuracy of the simulation and accounting for the agent's responsiveness in realistic city or industrial environments.

3. Research on State-of-the-Art Reinforcement Learning Methods

Future research will explore more sophisticated DRL methods like Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), or Asynchronous Advantage Actor-Critic (A3C), which are better suited to deal with greater and potentially infinite state and action sets of large-scale complex fog networks, resulting in faster convergence and optimal policies.

4. Multi-Agent Reinforcement Learning (MARL) Research

As a result of fog computing's decentralized aspect, multi-agent reinforcement learning research, in which cohorts or single fog nodes are treated as cooperative or competitive learning agents, can prove to be helpful in creating more scalable and fault-resilient resource management policies.

5. Evoluted and Distributed Attack Instances

Inclusion of evoluted and more distributed instances of security attacks (e.g., sophisticated jamming, data poisoning, distributed denial of service (DDoS) by a set of compromised devices, or node impersonation by malicious nodes) will assist in evaluating and hopefully improving the robustness and resiliency of Q-learning-based resource management against adversarial cyber environments [22].

6. Dynamic Capacity Allocation and Infrastructure Flexibility

Adaptation strategy capacity building with the ability of re-arranging broker capacity, access point coverages, or even virtual fog node spawning/placement dynamically based on expected load and Q-learning decision can make the system resilient to many requests and infrastructure bottlenecks.

7. In-Depth Chapman Probability Integration Discussion

More theoretical and empirical research needs to be carried out to thoroughly investigate the integration process of the Chapman probability idea both qualitatively and

RESEARCH ARTICLE

quantitatively in the state representation, transition model, or reward function of the Q-learning algorithm for its further advancement in stochastic fog conditions.

8. In-depth Energy Consumption Analysis

An in-depth analysis of the energy consumption for individual fog nodes as well as for the entire network across various resource management approaches and methods (e.g., the Q-learning method) must be presented to present an exhaustive performance analysis.

9. Security and Privacy of RL-Based Fog

Investigating how security and privacy can be ensured during training and decision-making for the data in Q-learning-based fog networks, for instance, through mechanisms like federated learning or differential privacy, is one of the essential aspects of constructing reliable systems [27, 5, 12].

7. CONCLUSION

This paper introduced and analyzed a Q-learning-based optimization platform for resource management in a simulated hierarchical fog computing network. It focused on mitigating issues of dynamic task scheduling and resource allocation in fog environments with differential user load and mobility. By utilizing the performance of the simulated FogNet with and without including the Q-learning mechanism, the results validate the high potential in employing reinforcement learning for improving the basic network performance indicators.

The results repeatedly indicate the superiority of the Q-learning method over the baseline static resource allocation policy. In particular, the simulation results provide improved throughput, reduced average response time, and reduced packet drop ratios, especially for moderate to heavy loads [14, 30]. The gains verify that the Q-learning agent is able to learn efficiently to make adaptive optimal resource allocation and task offloading decisions between the hierarchical master-slave fog nodes from real-time network observations and performance feedback.

The Q-learning adaptive feature enables the fog network to dynamically adapt to traffic conditions, user movement, and localized congestion. The capability is important in minimizing system bottlenecks, task load balancing, and ultimately enhancing the general Quality of Service (QoS) provided to end-users within a dynamic environment [27]. Simulation facilitated the belief that learning-based control can result in smarter use of resources compared to pre-defined static rules.

While the simulation gave a sanitized setting to illustrate these advantages, subsequent studies need to pay attention to the realities of implementing Q-learning architectures in production fog computing systems. It will be important to address challenges pertaining to computational overhead, model convergence rate when scaled, and the engineering effort of constructing good reward functions for successful use. Additional studies in the field of advanced RL algorithms, like Deep Rein-force Learning or Multi-Agent Reinforcement Learning, and incorporating secure security and privacy functionalities would further enhance RL-based fog networks to become more scalable, flexible, and reliable.

In conclusion, this article reaffirms the effectiveness of Q-learning as a potential method for resource optimization management in fog computing [31]. The validated improvement in throughput, response time, and drop rate improvement opens the door to further utilization of reinforcement learning methods to build more intelligent and robust fog network topologies to cope with the increasing requirements of IoT and edge applications.

REFERENCES

[1] S. E. Chafi, “Resource Placement Strategy Optimization for IoT,” TELKOMNIKA, vol. 21, no. 5, pp. 1109–1119, 2023.
[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog Computing and Its Role in the Internet of Things,” in Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, vol. 12, no. 1, pp. 13–16, 2012.
[3] S. Yi, C. Li, and Q. Li, “Survey of Fog Computing: Concepts, Applications and Issues,” Proceedings of the IEEE International Conference on Mobile Data Management (MDM), vol. 1, no. 1, pp. 12–15, 2015.
[4] H. Chouat, “Adaptive configuration of IoT applications in the fog infrastructure,” IEEE Transactions on Network and Service Management, vol. 20, no. 4, pp. 4333–4346, 2023
[5] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “Edge Computing: Vision and Challenges,” IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, 2016.
[6] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A Brief Survey of Deep Reinforcement Learning,” IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26–38, 2017.
[7] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, “Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption,” IEEE Internet of Things Journal, vol. 3, no. 6, pp. 1171–1181, 2016.
[8] Y. Sun, J. Zhang, C. Guo, and Y. Fang, “Predictive Resource Management for Fog Computing: Proactive Fault-Tolerant Approach,” IEEE Transactions on Network and Service Management, vol. 16, no. 4, pp. 1614–1625, 2019.
[9] H. Gupta, “Intelligent Resource Management in Fog Computing,” IEEE IoT Journal, vol. 7, no. 11, pp. 10711–10721, 2020.
[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “Latency Optimization for Resource Allocation in Mobile-Edge Computing Networks,” IEEE Transactions on Wireless Communications, vol. 17, no. 8, pp. 5502–5515, 2018.
[11] M. Laroui, “Edge and Fog Computing in IoT: Challenges and Future Directions,” Future Internet, vol. 13, no. 2, article 44, pp. 1-22, 2021.
[12] P. Zhan, “Security and trust issues in Fog Computing,” IEEE Security & Privacy, vol. 16, no. 4, pp. 14–21, 2018.
[13] P. Maiti, “Energy-Efficient Resource Management in Fog Computing,” IEEE Transactions on Sustainable Computing, vol. 5, no. 3, pp. 410–421, 2020.
[14] B. Costa, “AI-Driven Service Orchestration in Fog Computing,” Journal of Supercomputing, vol. 78, no. 6, pp. 8411–8435, 2022.

RESEARCH ARTICLE

[15] Z. Zhao, R. Xie, J. Li, and Y. Zhang, "Deep Reinforcement Learning for Resource Management in Network Slicing," IEEE Transactions on Network and Service Management, vol. 17, no. 4, pp. 2062–2076, 2020.
[16] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint Optimization of Radio and Computational Resources for Multicell Mobile-Edge Computing," IEEE Transactions on Signal and Information Processing over Networks, vol. 1, no. 2, pp. 89–103, 2015.
[17] R. Roman, J. Lopez, and M. Mambo, "Mobile Edge Computing, Fog et al.: A Survey and Analysis of Security Threats and Challenges," Future Generation Computer Systems, vol. 78, no. 1, pp. 680–698, 2018.
[18] A. M. Lone and R. B. Pachpor, "Security Challenges in Fog Computing: A Survey," Journal of King Saud University – Computer and Information Sciences, vol. 34, no. 7, pp. 5075–5087, 2022.
[19] J. Yakubu, "Security Challenges in Fog Computing," ACM Computing Surveys, vol. 54, no. 1, article 13, pp. 1–36, 2021.
[20] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog Computing for the Internet of Things: Security and Privacy Issues," IEEE Internet Computing, vol. 21, no. 2, pp. 34–42, 2017.
[21] M. Aazam, I. Khan, and E.-N. Huh, "Fog Computing-Based Smart Gateway for Cloud of Things," IEEE Consumer Electronics Magazine, vol. 7, no. 5, pp. 36–42, 2018.
[22] E. Mahdipor, "Load Balancing in Fog Networks," Future Internet, vol. 14, no. 2, article 60, pp. 1–19, 2022.
[23] R. Mahta, "Cloud, Fog or Edge: Where to Compute?" IEEE Internet Computing, vol. 25, no. 2, pp. 28–36, 2021.
[24] H. Patel, "A Survey on Cloud-Fog-Edge Continuum for IoT," ACM Computing Surveys, vol. 54, no. 7, article. 141, pp. 1–36,2021.
[25] S. K. Singh, "Resource Management Techniques in Fog Computing: A Survey,"IEEE Communications Surveys & Tutorials, vol. 22, no. 1, pp. 426–448, 2020.
[26] S. Jain, "Fog computing in enabling 5G-driven emerging technologies for development of sustainable smart city infrastructures," IEEE Wireless Communications, vol. 29, no. 1, pp. 80–87, 2022.
[27] Y. Zhang, L. Wang, and H. Chen, "Federated Learning-Based Intrusion Detection for IoT Networks: A Survey," IEEE Internet of Things Journal, vol. 9, no. 12, pp. 9037–9055, 2022.
[28] K. Lim, "Blockchain for Secure Fog Computing," IEEE Internet of Things Journal, vol. 8, no. 1, pp. 568–578, 2021.
[29] C. Springer, "Enabling IoT/M2M System Scalability with Fog Computing," IEEE Transactions on Cloud Computing, vol. 8, no. 3, pp. 843–856, 2020.
[30] H. Weng, "A Lightweight Anonymous Authentication and Secure Communication Scheme for Fog Computing Services," IEEE Transactions on Dependable and Secure Computing, vol. 18, no. 1, pp. 223–235, 2021.
[31] H. Chen, W. Xu, and X. Wang, "Industrial Internet of Things: A Survey on the Enabling Technologies, Applications, and Challenges," IEEE Communications Surveys & Tutorials, vol. 21, no. 2, pp. 1463–1493, 2019.

Authors

Nada M. Sallam received her bachelor's degree in 2012 from Mansoura university, Faculty of Engineering, Computers and Control Systems Engineering Department. Average Grade: Very Good with honors Last year Grade: Very Good Graduation Project: Embedded car security system Grade: Excellent, in 2013 Holds a Premaster in control department, Mansoura University, Faculty of Engineering, with a very good grade with honors. In 2015, she registered her Master in control engineering, Mansoura University, Faculty of Engineering. Holds a master's degree in control engineering to the thesis under the title " Proportional Integral Control for Electromechanical System Using Neural Network", Mansoura University Faculty of Engineering, Computers and Control Systems Engineering Department in 2019. In 2020, the University Council approved the registration for a Pre-PhD degree, and she successfully completed a Pre Ph.D. degree. In 2020, Accreditation of a qualifying exam result. In 2021, she registered for a Ph.D. in Computers and Control Systems Engineering Department, Mansoura University, Faculty of Engineering. In 2023, she holds a PhD degree in Computers and Control Systems Engineering Department to the thesis under the title " New Leukemia Diagnosis Strategy Using Machine Learning Techniques, Mansoura University Faculty of Engineering, Computers and Control Systems Engineering Department. Dr. nada is specialized in the field of computers and control systems engineering dept. she published many research papers in many international journals and attended many conferences inside and outside Egypt.

Dr. Samah Osman An Experienced Assistant Professor with a strong passion for teaching and a solid background in Computer Science. Committed to leveraging extensive education and expertise to create engaging, research-driven learning experiences for students. Dedicated to fostering a collaborative classroom environment that encourages critical thinking and innovation. Actively involved in curriculum development and mentorship, guiding students to explore their interests and achieve their academic goals. Adept at integrating modern technologies and methodologies to enhance educational outcomes, while also promoting interdisciplinary approaches that connect Computer Science with real-world applications.

How to cite this article:

Nada M. Sallam, Samah Osman, "Optimizing Fog Computing Performance Using Q-Learning: A Simulation-Based Study", International Journal of Computer Networks and Applications (IJCNA), 12(3), PP: 336-362, 2025, DOI: 10.22247/ijcna/2025/22.