International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2025/47 Volume 12, Issue 5, September – October (2025)

RESEARCH ARTICLE

A Multi Objective Performance Driven Weight Tuning Based Load-Balancing Mechanism in IoT–Fog–Cloud Environment

M. Parveen Taj
Department of Computer Science, PPG College of Arts and Science, Coimbatore, Tamil Nadu, India.
✉ tajroshan1phd@gmail.com

N. Muthumani
Department of Computer Science, PPG College of Arts and Science, Coimbatore, Tamil Nadu, India.
principalppgcas@ppg.edu.in

Received: 10 July 2025 / Revised: 24 September 2025 / Accepted: 05 October 2025 / Published: 30 October 2025

Abstract – An integrated Internet of Things (IoT), Fog, and Cloud computing (IoT-Fog-Cloud) has been very popular lately as a way to make networks more stable and efficient. However, if network resources are not managed well or Load Balancing (LB) is not done right, Quality of Service (QoS) can be less effective. More and more traffic on fog gateways slows things down and uses more energy, which is bad for real-time apps. This paper suggests a new Multi-Objective Performance-Driven Weight Tuning-Based LB (MOPDWTLB) method for IoT-Fog-Cloud networks to solve this problem. The main purpose is to use a Multi-Objective Optimization (MOO) technique to give weights to nodes and make sure that the load is spread out evenly among all nodes based on their performance. First, a three-layer structure is made up of IoT, fog, and cloud nodes. A data-driven profiling method is used to describe the performance parameters of several types of fog nodes, such as Throughput (TP), Energy Efficiency (EE), Responsiveness (RPN), and Cost Efficiency (CE). Then, a weighted round-robin strategy is used to improve these performance indicators based on how important they are and to give the best weights to nodes for effective LB. In addition, this plan can be used for the MOO by using a weighted sum method to find the best weights for policies with more than one goal. A coordinated distributed approach is also utilized to make the load balancer more reliable and speed up the network. Finally, the results of the simulation show that the MOPDWTLB is better than the standard LB schemes when it comes to TP, Latency, Response Time (RT), network lifetime, energy, memory, and bandwidth use.

Index Terms – IoT-Fog-Cloud, Heterogeneity, Quality of Service, Load Balancing, Weighted Round Robin, Multi-Objective Optimization.

1. INTRODUCTION

Recently, IoT technology has enabled various applications, including power grids, self-driving cars, medical diagnostics, industrial automation, smart agriculture, and connected homes [1]. The emergence of next-generation IoT applications necessitating edge device processing for rapid responses has heightened the focus on reducing execution time, latency, energy usage, and cost, while maximizing resource use [2]. Cloud computing has emerged as a predominant technology owing to its potential to provide extensive processing power and storage capabilities for managing the substantial data generated by IoT devices [3, 4]. Nonetheless, the traditional centralized cloud computing architecture has latency challenges that impede the effective execution of time-sensitive operations [5]. Fog computing manages data generated by IoT devices by using various data sources inside the local fog cluster or transmitting the data to cloud data centers for processing [6]. Consequently, the integration of fog and cloud computing models facilitates the use of both fog resources and cloud data center resources, providing a sustainable, efficient, and intelligent framework for IoT applications.

The rising prevalence of IoT applications has intensified attention to effective task allocation with LB in the IoT-Fog-Cloud paradigm. LB is crucial, facilitating equitable job allocation across Virtual Machines (VMs) to avert both overload and underutilization while sustaining consistent performance across data centers [7]. However, enhancing workload distribution to reduce execution time, latency, and energy consumption continues to be a formidable problem in this setting. Round robin, min-min, and max-min are LB algorithms that have been used to address this problem for decades [8]. These simple methods boost resource usage by processing incomplete requests in chronological order to the current host. Contrarily, these algorithms are solely suitable for homogeneous systems. Allocating equitable resources across fog and cloud nodes using these algorithms for IoT

RESEARCH ARTICLE

applications is a significant problem, particularly due to the loosely interconnected and extremely heterogeneous nature of fog/cloud devices.

Therefore, an effective IoT-Fog-Cloud architecture requires an improved resource allocation method that incorporates both fog and cloud characteristics. Recent studies are mostly focused on developing LB algorithms for heterogeneous IoT-Fog-Cloud network scenarios. The authors presented a performance-oriented weighted round-robin technique-based load balancing strategy in [9], which takes energy and cost considerations into account for heterogeneous serverless edge computing. Nonetheless, self-adaptive LB strategies are still necessary for various targets or fluctuating request arrivals from IoT devices. In [10], the authors devised a heuristic-based resource allocation strategy for heterogeneous fog computing systems, based on the load produced by groups of edge nodes. This approach also lacks essential aspects like energy efficiency, cost, deadlines, etc., which impact the optimal allocation decisions due to fluctuating task arrival patterns, execution durations, and QoS demands.

1.1. Problem Definition

Poor management of fog resources and LB can lower QoS and waste more energy. More traffic on fog nodes or gateways has caused longer delays, which are not acceptable for real-time applications. As more and more IoT components are employed, a system is needed to deal with the problems that come with more load, energy use, and delays caused by more data traffic and processing needs. On the other hand, the fact that fog nodes are different from each other in terms of hardware and software makes LB a big problem in these systems.

1.2. Main Contributions

This paper develops the MOPDWTLB scheme in IoT-Fog-Cloud systems. The main idea is to provide weights to nodes such that the load is evenly distributed according to how well each node performs. The main contribution of this study is listed here.

1. Initially, a 3-layer structure consisting of IoT (i.e., edge), fog, and cloud nodes is built.
2. Then, a data-driven profiling method is adopted to characterize the performance metrics of heterogeneous fog nodes, such as TP, EE, RPN, and CE. A weighted round-robin technique is applied to assign weights between nodes by optimizing these performance metrics according to their observed significance for LB.
3. Moreover, this weighted scheme is extended to the MOO, depending on the weighted sum approach to determine the appropriate weights for multi-objective policies, which concurrently optimizes multiple performance metrics. Also, a coordinated distributed method is presented to improve the load balancer's reliability and network efficiency.
4. Thus, by unifying these approaches, this study enables continuous functioning and optimal LB in the heterogeneous IoT-Fog-Cloud network.

The remaining sections are prepared as follows: Section 2 presents the related studies. Section 3 explains the MOPDWTLB scheme, and Section 4 illustrates its effectiveness. Section 5 summarizes the work and provides upcoming improvements.

2. LITERATURE SURVEY

Along with an LB method, Asghar et al. [11] suggested a fog-based health monitoring system with a three-tier structure. In tier 1, sensors and cellphones were used to detect and send the patient's data to the fog nodes. Once processed, the fog node in tier 2 sends a notification to the patient regarding their medical state and transmits the findings to a cloud server in tier 3 for storage. Moreover, an LB method was implemented to distribute incoming requests across fog nodes in an equitable manner based on the traffic load and computing load of all fog nodes. This method allowed the fog-based health monitoring system to efficiently handle a high volume of user requests. However, latency and network usage remained high, with an increase in requests.

Alqahtani et al. [12] presented the LB Service Scheduling Approach (LBSSA), a dependable scheduling strategy for allocating client requests to cloud-fog resources. They evaluated LB between resources while allocating requests to them, classifying them as real-time, critical, or time-sensitive. Additionally, request scheduling took into account the resource failure rate to ensure high dependability for requested services. This strategy can handle a variety of requests from IoT devices. However, the scheduling mechanism ignores the delay of request processing.

Singh [13] devised an excellent LB system named Fuzzy Golden Eagle LB (FGELB) for fog computing. It had three phases: work prioritization, resource sorting and scheduling, and power management. First, the incoming tasks were sorted based on deadline period, work volume, and specified priority using a fuzzy approach to ensure that the vital task was completed without latency. Following that, the Golden Eagle Optimization Algorithm (GEOA) was used to sort and distribute the resources. The sorted resources perform tasks in the resource engine, while the power engine manages energy by activating or deactivating resources based on their needs. However, this strategy did not account for TP and RT.

Javadpour et al. [14] developed an energy-efficient embedded LB that uses Dynamic Voltage and Frequency Scaling (DVFS) computing in cloud data centers. An approach was developed to prioritize activities based on completion time

RESEARCH ARTICLE

and classify Physical Machines (PMs) based on their arrangement state. It allocated tasks to PMs in the same priority classifications as the user. It lowered the power consumption of computers doing low-priority jobs using the DVFS approach. It also offered job migration to guarantee workload balance and adapt to changes in machine classes depending on their ratings. However, it was computationally difficult.

Yakubu and Murali [15] created a layer fit strategy that consistently distributes work between fog and cloud depending on their priorities, preventing the fog layer from being oversaturated with delay-tolerant activities. A Modified Harris-Hawks Optimization (MHHO) algorithm was applied to determine the best available resource for a job within a tier based on the completion period, execution cost, and energy use. In this approach, a unique energy update function and LB mechanism were used to keep the HHO population from slipping into a local optimum and to prevent resource overload in the environment. However, it incurred high energy usage and overall execution costs.

Singh et al. [16] proposed an IoT task allocation strategy to reduce energy usage during uploading, downloading, and processing. A unique modified version of the JAYA metaheuristic scheme, entitled MAYA, was given to address the energy-efficient workload allocation of sensors in the fog-cloud system based on request deadlines, transmission energy consumption, and computing requirements. In contrast, resource utilization was not taken into account, which influences network scalability when increasing the number of jobs or fog nodes.

Ibrahim et al. [17] suggested a Delay-Aware and LB called DALBFog allocation system for deadline-constrained IoT applications in fog computing. This system sought to schedule clients' delay-sensitive IoT activities in such a manner that it decreases delay, increases job acceptance, reduces load imbalance, and improves resource utilization of fog resources with reduced average response times. However, it did not account for energy consumption, which has an influence on scalability in large-scale networks with a high number of tasks.

Shamsa et al. [18] introduced the LB strategy for IoT/fog/cloud systems, which incorporates local schedulers based on workload prediction and the existence of volatile mobile nodes as dynamic resources. The network environment was first segmented into a grid of equal-sized cells. The entire state of intra-cell resources, such as overloaded, underloaded, or normal, was then assessed using workload prediction and available adaptive resources. Furthermore, an intensive search was conducted to shift extra tasks from overcrowded cells to underloaded ones, hence avoiding workflow deadlines and boosting system efficiency. However, energy usage and delay were not assessed.

Mahapatra et al. [19] created a fuzzy logic scheme to determine desired levels of offloading depending on resource heterogeneity and network demands, such as network bandwidth, task size, resource usage, and latency sensitivity. They employed a Binary Linear-Weight JAYA (BLWJAYA) work allocation system to route accepted IoT queries to compute-intensive fog nodes or VMs. This technique was scalable in terms of managing data unpredictability and latency as a result of the job offloading criteria inside the fog layer. However, they failed to address task interdependence, resulting in higher latency and energy usage.

Shaik et al. [20] introduced a hybrid Particle Swarm Optimization (PSO) with Simulated Annealing (SA) strategy called the PSOSA-LB algorithm for fog-cloud networks. The goal was to reduce execution period, delay, and energy usage while ensuring a balanced workload allocation. The PSO was used to drive solution space exploration, while SA was used to prevent premature convergence and escape local optima. A load imbalance adjustment factor was added to the PSO velocity update to ensure LB. However, the system must constantly monitor fog and cloud resources, requiring increased communication and energy use, which may be challenging in areas with limited bandwidth or large delays between fog and cloud levels.

Alaanzy et al. [21] developed an Optimized LB (OLB) method to efficiently distribute workloads among fog nodes and eliminate communication and processing delays. They used a three-tier architecture, which included sensor data collection (patient-worn sensors transmit vital signs), a fog layer for real-time analysis near base stations, and cloud storage and user access via mobile devices. The key advantage was its capacity to bring processing resources closer to consumers, such as patients, allowing for more efficient handling of large volumes of sensor data. However, it has a high energy usage and execution time.

To categorize the requests and identify the target layers for processing in fog-cloud systems, Srichandan et al. [22] introduced the Adaptive Neuro-Fuzzy Inference System (ANFIS). Additionally, such requests were scheduled at the target layer using a Chaotic Honey Badger Algorithm (CHBA). To improve the convergence of HBA, an Opposition-based Learning (OBL) scheme was used with a chaotic mapping function. However, it has high resource and energy usage due to load imbalance among nodes.

In fog-cloud circumstances, Bharathi et al. [23] developed the One-to-One-based optimizer with Priority and LB (O2O-PLB) method to improve resource allocation. Task assignment was established using a priority-driven resource allocation method. It guaranteed that high-priority jobs were assigned the necessary resources without delay, improving system responsiveness and lowering latency. The LB mechanism was implemented into the resource allocation

RESEARCH ARTICLE

method to ensure that jobs were distributed fairly among fog and cloud resources. This prevented the overloading of any one resource, thereby optimizing resource use and enhancing overall system stability. By improving resource allocation across both paradigms, scalability was increased in distributed IoT systems. However, the response time and latency were also high. Rateb et al. [24] examined the workflow scheduling problem of IoT devices inside the fog-cloud environment. A combination of Aquila and Salp Swarm Algorithms (ASSA) was used to choose the optimal VMs. The Reducing MakeSpan Time (RMST) approach was then used to decrease the makespan of the process on selected VMs. Furthermore, by using VM merging and the Dynamic Voltage Frequency Scaling (DVFS) technique on the output from RMST, energy usage was reduced. Nonetheless, bandwidth and memory use were adversely affected by uneven load circumstances.

Khaledian et al. [25] introduced the Hybrid Markov Chain-Based Dynamic Scheduling (HMCDS) framework for effective job scheduling in fog-cloud situations. This HMCDS technique took advantage of the combined resources of the fog and cloud layers, using Markov chain-based load prediction to forecast resource needs and the Arithmetic Optimization Algorithm (AOA) for optimal task-to-resource mapping. This integrated method aimed to enhance the efficiency of resource allocation, minimize job completion delays, and extend the overall makespan. However, it has a high computational cost, which affects scalability, resulting in longer RT and execution times in large-scale IoT deployments with a large number of tasks and fog nodes. Table 1 summarizes the above-studied algorithms according to their benefits, drawbacks, and performance metrics.

Table 1 Summary of Current LB Algorithms in IoT-Fog-Cloud Systems

Ref. No.	Methods	Benefits	Drawbacks	Performance metrics
[11]	LB scheme	It can choose the suitable fog node to schedule incoming requests to balance the load among fog nodes.	Latency and network usage increased with a higher number of requests.	Latency = 12.81 milliseconds (ms); Network usage = 341668 kilobytes (kB)
[12]	LBSSA	It achieved the lowest running time and provided highly reliable services.	The scheduling method does not address the latency of request processing.	No. of requests = 2000: No. of used computing resources = 75; Average resource utilization = 93%; Running period = 59 seconds (s)
[13]	FGELB	It reduced the failure rate and waiting time.	This method did not take into account throughput and response time.	No. of fog resource = 2000: Computational cost = $6.2x10^4$ $; No. of fog nodes = 20: Communication overhead = $2.2x10^4$ bits; Energy consumption = 1.5 millijoules (mJ)
[14]	DVFS	It decreases energy consumption significantly.	It has a computational complexity.	Time = 900 sec: Power = 34 kilowatts per hour (kW/h)
[15]	MHHO	It achieved a lower makespan.	Energy consumption and average execution costs were high.	No. of VMs = 20: Execution cost = 18x100000; Average makespan time = 4.4; Energy consumption = 70000
[16]	MAYA	It effectively reduced energy consumption and computation time.	Resource utilization was not taken into account, which influences network scalability when increasing the quantity of jobs or fog nodes.	Energy consumption reduction = 93.84%; Computation time reduction = 91.08%

RESEARCH ARTICLE

[17]	DALBFog	It reduces the delay and load imbalance significantly.	It did not account for energy consumption, which has an influence on scalability in large-scale networks with a high number of tasks.	-
[18]	Load scheduler with exhaustive search	It balances the workload efficiently and enhances the job completion rate.	Delay and energy usage were not analyzed, which impacts the network QoS efficiency.	-
[19]	Fuzzy logic and BLWJAYA	It effectively addressed the unpredictability of data and the latency owing to the task offloading condition.	They did not consider task dependencies, leading to increased latency and energy consumption.	Resource utilization = 96.29; Latency = 164.73; Energy consumption = 262.82; LB rate = 261.83
[20]	PSOSA-LB	It provided a robust and scalable solution for dynamic resource management.	It requires continuous monitoring of fog and cloud resource statuses, leading to increased communication overhead and energy consumption.	No. of tasks = 1000: Execution time = 35 s; Energy consumption = 1220 kilojoules (kJ); Latency = 85 ms; Load imbalance = 11%
[21]	OLB	It achieved better adaptability and resource efficiency.	It has high energy consumption and execution time.	Latency = 70.27 ms; Network usage = 332.27 Megabytes (MB); Energy consumption = $25x10^5$
[22]	ANFIS-CHBA-OBL	Secure and distributed QoS-aware scheduling in fog systems.	Scheduling requests on available nodes requires predicting the loads of each computing node, which is a difficult task.	Latency rate = 133 ms; Utilization = 59%; Service delay = 1.00E+05 ms; Energy consumption = 98 kJ
[23]	O2O-PLB	It achieved a lower response time and latency.	Task failure rate remained high.	No. of tasks = 1000: Response time = 76 ms; Latency = 60 ms; Load imbalance = 0.5; Task failure rate = 3.8%
[24]	ASSA	It reduced total energy usage while maintaining reliability and deadline demands.	Bandwidth and memory use were adversely affected by uneven load circumstances.	Energy consumption = 28%;
[25]	HMCDS	It has high scalability and adaptability.	It has a high computational complexity, which influences scalability, such as higher RT and execution times in large-scale IoT installations with a large number of jobs and fog nodes.	Delay = 68.52 s;

RESEARCH ARTICLE

2.1. Research Gap

The literature analysis reveals that existing algorithms for IoT-Fog-Cloud contexts struggle with latency, throughput, execution time, and energy consumption, limiting their efficacy in real-time applications. Furthermore, these algorithms fail to optimally balance throughput and resource consumption among several fog nodes, resulting in poor performance. Multiple QoS parameters, such as response time, throughput, cost efficiency, and network latency, must all be optimized simultaneously. To address these deficiencies, this work offers the MOPDWTLB method, which is optimized for IoT-Fog-Cloud scenarios.

In contrast to prior LB algorithms, which often concentrate on single-objective optimization, the proposed MOPDWTLB scheme optimizes multiple QoS measures, including TP, EE, RT, and CE, concurrently. This scheme relies on a three-layer IoT-Fog-Cloud structure, in which fog nodes are empirically profiled to quantify performance metrics under various scenarios, followed by dynamic weight tuning using a weighted sum MOO scheme and a weighted round-robin strategy, and integrated with a coordinated distributed LB mechanism to improve reliability and efficiency. As a result of these techniques, the MOPDWTLB scheme is more adaptable to real-time, heterogeneous IoT applications than previous systems.

3. PROPOSED METHODOLOGY

This section describes the MOPDWTLB scheme for the IoT-Fog-Cloud network in detail.

3.1. Network Model

This study considers the following concepts when developing and deploying the MOPDWTLB for the dynamic IoT-Fog-Cloud network:

- Fog nodes are given precedence over cloud nodes. Data flows are directed to cloud nodes only when fog devices or gateways are overloaded.
- Cloud nodes are operated if fog nodes are overloaded or low on energy.
- Energy saving is achieved by deactivating some fog nodes or gateways while the computational demand is low.

Consider an IoT-Fog-Cloud network architecture shown in Figure 1, which consists of a central cloud node, K edge nodes, and N fog nodes. All edge nodes produce multiple task requests. The traffic patterns include M/M/1, M/M/c, and M/M/∞ queues at the edge, fog, and cloud levels, respectively.

Edge nodes use a wireless channel to send requests to the fog nodes. The fog nodes can handle the received requests if their maximum acceptance rate exceeds the request rate. Or else, the fog nodes can forward the traffic to the central cloud for processing.

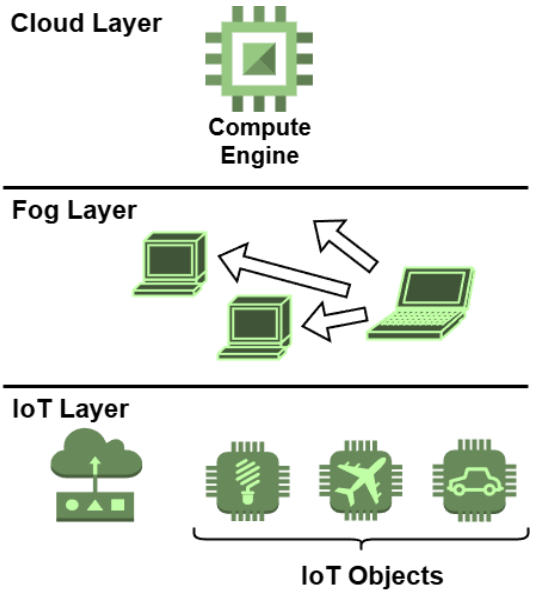

Figure 1 Architecture of IoT-Fog-Cloud Network

3.2. Problem Formulation

In the heterogeneous IoT-Fog-Cloud scenario, a major challenge is to balance the workload among nodes. Assume a collection of fog nodes denoted as F, with each node labeled as f_j. The performance of f_j is represented as $P_{j,p}$ for a particular performance metric m_p. This m_p is a fragment of a massive collection M that includes metrics like TP, latency, etc. The goal is to find the optimal weight $w_{j,p}$ for f_j based on a specific m_p to maximize overall performance, according to m_p. This optimization problem is modeled to consider the computational abilities and resource constraints of fog nodes, which is expressed in equations (1) and (2).

$$\max \sum_{j=1}^{|F|} w_{j,p} \times P_{j,p} \quad (1)$$

$$\text{Subject to } \sum_{j=1}^{|F|} w_{j,p} = 1, w_{j,p} \geq 0, \forall j \in F \quad (2)$$

This is especially true for heterogeneous IoT-Fog-Cloud situations, where the assumptions of linear combination and independence of node performances are essential to these equations. This feature represents relative strengths in ensuring that nodes are not overloaded and that all nodes get an equitable share of the workload.

3.3. Proposed Multi-Objective Performance-Driven Weight Tuning-Based Load Balancing (MOPDWTLB) Scheme

The MOPDWTLB scheme addresses the LB problem by adjusting weights based on performance metrics. It involves characterizing performance metrics for fog nodes to quantify their performance and utilizes a weighted round-robin

RESEARCH ARTICLE

scheduling scheme to meet performance objectives in real time. By using the MOO, it can handle multiple performance metrics and optimize them simultaneously. Moreover, it includes a coordinated distributed method tailored to the serverless edge environment to boost the reliability and efficiency of the network. MOPDWTLB provides better load balancing in fog and cloud as it utilized multiple parameters with multi objective function to provide the weight of the nodes for assigning incoming tasks arrive frequently.

3.3.1. Determination of Performance Metrics

3.3.1.1. Response Time

Data transfer rate $\left(C_{i,j}(t)\right)$ between i^{th} edge node and j^{th} fog node at time t is defined by Shannon's equation as follows:

$$C_{i,j}(t) = B \log_2 \left(1 + \frac{\mathcal{P}_i^c(t) G_{i,j}(t)}{B \times N_0}\right) \quad (3)$$

In equation (3), $G_{i,j}(t)$ refers to the channel gain, $\mathcal{P}_i^c$ is the communication power of the edge node i at time t, B denotes the channel bandwidth, and N_0 is the noise power spectral density. It estimates transmission efficiency by emphasizing how power and bandwidth impact transfer speeds. Utilizing the M/M/1 model, the transmission delay $\left(TD_j\right)$, such as transmission and queuing delays, for j^{th} fog node is determined by equation (4). This study chooses the M/M/1 queuing model because of its applicability for single-server networks with Poisson task arrivals, signifying the varying request patterns in fog nodes.

$$TD_j = \frac{1}{\frac{C_{i,j}(t)}{\rho_i(t) S_i(t)} - \rho_j(t) \varphi_j(t)} \quad (4)$$

In equation (4), $\rho_i(t)$ and $\rho_j(t)$ are the offloading percentages for the edge node i and fog node j at t, respectively, S_i represents the data size of i^{th} job, and $\varphi_j(t)$ is the task arrival rate at j^{th} fog node at t. These parameters are primarily utilized to obtain the fraction of tasks offloaded and the volume of data processed, which may differ according to the node's capacity and network load.

Consider j^{th} fog node with a traffic rate t_j and service rate γ_j, the computation latency $\left(CL_j\right)$ is calculated by equation (5).

$$CL_j = \frac{1}{\gamma_j - t_j} \quad (5)$$

Equation (5) assumes that the queues are stable and measures the processing delays in fog nodes, which affects the total response time in real-time situations. Therefore, the RT is calculated by equation (6) based on the assumption that delays are additive and independent in heterogeneous network scenarios.

$$RT = TD_j + CL_j \quad (6)$$

3.3.1.2. Energy Utilization

The energy spent for each request by the j^{th} fog node at t, i.e., $E_j(t)$ is determined by equation (7).

$$E_j(t) = CL_j \times \upsilon \times \left(F_j(t)\right)^3 \quad (7)$$

In equation (7), υ is a constant, $F_j(t)$ is the CPU rate of the j^{th} fog node at t. Here, the constant value defines the resource's energy characteristics, while the CPU rate defines the processing ability.

3.3.1.3. Throughput (TP)

It determines the number of requests or tasks processed by the fog nodes per unit interval. It is utilized to quantify the processing capacity of fog nodes and is computed by equation (8).

$$TP = \frac{No.of\ tasks}{Time} \quad (8)$$

3.3.1.4. Cost Efficiency

It defines the amount of memory required by a task to execute by fog nodes in a given period. This metric guarantees effective resource distribution and decreases memory overhead in fog nodes.

3.3.2. Performance Metrics Characterization

Consider $M = \left\{TP, \frac{energy}{request}, RT, CE, CPU\ memory\right\}$ is a collection of performance metrics, where $m_p \in M$ has a measurement unit (U). $U = \left\{\frac{request}{s}, \frac{kJ}{request}, s, memory - s, MB\right\}$, where $u_p \in U$ defines the unit of $m_p \in M$, e.g., the energy/request is determined in kJ/request. The goal is to assess the performance metrics of fog nodes in F based on various measures in M. To do this, individual nodes are tested in isolation mode through profiling tests. Each node runs the same application under various runtime scenarios, i.e., heterogeneous conditions. In this mode, a node creates and transfers requests to a job positioned on a similar node. One request is synchronously made after another, waiting for a response before sending the next. Profiling is carried out for an adequate duration and continues for several iterations to ensure accurate results.

The outcomes from the profiling tests are represented by R, where $r_{j,p} \in R$ indicates the outcome for a certain $m_p \in M$ and a node $f_j \in F$, such as TP, etc. It is crucial to observe that $|R| = |M| \times |F|$. Some metrics, like TP, are better with larger values, while others, like RT, are better with smaller values. Initially, the result $r_{j,p}$ is converted into a metric value $c_{j,p}$ to ensure that a higher $c_{j,p}$ is often preferable for $m_p \in M$. This conversion is determined by a ranked MOO goal O, which defines whether a conversion is needed for the chosen

RESEARCH ARTICLE

measures $O = \{0,1,1,1,1,0\}$. This means that no conversion is needed for TP, yet for another measure where lesser values are preferred, a conversion is necessary.

Once the conversion is completed, the picked collection of measures is called {TP, EE, RT, CE, and CPU memory}. For each $r_{j,p} \in R$, R and O are used to determine a converted value $c_{j,p}$ as given in equation (9). This normalization equation makes sure that all measures are maximized, which helps with fair weight assignment in multi-objective optimization for heterogeneous fog nodes, and also scales raw results to a similar range, assuming linear scaling is enough for relative comparisons.

$$c_{j,p} = \begin{cases} \frac{1}{r_{j,p}}, & if\ o_p = 1 \\ r_{j,p}, & or\ else \end{cases} \quad (9)$$

3.3.3. Weight Adjustment

Once $c_{j,p}$ values are obtained, the next step is to convert them into load balancer weights. The aim is to allocate a weight to all nodes based on their effectiveness relative to every other node for a specific m_p. Consider W is the collection of computed weights: $W = \{w_{j,p}, \forall f_j \in F, \forall m_p \in M\}$. For m_p, $w_{j,p} \in W, \forall f_j \in F$ is calculated in equation (10).

$$w_{j,p} = \frac{c_{j,p}}{\sum_{j=1}^{|F|} c_{j,p}} \quad (10)$$

Equation (10) ensures that the weight assigned to f_j is calculated using the percentage of certain m_p value to the sum of all metric gauge values for that m_p. This means that the sum of weights for all devices under a particular metric will always be equal to 1. In other words, $\sum_{j=1}^{|F|} w_{j,p} = 1$. It improves fairness by allocating greater tasks to nodes with higher performance and aids LB by dynamically representing node abilities in IoT-Fog-Cloud heterogeneity.

3.3.4. Multi-Objective Load Balancing

The weight adjustment method explained in the above section is intended to allocate weights to maximize a single metric, such as TP. However, in real-time scenarios, it is essential to optimize several metrics concurrently, such as EE and TP. Additionally, it may be necessary to address the minor impacts of the desired metric. For instance, if the target application requires a fast RT but other applications running on the nodes impact energy consumption, multi-objective strategies are employed to manage this coexistence.

The suggested algorithm for MOO relies on the weighted sum approach, in which the multi-objective dilemma is simplified into a unidimensional dilemma by distributing weights to all metrics. These weights define the significance of all metrics. In this framework, a novel collection of weights is adopted for m_p and the weighted sum of $w_{j,p}$ values are calculated for f_j across every m_p in the optimization.

Consider N is a subcollection of M containing metrics $N \subseteq M$, such as TP, and $\psi_k \in \Psi$ is the multi-objective weight for all metrics $n_k \in N$. The weight ψ_k is determined according to the user's choice for n_k. The function $fn(j,k)$ assigns $w_{j,p}$ computed in equation (10) for $n_k = m_p$ for $f_j \in F$. A novel multi-objective weight s_j for f_j is determined by equation (11).

$$s_j = \sum_{k=1}^{|N|} \psi_k \times fn(j,k) \quad (11)$$

It is important to observe that $\psi_k \geq 0$ for every k, guaranteeing non-negativity of the multi-objective weights. Additionally, $\sum_{k=1}^{|N|} \psi_k = 1$, ensuring that the weights add up to 1. With this weighted sum, various metrics are combined into one score, allowing for simultaneous optimization in dynamic IoT-Fog-Cloud environments.

3.3.5. Weighted Round-Robin Strategy

To meet the criteria of the LB, a central load balancer is utilized. It employs the Earliest Deadline First (EDS) allocation scheme to implement a round-robin technique, as described in equation (12).

Upon receiving a fresh request, the load balancer chooses f_j with the earliest estimated deadline and forwards the request to that node.

$$fog\ node, f_j = \underset{f_j \in F}{\operatorname{argmin}}\ deadline(f_j) \quad (12)$$

The deadline for each f_j is calculated by equation (13).

$$\text{Where } deadline = \frac{CT_{f_j} + 1.0}{w_{f_j}} \quad (13)$$

In equation (13), CT_{f_j} is the current time of f_j and w_{f_j} is the weight of f_j. The w_{f_j} is statistically assigned to each f_j according to the obtained values by performance characterization in equation (10). The CT_{f_j} begins at 0 and increases by 1 with all requests' allocation to the fog node. Equations (12) and (13) modify round-robin scheduling with weights to improve fairness and reduce variance in IoT-Fog-Cloud systems, resulting in better load distribution among nodes.

3.3.6. Coordinated Load Balancing Using Optimal Weight Values of Fog Nodes

In a centralized LB framework, received requests are sent to a centralized gateway, as illustrated in Figure 2, which allocates them to various jobs installed between nodes. Conversely, this setup may face difficulties like node failure, scalability challenges, and high delay in large-scale systems.

RESEARCH ARTICLE

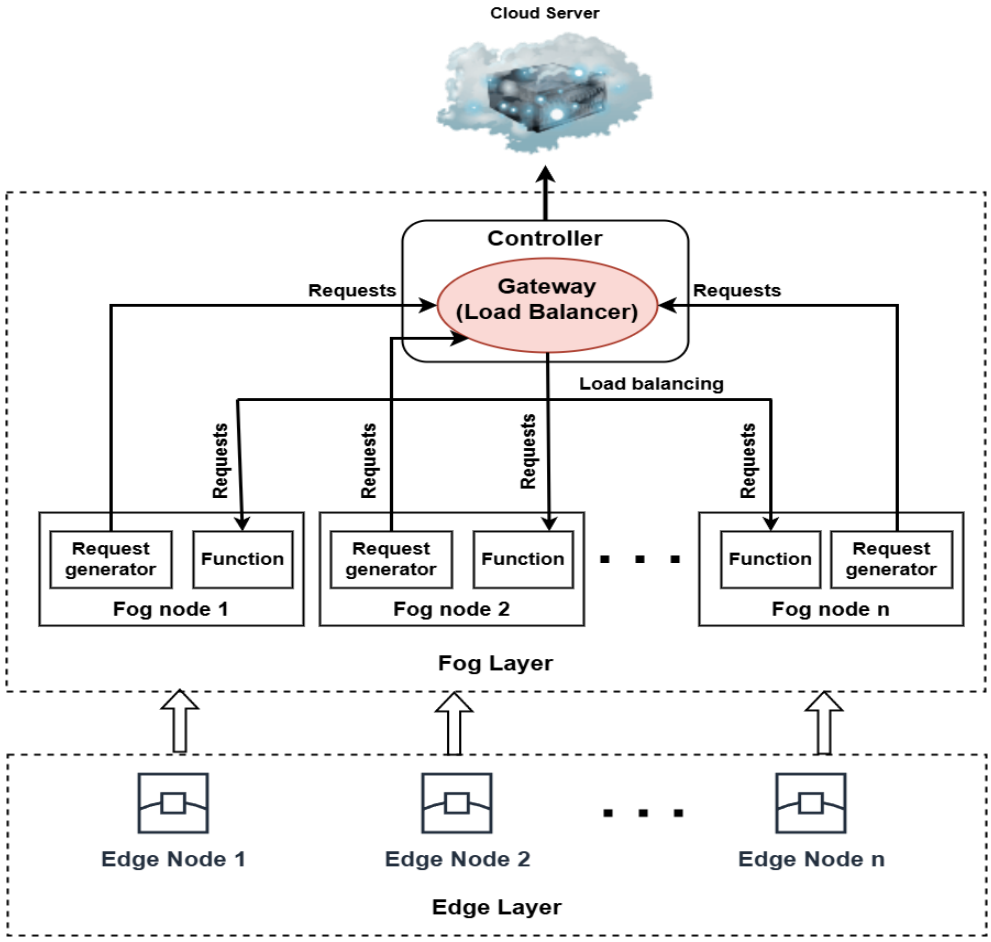

Figure 2 Centralized Load Balancing Framework

To address these issues, a coordinated distributed LB approach is implemented, as depicted in Figure 3, where all nodes have an LB agent responsible for redirecting requests to jobs on fog nodes based on a strategy determined by the central coordinator. This design aims to eliminate the need to route all traffic via a central gateway, improving system efficiency and reliability in large-scale networks. The central coordinator can manage rule dissemination to fog nodes, enabling the transition to an adaptive load balancer when needed. The entire processes in this study are summarized in Algorithm 1.

Input: IoT-Fog-Cloud nodes

Output: Optimized load distribution among fog and cloud devices

1. Begin
2. Construct a 3-level network;
3. for(each fog node)
4. Determine TP, EE, RT, and CE using equations (3) to (8);
5. Normalize the calculated values of each metric;
6. Determine the weight value for each metric;
7. Compute the final weight value for the selected metrics using a weighted sum approach;
8. end for
9. Initialize request queue;
10. while(request queue is not empty)
11. Choose a fog node with the highest weight value;
12. Send a request to the selected node and update its weight dynamically based on load;
13. If the fog node is overloaded, then forward the request to the cloud;
14. Apply coordinated LB mechanism;
15. if(centralized LB has low scalability or single point failure)
16. Deploy distributed LB agents at each fog node;

RESEARCH ARTICLE

17. Each agent redirects requests locally based on the weights;
18. Update weights dynamically according to real-time monitoring;
19. else
20. Use a centralized LB approach;
21. end if
22. end while
23. End

Algorithm 1 MOPDWTLB for IoT-Fog-Cloud Networks

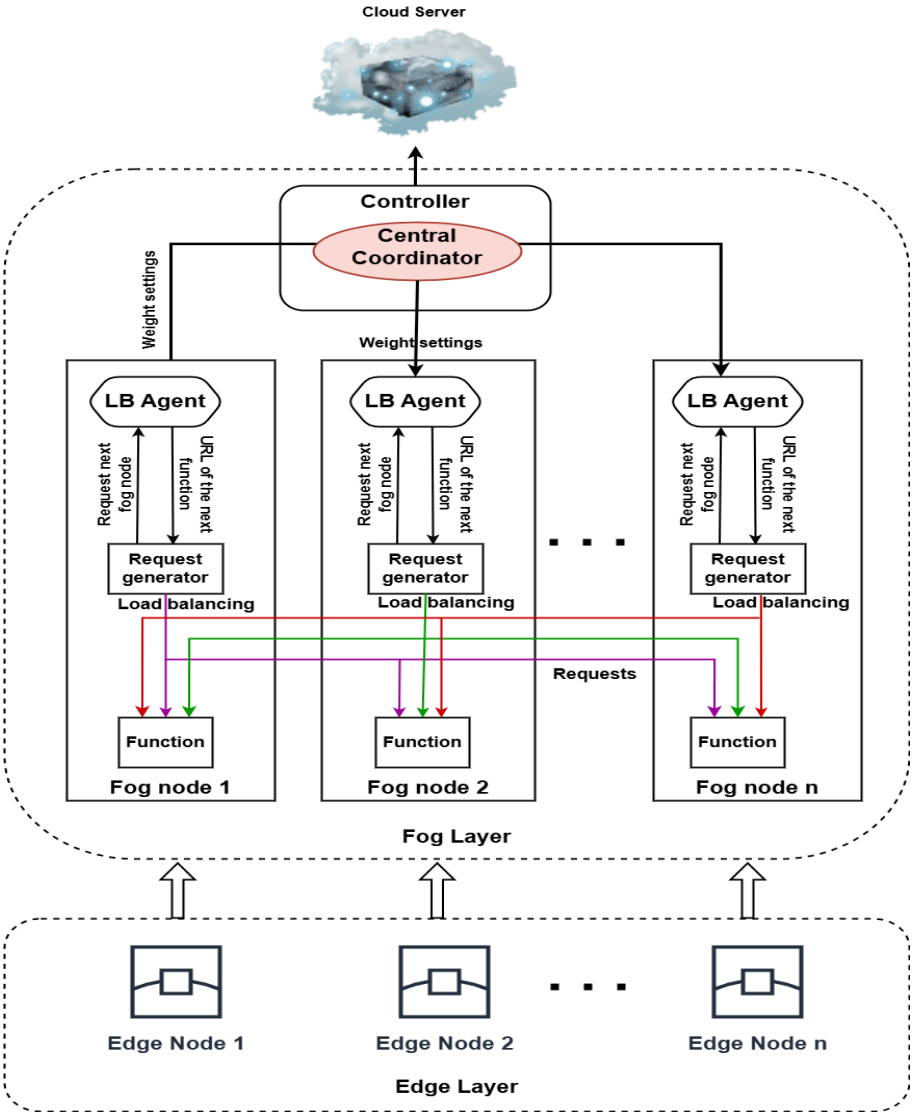

Figure 3 Coordinated Distributed Load Balancing Using MOPDWTLB Scheme

4. SIMULATION RESULTS

This section evaluates the effectiveness of the MOPDWTLB scheme compared to the existing LB algorithms, including FGELB [13], DALBFog [17], PSOSA-LB [20], OLB [21], and O2O-LB [23].

Performance metrics considered for evaluation are TP, average delay, RT, network lifetime, energy utilization, memory usage, and bandwidth utilization.

4.1. Simulation Setup

The experiments were conducted using Intel processors and 16 GB of RAM on a Windows operating system. The simulated Core i5-4210 CPU duration was assigned to 1000 seconds to accommodate load changes and recognize idle fog gateways and overload scenarios. In this study, 20 nodes were utilized, each producing data at a rate of 10 tasks/s, providing a sum of 200 tasks/s. Also, each node transferred data at a rate

RESEARCH ARTICLE

of 2 MB/s, resulting in a total data transfer rate of 40 MB/s across all nodes. To ensure reliable and practical results, the simulation was repeated 1200 times with varying parameters. In Table 2, the simulation parameters and their corresponding values are presented.

Table 2 Simulation Parameters

Parameter	Values
No. of end nodes	50-100
No. of fog nodes	10-100
No. of gateway devices	2
CPU rate in cloud	100×10^9 cycles/s
Average number of tasks	200 tasks/s
Max. channel bandwidth	30 Mega Hertz (MHz)
No. of cloud nodes	5
CPU rate in IoT nodes	500×10^6 cycles/s
CPU rate in fog nodes	50×10^9 cycles/s
Processing power utilization	0.5 Joule (J)
Transfer energy of IoT nodes	0.2 milliwatts (mW)

4.2. Throughput (TP)

In Figure 4, a comparison of average TP for different LB algorithms with varying numbers of tasks is plotted. The MOPDWTLB outperforms FGELB, DALBFog, PSOSA-LB, OLB, and O2O-LB in terms of TP efficiency for each quantity of tasks. For instance, with 1000 tasks, the MOPDWTLB increases TP by 29.73%, 8.52%, 12.94%, 9.09%, and 3.23% compared to the existing algorithms, respectively. This improvement in TP is because of adopting a weighted load distribution that effectively distributes workloads or tasks to fog nodes based on different performance metrics, in which TP is considered as one of the metrics. As well, the architectural adjustment with distributed LB prevents congestion and minimizes latencies, leading to higher request processing rates across different task counts.

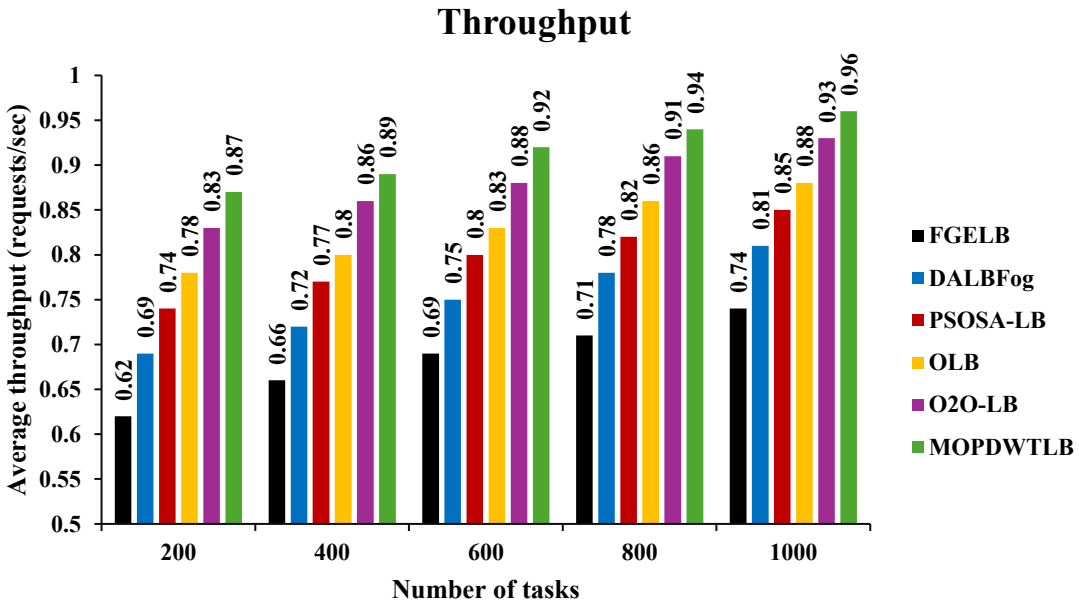

Figure 4 Throughput vs. No. of Tasks

4.3. Delay

The delay in IoT-Fog-Cloud operation is a combination of computational and transmission delays. It is calculated by equation (14).

$$Average\ delay = \frac{TD_j + CL_j}{Total\ number\ of\ nodes} \quad (14)$$

Figure 5 illustrates a comparison of the average delay for different LB algorithms with varying numbers of tasks. The MOPDWTLB reduces the delay compared to the FGELB, DALBFog, PSOSA-LB, OLB, and O2O-LB algorithms for each quantity of tasks. For example, with 1000 tasks, the MOPDWTLB minimizes the average delay by 53.64%, 40%, 35.44%, 25%, and 15% compared to the existing algorithms, respectively. This reduction is achieved by applying the

RESEARCH ARTICLE

weighted round-robin scheduling that dynamically allocates workloads to the least-loaded nodes, resulting in lower queuing and processing delays compared to the previous algorithms.

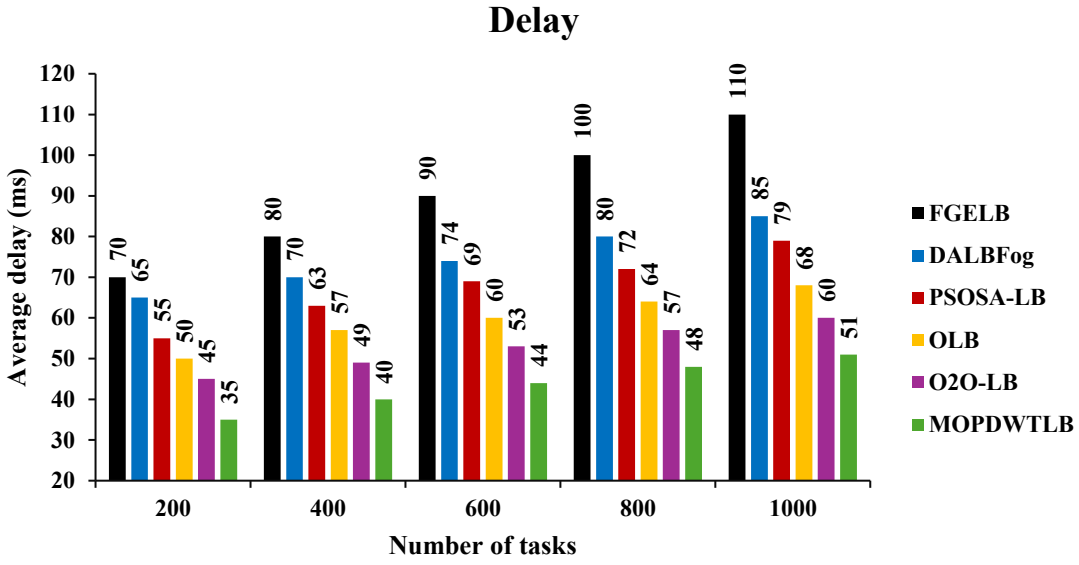

Figure 5 Delay vs. No. of Tasks

4.4. Energy Utilization

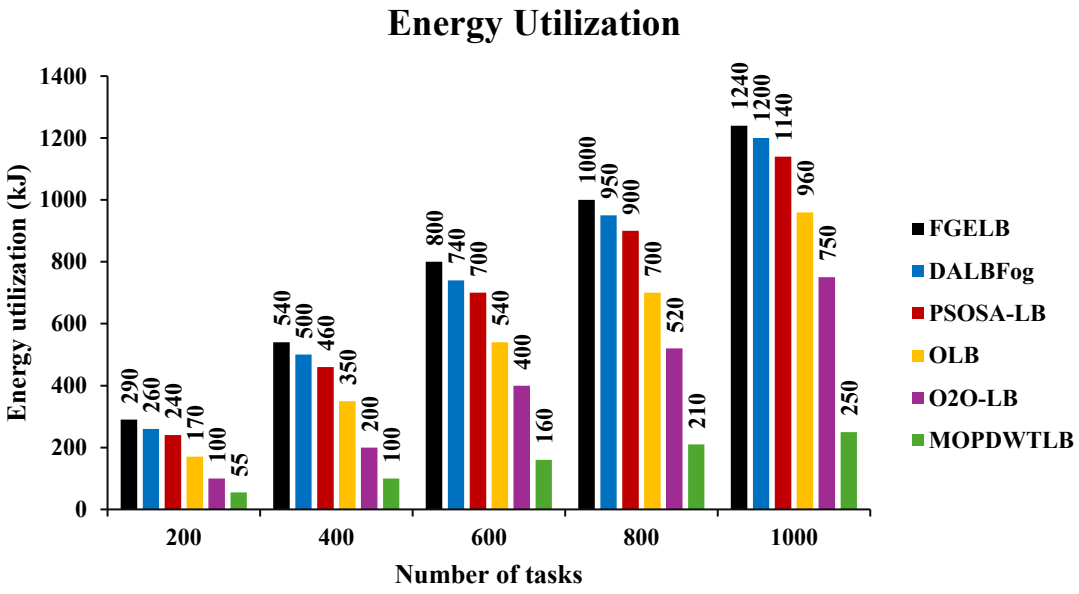

Figure 6 Energy Utilization vs. No. of Tasks

It refers to the total power spent by the IoT-Fog-Cloud system. Figure 6 shows a comparison of energy utilization for different LB algorithms with varying numbers of tasks. The MOPDWTLB decreases the energy utilization compared to the FGELB, DALBFog, PSOSA-LB, OLB, and O2O-LB algorithms for each quantity of tasks. For example, with 1000 tasks, the MOPDWTLB decreases the energy utilization by 79.84%, 79.17%, 78.07%, 73.96%, and 66.67% compared to other algorithms, respectively. This is achieved by optimizing load distribution based on EE as one of the key objectives and

RESEARCH ARTICLE

utilizing the distributed LB scheme, which helps in avoiding excessive energy consumption compared to the earlier algorithms.

4.5. Response Time (RT)

In Figure 7, a comparison of RT for different LB algorithms with varying numbers of tasks is shown. The MOPDWTLB reduces the RT compared to the FGELB, DALBFog, PSOSA-LB, OLB, and O2O-LB for each quantity of tasks. For example, with 1000 tasks, the RT of MOPDWTLB is decreased by 30.29%, 24.69%, 17.57%, 12.23%, and 6.15% compared to the existing algorithms, respectively. This reduction is achieved by employing multi-objective weight adjustment and data-driven profiling, which allocates tasks to nodes based on optimal weights with lower processing delays.

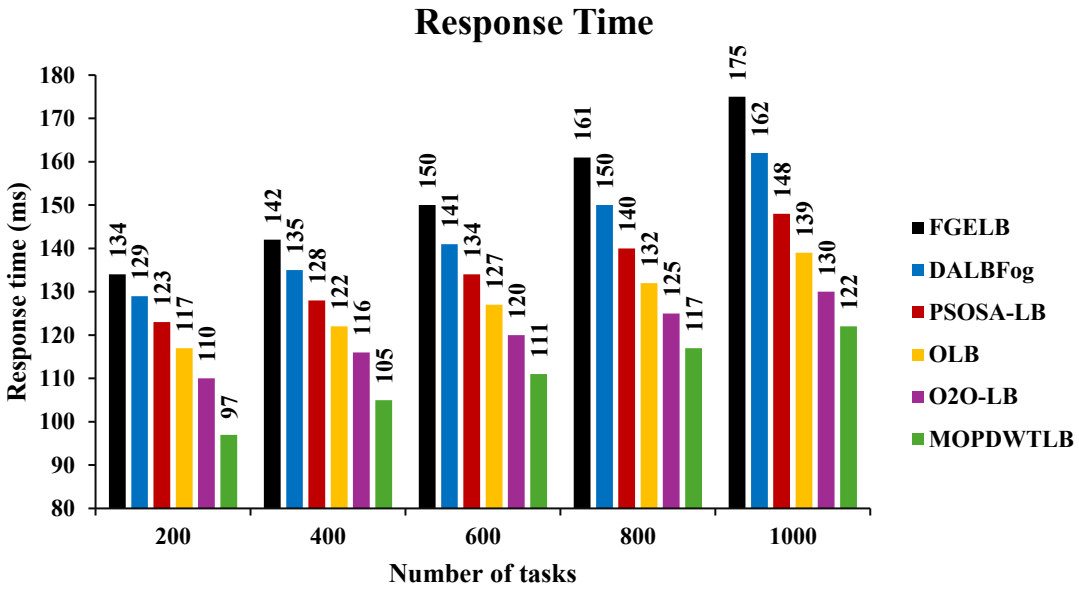

Figure 7 Response Time vs. No. of Tasks

4.6. Memory Utilization

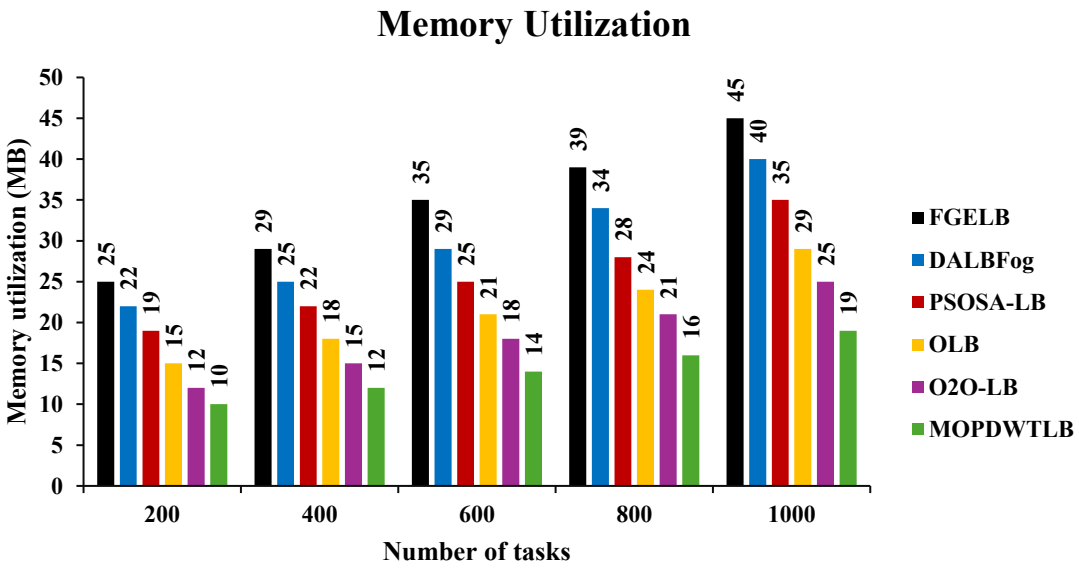

Figure 8 Memory Utilization vs. No. of Tasks

RESEARCH ARTICLE

It is the maximum memory requirement of each VM for task execution. In Figure 8, a comparison of memory utilization for different LB algorithms with varying numbers of tasks is illustrated. The MOPDWTLB reduces the memory utilization compared to the FGELB, DALBFog, PSOSA-LB, OLB, and O2O-LB algorithms for each quantity of tasks. For example, with 1000 tasks, the memory utilization of MOPDWTLB is 66.67%, 52.5%, 45.71%, 34.48%, and 24% lower than the other algorithms, respectively. This is due to the optimized distribution of workloads among available fog nodes according to the optimal weight values, which alleviates overloading and guarantees efficient allocation of resources compared to the previous algorithms.

4.7. Bandwidth Utilization

It is the maximum bandwidth requirement of each VM for task execution. Figure 9 compares bandwidth utilization for different LB algorithms with varying numbers of tasks. The MOPDWTLB decreases the bandwidth utilization compared to the FGELB, DALBFog, PSOSA-LB, OLB, and O2O-LB algorithms for each quantity of tasks. For example, with 1000 tasks, the bandwidth utilization of MOPDWTLB is 26.29%, 22.97%, 18.57%, 14.5%, and 8.06% lower than the other algorithms, respectively. This reduction is achieved by optimizing the weight values of each node and distributing workloads across fog nodes efficiently, which prevents redundant data transfers and increases bandwidth efficiency.

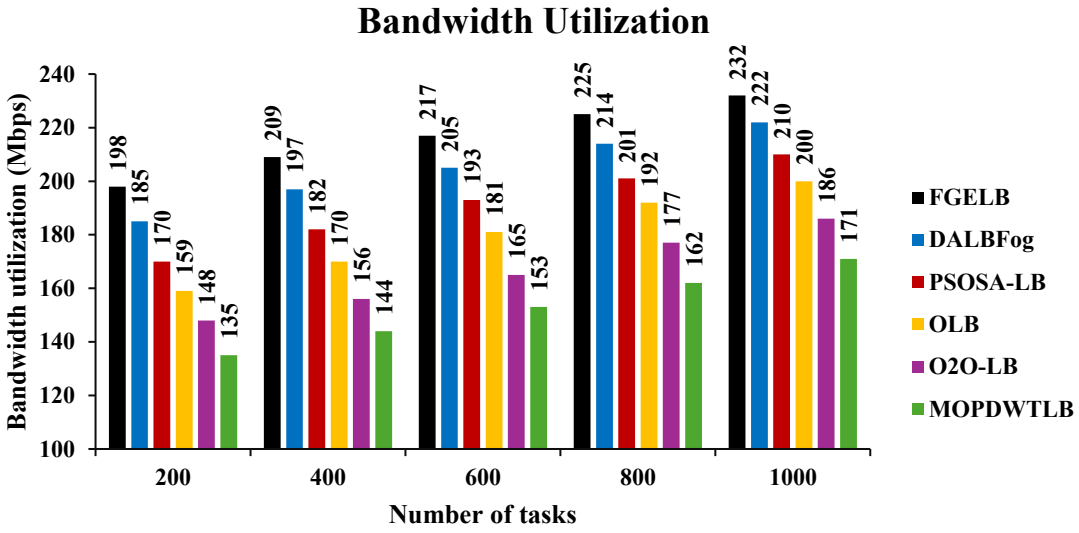

Figure 9 Bandwidth Utilization vs. No. of Tasks

4.8. Network Lifetime

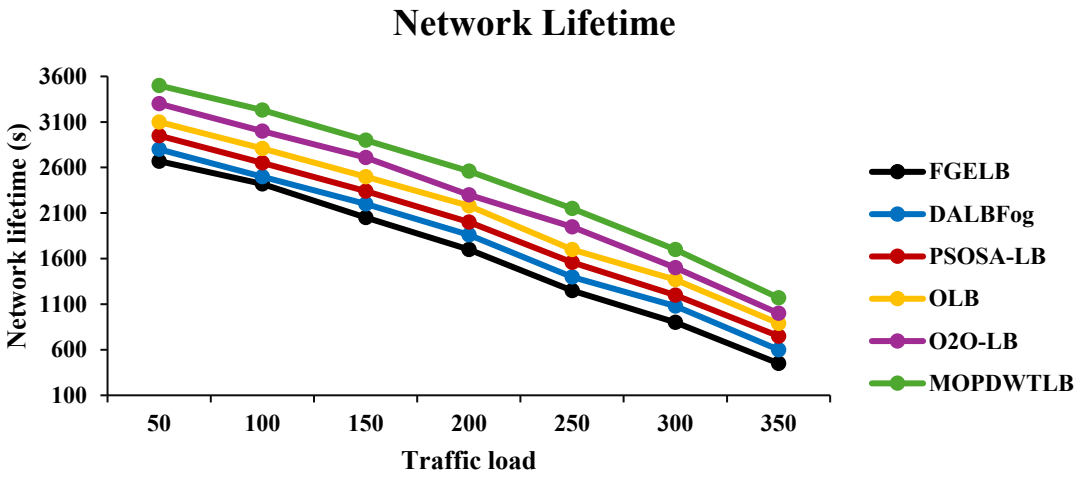

Figure 10 Network Lifetime vs. Traffic Load

RESEARCH ARTICLE

It determines the system longevity as given in equation (15).

$$\text{Lifetime} = \frac{\text{Residual energy of nodes}}{\text{Drain rate}} \quad (15)$$

Figure 10 evaluates the system longevity of different LB algorithms with varying numbers of traffic loads. The MOPDWTLB outperforms the FGELB, DALBFog, PSOSA-LB, OLB, and O2O-LB algorithms. For instance, with a traffic load of 50, the network lifetime for MOPDWTLB is improved by 31.09%, 25%, 18.64%, 12.9%, and 6.06% compared to the other algorithms, respectively.

This is achieved by effectively distributing the LB across fog nodes based on optimal weight values, which avoids overloading, reduces energy usage, and prolongs each device's lifetime compared to the previous algorithms.

4.9. Execution Time

It is the total time taken by the computing system to complete the execution of a given task from start to finish. It involves the time needed for task processing, scheduling, and execution. Figure 11 depicts the execution time of various LB algorithms for processing and scheduling 1000 tasks in the network. It is observed that the MOPDWTLB reduces the execution time by 67.03%, 62.5%, 55.88%, 49.15%, and 41.18% compared to the FGELB, DALBFog, PSOSA-LB, OLB, and O2O-LB algorithms. This is due to the use of a multi-objective weight-tuning strategy, which assigns tasks according to the node performance metrics. Also, it applies the weighted round-robin scheduling to reduce processing delay and overhead across the network.

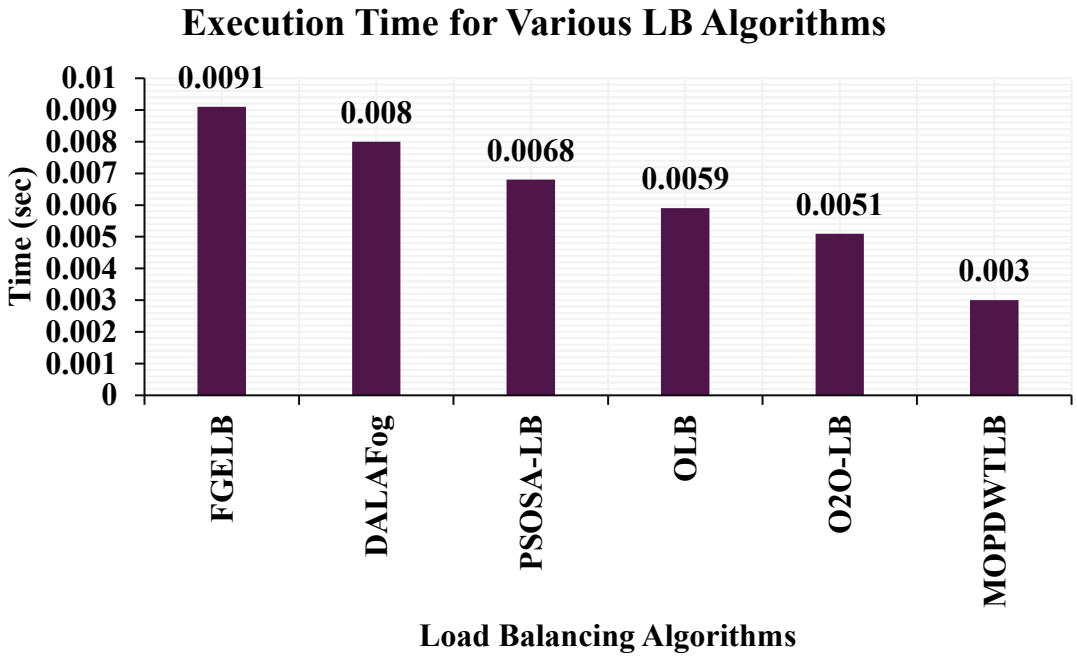

Figure 11 Execution Time Analysis for Different LB Algorithms

4.10. Ablation Studies

Table 3 compares the proposed MOPDWTLB to the LBSSA [12] in terms of computational resources, average resource usage, average LB variation, and running time for various requests from customers. Ten VMs are housed in each of the three data centers that make up the cloud in this scenario. 100 fog nodes are taken into consideration, each of which is linked to 10 to 20 IoT devices. Between 500 and 2000 requests per hour are considered. The computational capacity of each fog node's resources, which range from 1 to 10 Million Instructions Per Second (MIPS), allows it to fulfil requests. It is expected that the computational capability of each data center falls between 10 and 1000 Global Investment Performance Standards (GIPS). It is expected that the link between fog nodes will be 1 gigabit per second (Gb/s) and that the link between fog nodes and data centers will be 10 Gb/s.

The results demonstrated that the proposed MOPDWTLB reduces the amount of computing resources used to service customer requests by continuously checking for the best resources to schedule all requests. The MOPDWTLB serves all requests using the fewest computational resources possible. The average resource consumption is given as a proportion of computer resources used to serve client requests. The MOPDWTLB utilizes more resources than the LBSSA due to its sophisticated LB approach. The average

RESEARCH ARTICLE

load variance is connected to resource utilization, indicating that the MOPDWTLB has the lowest load variance value when compared to the LBSSA, owing to the effective distribution of submitted requests among available computing resources. Additionally, the running time is defined as the time spent arranging a specified number of requests to the resources. It is noted that the MOPDWTLB has a reduced running time due to fewer loop repeats for request scheduling.

Table 4 compares the proposed MOPDWTLB and MHHO [15] in terms of average makespan time, execution cost, and energy consumption. In this scenario, the fog layer has 20 VMs with randomly generated processing and memory capacity. Also, the test is performed using 100 randomly generated tasks with different properties such as the number of instructions, memory needed, and deadline. Based on the results, it can be realized that the MOPDWTLB algorithm has reduced makespan, execution cost, and energy consumption compared to the MHHO by employing dynamic weight adjustments for effective LB. This proves that the LB strategy of MOPDWTLB can enhance the scheduling of task requests to the most appropriate resources effectively based on the node parameters.

Table 3 Comparative Analysis of Proposed MOPDWTLB with LBSSA

Metrics	Algorithms	No. of customers' requests			
		500	1000	1500	2000
No. of used computing resources	LBSSA [12]	30	55	70	75
	Proposed MOPDWTLB	45	70	80	85
Average resource utilization (%)	LBSSA [12]	90.0	91.0	91.0	93.0
	Proposed MOPDWTLB	91.6	92.4	92.9	94.3
Average LB variance × 0.01	LBSSA [12]	1.5	1.8	1.8	1.9
	Proposed MOPDWTLB	1.0	1.3	1.3	1.5
Running time (s)	LBSSA [12]	1.0	2.0	5.0	14.0
	Proposed MOPDWTLB	0.8	1.4	3.6	11.3

Table 4 Comparative Analysis of Proposed MOPDWTLB with MHHO

Metrics	Algorithms	No. of VMs			
		5	10	15	20
Average makespan time	MHHO [15]	8.2	7.0	6.0	4.4
	Proposed MOPDWTLB	7.0	5.5	5.0	3.0
Execution cost × 100000	MHHO [15]	17.5	17.5	17.0	17.5
	Proposed MOPDWTLB	15.2	15.0	15.2	14.9
Energy consumption	MHHO [15]	40000	55000	69000	70000
	Proposed MOPDWTLB	34000	48000	53000	59000

Table 5 shows the performance of the proposed MOPDWTLB with PSOSA-LB [20]. In this scenario, the fog layer comprises 20 fog nodes, and the cloud layer has 5 cloud servers. A total of 1000 tasks are created within an interval of 10s and submitted to the network in batches for execution. It is observed that the MOPDWTLB achieved the lowest execution time, energy use, latency, and load imbalance for different task quantities compared to the PSOSA-LB. This is owing to the use of multi-objective weight-tuning and weighted round-robin strategies for LB depending on node characteristics.

Table 6 presents the performance of the proposed MOPDWTLB with O2O-PLB [23]. In this scenario, 20 and 5 resources are considered in the fog and cloud layers, respectively. Each node produces 500 tasks per time-sensitive and time-insensitive categories, which are submitted to the network in batches throughout the simulation. The results indicate that the MOPDWTLB attained the lowest response time, latency, and workload imbalance compared to the O2O-PLB for different task counts. This is because of applying a multi-objective weight-tuning strategy for LB in the network during task scheduling.

RESEARCH ARTICLE

Table 5 Comparative Analysis of Proposed MOPDWTLB with PSOSA-LB

Metrics	Algorithms	No. of tasks				
		200	400	600	800	1000
Execution time (s)	PSOSA-LB [20]	7.0	13.5	19.5	26.0	33.5
	Proposed MOPDWTLB	5.9	11.7	17.0	24.4	31.2
Energy consumption (kJ)	PSOSA-LB [20]	240	480	700	920	1160
	Proposed MOPDWTLB	215	450	660	875	1080
Latency (ms)	PSOSA-LB [20]	55	64	68	72	78
	Proposed MOPDWTLB	35	40	44	48	51
Load imbalance (%)	PSOSA-LB [20]	5.6	6.2	7.0	7.6	8.0
	Proposed MOPDWTLB	4.9	5.4	5.8	6.3	6.9

Table 6 Comparative Analysis of Proposed MOPDWTLB with O2O-PLB

Metrics	Algorithms	No. of tasks				
		200	400	600	800	1000
Response time (ms)	O2O-PLB [23]	22	32	42	58	70
	Proposed MOPDWTLB	18	24	30	41	53
Latency (ms)	O2O-PLB [23]	20	30	40	50	60
	Proposed MOPDWTLB	15	20	28	38	45
Load imbalance (%)	O2O-PLB [23]	4.0	3.5	2.5	1.0	1.0
	Proposed MOPDWTLB	3.2	2.8	2.0	0.6	0.6

4.11. Discussion

These comparative results indicate that the MOPDWTLB scheme significantly performs better than the current algorithms, such as FGELB, DALBFog, PSOSA-LB, OLB, O2O-LB, LBSSA, and MHHO, based on the TP, delay, RT, energy use, memory use, bandwidth use, network lifetime, execution time, makespan, and load imbalance. These improvements are achieved due to optimizing multiple objectives simultaneously through a weighted-sum approach rather than previous single-objective optimization models. Also, data profiling assigns specific weights to the heterogeneous nodes in the fog, ensuring that workloads are placed on high-performing nodes, which minimizes response time and memory usage. Moreover, the weighted round robin scheduling and coordinated distributed architecture reduce bandwidth consumption and increase network lifetime. Thus, it can be concluded that MOPDWTLB is highly capable of handling heterogeneous IoT-fog-cloud settings with variable workloads, making it ideal for latency-sensitive networks, such as those in smart healthcare and industrial automation.

5. CONCLUSION

The MOPDWTLB scheme was introduced in this paper for optimizing load distribution in IoT-Fog-Cloud networks. This algorithm utilized a 3-level structure and a data-driven profiling technique to define performance metrics for fog nodes. A weighted round-robin scheme was then employed to allocate weights to nodes based on these metrics. It was then extended to the MOO dilemma to optimize multiple performance metrics simultaneously. An architectural adjustment scheme was also implemented to meet the fog-cloud environment's specific criteria. Moreover, simulation results demonstrated that the MOPDWTLB achieves superior performance metrics compared to existing LB algorithms in IoT-Fog-Cloud networks, with an average TP of 0.96 requests/sec, average delay of 51ms, energy utilization of 250 kJ, RT of 122ms, memory usage of 19 MB, and bandwidth utilization of 171 Megabits per second (Mbps) for 1000 tasks in the network.

REFERENCES

[1] R. Chataut, A. Phoummalayvane, and R. Akl, “Unleashing the power of IoT: a comprehensive review of IoT applications and future prospects in healthcare, agriculture, smart homes, smart cities, and industry 4.0,” Sensors, vol. 23, no. 16, p. 7194, Aug. 16, 2023.

[2] V. K. Prasad, D. Dansana, M. D. Bhavsar, B. Acharya, V. C. Gerogiannis, and A. Kanavos, “Efficient resource utilization in IoT and cloud computing,” Inf., vol. 14, no. 11, p. 619, Nov. 19, 2023.

[3] S. Rani, P. Bhambri, and A. Kataria, “Integration of IoT, big data, and cloud computing technologies: trend of the era,” in Big Data, Cloud Comput. IoT, pp. 1–21, Apr. 19, 2023.

[4] L. Rosa, L. Foschini, and A. Corradi, “Empowering cloud computing with network acceleration: a survey,” IEEE Commun. Surv. Tutor., vol. 26, no. 4, pp. 2729–2768, Mar. 14, 2024.

RESEARCH ARTICLE

[5] A. Avan, A. Azim, and Q. H. Mahmoud, "A state-of-the-art review of task scheduling for edge computing: a delay-sensitive application perspective," Electron., vol. 12, no. 12, p. 2599, Jun. 8, 2023.
[6] R. Das and M. M. Inuwa, "A review on fog computing: issues, characteristics, challenges, and potential applications," Telemat. Inform. Rep., vol. 10, p. 100049, Jun. 1, 2023.
[7] M. H. Kashani and E. Mahdipour, "Load balancing algorithms in fog computing," IEEE Trans. Serv. Comput., vol. 16, no. 2, pp. 1505–1521, May 11, 2022.
[8] Y. Lohumi, D. Gangodkar, P. Srivastava, M. Z. Khan, A. Alahmadi, and A. H. Alahmadi, "Load balancing in cloud environment: a state-of-the-art review," IEEE Access, vol. 11, pp. 134517-134530, Nov. 27, 2023.
[9] M. S. Aslanpour, A. N. Toosi, M. A. Cheema, M. B. Chhetri, and M. A. Salehi, "Load balancing for heterogeneous serverless edge computing: a performance-driven and empirical approach," Future Gener. Comput. Syst., vol. 154, pp. 266-280, May 1, 2024.
[10] S. R. Hassan, A. U. Rehman, N. Alsharabi, S. Arain, A. Quddus, and H. Hamam, "Design of load-aware resource allocation for heterogeneous fog computing systems," PeerJ Comput. Sci., vol. 10, p. e1986, Apr. 18, 2024.
[11] A. Asghar, A. Abbas, H. A. Khattak, and S. U. Khan, "Fog based architecture and load balancing methodology for health monitoring systems," IEEE Access, vol. 9, pp. 96189–96200, Jul. 1, 2021.
[12] F. Alqahtani, M. Amoon, and A. A. Nasr, "Reliable scheduling and load balancing for requests in cloud-fog computing," Peer-to-Peer Netw. Appl., vol. 14, no. 4, pp. 1905–1916, Jul. 2021.
[13] S. P. Singh, "Effective load balancing strategy using fuzzy golden eagle optimization in fog computing environment," Sustain. Comput. Inform. Syst., vol. 35, p. 100766, Sep. 1, 2022.
[14] A. Javadpour, A. K. Sangaiah, P. Pinto, F. Ja'fari, W. Zhang, A. M. H. Abadi, and H. Ahmadi, "An energy-optimized embedded load balancing using DVFS computing in cloud data centers," Comput. Commun., vol. 197, pp. 255–266, Jan. 1, 2023.
[15] I. Z. Yakubu and M. Murali, "An efficient meta-heuristic resource allocation with load balancing in IoT-fog-cloud computing environment," J. Ambient Intell. Humaniz. Comput., vol. 14, no. 3, pp. 2981–2992, Mar. 2023.
[16] S. Singh, E. E. Sham, and D. P. Vidyarthi, "Optimizing workload distribution in fog-cloud ecosystem: a JAYA based meta-heuristic for energy-efficient applications," Appl. Soft Comput., vol. 154, p. 111391, Mar. 1, 2024.
[17] M. Ibrahim, Y. Lee, and D. H. Kim, "DALBFog: deadline-aware and load-balanced task scheduling for the internet of things in fog computing," IEEE Syst. Man Cybern. Mag., vol. 10, no. 1, pp. 62–71, Jan. 9, 2024.
[18] Z. Shamsa, A. Rezaee, S. Adabi, A. M. Rahimabadi, and A. M. Rahmani, "A distributed load balancing method for IoT/fog/cloud environments with volatile resource support," Clust. Comput., vol. 27, no. 4, pp. 4281–4320, Jul. 2024.
[19] A. Mahapatra, S. K. Majhi, K. Mishra, R. Pradhan, D. C. Rao, and S. K. Panda, "An energy-aware task offloading and load balancing for latency-sensitive IoT applications in the fog-cloud continuum," IEEE Access, vol. 12, pp. 14334–14349, Jan. 22, 2024.
[20] M. B. Shaik, K. S. Reddy, K. Chokkanathan, S. A. A. Biabani, P. Shanmugaraja, and D. D. Brabin, "A hybrid particle swarm optimization and simulated annealing with load balancing mechanism for resource allocation in fog-cloud environments," IEEE Access, vol. 12, pp. 172439–172450, Nov. 1, 2024.
[21] M. A. Ala'anzy, R. Zhanuzak, R. Akhmedov, N. Mohamed, and J. Al-Jaroodi, "Dynamic load balancing for enhanced network performance in IoT-enabled smart healthcare with fog computing," IEEE Access, vol. 12, pp. 188957–188975, Dec. 12, 2024.
[22] S. K. Srichandan, S. K. Majhi, S. Jena, K. Mishra, and R. Bhat, "A secure and distributed placement for quality of service-aware IoT requests in fog-cloud of things: a novel joint algorithmic approach," IEEE Access, vol. 12, pp. 56730-56748, Apr. 18, 2024.
[23] V. C. Bharathi, S. S. Abuthahir, M. Ayyavaraiah, G. Arunkumar, U. Abdurrahman, and S. A. A. Biabani, "O2O-PLB: a one-to-one-based optimizer with priority and load balancing mechanism for resource allocation in fog-cloud environments," IEEE Access, vol. 13, pp. 22146–22155, Jan. 29, 2025.
[24] R. Rateb, A. A. Hadi, V. M. Tamanampudi, L. Abualigah, A. E. Ezugwu, A. I. Alzahrani, ... and H. Jia, "An optimal workflow scheduling in IoT-fog-cloud system for minimizing time and energy," Sci. Rep., vol. 15, no. 1, p. 3607, Jan. 29, 2025.
[25] N. Khaledian, S. Razzaghzadeh, Z. Haghbayan, and M. Völp, "Hybrid Markov chain-based dynamic scheduling to improve load balancing performance in fog-cloud environment," Sustain. Comput. Informatics and Syst., vol. 45, p. 101077, Jan. 1, 2025.

Authors

M. Parveen Taj is currently working as an Assistant Professor in the Department of Computer Science at Dr. G.R.D. College of Science, Coimbatore. She previously served as an Assistant Professor in the Department of Computer Science at Sri Jayendra Saraswathy Maha Vidyalaya College of Arts & Science, Coimbatore, from 2005 to 2024. She is pursuing a Ph.D. in Computer Science (part-time) at PPG College of Arts & Science and has completed an M.Phil. in Computer Science with a specialization in Advanced Networking. She has guided 15 M.Phil. research scholars in the areas of Data Warehousing and Mining, and Advanced Networking.

Dr. N. Muthumani received her Ph.D. in Computer Science from Mother Teresa Women's University, Kodaikanal, and has qualified for the UGC-NET in Computer Science, demonstrating her strong commitment to academic excellence and research. She currently serves as the Principal of PPG College of Arts and Science, Coimbatore, Tamil Nadu, a role she has held since November 2020. Previously, she served as Professor and Head at Sri Ramakrishna College of Arts and Science (2008–2020), Assistant Professor at KG College of Arts and Science (2005–2008), and Lecturer at RVS College of Arts and Science (2000–2005). Her areas of expertise include academic administration, accreditation processes, curriculum design, and faculty development. Dr. Muthumani has published more than 25 research articles in reputed journals, authored 8 books, and contributed 5 chapters to edited volumes. She has presented over 20 research papers at national and international conferences and has served as a resource person in more than 100 faculty development programs, inspiring and mentoring educators across the academic spectrum.

How to cite this article:

M. Parveen Taj, N. Muthumani, "A Multi Objective Performance Driven Weight Tuning Based Load-Balancing Mechanism in IoT–Fog–Cloud Environment", International Journal of Computer Networks and Applications (IJCNA), 12(5), PP: 791-808, 2025, DOI: 10.22247/ijcna/2025/47.