electronics

MDPI

Article

An Optimized, Dynamic, and Efficient Load-Balancing Framework for Resource Management in the Internet of Things (IoT) Environment

Mohammed Shuaib [1], Surbhi Bhatia [2,*], Shadab Alam [1], Raj Kumar Masih [1], Nayef Alqahtani [3,*], Shakila Basheer [4] and Mohammad Shabbir Alam [1]

1 Department of Computer Science, Jazan University, Jazan 45142, Saudi Arabia
2 Department of Information Systems, College of Computer Sciences and Information Technology, King Faisal University, Al-Ahsa 31982, Saudi Arabia
3 Department of Agricultural Systems Engineering, College of Agricultural and Food Sciences, King Faisal University, Al-Hofuf 31982, Saudi Arabia
4 Department of Information Systems, College of Computer and Information Science, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh 11671, Saudi Arabia
* Correspondence: sbhatia@kfu.edu.sa (S.B.); nmalqahtani@kfu.edu.sa (N.A.)

Abstract: Major problems and issues in Internet of Things (IoT) systems include load balancing, lowering operational expenses, and power usage. IoT devices typically run on batteries because they lack direct access to a power source. Geographical conditions that make it difficult to access the electrical network are a common cause. Finding ways to ensure that IoT devices consume the least amount of energy possible is essential. When the network is experiencing high traffic, locating and interacting with the next hop is critical. Finding the best route to load balance by switching to a less crowded channel is hence crucial in network congestion. Due to the restrictions indicated above, this study analyzes three significant issues—load balancing, energy utilization, and computation cost—and offers a solution. To address these resource allocation issues in the IoT, we suggest a reliable method in this study termed Dynamic Energy-Efficient Load Balancing (DEELB). We conducted several experiments, such as bandwidth analysis, in which the DEELB method used 990.65 kbps of bandwidth for 50 operations, while other existing techniques, such as EEFO (Energy-Efficient Opportunistic), DEERA (Dynamic Energy-Efficient Resource Allocation), ELBS (Efficient Load-Balancing Security), and DEBTS (Delay Energy Balanced Task Scheduling), used 1700.91 kbps, 1500.82 kbps, 1300.65 kbps, and 1200.15 kbps of bandwidth, respectively. The experiment's numerical analysis showed that our method was superior to other ways in terms of effectiveness and efficiency.

Keywords: internet of things; resource management; load balancing; energy efficiency; cost efficient

check for updates

Citation: Shuaib, M.; Bhatia, S.; Alam, S.; Masih, R.K.; Alqahtani, N.; Basheer, S.; Alam, M.S. An Optimized, Dynamic, and Efficient Load-Balancing Framework for Resource Management in the Internet of Things (IoT) Environment. *Electronics* **2023**, *12*, 1104. https://doi.org/10.3390/electronics12051104

Academic Editor: Antoni Morell

Received: 10 January 2023
Revised: 9 February 2023
Accepted: 16 February 2023
Published: 23 February 2023

1. Introduction

A network that connects diverse gadgets and programs is the Internet of Things (IoT) paradigm. The main objectives of the Internet of Things are to provide resources based on end-user needs and to transform the massive amounts of heterogeneous data produced by these numerous IoT devices into usable information [1]. Both industry and academia are interested in the IoT, which will improve people's daily lives [2]. As the IoT and mobile communication technologies advance, IoT devices and apps proliferate, providing end users with rich, straightforward services [3,4]. However, IoT device computational and energy demands have increased [5,6]. Because of their limited computing and energy capabilities, IoT devices cannot efficiently service these applications [7]. To compensate for the lack of computing resources, IoT devices can use cloud computing (CC) to outsource computing tasks to cloud servers for processing, which uses much computational power [8,9]. In contrast, long-distance task transfer will necessitate significant transmission delays.

Several obstacles, including a broad range of client requirements, a variety of device types, crucial communication requirements, restricted network bandwidth, limited computing power, operational costs, etc., affect the efficacy of the IoT network. A workable solution to the IoT resource allocation problem (IRAP) will greatly boost system performance, making it one of the most important topics. Resource scheduling and allocation are essential for managing data centers, since they aid in load balancing, resource utilization, and reducing carbon emissions [10]. Data center computers frequently process applications, and intelligent sensor data is frequently transferred to cloud data centers. Because of the ever-increasing resource requirements of IoT applications, which lead to explosively rising energy consumption and computing node performance deterioration due to data transmission and computing node movement [11], the question of how to implement IoT applications has become crucial. Fog computing shifts the processing of IoT applications to the network's edge rather than to cloud platforms.

In IoT systems, the sensed data from the sensors are sent and stored in the cloud layer for further processing, but this process consumes a lot of resources, and hence the issue of latency arises. A cloud-based IoT environment cannot handle these latency-sensitive conditions [12]. Fog computing is a new paradigm of allocating processing and storage resources to resource-constrained IoT devices. In fog computing, several small devices with a small computing capacity named edge are deployed near the IoT sensor layer. The processed data is then forwarded to the cloud layer for final storage, which resolves the latency issue and provides more computing power than basic IoT sensors. All devices or sensors in a fog computing system can delegate their tasks to nearby fog nodes where high computational power or storage is required instead of forwarding them to the cloud module at a more considerable distance [13]. Fog computing can considerably cut the communication time between IoT nodes and computer servers compared to cloud computing. As a result, fog computing adoption in this sector is critical.

Fog computing has become one of the most effective models for processing IoT applications as an outcome of the growth of IoT applications. IoT applications are run in the fog environment by edge computing nodes, intermediary computing nodes, and actual computers on cloud platforms [14]. This study proposed a dynamic resource allocation strategy called DEELB to facilitate dynamic load balancing for all fog and cloud computing nodes. Fog computing shifts processing capacity away from centralized data centers and closer to the network's edge to better serve IoT applications.

Allocating resources to the cloud should prevent bottlenecks, maximize consumption, and lessen the impact of under and over-utilization [15]. Allocating resources in a fog system, where both cloud and fog computing nodes respond to applications, is challenging [16]. In contrast to cloud computing nodes, fog computing nodes are located throughout the edge network rather than in a central data center [17]. The resource needs of computer nodes shift due to the demands of IoT applications on their processors, storage spaces, and data transfer rates. Load balancing necessitates resource allocation to accommodate the unpredictable resource needs of IoT applications.

Load balancing in fog computing is possible thanks to the "Dynamic Resource Allocation Strategy" (DRAM) technique. Optimizing the distribution of work across all available compute nodes is essential in cloud and fog computing environments. The four steps of the DRAM approach are:

1. Segmenting fog processes
2. Identifying nodes with extra storage space
3. Dynamically allocating resources to various parts of the fog web
4. Balancing workloads with global resource allocation

The DRAM technique effectively distributes the load across different types of computer nodes in cloud and fog settings; however, it was devised without considering critical considerations like energy consumption and processing costs.

Because IoT devices do not have direct access to a power source, they are generally powered by batteries. It is frequently brought on by geographic locations that make it impossible to access the electrical network. Finding methods to guarantee minimal energy use by IoT devices is indispensable. Finding and communicating with the next hop is crucial when the network is under heavy traffic. Hence, finding the best route is essential to perform load balancing by selecting the less busy channel when traffic congestion occurs in the network. Figure 1 shows a basic representation of load balancing in fog computing architecture. In IoT systems, load balancing, energy use, and computing cost are three major issues. Therefore, load balancing, reducing operational costs, and power consumption are essential topics and issues. We want to lower the energy consumption of IoT devices. One of the most significant issues is the IoT resource allocation problem (IRAP), and a workable solution will significantly enhance system performance [18]. This paper discusses and proposes a solution to resolve the issues of load balancing, energy use, and computing cost in the form of dynamic energy-efficient load balancing (DEELB). Further, its effectiveness is evaluated using simulated experiments. Our main contributions can be summarized as follows:

- This paper discusses three massive problems: load balancing, energy use, and computing cost.
- Present a particular technique in this work called dynamic energy-efficient load balancing (DEELB) to handle these IoT resource allocation issues.

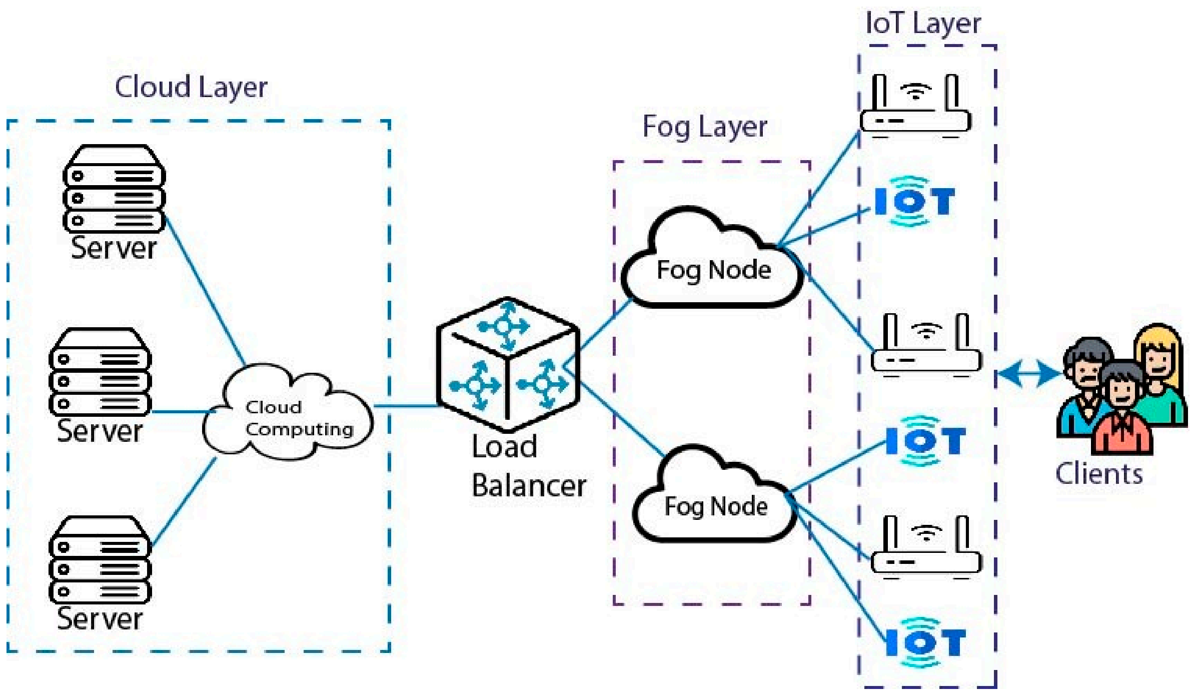

Figure 1. Representation of Load Balancing in Fog Computing.

This article continues as follows: Section 2 investigates previous studies and assesses their strengths and weaknesses. The proposed system is described in Section 3. Section 4 compares the DEELB–IoT to the existing system using performance metrics. This section summarizes and illustrates the findings using graphs from previous research. Section 5 concludes with a summary of all proposed and future work.

Background

Users can exchange data and access a variety of resources by using the technology known as cloud computing. Users are only charged for the resources that they actually use. Public data and distributed resources are stored in the cloud, and data storage capacity is growing quickly. As a result, load balancing is the main task in the cloud environment. By avoiding overloads, load balancing assists in the distribution of dynamic workload.

The “cloud computing” distributed computing approach is centered on granting simple access to virtualization hardware and software infrastructure to a broad range of users. Web services, applications, networking, and distributed computing virtualization are all included [19]. Users of today’s parallel, distributed, and virtualized computing systems are becoming increasingly interested in cloud computing. It seems to be a popular choice for giving affordable and straightforward access to particular IT resources. Because of virtualization, cloud computing may use the same physical infrastructure to service a big client base with diverse processing needs [20]. The major issue with the cloud environment is the latency that can not be suitable for time-critical systems such as healthcare.

Edge computing tries to move the processing capabilities at the edge node instead of depending on the cloud layer. Another paradigm, named fog computing, has been proposed to resolve the latency issue in cloud-based IoT systems. There are many suggestions that fog computing is an implementation of edge computing [21], and some researchers consider fog computing as an advanced setup of edge computing [22]. Fog computing is a decentralized setup that acts as an intermediary between the cloud and IoT sensor layers. Fog computing can provide storage and processing capabilities to the IoT layer in a decentralized manner that is closer to the sensors. The fog layer can process and temporarily store the sensed data before transmitting it to the cloud layer for final storage and processing, which helps greatly in reducing the time delay and solves the latency problem [22]. Still, the fog systems can suffer the same bottleneck if a single node is requested for more and more services compared to other nodes; when a heavy load comes, it will create a similar latency issue as that discussed for cloud computing. Load balancing is the solution to avoid heavy dependence on a single fog node and distribute the task properly [23]. A “load balancing” process involves distributing a heavier processing load among a smaller number of processing nodes to improve system performance. It is the practice of distributing load among many distributed system nodes in a distributed system environment to increase resource utilization and task response time. A good load-balancing algorithm should avoid overloading or underloading nodes. Due to extra restrictions like security, reliability, throughput, and other factors, choosing a load-balancing technique in a cloud computing environment is challenging. In order to reduce job response times, a load-balancing method for cloud computing distributes the entire system load. The algorithm must also guard against overloading any node [24].

In order to achieve a high level of user satisfaction and proper resource utilization, load balancing assists in the equitable allocation of computing resources. Low resource consumption is made possible by effective load balancing and high resource utilization. Fault tolerance can be implemented with the help of this.

Load balancing is a technique that has assisted networks and resources in providing the highest throughput while having the shortest response time. In cloud computing, load balancing occurs on two levels.

- A mapping is created at the virtual machine level between applications loaded in the cloud on the virtual machine. The load balancer assigns the required virtual machine to actual machines to balance the load on various PC programs [25].
- A virtual machine-to-host resource mapping that enables the host to handle numerous incoming application requests.

2. Related Literature

Liaqat et al. [26] proposed using the virtual machine (VM) approach for load balancing in the cloud computing environment. This study reviewed the OpenStack scheduler using the VM approach. It considered CPU utilization as a parameter for VM deployment and shows that it helps in better load balancing. A wireless energy-powered cooperative relay network is the subject of research by Lan et al. [27]. It suggests that the energy efficiency of a novel buffer-aided power grid can be asymptotically improved using the Lyapunov optimization technique. Yuan et al. [28] presented a Geography-Aware Task Scheduling (GATS) approach for task scheduling to maximize resource utilization for data centers.

It suggested that it will provide a high throughput that will incidentally result in more profitability. Mishra et al. [29] reviewed the different types of load-balancing approaches in the cloud computing environment. Further, it examines the various parameters used for validating the load balancing parameters by different researchers and presents their own parameters for evaluating load balancing in cloud computing.

Kaur et al. [30] investigated resource optimization issues for single-hop network topology and a broad cooperative multi-hop network architecture using a range of cooperative methodologies. He creates an ideal strategy for each source-to-destination communication link based on a dual optimization framework, decreasing transmit power through channel allocation and the best choice of relays while sustaining quality of service (QoS) standards. He provides an energy-efficient power-routing method and generates closed-form equations for the outage probability of the cooperative relaying protocol in order to maximize data transmission.

According to Rui et al. [31], load balancing is essential in any distributed environment, including the cloud and the IoT. This study has provided a new method to resolve this problem using the Simplified Swarm Optimization (SSO) algorithm, given that the NPhard problem and the SSO algorithm is a potent meta-hybrid algorithm inspired by shark-catching behavior and their capacity to recognize and perceive the smell of the fishing line even from a distance. The study also proposed using fuzzy logic and Dynamic Voltage and Frequency Scaling (DVFS) for managing the load distribution to save energy consumption for IoT nodes if their frequency decreased.

Wang et al. [32] review the applications of the Industrial Internet of Things (IIoT) and security issues. The study further proposes a lightweight signature protocol that can be used to manage the load and provide security that will consume a fraction of energy compared to traditional systems. Iwendi et al. [33] review the IoT sensors' energy consumption issues and highlight the challenges. Further, the study proposes a mechanism using Whale Optimization Algorithm (WOA) for cluster head selection and load balancing to minimize energy and resource consumption.

Lin et al. [34] highlight the importance of energy efficiency and security in satellite communication. Further, they discuss the role of Reconfigurable Intelligent Surfaces (RIS) in minimizing the total transmission power using base stations that focus on 5G and 6G communication. Lin et al. [35] further review this aspect and presents refracting RIS approach for resolving the issue of high energy consumption by base stations.

According to Xiang et al. [36], local edge servers implement an IoT service. Because edge servers are limited in terms of resources and energy, he should proceed with caution when implementing related Artificial Intelligence of Things (AIoT) services, especially when they can be combined to create complex applications. The author investigated the relationship between MEC-based service system energy costs and AIoT application efficiency in this study. He used directed acyclic graphs to model complex AIoT applications (DAGs). He also conducted extensive experiments using real simple or complex workflow data sets, such as the Alibaba Cloud and the Montage project, to evaluate the results of our strategy. The findings demonstrated that the proposed approach could successfully generate just solutions, and all factors influencing the findings were carefully considered.

Lim [37] reviewed the overview of cloud computing and fog computing concepts and migration issues of IoT nodes. It further discusses the aspects of task scheduling in cloud computing reliability by using fog computing orchestration and managing the resources. Costa et al. [38] reviewed the issues with cloud computing and the advantages of the fog computing paradigm. They review the aspects of fog orchestration for managing the computational resources of heterogeneous IoT nodes by various researchers. They further provided possible challenges in the domain. Table 1 below summarizes the current works in the domain of DEELB in an IoT environment.

Table 1. Literature Review on DEELB in the IoT environment.

Author	Techniques/Methodology	Year
Liaqat et al. [26]	Dynamic load balancing using VM for Cloud	2019
Lan et al. [27]	The buffer-aided transmission system uses power-splitting relaying which consumes less energy	2019
Yuan et al. [28]	Geography-Aware Task Scheduling	2020
Mishra et al. [29]	Review of load balancing approached in cloud computing	2020
Rui et al. [31]	Shark Smell Optimization algorithm (SSO)	2020
Wang et al. [32]	Lightweight Certificateless Signature (CLS) protocol	2021
Kaur et al. [30]	Techniques for managing loads and resources in a diverse and homogenous cloud environment	2021
Iwendi et al. [33]	Whale Optimization Algorithm (WOA)	2021
Lim [37]	Fog computing orchestration	2021
Lin et al. [34]	Reconfigurable Intelligent Surface (RIS)	2022
Xiang et al. [36]	Directed Acyclic Graphs (DAGs)	2022
Costa et al. [38]	Systematic literature review of fog orchestration	2022

3. Proposed System

Load balancing in a fog environment might be challenging; thus, we provide a DEELB strategy to do just that. It is an efficient approach for balancing the load on the system by allocating resources to users based on their energy consumption and processing requirements. It is through the following steps that Figure 2's DEELB plan is applied:

1. The user will send the Task Manager "n" tasks. Based on the instructions it provides, each activity's computing cost and energy consumption are seen as being preset.
2. In this scenario, the information provider will register n resources via the Resource Registration and Inventory Protocol. Each task must also have specified computational expenses and energy requirements for each resource based on the work instructions.
3. The Tasks Manager sends task information to the Resource Scheduler. Jobs are arranged in descending order based on cost and energy calculations: S1S2R2 > R3 . . . Rn.
4. RIP provides information about resources that are readily available to the Resource Scheduler. The available resources are arranged according to computational cost and energy consumption: R1 > R2 > R3 . . . Rn.
5. The Resource Engine needs data from the Resource Scheduler regarding jobs and available resources to carry out its task.
6. When tasks are assigned to available resources, the Resource Engine starts the execution process and updates the Resource Load Manager on its status.
7. After a process, the Resource Load Manager will inform the Resource Power Manager of the current resource load.
8. The Resource Power Manager decides whether or not the resource is powered on based on the resource's usage.
9. If everything goes as planned, the Resource Engine will collect and provide the data to the user.

When implementing the suggested DEELB technique, jobs are forwarded to the Tasks Manager. Information aggregators track hardware and software in a cloud data center. The Resource Scheduler receives all of this data regarding jobs and available resources. Tasks are prioritized by the resource scheduler in light of historical data and projected demands.

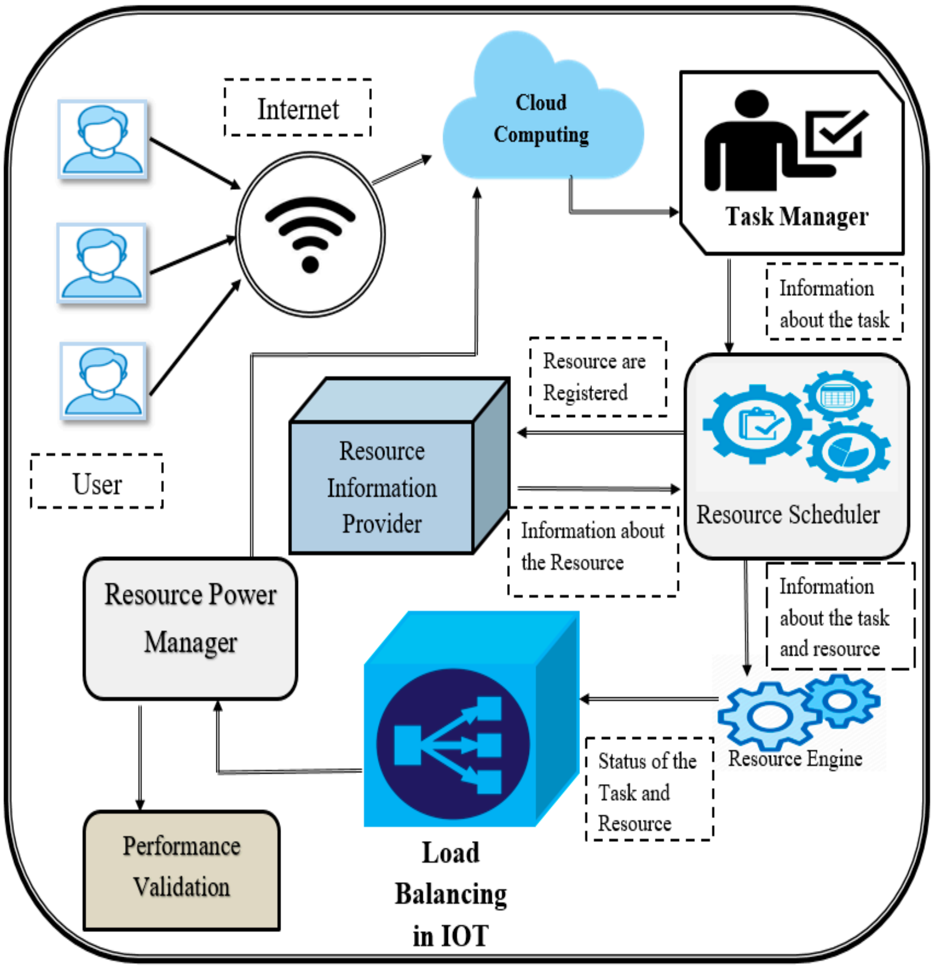

Figure 2. The proposed DEELB method's architecture.

The Resource Engine distributes tasks among the team's active members using the data from the Resource Scheduler. When a task is finished, the Resource Power Manager and the Resource Load Manager are informed about the resources that are still available. Powering and disabling resources fall under the purview of the Resource Power Manager. When a job is finished, the Resource Engine notifies the client of the outcome. Figure 2 displays the suggested DEELB approach's block diagram. The sequence in which the user submits a task to the Task Manager, who then obtains task data from the Resource Scheduler, is shown in detail in the Figure 2. After the Resource Scheduler requests details about registered resources, the Resource Information Provider responds with details about the resources that are now available. The Resource Engine receives job and resource data from the Resource Scheduler.

The Resource Load Manager receives that data from the Resource Engine to keep the resource power management up-to-date on job and resource status. Managed resources in the cloud fall under the purview of this position. The task is given to the User to complete. This operation is carried out in a secure and risk-free manner.

3.1. Energy Efficiency Problem Formulation

This optimization problem's purpose is to maximize total energy efficiency (EE^T). As illustrated below, the restrictions are the same as in the preceding optimization problem:

$$\max_{a_k,u,n,P_k,u,n} \quad EE^T \tag{1}$$

Energy efficiency is a metric that assesses how well resources are used. In bits/Hz/joule, the spectral efficiency over power consumption is assessed. This is, in some senses, a multi-objective problem in which both throughput and power are maximized [39]. As a result, this challenge aims to transport data at the maximum possible data rate while spending the least amount of power while meeting the constraints.

3.2. DEELB Strategy Components

The following components make up the DEELB strategy:

(1) User: A User is a person or thing that makes requests for things to be done. A task may be submitted simultaneously by one or more people.
(2) Task Manager: User tasks are collected by the Task Manager and sent to the Resource Scheduler for scheduling. The Task Manager is in charge of making sure that the tasks submitted are legitimate. The Task Manager also keeps track of how much energy and money each task uses in the calculation.
(3) Resource Information Provider: In addition to registration, the Resource Information Provider (RIP) also disseminates knowledge about the resources that are accessible. Additionally, the data includes the resources' computed costs and energy usage.
(4) Resource Scheduler: The Resource Scheduler gathers resource information from the Resource Information Provider and task information from the Task Manager. The Resource Scheduler subsequently sorts the submitted jobs.

3.3. Load-Balancing Model in the IoT Environment

Fog computing is a liberating concept for balancing workloads in a cloud context. The resources could not be utilized to their full capacity because of execution time and compute node property variances in the fog. We put much effort into preventing the compute nodes in the fog environment from being overworked or underworked.

If $t_m(1 \le m \le M)$ is assigned to the computing node $q_n(1 \le n \le N)$ at time instant s, then let $K_m^n(s)$ be the binary variable used to determine (2):

$$K_m^n(s) = \begin{cases} 1, & if\ t_m\ is\ assigned\ to\ q_n \\ 0, & otherwise \end{cases} \tag{2}$$

The fog service's distribution judgment estimates the resource utilization for the nth computing node q_n at time s in (3):

$$bx_n(s) = \frac{1}{a_n} \sum_{m=1}^{M} K_m^n(s) \cdot b_m \tag{3}$$

where a_n is the computing node q_n's capacity. Using the service distribution, we can calculate how many services are running at a time s in (4):

$$mt_n(s) = \sum_{m=1}^{M} K_m^n(s) \tag{4}$$

For each kind of processing node, the load-balance variation must be specified. In order to provide fog services, assume there are W distinct categories of cloud and fog nodes (5).

Resource utilization is intrinsically linked to load-balancing fluctuations. In order to quantify resource consumption, we must ascertain the number of compute nodes of type w that are actively being used at instant s (5):

$$c_w(s) = \sum_{n=1}^{N} f_n(s) \cdot \beta_w^n \tag{5}$$

where β_w^n is used to judge whether q_n is a type w computing node, which is described in (6), and $f_n(s)$ is used to judge whether q_n is empty at s, presented in (7).

$$\beta_w^n = \begin{cases} 1, & if\ q_n\ is\ a\ \ wth\ type\ computing\ node, \\ 0, & otherwise. \end{cases} \tag{6}$$

$$f_n(s) = \begin{cases} 1, & if\ mt_n > 0, \\ 0, & otherwise \end{cases} \tag{7}$$

At time s, the resource consumption for computing nodes of type w is calculated by (8):

$$BX_w(s) = \frac{1}{c_w(s)} \sum_{n=1}^{N} \sum_{m=1}^{M} K_m^n(s) \cdot bx_n(s) \cdot f_n(s) \tag{8}$$

For this additional purpose (variance of load balance of at time q_n instant s), the load-worth balance will rise or fall depending on how resources are used. The standard deviation of q_n can be determined by using the following Equation (9):

$$dr_n(s) = \left(bx_n(s) - \sum_{w=1}^{W} BX_w(s) \cdot \beta_w^n \right)^2 \tag{9}$$

The standard deviation of all compute nodes of type w is summed to get an average of ten (10).

$$DR_w(s) = \frac{1}{c_w(s)} \sum_{n=1}^{N} \sum_{m=1}^{M} K_m^n(s) \ \cdot \ dr_n(s) \cdot f_n(s) \tag{10}$$

Load-balance variance in cloud and fog conditions during the execution period $[S_0, S]$ might be determined using (11):

$$DR_w = \frac{1}{S - S_0} \int_{T_0}^{T} DR_w(s)\ rs \tag{11}$$

Given these facts, min $\ DR_w, \quad \forall w = 1, \ldots, W$, and $t \cdot s \cdot c_w(s) \leq \sum_{m=1}^{M} \beta_w^n$, the following expression is obtained for the difficulty of reducing the load-balance variance:

$$\sum_{m=1}^{M} K_m^n b_m \leq a_n, \tag{12}$$

Here, $\sum_{m=1}^{M} b_m$ represents the resource requirements for all services, $\sum_{n=1}^{N} a_n$ displays the resource requirements for all services assigned to the type m computer node, and Formula (12) depicts the computing node capacity. We can see from (12) that the main objective of this study was to resolve an optimization problem with several constraints [36].

4. Result and Discussion

This section contains in-depth information on simulation toolkits and application modeling. CloudSim [31] was used to help model and create fog and cloud computing environments. It evaluates the effectiveness of cloud resource management rules for various application and service types in varied load, energy performance, and system size scenarios. Data centers and numerous nodes/VMs can also be created. It offers “large-scale cloud computing infrastructure modelling and simulation” and a single physical data center built on computer nodes. It is a stand-alone framework for “modelling data centres, scheduling, service brokers, and allocation algorithms”. We propose a simulation-based project to assess the provided solution, DEELB, using the CloudSim cloud computing simulator. Because DEELB is based on the most pertinent existing work, DRAM, in which comparable

CloudSim settings were used, its significance could be assessed by simulation. However, we will test the suggested model using real datasets on a testbed. We are currently comparing the performance of our approach to that of other methods on the same datasets.

4.1. Experiment Setup

Any cloud scheduling heuristic's performance is evaluated using a variety of methodologies, including analytical, experimental, and simulation techniques. The challenge with experiments on actual clouds is the cost and frequent need for a specialist for the setup. Furthermore, using the cloud platform for simulations results in high monitoring costs. The analytical methods' drawback is that they cannot assess the suggested scheduling schemes in their entirety. Tools for simulation are frequently used to evaluate the effectiveness of contemporary approaches. This study used Cloudsim to compare the suggested technique to the four available contemporary approaches (EEFO, DEERA, ELBS, and DEBTS). The simulations were run on a system outfitted with an Intel Core i5-8500 (TM) CPU (3.00 GHz) and 20 GB of RAM.

4.2. Evaluation Parameters

(a) Delay

Table 2 and Figure 3 describes the delay analysis of the DEELB methodology using known methods. The data clearly show that the DEELB method outperformed the other techniques in all aspects. For example, with a large number of operations, the DEELB method has only 10.352 s of delay, while the other existing techniques like EEFO, DEERA, ELBS, and DEBTS have a delay of 26.832 s, 22.851 s, 17.877 s, and 13.453 s, respectively.

Table 2. Delay analysis of the DEELB method with existing systems.

No. of Operations	EEFO	DEERA	ELBS	DEBTS	DEELB
10	26.832	22.851	17.877	13.453	10.352
20	27.356	23.762	18.965	13.211	11.467
30	27.931	22.565	19.453	14.671	11.132
40	29.652	24.683	20.732	15.872	11.763
50	28.781	25.762	19.954	16.132	12.852

Similarly, for 50 operations, the DEELB method has a delay of 12.852 s, while the other existing techniques such as EEFO, DEERA, ELBS, and DEBTS have 30.781 s, 25.762 s, 21.654 s, and 16.132 s of delay, respectively.

(b) Response Time

Table 3 and Figure 4 evaluate the DEELB technique's response time compared to that of already-existing approaches. According to the results, the DEELB method is head and shoulders above the rest of the approaches. For example, with a large number of operations, the DEELB method took only 3.675 s to respond, while the other existing techniques like EEFO, DEERA, ELBS, and DEBTS had a response time of 8.872 s, 8.165 s, 6.256 s, and 5.111 s, respectively. Similarly, for 50 operations, the DEELB method had a response time of 4.732 s, while the other existing techniques like EEFO, DEERA, ELBS, and DEBTS had 9.652 s, 8.653 s, 7.656 s, and 6.132 s of response time, respectively.

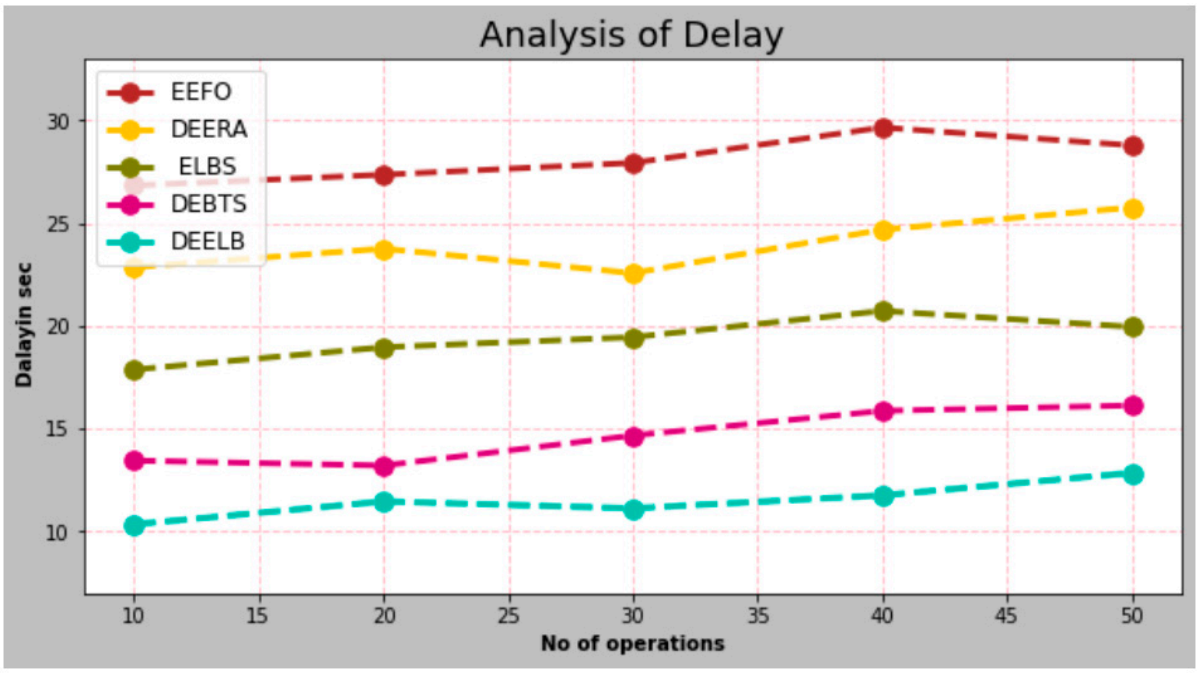

Figure 3. Delay analysis of the DEELB method with existing systems.

Table 3. Response-time analysis of the DEELB method with existing systems.

No. of Operations	EEFO	DEERA	ELBS	DEBTS	DEELB
10	8.872	8.165	6.256	5.111	3.675
20	9.165	8.675	6.326	5.675	3.436
30	9.321	8.132	6.123	5.732	4.723
40	9.956	8.756	7.145	5.832	4.263
50	9.652	8.653	7.656	6.132	4.732

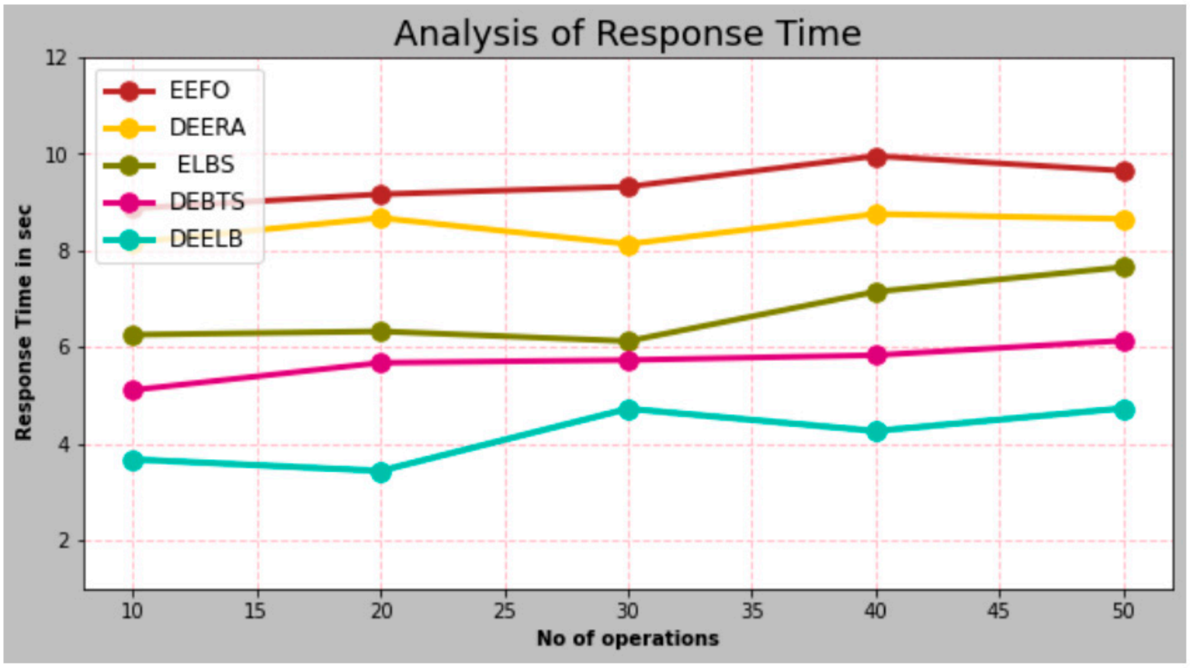

Figure 4. DEELB technique response time analysis with existing systems.

(c) Packet Loss Ratio

Table 4 and Figure 5 explain the packet loss ratio analysis of the DEELB approach in comparison to other existing techniques. The data clearly demonstrate that the proposed method had the lowest PLR in all aspects when compared to the other methods. For example, with 10 operations, the proposed method had a PLR of 30.17%, 49.67%, 44.93%, 39.45%, and 34.78% for EEFO, DEERA, ELBS, and DEBTS, respectively. The DEELB method had greater performance with less PLR. Similarly, with 50 operations, the proposed method had 33.45% PLR, whereas the methods like EEFO, DEERA, ELBS, and DEBTS had PLR of 52.85%, 48.65%, 43.72%, and 38.65%, respectively.

Table 4. Packet loss ratio analysis of the DEELB method with existing systems.

No. of Operations	EEFO	DEERA	ELBS	DEBTS	DEELB
10	49.67	44.93	38.45	34.78	30.17
20	50.73	43.65	40.23	35.61	29.65
30	48.72	46.83	41.83	35.92	31.82
40	51.65	47.91	40.65	37.88	32.17
50	50.85	45.65	43.72	38.65	33.45

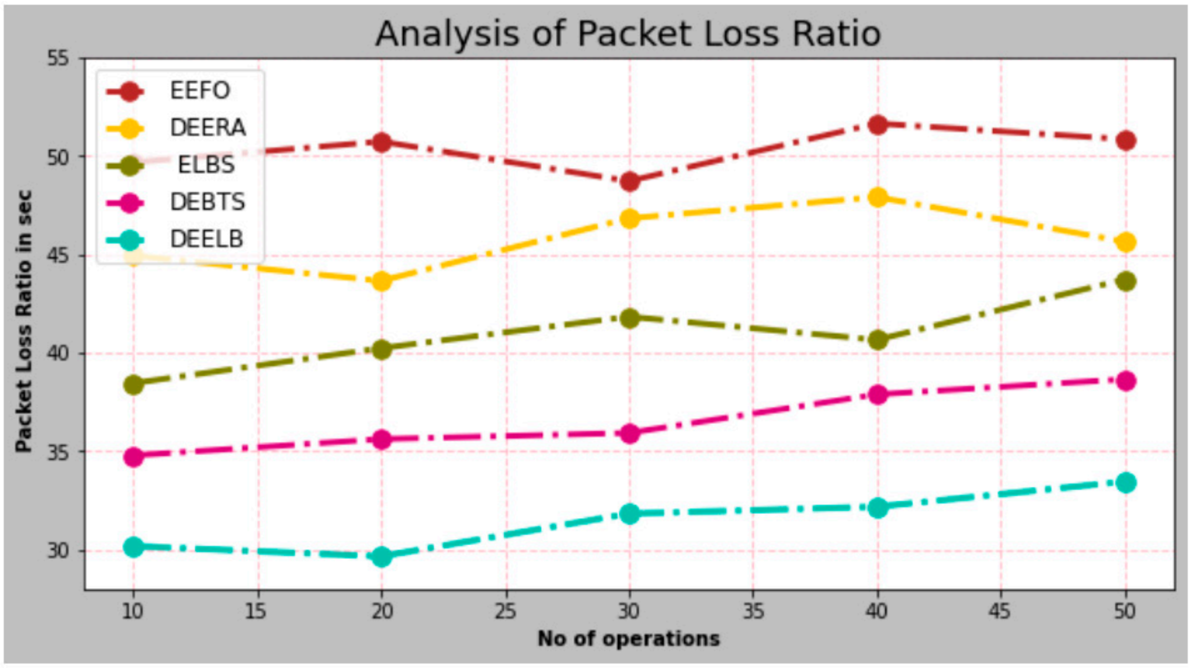

Figure 5. Packet loss ratio analysis of DEELB method with existing systems.

(d) Bandwidth Utilization

The bandwidth utilization analysis of the DEELB technique using current techniques is shown in Table 5 and Figure 6. The data unmistakably demonstrate that the DEELB method performed better than the other techniques in every way. For example, with 10 operations, the DEELB method took only 900.32 kbps of bandwidth utilization, while the other existing techniques such as EEFO, DEERA, ELBS, and DEBTS had a bandwidth utilization of 1500.97 kbps, 1400.65 kbps, 1200.65 kbps, and 1000.82 kbps, respectively. Similarly, for 50 operations, the DEELB method had a bandwidth utilization of 990.65 kbps, while the other existing techniques such as EEFO, DEERA, ELBS, and DEBTS had 1700.91 kbps, 1500.82 kbps, 1300.65 kbps, and 1200.15 kbps of bandwidth utilization, respectively.

Table 5. Bandwidth utilization analysis of the DEELB method with the existing systems.

No. of Operations	EEFO	DEERA	ELBS	DEBTS	DEELB
10	1500.97	1400.65	1200.65	1000.82	900.32
20	1600.12	1400.91	1200.83	1000.96	900.75
30	1600.72	1500.15	1200.96	1100.12	910.65
40	1600.85	1500.65	1300.13	1100.65	970.12
50	1700.91	1500.82	1300.65	1200.15	990.65

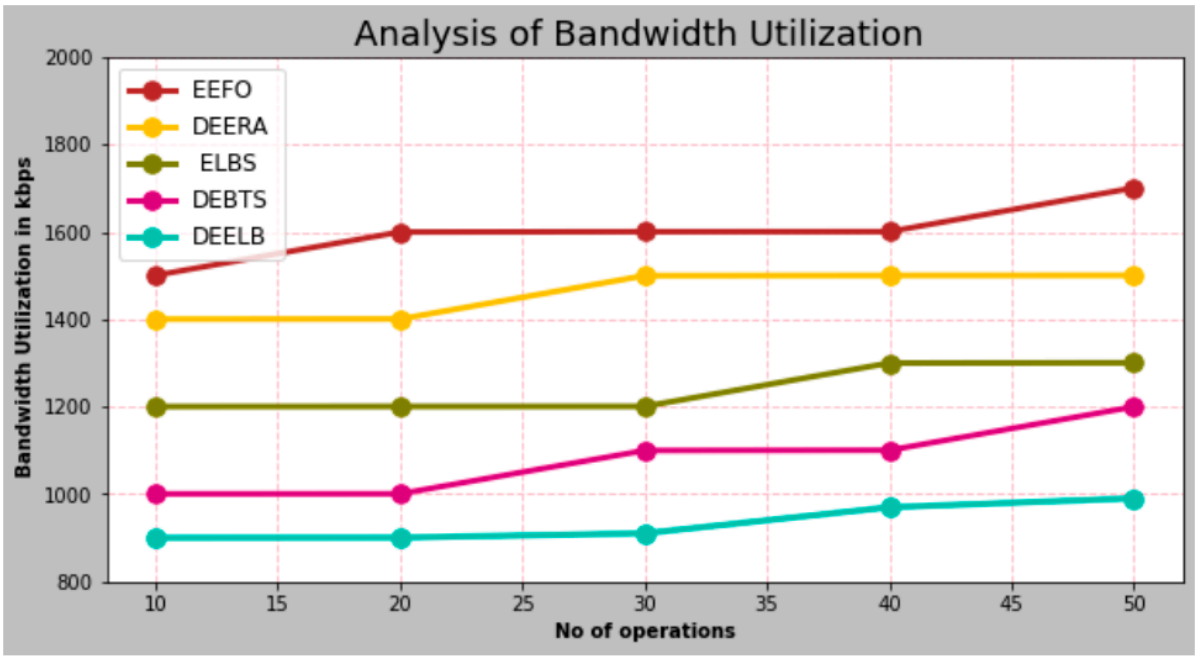

Figure 6. Bandwidth utilization analysis of DEELB method with existing systems.

(e) Throughput

The throughput study of the DEELB methodology with existing methods is described in Table 6 and Figure 7. The data clearly show that the proposed method outperforms the other techniques in all aspects. For example, with operations, the DEELB method has a throughput of 1373.98 kbps, while the other existing methods such as EEFO, DEERA, ELBS, and DEBTS have a throughput of 850.18 kbps, 935.86 kbps, 1083.36 kbps, and 1195.75 kbps, respectively.

Table 6. Throughput analysis of the DEELB method with the existing systems.

No. of Operations	EEFO	DEERA	ELBS	DEBTS	DEELB
10	850.18	935.86	1083.36	1195.75	1373.98
20	826.95	976.48	1107.48	1209.56	1392.74
30	850.84	952.32	1113.65	1200.85	1380.67
40	806.72	1040.57	1178.74	1284.86	1470.73
50	913.87	1070.65	1152.85	1346.29	1500.65

Similarly, with 50 operations, the proposed method had 1500.65 kbps of throughput while the other existing methods, EEFO, DEERA, ELBS, and DEBTS, had a throughput of 913.87 kbps, 1070.65 kbps, 1152.85 kbps, and 1346.29 kbps, respectively. This proves that the DEELB technique has higher performance with greater throughput.

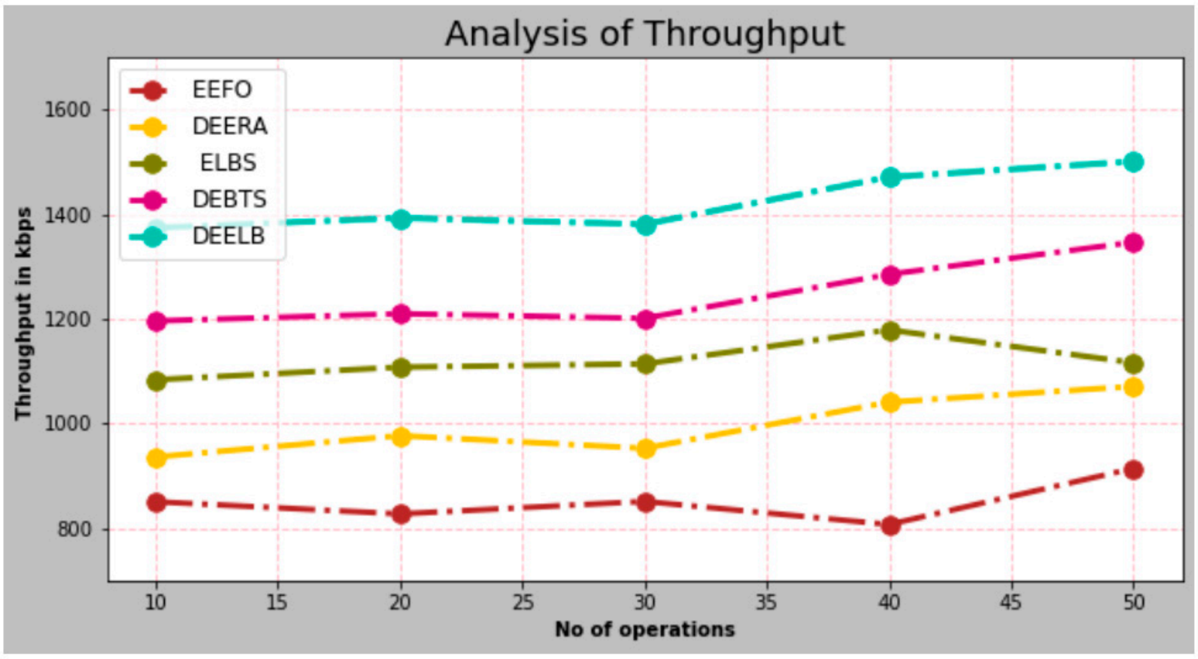

Figure 7. Throughput analysis of the DEELB method with existing systems.

(f) Reliability

Figure 8 and Table 7 illustrate a comparative reliability examination of the DEELB approach with other existing methods. The figure shows that the IoT approach has resulted in higher performance with reliability. For example, with 10 operations, the reliability was 88.178% for DEELB, whereas the EEFO, DEERA, ELBS, and DEBTS models obtained reliability of 80.153%, 82.657%, 84.297%, and 85.612%, respectively. However, the DEELB model has shown maximum performance with a varied number of operations. Similarly, under 50 operations, the reliability value of DEELB is 91.652%, while it is 81.749%, 84.132%, 85.477%, and 87.322% for EEFO, DEERA, ELBS, and DEBTS models, respectively.

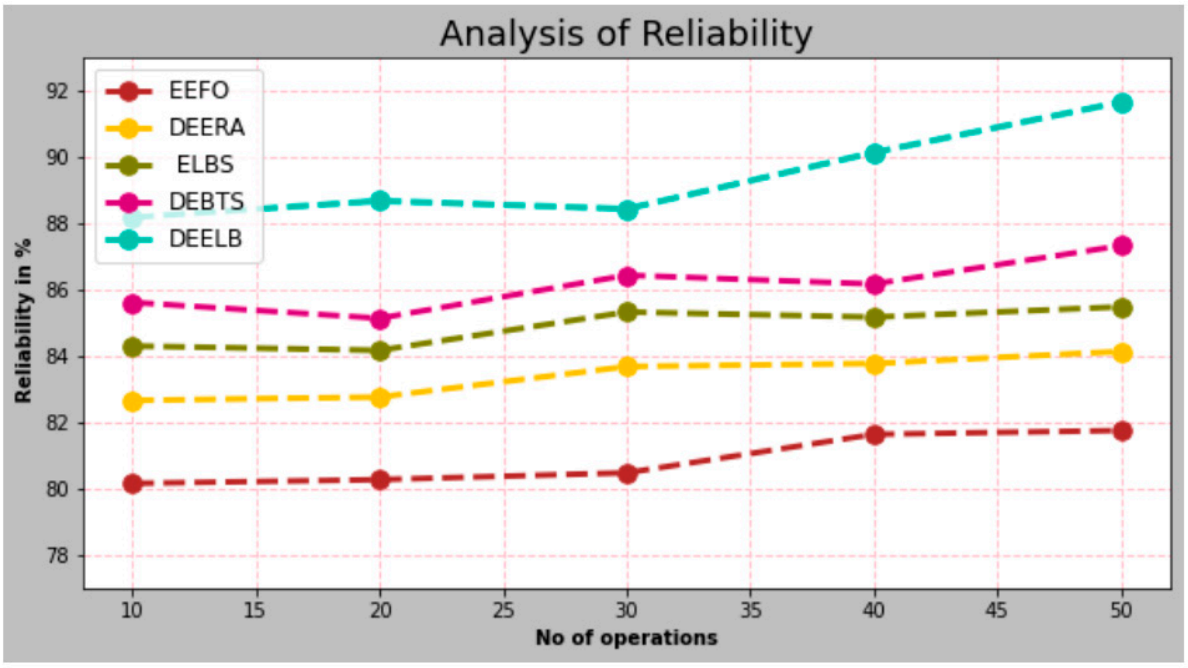

Figure 8. Reliability analysis of the DEELB method with existing systems.

Table 7. Reliability analysis of the DEELB method with existing systems.

No. of Operations	EEFO	DEERA	ELBS	DEBTS	DEELB
10	80.153	82.657	84.297	85.612	88.178
20	80.267	82.757	84.165	85.132	88.675
30	80.472	83.678	85.322	86.432	88.432
40	81.632	83.769	85.168	86.167	90.132
50	81.749	84.132	85.477	87.322	91.652

(g) Packet Delivery Ratio

The ratio of the total number of packages sent from the sender node to the total number of packages received at the destination is what is meant when someone uses the term “package delivery ratio”. A comparison of the DEELB analysis with other approaches, including EEFO, DEERA, ELBS, and DEBTS, is shown in Table 8 and Figure 9. The PDR for 10 nodes for the DEELB method was 89.97%, whereas the PDR for earlier processes such as the EEFO, DEERA, ELBS, and DEBTS processes was 78.34%, 88.32%, 72.32%, and 82.76%, respectively. The PDR of the suggested technique was 93.65% with 50 nodes, compared to the other current methods’ range of 74.21% to 89.29%.

Table 8. Packet delivery ratio analysis of the DEELB method with existing systems.

No. of Operations	EEFO	DEERA	ELBS	DEBTS	DEELB
10	78.34	88.32	72.32	82.76	89.97
20	79.02	87.74	73.24	83.18	90.32
30	80.34	88.53	74.65	84.37	90.75
40	82.76	87.32	75.64	85.29	92.76
50	81.27	89.29	74.21	85.31	93.65

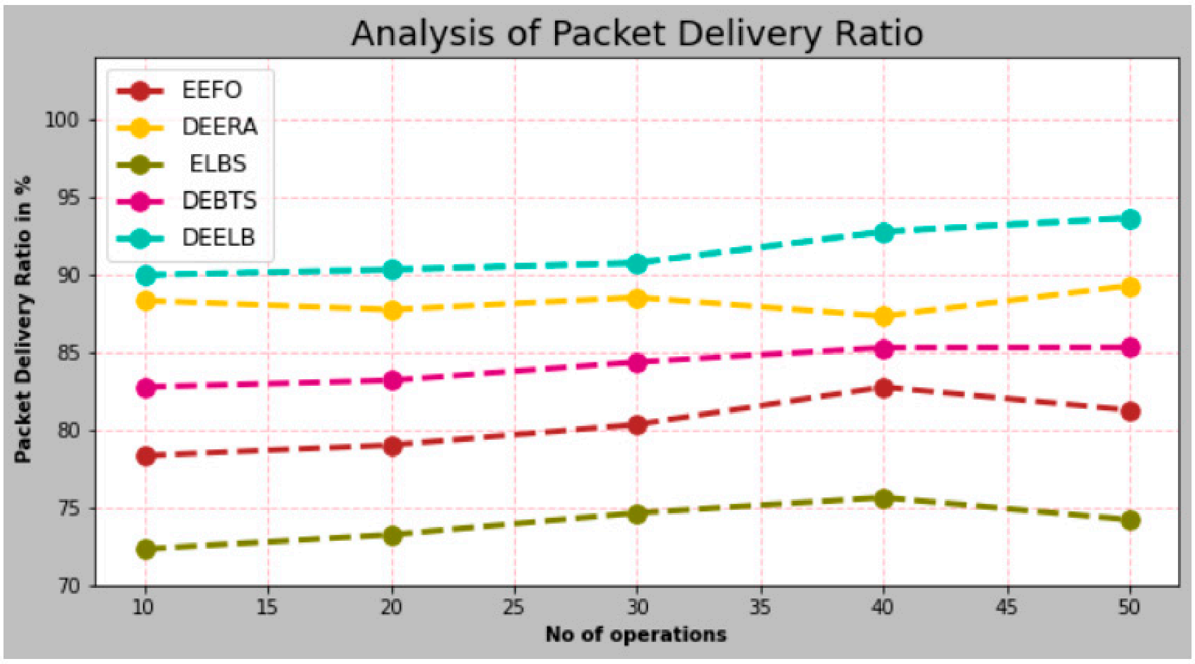

Figure 9. Packet delivery ratio analysis of the DEELB method with the existing systems.

5. Future Research Work

- Service Migration: In the future, we will try to analyze the detrimental effects of service migration on various factors, such as traffic for different computer node types, service migration cost, service migration performance deterioration, and data transmission. We will also devise a plan to balance service migration's advantages and disadvantages.
- Security Concern: Security concerns and standardization are significant challenges to dynamic load balancing and orchestration. While orchestration provides dynamic allocation and mobility for IoT nodes, it will require further hopping of data that may cause security breaches that have not been thoroughly examined yet. There are also many heterogeneous devices in a fog environment, so it is not easy to provide a standard procedure, and less effort has been expended towards such standardization. These aspects can also be considered for future research. Besides these future plans, blockchain integration in fog computing and its effect on load-balancing mechanisms can also be studied in detail [38].
- Machine Learning Application: Traditional load-balancing approaches are generally not very efficient for dynamic conditions. Machine learning (ML) can be suitable for implementing dynamic load balancing when conditions change swiftly. Some research has been done in this domain, but further research is needed, especially in heterogeneous environments. A detailed study to find the best-suited ML approach in such conditions is needed as well [40].

6. Conclusions

In recent years, discussions about the IoT have dominated the global technology scene (IoT). As IoT applications proliferate, fog computing is becoming useful for orchestrating them. Applications for the IoT are executed in fog environments by combining the processing power of cloud-based intermediary nodes with that of traditional client devices. This study aimed to create DEELB, a generic approach of dynamic resource allocation for load balancing that may be used with any type of cloud- or fog-based computer node. First, it is important to pinpoint means to guarantee that IoT gadgets consume as little power as possible. Locating and interacting with the next hop is crucial when the network is crowded. As a result, when network traffic congestion occurs, finding the right route is crucial for executing load balancing by selecting the less congested channel is critical. As a result, we propose a complete methodology called dynamic energy-efficient load balancing (DEELB) in this work to solve all of these resource allocation challenges in the IoT. The simulation findings demonstrated that the suggested technique is a successful resource allocation strategy for fog load balancing, with reductions in packet loss ratio analysis and delay charges of 30.17% and 10.352%, respectively. Further, this study proposes some future research directions for specific domains like service mitigation, security concerns for orchestration, and machine learning applications for dynamic load balancing in changing conditions.

Author Contributions: Conceptualization, S.A., S.B. (Surbhi Bhatia) and M.S.; formal analysis, R.K.M., N.A. and S.B. (Shakeela Basheer); investigation, S.B. (Shakeela Basheer), M.S.A., S.A. and M.S.; project administration, S.B. (Surbhi Bhatia) and S.B. (Shakeela Basheer); resources, S.B. (Shakeela Basheer); supervision, S.B. (Surbhi Bhatia) and N.A.; visualization, M.S.A. and R.K.M.; writing—original draft, M.S., S.A. and S.B. (Surbhi Bhatia); writing—review and editing, R.K.M., N.A., S.B. (Shakeela Basheer) and M.S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R195), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lin, Z.; Lin, M.; De Cola, T.; Wang, J.-B.; Zhu, W.-P.; Cheng, J. Supporting IoT with Rate-Splitting Multiple Access in Satellite and Aerial-Integrated Networks. *IEEE Internet Things J.* **2021**, *8*, 11123–11134. [CrossRef]
2. Niu, H.; Lin, Z.; Chu, Z.; Zhu, Z.; Xiao, P.; Nguyen, H.X.; Lee, I.; Al-Dhahir, N. Joint Beamforming Design for Secure RIS-Assisted IoT Networks. *IEEE Internet Things J.* **2022**, *10*, 1628–1641. [CrossRef]
3. Qi, L.; Lin, W.; Zhang, X.; Dou, W.; Xu, X.; Chen, J. A Correlation Graph Based Approach for Personalized and Compatible Web APIs Recommendation in Mobile APP Development. *IEEE Trans. Knowl. Data Eng.* **2022**. Early Access. [CrossRef]
4. Li, J.; Peng, H.; Cao, Y.; Dou, Y.; Zhang, H.; Philip, S.Y.; He, L. Higher-Order Attribute-Enhancing Heterogeneous Graph Neural Networks. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 560–574. [CrossRef]
5. Huang, J.; Tong, Z.; Feng, Z. Geographical POI Recommendation for Internet of Things: A Federated Learning Approach Using Matrix Factorization. *Int. J. Commun. Syst.* **2022**, e5161. [CrossRef]
6. Huang, J.; Wang, M.; Wu, Y.; Chen, Y.; Shen, X. Distributed Offloading in Overlapping Areas of Mobile-Edge Computing for Internet of Things. *IEEE Internet Things J.* **2022**, *9*, 13837–13847. [CrossRef]
7. Fu, S.; Gao, J.; Zhao, L. Collaborative Multi-Resource Allocation in Terrestrial-Satellite Network towards 6G. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 7057–7071. [CrossRef]
8. Qu, G.; Wu, H.; Li, R.; Jiao, P. DMRO: A Deep Meta Reinforcement Learning-Based Task Offloading Framework for Edge-Cloud Computing. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 3448–3459. [CrossRef]
9. Wu, H.; Zhang, Z.; Guan, C.; Wolter, K.; Xu, M. Collaborate Edge and Cloud Computing with Distributed Deep Learning for Smart City Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 8099–8110. [CrossRef]
10. Xu, X.; Zhang, X.; Khan, M.; Dou, W.; Xue, S.; Yu, S. A Balanced Virtual Machine Scheduling Method for Energy-Performance Trade-Offs in Cyber-Physical Cloud Systems. *Futur. Gener. Comput. Syst.* **2020**, *105*, 789–799. [CrossRef]
11. Luo, E.; Bhuiyan, M.Z.A.; Wang, G.; Rahman, M.A.; Wu, J.; Atiquzzaman, M. Privacyprotector: Privacy-Protected Patient Data Collection in IoT-Based Healthcare Systems. *IEEE Commun. Mag.* **2018**, *56*, 163–168. [CrossRef]
12. Singh, J.K.; Goel, A.K. Data Security Through Fog Computing Paradigm Using IoT. In Proceedings of the Academia-Industry Consortium for Data Science: AICDS 2020, Wenzhou, China, 19–20 December 2020; Springer: Singapore, 2022; pp. 95–103.
13. Sabireen, H.; Neelanarayanan, V. A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges. *Ict Express* **2021**, *7*, 162–176.
14. Alam, S.; Siddiqui, S.T.; Ahmad, A.; Ahmad, R.; Shuaib, M. Internet of Things (IoT) Enabling Technologies, Requirements, and Security Challenges. In *Lecture Notes in Networks and Systems*; Springer: Singapore, 2020; Volume 94, pp. 119–126.
15. Xu, X.; Dou, W.; Zhang, X.; Chen, J. EnReal: An Energy-Aware Resource Allocation Method for Scientific Workflow Executions in Cloud Environment. *IEEE Trans. Cloud Comput.* **2015**, *4*, 166–179. [CrossRef]
16. Alam, S.; Shuaib, M.; Ahmad, S.; Jayakody, D.N.K.; Muthanna, A.; Bharany, S.; Elgendy, I.A. Blockchain-Based Solutions Supporting Reliable Healthcare for Fog Computing and Internet of Medical Things (IoMT) Integration. *Sustainability* **2022**, *14*, 15312. [CrossRef]
17. Rahmani, M.K.I.; Shuaib, M.; Alam, S.; Siddiqui, S.T.; Ahmad, S.; Bhatia, S.; Mashat, A. Blockchain-Based Trust Management Framework for Cloud Computing-Based Internet of Medical Things (IoMT): A Systematic Review. *Comput. Intell. Neurosci.* **2022**, *2022*, 9766844. [CrossRef]
18. Xie, Y.; Gui, F.-X.; Wang, W.-J.; Chien, C.-F. A Two-Stage Multi-Population Genetic Algorithm with Heuristics for Workflow Scheduling in Heterogeneous Distributed Computing Environments. *IEEE Trans. Cloud Comput.* **2021**. Early Access. [CrossRef]
19. Shuaib, M.; Samad, A.; Alam, S.; Siddiqui, S.T. Why Adopting Cloud Is Still a Challenge?—A Review on Issues and Challenges for Cloud Migration in Organizations. In *Advances in Intelligent Systems and Computing*; Springer: Singapore, 2019; Volume 904, pp. 387–399.
20. Hosseinian-Far, A.; Ramachandran, M.; Slack, C.L. Emerging Trends in Cloud Computing, Big Data, Fog Computing, IoT and Smart Living. In *Technology for Smart Futures*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 29–40.
21. Li, C.; Xue, Y.; Wang, J.; Zhang, W.; Li, T. Edge-Oriented Computing Paradigms: A Survey on Architecture Design and System Management. *ACM Comput. Surv.* **2018**, *51*, 1–34. [CrossRef]
22. De Donno, M.; Tange, K.; Dragoni, N. Foundations and Evolution of Modern Computing Paradigms: Cloud, Iot, Edge, and Fog. *IEEE Access* **2019**, *7*, 150936–150948. [CrossRef]
23. Asghar, A.; Abbas, A.; Khattak, H.A.; Khan, S.U. Fog Based Architecture and Load Balancing Methodology for Health Monitoring Systems. *IEEE Access* **2021**, *9*, 96189–96200. [CrossRef]
24. Gupta, H.; Vahid Dastjerdi, A.; Ghosh, S.K.; Buyya, R. IFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments. *Softw. Pract. Exp.* **2017**, *47*, 1275–1296. [CrossRef]
25. Iwendi, C.; Khan, S.; Anajemba, J.H.; Bashir, A.K.; Noor, F. Realizing an Efficient IoMT-Assisted Patient Diet Recommendation System through Machine Learning Model. *IEEE Access* **2020**, *8*, 28462–28474. [CrossRef]
26. Liaqat, M.; Naveed, A.; Ali, R.L.; Shuja, J.; Ko, K.-M. Characterizing Dynamic Load Balancing in Cloud Environments Using Virtual Machine Deployment Models. *IEEE Access* **2019**, *7*, 145767–145776. [CrossRef]
27. Lan, X.; Zhang, Y.; Chen, Q.; Cai, L. Energy Efficient Buffer-Aided Transmission Scheme in Wireless Powered Cooperative NOMA Relay Network. *IEEE Trans. Commun.* **2019**, *68*, 1432–1447. [CrossRef]

28. Yuan, H.; Bi, J.; Zhou, M. Geography-Aware Task Scheduling for Profit Maximization in Distributed Green Data Centers. *IEEE Trans. Cloud Comput.* **2020**, *10*, 1864–1874. [CrossRef]
29. Mishra, S.K.; Sahoo, B.; Parida, P.P. Load Balancing in Cloud Computing: A Big Picture. *J. King Saud Univ. Inf. Sci.* **2020**, *32*, 149–158. [CrossRef]
30. Kaur, R.; Verma, S.; Jhanjhi, N.Z.; Talib, M.N. A Comprehensive Survey on Load and Resources Management Techniques in the Homogeneous and Heterogeneous Cloud Environment. *Proc. J. Phys. Conf. Ser.* **2021**, *1979*, 12036. [CrossRef]
31. Rui, X.; Wu, J.; Zhao, J.; Khamesinia, M.S. Load balancing in the internet of things using fuzzy logic and shark smell optimization algorithm. *Circ. World* **2020**, *47*, 335–344. [CrossRef]
32. Wang, W.; Xu, H.; Alazab, M.; Gadekallu, T.R.; Han, Z.; Su, C. Blockchain-Based Reliable and Efficient Certificateless Signature for IIoT Devices. *IEEE Trans. Ind. Inform.* **2021**, *18*, 7059–7067. [CrossRef]
33. Iwendi, C.; Maddikunta, P.K.R.; Gadekallu, T.R.; Lakshmanna, K.; Bashir, A.K.; Piran, M.J. A Metaheuristic Optimization Approach for Energy Efficiency in the IoT Networks. *Softw. Pract. Exp.* **2021**, *51*, 2558–2571. [CrossRef]
34. Lin, Z.; An, K.; Niu, H.; Hu, Y.; Chatzinotas, S.; Zheng, G.; Wang, J. SLNR-Based Secure Energy Efficient Beamforming in Multibeam Satellite Systems. *IEEE Trans. Aerosp. Electron. Syst.* **2022**. Early Access. [CrossRef]
35. Lin, Z.; Niu, H.; An, K.; Wang, Y.; Zheng, G.; Chatzinotas, S.; Hu, Y. Refracting RIS-Aided Hybrid Satellite-Terrestrial Relay Networks: Joint Beamforming Design and Optimization. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 3717–3724. [CrossRef]
36. Xiang, Z.; Zheng, Y.; He, M.; Shi, L.; Wang, D.; Deng, S.; Zheng, Z. Energy-Effective Artificial Internet-of-Things Application Deployment in Edge-Cloud Systems. *Peer-Peer Netw. Appl.* **2022**, *15*, 1029–1044. [CrossRef]
37. Lim, J. Scalable Fog Computing Orchestration for Reliable Cloud Task Scheduling. *Appl. Sci.* **2021**, *11*, 10996. [CrossRef]
38. Costa, B.; Bachiega, J., Jr.; de Carvalho, L.R.; Araujo, A.P.F. Orchestration in Fog Computing: A Comprehensive Survey. *ACM Comput. Surv.* **2022**, *55*, 1–34. [CrossRef]
39. Mittal, M.; Iwendi, C.; Khan, S.; Rehman Javed, A. Analysis of Security and Energy Efficiency for Shortest Route Discovery in Low-energy Adaptive Clustering Hierarchy Protocol Using Levenberg-Marquardt Neural Network and Gated Recurrent Unit for Intrusion Detection System. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e3997. [CrossRef]
40. Seyfollahi, A.; Taami, T.; Ghaffari, A. Towards Developing a Machine Learning-Metaheuristic-Enhanced Energy-Sensitive Routing Framework for the Internet of Things. *Microprocess. Microsyst.* **2023**, *96*, 104747. [CrossRef]