

Energy-Efficient Load Balancing in Fog Computing Using a Hybrid Dijkstra Greedy Approach

Mahmoud A. El-Sakhawy, Ghali Abdullah, Imane Aly Saroit Ismail, Shaimaa M. Mohamed, and Ghada Abdelhady, *Member, IAENG*

Abstract - Fog computing has become a key paradigm for latency-sensitive Internet of Things (IoT) applications by bringing computation closer to data sources. It extends processing capabilities to the network edge, thereby reducing reliance on centralized cloud infrastructures and enabling real-time data handling. However, the distributed and heterogeneous characteristics of fog environments create significant challenges in resource management and energy efficiency, especially when distributing workloads across dynamically changing nodes.

To overcome these challenges, this paper aims to distribute workloads efficiently among fog nodes. It introduces a Hybrid Energy-Aware Load Balancing (EALB) algorithm that integrates Dijkstra's shortest path algorithm for routing with a greedy resource allocation strategy. To evaluate the proposed algorithm, simulations were conducted across multiple deployment scales, from small IoT setups to large smart city environments. Results demonstrate that, unlike existing approaches, the proposed method simultaneously accounts for network topology and resource availability when making load balancing decisions.

The results indicate notable improvements in average latency (5.36% in medium-scale and 4.55% in large-scale deployments), together with a manageable trade-off in power consumption. By combining shortest-path routing with greedy allocation, the hybrid model improves latency performance as the system scales, besides a slight increase in energy usage. In small-scale scenarios, the Greedy algorithm achieves superior performance in both latency (13.4% reduction in total latency) and energy consumption (5.57% lower power use). However, in large-scale environments, the Hybrid EALB algorithm reaches substantial latency gains with only a marginal increase in power consumption (0.36%). Overall, the EALB approach enables more intelligent routing and task allocation through its integrated strategy, effectively balancing latency and energy efficiency. These results show how useful hybrid optimization approaches can be when building sustainable fog computing architectures. They also offer practical guidance on choosing the right load balancing strategy, depending on the system's size, network structure, and specific performance needs.

Index Terms - Fog Computing, Load Balancing, Energy-Aware Algorithms, Dijkstra Algorithm, Resource Allocation.

Manuscript received July 22, 2025; revised July 14, 2026.

Mahmoud A. El-Sakhawy is an Assistant Professor in the Cyber Security Department, IT College, Amman Arab University, Jordan Street, Amman, Jordan (phone: 00962796012872; fax: +96264791414; e-mail: m.elsakhawy@aau.edu.jo).

Ghali Abdullah is a postgraduate student at the Faculty of Computing & Artificial Intelligence, Cairo University, Egypt (e-mail: ghalyabdullah022@gmail.com).

Imane Aly Saroit Ismail is a Professor in the IT Department, Faculty of Computing & Artificial Intelligence, Cairo University, Egypt (e-mail: i.saroit@fci-cu.edu.eg).

Shaimaa M. Mohamed is an Assistant Professor in the IT Department, Faculty of Computing & Artificial Intelligence, Cairo University, Egypt (e-mail: s.mosaad@fci-cu.edu.eg).

Ghada Abdelhady is an Associate Professor in the Computer Systems Engineering Department, Faculty of Engineering, October University for Modern Sciences and Arts, Egypt (e-mail: gabdelmouez@msa.edu.eg).

I. INTRODUCTION

The rapid growth of Internet of Things (IoT) devices has caused a dramatic increase in the amount of data generated at the edge of networks. Traditional cloud computing architectures have difficulty managing this rapid increase in data effectively due to bandwidth limitations, strict latency requirements, and increasing energy consumption. To address these challenges, fog computing has emerged as a promising solution. By extending cloud capabilities closer to the network edge, fog computing enables real-time data processing and analytics near the data sources themselves [1].

Fog computing forms a distributed system made up of heterogeneous devices with different levels of computational power, storage capacity, and energy availability. Efficiently managing resources across these distributed fog nodes is a complex task, especially when trying to balance workloads while minimizing energy consumption. Load balancing plays a critical role in this context, as it improves resource utilization, enhances system responsiveness, and prevents nodes from becoming overloaded or underutilized. At the same time, energy efficiency has become a key concern in the design of distributed systems. As fog deployments expand to support the growing IoT landscape, their overall energy usage continues to rise.

Recent studies show that data centers in the United States consumed around 61 billion kWh in 2006. Also, global energy demand is projected to reach 10,300 TWh annually by 2030 [2]. These figures highlight the urgent need for energy-efficient strategies in fog computing environments. One of the main challenges in fog computing is load balancing (LB). Although many LB techniques have proven effective in cloud environments, directly applying them to fog systems is often inefficient due to architectural and operational differences. Ensuring high resource utilization, minimizing bottlenecks, avoiding overload and underload situations, and improving response times remain open research issues in fog-based systems [3].

While cloud-based load balancing methods can be adapted for fog computing, they must be redesigned to take into account the dynamic nature, resource constraints, and heterogeneity of fog nodes [4]. Effective load balancing in fog systems significantly enhances task execution efficiency and overall network performance by making better use of both end-user devices and intermediary fog nodes.

To address these issues, this study proposes a Hybrid Energy-Aware Load Balancing (EALB) algorithm that combines Dijkstra's shortest path algorithm with a greedy

resource allocation strategy. The objectives of the proposed algorithm are to:

- Minimize data transmission latency through efficient route selection between source and destination nodes.
- Reduce power consumption by implementing adaptive and intelligent load distribution.
- Dynamically balance computational workloads based on real-time node availability and utilization.

Unlike many existing approaches, the proposed method considers both network topology and resource availability simultaneously during decision-making. By combining optimal path selection using Dijkstra's algorithm with greedy-based resource allocation, the hybrid model improves energy efficiency while maintaining or enhancing key performance metrics as the system scales.

The proposed algorithm is evaluated through extensive simulations conducted on AWS across three fog deployment scales: small-scale environments (such as smart homes), medium-scale systems (e.g., factory-level deployments), and large-scale scenarios (such as smart cities). The results show up to a 0.85% reduction in power consumption in large-scale environments, along with significant improvements in average latency; 2.36% in medium-scale and 6.29% in large-scale deployments.

The remainder of this paper is organized as follows. Section 2 reviews related work on fog computing, load balancing strategies, and energy-aware system design. Section 3 presents the proposed system architecture and details the Hybrid EALB algorithm. Section 4 describes the experimental setup and evaluation methodology. Section 5 analyzes and discusses the simulation results. Finally, Section 6 concludes the paper and outlines future research directions.

II. LITERATURE REVIEW

This section reviews the fundamentals of fog computing architecture and explains its role in extending cloud services closer to the network edge. It also examines the main load balancing algorithms in fog environments; they are divided into three main categories: static, dynamic, and nature-inspired algorithms. Finally, it discusses energy-aware system design and its importance in improving efficiency and sustainability in fog-based infrastructures.

A. Fog Computing

Fog computing extends cloud services to the network edge by placing computing, storage, and networking resources closer to data sources and end users. The concept was first introduced by Cisco Systems to describe edge-based resources such as routers, switches, and base stations [1]. Unlike traditional cloud computing, which relies on centralized large-scale data centers, fog computing distributes resources across a hierarchical structure that spans from the cloud to the edge.

A typical fog computing architecture consists of three layers:

1. **IoT Layer** – consists of sensors and end devices that generate data.
2. **Fog Layer** – composed of intermediate fog nodes with limited computing and storage capacities.

3. **Cloud Layer** – consists of centralized, high-capacity data centers [5].

This layered structure helps fog computing to address several limitations of traditional cloud systems:

- **Reduced Latency:** Processing data close to its source reduces communication delays, which is essential for real-time applications.
- **Bandwidth Optimization:** Local data processing decreases the amount of data sent to the cloud, so reducing network congestion.
- **Location Awareness:** Fog nodes support geo-distributed services through localized processing.
- **Mobility Support:** The fog layer ensures smooth communication with mobile devices.
- **Enhanced Privacy and Security:** Sensitive data can be processed locally before being transmitted to the cloud, offering better control and protection.

However, the distributed and heterogeneous nature of fog computing also creates significant challenges in resource management and load balancing, both of which are essential for maintaining optimal system performance [5].

B. Load Balancing Algorithms

Load balancing involves distributing workloads across multiple computing resources to improve resource utilization, maximize throughput, reduce response time, and prevent individual nodes from becoming overloaded. In fog environments, load balancing is complex due to differences in node capabilities, changing network conditions, and limited resources.

Load balancing techniques are generally divided into three main categories: static, dynamic, and nature-inspired algorithms [6].

1) Static Load Balancing Algorithms

Static algorithms assign workloads based on predefined system information. They do not adapt to real-time changes, which makes them less suitable for highly dynamic environments. Common examples include:

- **Round Robin:** Tasks are assigned sequentially to servers without considering their current load. Although simple, it can lead to imbalance in heterogeneous systems [5].
- **Weighted Round Robin:** Assigns weights to servers according to their capacity, enabling more powerful nodes to process a greater number of tasks [5].
- **IP Hash:** Tasks are distributed based on client IP addresses to maintain session consistency. However, this may cause imbalance if client requests are not uniformly distributed [7].

2) Dynamic Load Balancing Algorithms

Dynamic algorithms adjust to real-time system conditions. They make decisions based on current resource usage, so they are more suitable for fluctuating workloads. Examples include:

- **Least Connections:** Assigns new tasks to the server with the fewest active connections, which is effective during high traffic [5].
- **Weighted Least Connections:** Enhances the least connections strategy by adding server capacity into consideration [6].

- **Least Response Time:** Selects the server with the least response time, so improving performance [6].

3) Nature-Inspired Load Balancing Algorithms

Nature-inspired algorithms are based on biological or swarm behaviors. They are well-suited for complex, dynamic systems like fog environments. Some of these approaches are:

- **Ant Colony Optimization (ACO):** Inspired by the foraging behavior of ants, ACO uses virtual pheromone trails to identify efficient task allocation paths. Studies in [8] reported improved response times compared to traditional approaches.
- **Particle Swarm Optimization (PSO):** Based on bird flocking behavior, PSO uses a population of candidate solutions to identify optimal scheduling strategies. Research in [9] showed improvements in makespan and total cost in cloud-fog systems.
- **Genetic Algorithms (GA):** Modeled after evolutionary processes, GA iteratively improves solutions over generations. In [3]; a reinforcement learning and GA-based strategy achieved an 85.71% load balancing level.

C. Recent Developments in Fog Computing Load Balancing

Recent studies have proposed various strategies to enhance load balancing in fog environments.

In [3]; a Load Balancing and Optimization Strategy (LBOS) combining reinforcement learning and genetic algorithms was introduced. It distributed requests evenly across servers, achieving an 85.71% balancing level and improving allocation cost and response time.

In [4]; the Min-Min algorithm was implemented using the Cloud Analyst tool to minimize response time, showing notable efficiency gains in small cluster configurations.

A comparative study in [8] evaluated ACO and PSO for IoT task offloading and found that ACO outperformed both PSO and Round Robin in terms of response time.

In [10]; the Rock Hyrax Optimization (RHO) metaheuristic was applied to Smart Grid load balancing. It reduced processing time by 26%, response time by 15%, energy consumption by 29%, cost by 6%, and delay by 14% compared to conventional methods.

Authors in [11] focused on minimizing energy consumption while meeting task deadlines in IoT systems. Two semi-greedy algorithms, Priority-Aware Semi-Greedy (PSG) and PSG with Multi-Start (PSG-M), improved deadline satisfaction by 1.35 times and reduced deadline violations by 97.6%.

In [12]; a modified version of Dijkstra's algorithm was proposed for Software Defined Networking (SDN) to optimize routing metrics such as hop count, latency, and energy efficiency.

In [13], authors proposed Chaotic Grey Wolf Optimization (CGWO), which integrates chaos theory into the Grey Wolf algorithm to improve convergence and search performance. Results showed lower makespan, better resource utilization, and faster response time compared to ACO and Min-Max.

In [14], AEOSA proposed algorithm combined Artificial Ecosystem-Based Optimization (AEO) with the Salp Swarm Algorithm (SSA) for IoT-fog-cloud systems. The hybrid

approach improved makespan and throughput and outperformed Particle Swarm Optimization (PSO), Harris Hawks Optimization (HHO), Firefly Algorithm (FA) and the traditional AEO

In [15], CHMPAD proposed algorithm merged Chimp Optimization and Marine Predators algorithms with a disruption mechanism to balance exploration and exploitation. It achieved reduced makespan and higher throughput, demonstrating the strength of hybrid swarm-based scheduling in fog environments.

D. Energy-Aware System Design

Energy efficiency has become a major concern in modern computing systems, especially as infrastructure scales continue to grow. Energy-aware system design aims to reduce power consumption without compromising performance.

In fog computing, energy efficiency is especially critical due to distributed resources and the frequent reliance on battery-powered edge devices. Several strategies have been proposed:

- **Resource Consolidation:** Research in [16] demonstrated that effective consolidation policies can reduce energy consumption by approximately 12.4% on average.
- **Dynamic Resource Allocation:** Authors in [2] introduced an energy-aware cloud load balancing model that identifies optimal operating conditions to minimize energy usage while maintaining high utilization.
- **Workload Classification:** In [17]; load balancing strategies were developed based on factors such as carbon emissions and energy savings.
- **Energy-Efficient Routing:** In [18]; a Greedy Knapsack-based Energy Efficient Routing Algorithm (GKEERA) was proposed to distribute energy consumption more evenly and extend network lifetime.

Despite these contributions, limited research has addressed the joint optimization of network path selection and resource allocation to reduce energy consumption in fog systems. The proposed Hybrid Energy-Aware Load Balancing (EALB) algorithm fills this gap by combining Dijkstra's shortest path algorithm for route optimization with a greedy resource allocation strategy, so improving both latency and energy efficiency simultaneously.

III. PROPOSED SYSTEM

This section first describes the three-layer fog computing architecture and describes the role of each layer, then presents the proposed Hybrid EALB algorithm, and finally provides an analysis of its operation.

A. Three-layer fog computing architecture

The system follows a three-tier architecture composed of the sensor layer, fog layer, and cloud layer (as shown in Fig. 1). This hierarchical design allows efficient data transmission and distributed processing within the fog environment. It also provides the foundation for the proposed Hybrid Energy-Aware Load Balancing (EALB) algorithm.

- **The sensor layer** is the lowest layer in the architecture. It

consists of IoT devices and sensors that collect data from the physical environment. These devices usually have limited processing power and often run on batteries, so energy efficiency is very important. In some cases, they may perform simple preprocessing before sending the data to the fog layer.

- **The fog layer** is the middle layer and contains distributed fog nodes that provide local computing, storage, and networking services. A special controller node acts as the central coordinator for load balancing decisions. This controller runs the proposed Hybrid EALB algorithm to distribute tasks efficiently among the fog nodes. The main components of the fog layer include:
 - *Fog Nodes*: Heterogeneous distributed computing units that process data received from the sensor layer.
 - *Controller Node*: The management unit responsible for running the EALB algorithm and assigning tasks to fog nodes.
 - *Communication Infrastructure*: The network framework that connects fog nodes to each other and to the other layers.
- **The cloud layer**, is the top layer of the architecture. It provides virtually unlimited computing power and storage capacity. It is used for long-term data storage, large-scale data analysis, and computationally intensive tasks that cannot be handled by fog nodes. The cloud also acts as a backup processing unit and handles tasks that are not time-sensitive.

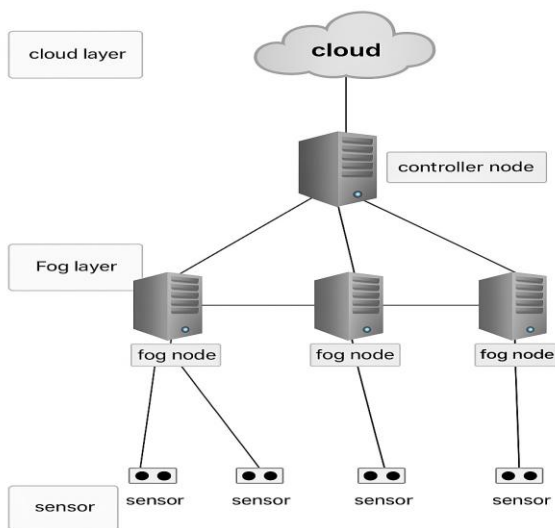


Fig. 1. System architecture of the proposed methodology

B. Hybrid EALB Algorithm

This subsection introduces the main contribution of this work: the Hybrid Energy-Aware Load Balancing (EALB) algorithm. The algorithm combines Dijkstra's shortest path algorithm with a greedy resource allocation strategy to achieve efficient and energy-aware load balancing in fog computing environments.

The Hybrid EALB algorithm operates in two main phases:

- **Path Optimization Phase (Dijkstra)**: Determines the shortest communication paths between nodes in order to

minimize transmission latency and communication overhead.

- **Resource Allocation Phase (Greedy)**: Assigns tasks to fog nodes based on proximity, available resources, and energy constraints.

When sensors send data to the fog layer, the controller node identifies the receiving node as the source. It then computes the shortest paths from this node to all other fog nodes. For each task, the controller selects the most suitable fog node by considering both path distance and available resources.

1) Dijkstra Component

The Dijkstra component focuses on identifying the shortest paths within the fog network. The network is modeled as a graph $G = (V, E)$, where V represents nodes and E represents communication links. Given a source node $s \in V$; the algorithm proceeds as follows:

- 1) Initialize distances: $d(s) = 0$ and $d(v) = \infty$, for all $v \neq s$
 - 2) Insert all nodes into a priority queue Q ordered by distance.
 - 3) While Q is not empty:
 - a) Extract node u with minimum distance from Q .
 - b) For each neighbor v of u :
 - Calculate potential new distance $d'(v) = d(u) + w(u, v)$
 - If $d'(v) < d(v)$, update $d(v) = d'(v)$.
- Where:
- $d(u)$: The currently known shortest distance from the source vertex to vertex u .
 - $w(u, v)$: The weight (cost) of the direct edge from vertex u to vertex v .
 - $d'(v)$: The newly calculated potential distance from the source to vertex traveling through u .

The output is a shortest-path table from the source node to all other nodes.

To better reflect real network conditions, the traditional Dijkstra algorithm is extended by including additional parameters such as bandwidth usage, link delay, and flow entries. A path load metric F is calculated as a weighted combination using Equation (1):

$$F = w_1 \cdot S + w_2 \cdot T_{avg} + w_3 \cdot V_{max} \quad (1)$$

where:

- S denotes the flow entries measured at the switch,
- T_{avg} represents the average link delay,
- V_{max} indicates the maximum bandwidth utilization,
- w_1 , w_2 , and w_3 are normalized weighting coefficients such that their sum equals 1.

This enhancement ensures that routing decisions consider both physical distance and network performance indicators.

2) Greedy Component

The greedy component selects the most suitable fog node for task assignment based on current resource availability. It makes the best local decision at each step, which leads to an efficient overall allocation over time.

The procedure works as follows:

1. Sort fog nodes according to their distance from the source (calculated using Dijkstra).

2. Filter nodes based on resource availability for the current task.
3. Select the closest node with sufficient resources.
4. Update the selected node's resource availability.
5. Repeat the process for subsequent tasks.

This strategy ensures that tasks are assigned to the nearest capable fog node while preventing overload conditions.

3) Integration of Dijkstra and Greedy Components

The integration of Dijkstra's path optimization and greedy resource allocation forms a hybrid framework that simultaneously addresses routing efficiency and workload distribution.

The workflow of the Hybrid EALB algorithm is as follows:

1. The controller node initializes and maintains updated information about all fog nodes and their resource capacities.
2. Upon task arrival from a sensor:
 - Compute shortest paths from the source node using Dijkstra's algorithm.
 - Filter nodes based on real-time resource availability.
 - Apply the greedy strategy to select the optimal node.
 - Assign the task and update node resource records.
3. Periodically recalculate paths to adapt to changes in network conditions.

This combined approach ensures efficient routing and balanced resource utilization, so reduces energy consumption while improving system performance.

Pseudocode for a General Greedy Algorithm:

```
Function greedyAlgorithm(problem):
    solution = empty
    while not problem.isSolved():
        choice =
        problem.bestLocalChoice() if
        problem.isFeasible(choice):
            solution.add(choice)
            problem.updateState(choice)
    return solution
```

C. System Analysis

1) Flowchart Diagram: Fig.2 illustrates the operational flow of the Hybrid EALB algorithm. The process starts when a packet arrives from a user. The system then checks the flow table, monitors load conditions, computes possible paths, and selects the optimal route for data transmission.

2) Sequence Diagram: Fig.3 illustrates the sequence of interactions between system components during execution of the algorithm. It shows how the user, fog server, and cloud layer communicate, including path generation, shortest-path computation, route updates, and load monitoring.

Overall, the structured three-layer architecture combined with the hybrid optimization strategy enables efficient, scalable, and energy-aware task management in fog computing environments.

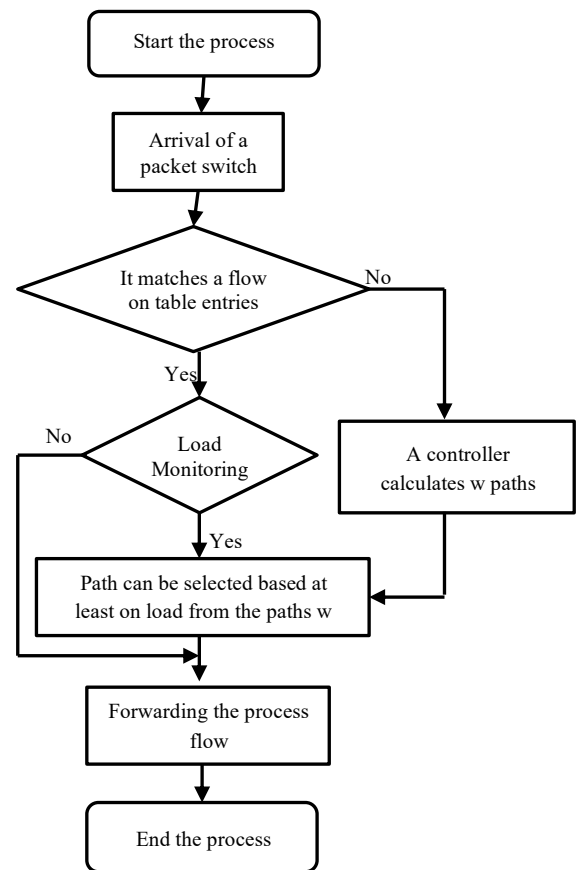


Fig. 2. Flow chart diagram of the Hybrid EALB algorithm

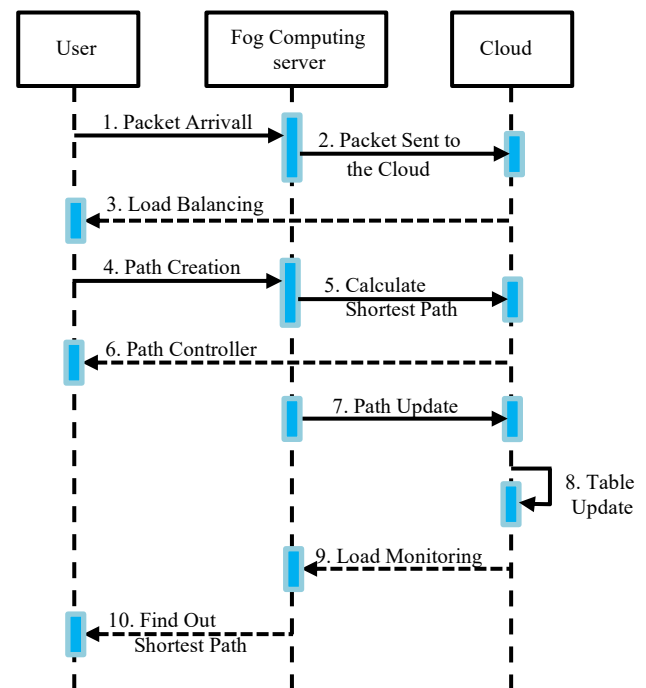


Fig. 3. Sequence diagram of system interactions

IV. EXPERIMENTAL SETUP AND METHODOLOGY

This section outlines the experimental environment, simulation parameters, implementation framework, and metrics used to evaluate the performance of the proposed Hybrid Energy-Aware Load Balancing (EALB) algorithm against the greedy one, across different fog computing deployment scales.

A. Experimental Setup

Overall, simulations provide a comprehensive evaluation framework for analyzing performance trade-offs and identifying scale-dependent behavior.

To evaluate the effectiveness of the Hybrid EALB algorithm, extensive simulations were performed using the Amazon Web Services (AWS) cloud platform with Python 3. The proposed approach was compared with the Greedy algorithm, under different deployment scales and performance metrics.

To ensure fair comparison, some simulation parameters were kept fixed across all experiments, as summarized in Table I [2], [3], [4], [19]. Others were changed, as described in Table II, to reflect realistic fog computing environments and allow consistent and controlled evaluation.

Table I – Fixed Simulation Parameters

Parameter	Value
Simulation Time	200 time-units
Fog Node Capacity	100 processing units
Fog Node Power Consumption Rate	0.2 units per task load
Task Load	Randomly generated (10–30 units)
Task Generation Interval	Randomly generated (1–5 time-units)

Table II – Scale-Specific Deployment Parameters

Scale	Environment Type / Example	Fog Nodes	IoT Devices	Tasks
Small	Smart Home Environment	10	500	1,000
Medium	Industrial Factory Environment	30	2,000	10,000
Large	Smart City Environment	50	10,000	50,000

The selected simulation duration; 200 time-units; is long enough to observe stable system behavior while keeping computation manageable. Each fog node had a processing capacity of 100 units, representing typical resource limitations in edge computing environments [19]. The power consumption rate was set to 0.2 units per task load, based on established energy-aware cloud models [2].

Task loads were randomly generated within a 10–30 units to emulate heterogeneous IoT workloads. Similarly, task generation intervals were randomized between 1 and 5 time-units to simulate bursty traffic patterns usually observed in IoT systems [3], [4].

The three deployment scenarios were designed to represent increasing levels of system complexity:

- *Small-scale configuration, 10 fog nodes, 500 IoT devices;* reflects smart home or small office deployments [20], [21].
- *Medium-scale configuration, 30 fog nodes, 2,000 IoT devices;* represents industrial environments with moderate device density [5], [7].
- *Large-scale configuration, 50 fog nodes, 10,000 IoT devices;* simulates smart city deployments with high device concentration and task volume [22].

The system size increases progressively from small to large scale. The number of fog nodes increases by roughly three times between each scale, while IoT devices increase by four

to five times [17]. This gradual scaling allows clear observation of how the algorithms behave as system complexity grows.

In the simulation model, IoT devices and fog nodes form a mesh network. Each fog node connects to multiple IoT devices and neighboring fog nodes. As mentioned, in the Hybrid EALB approach, routing paths are determined using the Dijkstra-based path component.

B. Implementation Details

The simulation framework was developed in Python. The implementation utilizes SimPy for discrete-event simulation modeling, and NetworkX for graph-based network representation and path computation.

The system consists of the following components:

- **Fog Node Class:** Represents fog nodes, including their processing capacity, current workload, and power consumption rate.
- **IoT Device Class:** Simulates IoT devices that generate tasks and select appropriate fog nodes according to the selected load balancing algorithm.
- **Network Graph Representation:** Models the connectivity between IoT devices and fog nodes. Edges are weighted to represent latency or communication distance.
- **Task Generation Module:** Generates tasks with heterogeneous computational requirements at random time intervals.
- **Load Balancing Algorithms Implemented:**
 - **Hybrid EALB:** Proposed method integrating Dijkstra's shortest path computation with greedy resource allocation.
 - **Greedy:** Selects the least-loaded fog node.

C. Performance Metrics

Both algorithms were evaluated using the following performance metrics:

- **Total Transmissions:** The total number of task allocation attempts, indicating the algorithm's operational activity and adaptability.
- **Total Latency:** The overall delay experienced by all task transmissions during the simulate, calculated using Equation (2)

$$Total\ Latency = \sum_{i=1}^N L_i \quad (2)$$

where: N represents the total number of transmissions.

L_i represents the latency of transmission i .

- **Average Latency:** The mean latency per task transmission, calculated using Equation (3):

$$Average\ Latency = \frac{Total\ Latency}{Total\ Transmissions} \quad (3)$$

Note that latency is a critical performance measure for delay-sensitive fog computing applications.

- **Power Consumption:** The total energy consumed during the simulation. It is a fundamental metric for ensuring sustainability in fog computing systems. It was calculated using Equation (4).

Power Consumption

$$= \sum_{i=1}^N (Task_Load_i * Power\ Rate) \quad (4)$$

where: N represents the number of tasks.

$Task_Load_i$ represents the computational load of task i.

$Power\ Rate$ represents the energy consumption rate of the fog node.

Together, these metrics provide a balanced evaluation of each algorithm's performance by measuring both efficiency (latency-related metrics) and sustainability (energy consumption).

In summary, this structured experimental framework allows thorough evaluation of the Hybrid EALB algorithm and enables detailed comparison across different fog computing deployment scenarios.

V. RESULTS AND DISCUSSION

This section presents a comparative analysis of the proposed Hybrid EALB algorithm and the Greedy algorithm. The evaluation is conducted across multiple deployment scales using performance metrics; power consumption, latency, total transmissions, and overall efficiency.

A. Metric-Based Performance Analysis

The simulation results show the clear differences between the Hybrid EALB and Greedy algorithms in small, medium, and large-scale fog environments. The analysis focuses on transmission count, latency, power usage, and efficiency.

1) Power Consumption: Fig. 4 compares the energy usage of the Hybrid EALB and Greedy algorithms across the evaluated deployment scales.

The results show that the Greedy algorithm consistently consumes slightly less power than the Hybrid EALB algorithm across all scales. When results are averaged over

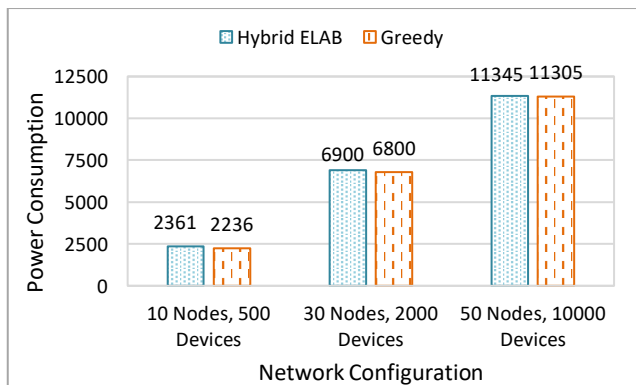


Fig. 4 Comparison of Power Consumption between the Hybrid EALB and Greedy algorithms across Different Scales

multiple simulation runs; the Greedy approach uses approximately 0.03% to 0.85% less energy compared to the Hybrid EALB algorithm. However, differences observed in individual runs are not statistically significant at $p < 0.05$ level. These findings suggest that although the Hybrid EALB algorithm improves routing efficiency; especially in larger deployments; its additional path optimization computations introduce a slight increase in energy consumption.

2) Latency Metrics: Figures 5 and 6 illustrate the total latency and average latency results for both algorithms, respectively. The latency analysis reveals a clear scale-dependent trend. In small-scale environments, the Hybrid EALB algorithm shows slightly higher latency than the Greedy approach. As the system grows to medium and large scales, the Hybrid EALB algorithm increasingly outperforms the Greedy method.

It is clear that in medium and large deployments, both total latency and average latency improvements are statistically significant. This demonstrates that the hybrid routing and resource allocation strategy becomes more effective as system complexity increases.

The Hybrid EALB algorithm performs a higher number of transmissions than the Greedy approach in medium and large deployments. This behavior indicates that the hybrid method performs more frequent routing evaluations and task allocation decisions. Despite the increased transmission volume, these decisions are more efficient, leading to better overall latency performance.

Overall, the results demonstrate that the Hybrid EALB algorithm offers improved latency performance at larger

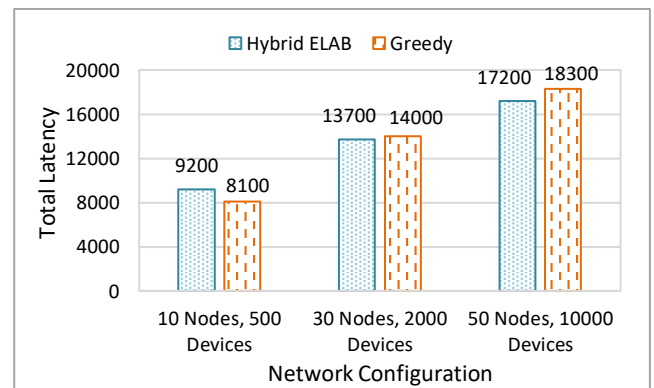


Fig. 5 Comparison of Total Latency between the Hybrid EALB and Greedy algorithms across Different Scales

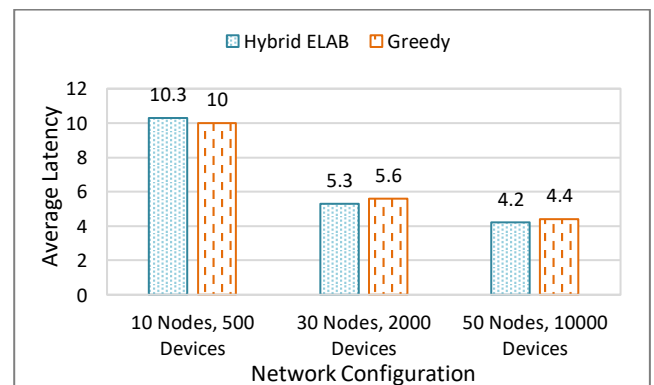


Fig. 6. Comparison of Average Latency between the Hybrid EALB and Greedy algorithms across Different Scales

3) Total Transmissions: Figure 7 presents the comparison of total transmissions generated by both algorithms across different scales.

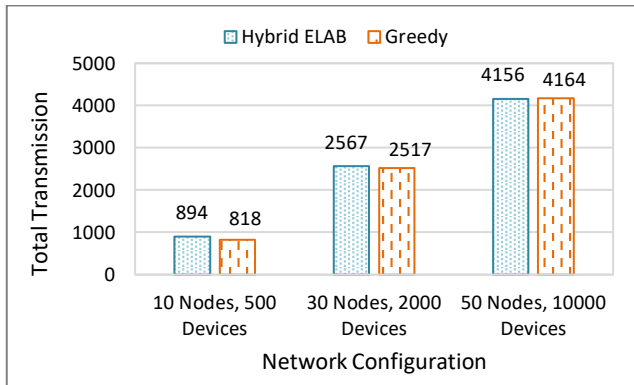


Fig. 7. Comparison of Total Transmissions between the Hybrid EALB and Greedy algorithms across Different Scales

Overall, the results demonstrate that the Hybrid EALB algorithm offers improved latency performance at larger scales, with only a minimal trade-off in power consumption, highlighting its suitability for complex and large-scale fog computing environments.

B. Results Analysis

A detailed statistical analysis provides further insight into the performance differences between the two algorithms across various scales. To ensure statistical robustness, 10 independent simulation runs were conducted for each algorithm at each deployment scale. Statistical significance was evaluated using a two-tailed t-test to compare the performance of the Hybrid EALB algorithm with the Greedy algorithm.

A two-tailed t-test is a statistical method used to determine whether there is a significant difference between two average values without assuming which one is greater. The test is calculated by subtracting the two means and dividing the result by the standard error to obtain the t-value. This t-value is then used with the t-distribution to determine the p-value, which represents the probability that the observed difference occurred by chance. If the p-value is small (commonly less than 0.05), the difference is considered statistically significant [23].

In order to compare the Hybrid EALB algorithm against the Greedy algorithm, Improvement percentage for each performance metric was calculated using Equation (5):

$$\text{Improvement Percentage} = \frac{\text{Greedy Value} - \text{Hybrid Value}}{\text{Greedy Value}} \times 100 \quad (5)$$

Positive improvement values indicate superior performance of the Hybrid EALB algorithm, whereas negative values indicate better performance by the Greedy algorithm. Fig.8 illustrates the overall percentage improvements of the Hybrid EALB algorithm relative to the Greedy approach across all evaluated metrics and deployment scales.

Table III summarizes the statistical evaluation for small, medium, and large environments.

The results presented in Table III demonstrate clear scale-dependent performance characteristics.

- The **Hybrid EALB algorithm** becomes increasingly advantageous as the fog environment grows larger, making it particularly suitable for medium and large-scale systems where low latency is critical.
- The **Greedy algorithm** consistently maintains slightly better energy efficiency, making it more suitable for small or highly power-constrained environments.

In conclusion, selecting an appropriate load balancing strategy should depend on deployment scale and system priorities, particularly whether latency or energy efficiency is the primary concern.

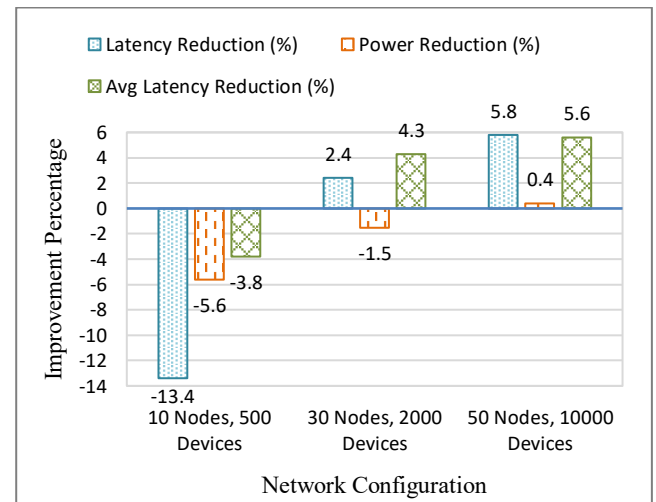


Fig. 8. Percentage Improvement of the Proposed Hybrid EALB Algorithm over the Greedy Algorithm for Total latency, Power Consumption, and Average Latency across Different Scales

VI. SCALE-SPECIFIC PERFORMANCE ANALYSIS

This section examines how the Hybrid EALB and Greedy algorithms perform in small-, medium-, and large-scale deployments.

A. Small-Scale Deployments (Smart Home Environment)

In small-scale environments, clear differences appear between the two algorithms:

- The proposed Hybrid EALB algorithm performs worse in terms of latency, with a 13.4% increase in total latency and a 3.0% increase in average latency compared to the Greedy algorithm.
- Power consumption is 5.57% higher with the Hybrid EALB approach.
- It performs 9.29% more transmissions to achieve comparable results.
- However, it shows a slightly better performance-per-watt ratio.

These results suggest that in small deployments, the additional computations required by Dijkstra-based path optimization create unnecessary overhead. As a result, the simpler Greedy algorithm is more suitable for small, latency-sensitive, and energy-constrained systems.

B. Medium-Scale Deployments (Industrial Factory Environment)

In medium-scale environments, the Hybrid EALB algorithm starts to show measurable benefits:

TABLE III: RESULT ANALYSIS OF KEY METRICS

Scale	Metric	Hybrid Mean	Greedy Mean	Improve. (%)	Significance levels
Small Scale	Total Transmissions	894.0	818.0	-9.29	ns: not significant
	Total Latency	9233.0	8140.0	-13.4	$p < 0.05$
	Power Consumption	2361.2	2236.6	-5.57	ns: not significant
	Average Latency	10.3	10.0	-3.00	$p < 0.05$
Medium Scale	Total Transmissions	2567.0	2517.0	-1.99	ns: not significant
	Total Latency	13718.0	14057.0	2.41	$p < 0.05$
	Power Consumption	6905.2	6806.2	-1.45	ns: not significant
	Average Latency	5.3	5.6	5.36	$p < 0.05$
Large Scale	Total Transmissions	4156.0	4164.0	0.19	ns: not significant
	Total Latency	17268.0	18328.0	5.78	$p < 0.01$
	Power Consumption	11345.0	11304.8	-0.36	ns: not significant
	Average Latency	4.2	4.4	4.55	$p < 0.01$

- It achieves statistically significant improvements of 2.41% in total latency and 5.36% in average latency.
- Power consumption increases slightly by 1.45%.
- Total transmissions increase by 1.99%.
- The performance-per-watt ratio improves compared to the Greedy algorithm.

As network complexity increases, the benefits of path optimization become more evident. The Hybrid EALB algorithm offers a better balance between latency reduction and additional energy consumption.

C. Large-Scale Deployments (Smart City Environment)

In large-scale environments, the Hybrid EALB algorithm demonstrates its strongest advantages:

- Highly statistically significant improvements in latency: total latency decreases by 5.78% and average latency decreases by 4.55%.
- Power consumption increases by only 0.36%.
- The number of transmissions is almost identical between both algorithms (0.19% difference).

These results indicate that the Hybrid EALB algorithm is particularly suitable for large-scale fog systems where response time is critical. At this scale, the latency gains are substantial, while the energy penalty becomes negligible.

D. Discussion and Practical Implications

The overall analysis provides several important insights:

1) *Scale-Dependent Performance Characteristics*: The most important observation is that performance depends strongly on deployment size.

- The Hybrid EALB algorithm becomes increasingly effective as the system scale grows, with the greatest benefits seen in large deployments.
- The Greedy algorithm consistently maintains slightly better energy efficiency across all scales.

Therefore, algorithm selection should depend on system size and performance priorities.

2) *Energy Efficiency Considerations*: Although the differences in power consumption (0.36% to 5.57%) are not statistically significant (at $p < 0.05$ level), even small energy differences can accumulate into considerable operational costs in large and long-term deployments. For highly power-constrained systems, the Greedy algorithm may still be the better choice.

3) *Practical Implementation Recommendations* Based on the results, the following recommendations can be made:

- Small-scale deployments (e.g., smart homes, small offices): The Greedy algorithm is clearly preferable. It provides 13.4% lower total latency, 3.0% lower average latency, and 5.57% lower power consumption.
- Medium-scale deployments (e.g., industrial facilities, campuses): Algorithm selection depends on priorities:
 - For *latency-critical* applications, the Hybrid EALB algorithm is preferable despite its 1.45% higher energy use.
 - For energy-sensitive systems, the Greedy algorithm remains advantageous.
- Large-scale deployments (e.g., smart cities, extensive industrial systems): The Hybrid EALB algorithm is recommended. It reduces total latency by 5.78% and average latency by 4.55%, while increasing power consumption by only 0.36%. In latency-sensitive applications such as emergency response systems, these benefits clearly outweigh the small energy increase.

Overall, these results highlight the importance of matching the load balancing strategy to the deployment scale and application requirements. The trade-off between latency improvement and energy consumption must be carefully considered in fog computing environments.

VII. CONCLUSION AND FUTURE WORK

This paper proposed a Hybrid Energy-Aware Load Balancing (EALB) algorithm designed for fog computing systems. The proposed approach integrates Dijkstra's shortest path algorithm with a greedy resource allocation strategy to balance latency optimization and energy awareness.

Extensive simulations were performed across three deployment scales; small, medium, and large; to evaluate Hybrid EALB's performance under varying network complexities. The results show that the proposed algorithm demonstrates clear scale-dependent behavior, with significant advantages in medium- and large-scale environments.

A. Main contributions

The main contributions of this work can be summarized as follows:

- 1) *Development of a Novel Hybrid Load Balancing Algorithm*: We proposed a Hybrid EALB approach that

integrates shortest-path routing with greedy allocation mechanisms to improve task distribution efficiency in fog computing systems.

- 2) **Identification of Scale-Dependent Performance Characteristics:** The evaluation revealed that the Hybrid EALB algorithm:
 - Achieves superior latency performance in medium- and large-scale deployments.
 - Performs less effectively in small-scale scenarios due to optimization overhead.
- 3) **Analysis of the Latency–Power Trade-off:** The study covered an important trade-off between latency reduction and energy consumption:
 - The Greedy algorithm consistently consumes less power.
 - The Hybrid EALB algorithm reduces latency in larger deployments.
 - The difference in power consumption decreases as system scale increases, narrowing from 5.57% in small-scale environments to 0.36% in large-scale deployments.
- 4) **Comprehensive Multi-Scale Evaluation:** The algorithm was evaluated across different fog scenarios (smart home, industrial environment, and smart city), providing practical guidance for selecting load balancing strategies based on:
 - Deployment scale
 - Performance priorities (latency versus energy efficiency)
 - Application-specific requirements

B. Limitations

Despite the positive results, some limitations exist:

- 1) Simulation-based evaluation using abstract units for latency and power consumption, potentially missing real-world complexities.
- 2) Assumption of relatively stable network conditions, whereas actual fog environments experience dynamic topology changes.
- 3) Simplified power modeling that may not fully capture nuanced energy consumption characteristics across different operational states.
- 4) Limited representation of heterogeneity in node capabilities compared to diverse real-world fog deployments.
- 5) Focus primarily on scaling node/device numbers without exploring other scalability factors like network density or connectivity patterns.

C. Future Work

Several directions for future research can extend and strengthen this work:

- 1) Deploying and evaluating the algorithm in real fog computing testbeds to validate simulation outcomes.
- 2) Incorporating adaptive mechanisms to dynamically respond to changing network conditions and workload variations.
- 3) Integrating machine learning techniques to predict workload patterns and proactively optimize task allocation.

- 4) Extending the model to consider energy harvesting capabilities in edge and fog nodes.
- 5) Embedding security and privacy-aware mechanisms into the load-balancing framework.
- 6) Evaluating performance in highly heterogeneous environments with diverse node capabilities and communication technologies.

ACKNOWLEDGMENT

This work was submitted in partial fulfillment of M.Sc. degree requirements in Cloud Computing Technology Master Program, Faculty of Computers and Artificial Intelligence, Cairo University, Egypt.

REFERENCES

- [1] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli, "Fog computing and its role in the internet of things," in *First Edition of the MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–15.
- [2] Ashkan Paya and Dan C. Marinescu, "Energy-aware load balancing and application scaling for the cloud ecosystem," *IEEE Transactions on Cloud Computing*, vol. 5, no. 1, pp. 15–27, 2017.
- [3] Fatma M. Talaat, Mohamed S. Saraya, Ahmed I. Saleh, Hesham A. Ali, and Shereen H. Ali, "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 4951–4966, 2020.
- [4] A. B. Manju and S. Sumathy, "Efficient load balancing algorithm for task preprocessing in fog computing environment," *Smart Intelligent Computing and Applications*, pp. 291–298, 2018.
- [5] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Latency-aware application module management for fog computing environments," *ACM Transactions on Internet Technology*, vol. 19, pp. 1–21, 2018.
- [6] Dalia Abdulkareem Shafiq, N.Z. Jhanjhi, and Azween Abdullah, "Load Balancing Techniques in Cloud Computing Environment: A Review," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 2, 2021, doi: 10.1016/j.jksuci.2021.02.007.
- [7] Simar Preet Singh, Rajesh Kumar, Anju Sharma, and Anand Nayyar, "Leveraging energy-efficient load balancing algorithms in fog computing," *Concurrency and Computation*, vol. 34, no. 13, 2020.
- [8] Mohamed K. Hussein and Mohamed H. Mousa, "Efficient task offloading for IOT- based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37191–37201, 2020.
- [9] Narayana Potu, Chandrashekar Jatoth, and Premchand Parvataneni, "Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 23, 2021.
- [10] Saurabh Singhal *et al.*, "Energy aware load balancing framework for smart grid using cloud and fog computing," *Sensors*, vol. 23, no. 7, 2023.
- [11] Sadoon Azizi, Mohammad Shojafar, Jemal Abawajy, and Rajkumar Buyya, "Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach," *Journal of Network and Computer Applications*, vol. 201, 2022.
- [12] Alina Buzachis, Antonio Celesti, Antonino Galletta, Jiafu Wan, and Maria Fazio, "Evaluating an Application Aware Distributed Dijkstra Shortest Path Algorithm in Hybrid Cloud/Edge Environments," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 2, pp. 289–298, 2022.
- [13] Shreyas J. *et al.*, "Chaotic grey wolf optimization-based framework for efficient task scheduling in cloud fog computing," *Bulletin of Electrical Engineering and Informatics*, vol. 14, no. 3, pp. 2066–2076, 2025.
- [14] Mohamed Elsayed Abd Elaziz, Laith Abualigah, and Ibrahim Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments," *Future Generation Computer Systems*, vol. 124, no. 9, 2021.
- [15] Ibrahim Attiya, Laith Abualigah, Doaa Elsadek, Samia Allaoua Chelloug, and Mohamed Abd Elaziz, "An Intelligent Chimp Optimizer for Scheduling of IoT Application Tasks in Fog Computing," *Mathematics*, vol. 10, April 2022.

- [16] Asfa Toor *et al.*, “Energy and performance aware fog computing: A case of DVFS and green renewable energy,” *Future Generation Computer Systems*, vol. 101, pp. 1112–1121, 2019.
- [17] Einollah Jafarnejad Ghomia, Amir Masoud Rahmania, and Nooruldeen Nasih Qaderb, “Load-balancing algorithms in cloud computing: A survey,” *Journal of Network and Computer Applications*, vol. 88, pp. 50–71, 2017.
- [18] T Venkata Naga Jayudu, M Rama Krishna Reddy, and C Shoba Bindu, “Greedy Knapsack Based Energy Efficient Routing in WMSNS,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 10, pp. 1455–1460, 2019.
- [19] Yogesh Sharma, Bahman Javadi, Weisheng Si, and Daniel Sun, “Reliability and energy efficiency in cloud computing systems: Survey and taxonomy,” *Journal of Network and Computer Applications*, vol. 74, pp. 66–85, 2016.
- [20] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, and Rajkumar Buyya, “iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environment,” *Software: Practice and Experience*, vol. 47, no. 1, pp. 1275–1296, 2016.
- [21] Arash Bozorgchenani, Daniele Tarchi, and Giovanni Emanuele Corazza, “An energy and delay-efficient partial offloading technique for fog computing architectures,” in *IEEE Global Communications Conference (GLOBECOM)*, Singapore, January 2017.
- [22] Paola G. Vinueza Naranjo, Zahra Pooranian, Mohammad Shojafar, Mauro Conti, and Rajkumar Buyya, “FOCAN: A Fog-supported smart city network architecture for management of applications in the Internet of Everything environments,” *Journal of Parallel and Distributed Computing*, vol. 132, pp. 274–283, 2019.
- [23] Ronald E. Walpole, Raymond H. Myers, Sharon L. Myers, and Keying Ye, *Probability and statistics for engineers and scientists*, 9th edition. Pearson, 2016.