Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.032485
Article

Tech Science Press

Classification of Request-Based Mobility Load Balancing in Fog Computing

D. Deepa and K. R. Jothi*

School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, 632014, India
*Corresponding Author: K. R. Jothi. Email: jothi.kr@vit.ac.in
Received: 19 May 2022; Accepted: 22 September 2022

Abstract: Every day, more and more data is being produced by the Internet of Things (IoT) applications. IoT data differ in amount, diversity, veracity, and velocity. Because of latency, various types of data handling in cloud computing are not suitable for many time-sensitive applications. When users move from one site to another, mobility also adds to the latency. By placing computing close to IoT devices with mobility support, fog computing addresses these problems. An efficient Load Balancing Algorithm (LBA) improves user experience and Quality of Service (QoS). Classification of Request (CoR) based Resource Adaptive LBA is suggested in this research. This technique clusters fog nodes using an efficient K-means clustering algorithm and then uses a Decision Tree approach to categorize the request. The decision-making process for time-sensitive and delay-tolerable requests is facilitated by the classification of requests. LBA does the operation based on these classifications. The MobFogSim simulation program is utilized to assess how well the algorithm with mobility features performs. The outcome demonstrates that the LBA algorithm's performance enhances the total system performance, which was attained by (90.8%). Using LBA, several metrics may be examined, including Response Time (RT), delay (d), Energy Consumption (EC), and latency. Through the on-demand provisioning of necessary resources to IoT users, our suggested LBA assures effective resource usage.

Keywords: Mobility; load balancing; classification; clustering; IoT devices

1 Introduction

In this technology world, the number of Internet of Things (IoT) devices reach up to 75 billion in 2025 and the amount of data volume created by these devices may reach 79.4 zettabytes [1]. The number of connected devices is increasing in the ratio of 25% to 50%, and in the upcoming years it may reach up to 150 billion in 2030 [2]. The traditional computation process of using IoT devices like traffic light controllers, smart homes, health monitoring, etc. was done in cloud computing. The time taken to send requests and responses is more because of the distance between the device and the cloud server, which causes delay. This delay and latency affect many time-sensitive applications. A new computing paradigm has been introduced to reduce the latency by extending the cloud computation near the source node in the name of fog computing. Fog computing architecture consists of three layers: large volume, wide variety, and high velocity of data generated from the IoT device layer 1. In layer 2 any device that has the

capability of computation, storage, and networking is defined as a fog node. The long-term storage and huge computation data are forwarded to the cloud server in layer 3 [3]. IoT devices are deployed with the IPv6 (Wireless Low-Power Personal Area Network (6LoWPAN)), sensor nodes keep monitoring the environmental values like patient heartbeat and forward sensed data to fog node through the gateway by using Constrained Application Protocol (CoAP). In the CoAP post method that utilizes the User Datagram Protocol (UDP) in the transport layer [4]. Content Caching scheme in the application layer improves the speed of delivery and solves the energy efficiency problem while transferring the data from the sensor to the fog node [5]. Fog node that receives the data from both homogenous groups like (cluster of Raspberry Pi) and the heterogeneous group like (Thinker, Udoo, Nvidila Jeston nano, Personal computers, Mobile, etc.). When the number of IoT devices increases, the devices need to be clustered to perform efficiently [6]. Overloaded fog servers may decrease the Quality of Service (QoS) and increase the latency, this affects many time-sensitive applications. If one server's capacity for handling requests is overloaded then the upcoming request may be given to a nearby fog node that can handle the request [7]. In general load balancing is divided into two types static and dynamic. Prior server information is gathered based on that load is distributed among the server, the current status of the server is unknown in the static load balancing (Ex. Min-Min, Min-Max). In dynamic load balancing, load distribution is based on the current status of server workload (Ex. Agent-Based, Hybrid, Real-Time, Nature-inspired). Handling load balancing techniques with priority-based execution that reduce the latency. While submitting the user request its classify with Higher Priority (HP) and Lower-Priority (LP). Load balancing with priority request handling improves the response time for many time-sensitive applications [8]. Request arriving at the fog server can be classified by Classes of Service (CoS) based on the basic principle component like bandwidth, reliability, location, storage, delay, etc., and executed many machine learning algorithms for better classification. Decision Tree (DT) provides a better classification for the n number of an attribute with a minimal amount of time [9].

From the above observations, clustering is one of the major contributions to fog nodes when it supports the mobility of IoT users. Based on the mobility of the user, a grouping of fog nodes can be done. Clustering of the fog nodes with the minimum number based on the number of IoT users. In general, not all IoT applications are time-sensitive, so without a precise classification of the IoT request, LBA can be less than optimal because of the complexity of dealing with the diversity of resources. The proposed method should satisfy both resource provisioning and QoS. The novel contribution of this work beyond the load balancing to minimize the response latency on the server-side are as below

- Clustering of fog nodes using the Optimized K-Means elbow clustering algorithm is proposed. In this method, the optimization of the clustering can be evaluated by the elbow method. Better clustering provides the minimum migration of nodes takes place.
- Classification of Request (CoR) is proposed for the IoT request. A Decision Tree machine learning algorithm is used to classify the request. Based on the user input characteristics the requests are divided into two types Class 1 (C1) and Class 2 (C2).
- CoR-based resource adaptive load-balancing algorithm is proposed to minimize the server-side latency problem. This algorithm first classifies the individual server characteristics. If the server capacity can handle the request then the load balancer assigns the task, otherwise, it searches for another server.

The rest of the paper is organized as follows. Section 2 discussed the related work based on various load balancing algorithms, Section 3 Optimized K-means clustering is done in the distributed fog computing environment, Section 4 describes the CoR using the decision tree algorithm, Section 5 describes the proposed load balancing algorithm, in Section 6 evaluated simulation results are described in the graphical representation.

2 Related Work

This section discusses the existing Load Balancing (LB) techniques. LB techniques improve the server capacity to handle the request by distributing the request across the multiple available servers.

- Load Balancing and optimized strategy (LBOS): LBOS uses dynamic resource allocation with reinforcement learning to continuously monitor the server load. If it exceeds its capacity load will be distributed to the remaining server. LBOS achieved (85.7%) level [10].
- Dynamic Energy-Efficient Resource Allocation (DEER): in this strategy task managers and resource schedulers dynamically allocate the. Information about the task and the resource is submitted to the resource scheduler by the task manager. The resource can be allocated based on the descending order as per the utilization. In this DEER minimize the energy consumption by 8.67% and computation cost by 16.77% [11].
- Load Balancing for Big Data on Heterogeneous Software Defined Network (LBBHD): this algorithm deploys the optimal assignment of data sharing in the heterogeneous network. They use the classification of the real-time object to improve the workload throughput by 2.2%, and load balancing error by 2.51% [12].
- Self-Similarity-Based Load Balancing (SSLB): load balancing concept is implemented in a large area. The similar nodes are grouped and elected one node as a cluster head. The cluster head is responsible for the communication for the load balancing process. Resource utilization of the node in the region of 1.7X and 1.2X of centralized and distributed under 100 nodes [13].
- Effective Load Balancing Strategy (ELBS): ELBS introduced fuzzy classification to make the priority of task assignment. Based on the priority load balancing the Average Turnaround Time (ATT) 9.5 s and CPU utilization is 99% [14].
- Fog Computing Architecture of Load Balancing (FOCALB): they proposed hybridizing by combining Grey Wolf Optimization (GWO) and Ant Colony Optimization (ACO) to enhance resource utilization by reducing execution time, cost, and energy. This proposed method reduces 25% the execution time and energy consumption by 14% [15].
- Workload Balancing Scheme (WBS): latency of the data flow in both communication and computing by implementing the alpha distribution algorithm. In this method, the latency is reduced up to 25.1% [16]. The proposed method parameters evaluation is compared with various existing methods mentioned in Table 1.

Table 1: Comparative analysis

Reference	Response time	Network management	Request classification	Energy consumption	Delay	Mobility Features
[10]	✓	✗	✗	✓	✓	✗
[11]	✓	✗	✗	✓	✗	✗
[12]	✓	✓	✗	✗	✓	✗
[13]	✓	✓	✗	✗	✗	✗
[14]	✗	✗	✓	✗	✓	✗
[15]	✓	✗	✗	✓	✗	✗
[16]	✓	✓	✗	✓	✓	✗
[17]	✓	✗	✓	✓	✓	✗

(Continued)

Table 1 (continued)

Reference	Response time	Network management	Request classification	Energy consumption	Delay	Mobility Features
[18]	✓	✓	✗	✗	✗	✗
[19]	✗	✗	✗	✓	✓	✗
[20]	✓	✗	✗	✓	✓	✗
[21]	✓	✗	✗	✓	✓	✗
[22]	✓	✗	✗	✓	✓	✗
[23]	✓	✓	✗	✗	✗	✗
Proposed	✓	✓	✓	✓	✓	✓

3 Optimized K-Means Elbow Clustering

To solve the traditional centralized fog scenario we propose an efficient way of grouping the nodes and making them more efficient in the distributed heterogeneous environment. Self-Similarity-based clustering is implemented in [24] but fog computing contains heterogeneous systems similarity-based grouping is not suitable. K-means++ clustering-based grouping is used in [25] but the optimization of the algorithm is not verified. The optimized K-means clustering is implemented to make an effective clustering. The network contains n number of fog nodes that can be grouped into n number of C clusters. The cluster can be formed based on the K-Means clustering algorithm. The form of a cluster in the network is adjacent nodes combined as one cluster created on the minimum distance between each node. in each group, one cluster head is selected to communicate the fractal. Cluster head selection criteria based on fog node range between the fog gateway and validate the threshold.

Algorithm 1: Optimized K-means Clustering

1: Initialization
2: fn(1….n) ← number of fog nodes
3: C (C1,C2,….Cn) ← number of clusters
4: G ← Fog Gateway
5: **for** n number of nodes from (1…n) **do**
6: Euclidean Distance(G,n)
7: Minimum distance d[fn(x,y), fn(a,b)]
8: C1 ← n
9: **end for**
10: **for** n number of clusters (c1…cn) **do**
11: C ← d(fnC1)
12: **end for**

The number of nodes in the network $fn(a_i = 1 \ldots n)$, these nodes can form n number of cluster $C = \{C1, C2, ..Cn\}$ The number of cluster in the network is calculated by the elbow method $C_i,\ i\epsilon\{1 \ldots 10\}$. To find the minimum distance between two fractals Euclidean distance is calculated.

Step 1: Select two cluster canters centroid at random.

$$fn(x_i = 1 \ldots n) \; and \; fn(y_i = 1 \ldots n) \tag{1}$$

Step 2: find the minimum distance between fractals.

$$d[fn(x,y),fn(a,b)] = \sqrt{fn(x-a)^2} + fn(x-b)^2 \tag{2}$$

Step 3: Group the fractals based on the minimum distance.

Step 4: After adding to the cluster, update the centroid value of the cluster.

$$C = \frac{C1 + C2}{2} \tag{3}$$

Step 5: Repeat steps 2 to 4 until no more fractals are available.

The performance of the K-Means clustering algorithm is based on how high efficiency it forms the cluster. To find the optimal value elbow method formulation is calculated.

$$C = \Sigma fn_{i \; in \; cluster \; 1} \; d(fn_i C_1)^2 + \Sigma fn_{i \; in \; cluster \; 2} \; d(fn_i C_2)^2 + \ldots + \Sigma fn_{i \; in \; cluster \; 10} (fn_i C_{10})^2 \tag{4}$$

where $\Sigma fn_{i \; in \; cluster \; 1} d(fn_i C_1)^2$ is the sum of the square of the distance between each fog node point and its centroid within the cluster point. The cluster range between {1 … 10} forms the best cluster for communication. If it exceeds 10 then the communication cost between the nodes is increased.

To find the optimal valve for the cluster, first it calculates the K-means clustering on given fog nodes, for example, it ranges from (1–10), for each value of the K (1–10) it calculates the cluster sum of the square. Then the values are plotted on the graph against number cluster (1–10) and cluster sum of square value. In the plotted graph sharp blend point represent the optimal cluster value. In Fig. 1 the sharp bend point 3 represents the optimal cluster value.

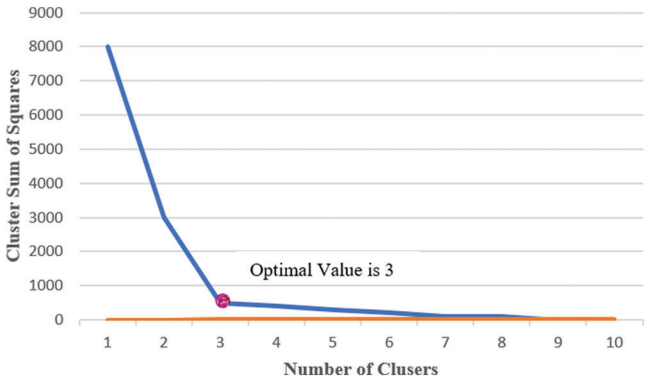

Figure 1: Optimized cluster value

The main advantage of K-means clustering is to find the optimal number of clusters in a particular region using the elbow method. An optimal number of cluster improve the performance of fog nodes. The limitation of the clustering is, cluster formation is based on the mobility nodes, so rapid changes can occur and re-clustering takes place when the user is moving from one location to another location.

The agreed fog node uses the user's past position with the appropriate speed and direction to calculate the user's future location under the assumption that each user updates their location every time t. The future location is projected using this data. If the user's anticipated position is outside of the current fog node, they

will locate the nearest one after making their location forecast. Users' current location: $l = t$, previous location $l = t - \theta$, and future location $l = t + \theta$. The future location that is dependent on the value is identified using the user's current location. The user-supplied X and Y values are used to determine the value. There are various ways to forecast the position using these X and Y values.

4 Classes of Request (CoR)

Fog computing deals with different heterogeneous applications involving the mobility nature. Improving the QoS with the integration of different environments is more challenging. To address these issues efficient handling of resource management provides better QoS. Whenever the IoT devices generate the request before handling the computation the request is classified based on their demand. To classify the request Decision Tree (DT) is implemented as represented in Fig. 2. There are several machine learning classifiers available like Artificial Neural Network (ANN), K-Nearest Neighbor (KNN), and Support Vector Machine (SVM). This classification algorithm is complex in prediction, noise sensitive, high prediction speed, and requires more memory.

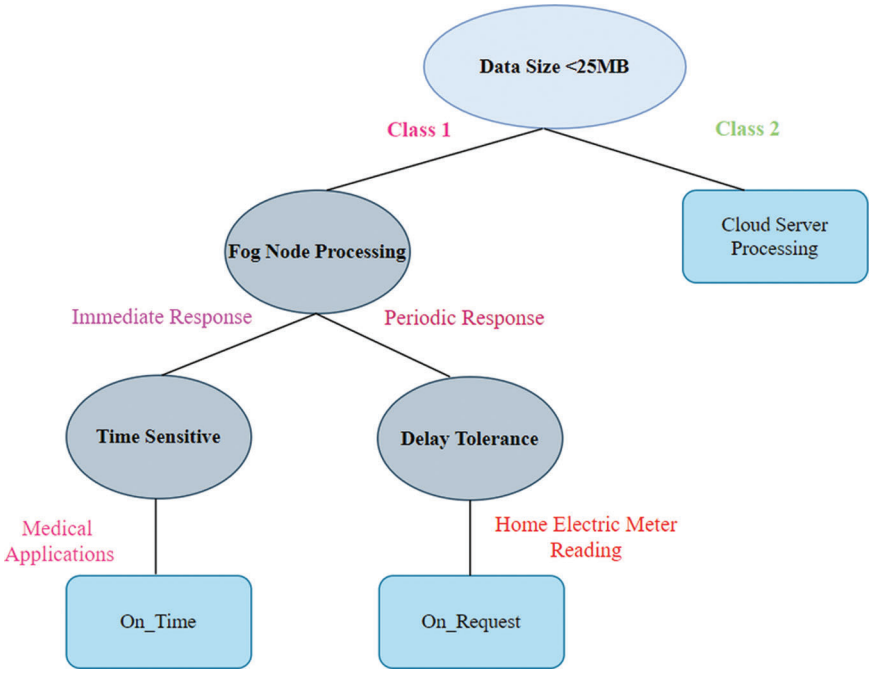

Figure 2: Decision tree classification of request

While the IoT device is connected to the fog node at that time itself the device should mention the response preference like time sensitivity and delay tolerance. If the IoT application belongs to time-sensitive then the request is given to class 1 and the computation authority is given to the fog node. Delay tolerance requests and large-size data like (5 GB) are classified as class 2 and the computation authority is handled by the cloud. Based on the classification LBA distribution takes place.

By the analysis of the classification mechanism, the Decision Tree mechanism is the best fit for fog computing. Because it supports doing computation with a heterogeneous device with minimal computing, and minimal storage, and ensures real-time application. The Decision Tree classification technique gives 99.986% average accuracy and 0.029% average time compared with ANN, KNN, Naïve Bayes, random forest, and SVM classification methods. DT takes less than 30 min to classify up to 1400 heterogeneous

applications with the highest level of accuracy [26]. When the IoT device generates the request if the requested data size is more than 5 MB and the bandwidth is 10000 Mbps request is forwarded to the cloud server for huge computation and long-term storage. Otherwise, the request is handled by the fog layer. Again the requested application is classified based on time sensitivity and delay tolerance. For example, if the application is based on medical-related then it is based on On_time processing, it pushes the fog nodes to do computation and gives an instant response. On the other hand, if the application is delay tolerance, for example, based on an electric bill update, it requires a monthly update. CoR for IoT request improve the efficiency of handling resource management and QoS. Classification of request is represented in Table 2.

Table 2: Classification of request

S.No.	Data size	Data requirement	Data type			Request classification		Authority	
			Text	Image	Video	Class 1	Class 2	Fog	Cloud
1	5 MB	Time sensitive	✗	✓	✗	✓	✗	✓	✗
2	3 MB	Time sensitive	✗	✗	✓	✓	✗	✓	✗
3	4 MB	Time sensitive	✓	✗	✗	✓	✗	✓	✗
4	2 MB	Time sensitive	✗	✓	✗	✓	✗	✓	✗
5	7 MB	Delay tolerance	✓	✗	✗	✗	✓	✗	✓
6	8 MB	Delay tolerance	✗	✗	✓	✗	✓	✗	✓
7	4 MB	Time sensitive	✗	✓	✗	✓	✗	✓	✗
8	5 MB	Delay tolerance	✗	✓	✗	✗	✓	✗	✓
9	1 MB	Time sensitive	✗	✗	✓	✓	✗	✓	✗
10	10 MB	Delay tolerance	✓	✗	✗	✗	✓	✗	✓

5 CoR Resource-Based Adaptive Load-Balancing Algorithm

This section introduces a proposed load balancing algorithm that supports users' mobility. As mentioned in Fig. 3 fog nodes are clustered as a group in the distributed environment with the optimized K-Means clustering algorithm with the elbow method. Then the user request from the IoT environment is classified based on the Decision Tree algorithm, the request is classified ad class 1 and class 2. The proposed load balancing algorithm effectively distributes the load to the server with minimum latency and also supports time-sensitive applications.

5.1 Fog Node Classification

Classification of the fog node provides better load balancing. Load Balancing (LB) is the process of distributing the workload among the servers. LB optimizes the response time and avoids the situation of the server like some servers overloaded, remaining idle for a long period of time. The load balancer is responsible for monitoring the system status, based on that LB decide which node is going to handle the upcoming request. The server status can be obtained from the Server Characteristics Table (SCT). The LB requires the system (i) Cache Size, (ii) total memory available, and (iii) System usage. To find the system's current status we need to calculate (1) the Capacity of the System (CoS), and (2) the Average Capacity of the system. By calculating these values, it can be able to find them (i) Balanced System, (ii) Underloaded system (iii) overloaded system.

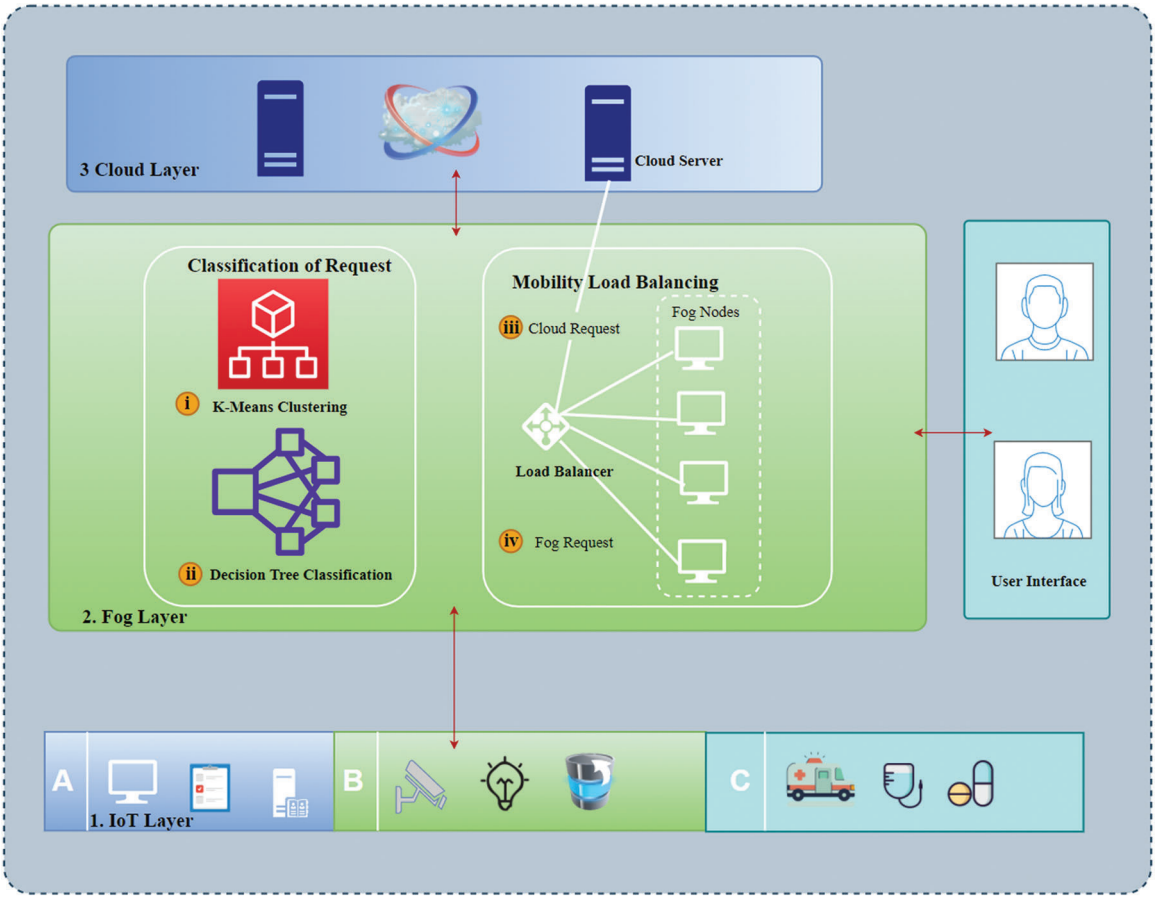

Figure 3: CoR based adaptive resource LBA

5.2 Calculating Capacity of System (CoS)

Calculating the capacity of the system plays an important role in load-balancing tasks. By this value, only the server is classified as balanced, overloaded, and underloaded. Each fog node has a different capacity based on the allocation of the work.

By using the mentioned parameters the capacity of the system is derived by

$$CoS = \frac{Cache\ Size * Memory}{Processor\ Usage} \tag{5}$$

After finding the capacity of the system, need to find the Average Capacity of the System (ACoS), and the threshold value of system capacity can be calculated by

$$ACoS = CoS|(Processor\ Usage \leq 12\%)' \tag{6}$$

By using a Genetic algorithm, to find the optimized system capacity can be evaluated by defining the values of ACoS = 1.5 and TCoS = 2.3. If the Capacity of the system is greater than the average capacity of the system then the system is in an overloaded condition. If the capacity of the system is greater than or equal to threshold capacity then the system is in an underloaded condition. Based on these conditions available system is selected for the load balancing.

6 Experimental Setup

Many existing simulation tools are used to evaluate the application behavior and performance in the fog computing environment. Fog computing supports only minimal mobility support. To evaluate the mobility-based load balancing performance, we used the MobFogSim tool. To the best of our knowledge still, now no one evaluated using this simulation. MobFogSim is extended from iFogSim and CloudSim. This tool has additional mobility support features like mobile nodes, wireless connections, and migration features [27]. With this, we can achieve all mobility features supported by the fog computing environment. In this simulation, a cloud with 4 data centers, and a set of 50 fog nodes, can be clustered based on their properties. Each fog node is connected with IoT devices ranging from 10 to 15 devices through the gateway. Fig. 4 illustrates that the simulation network consists of the user device, a fog node data center. When the request arrives LBA selects the nearest node to execute the task. The performance of the CoR-based Adaptive Resource LBA can be evaluated by the given parameters with mathematical calculations. Simulation parameters are represented in the "Table 3".

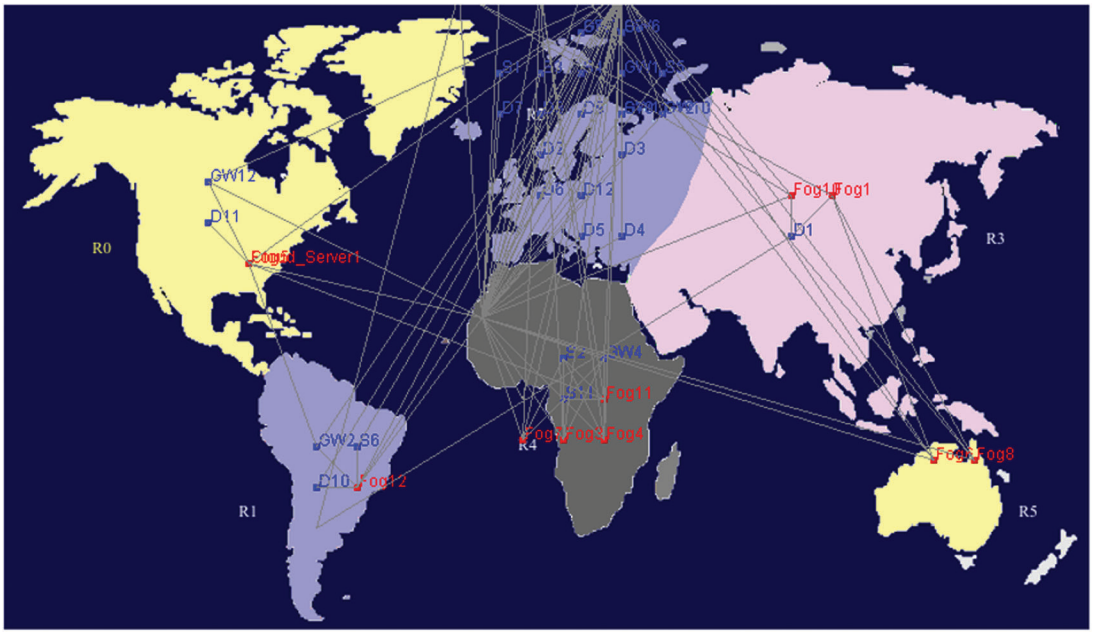

Figure 4: Fog node allocation in the region

Table 3: Simulation and system parameters

Parameters	Value
Language	Java
Development kit	JDK 1.8
Mobility model	Linear mobility model
Routing protocol Ipv6	AODV
Communication protocol	IPv6
The initial energy of fog node (J)	40
Transmission range (m)	60–250

(Continued)

Table 3 (continued)

Parameters	Value
Bandwidth (KBs)	1000
Storage (GB)	1
Fog node range	100
Fog node computing capacity (MIPS)	1 to 15
Delay (ms)	1
Storage cost	0.001
Memory cost	0.05
Topology	Fully connected
Receiving power (W)	1.1

Algorithm 2: CoR Resource-Based Adaptive LBA

1: **input**: Request ← R

2: fn(1….n) ← Number of fog nodes

3: CS ← Cloud server

4: **for** all (1….n) in fn **do**

5: Initiate CoR for all task

7: **if** R ← Fog nodes **then**

8: C1 ← R

9: **if** CoS ≥ T CoS **then**

10: n ← R

11: **else**

12: Repeat Step 3 until finding the appropriate fn

13: **Do** Task For all CoR

14: **if** R ← Cloud Server **then**

15: Compute task execution

16: **Output:** Balance Load Distribution & Fast Response Time (RT).

6.1 Response Time (RT)

Parameters evaluation Response Time (RT), delay (d), Energy Consumption (EC), and Load Balancing Rate (LBR) can be evaluated based on the classification of the request. The request is submitted to the fog node and the response is given back to the source node in a minimal amount of time. By calculating the time taken to compute the process, response time, delay, and load balancing rate can be evaluated.

Response time is defined as the amount of time taken to submit a request and processed it by the server and send a response back to the user. Response Time is calculated by the sum of time taken from submitting the request (Rs) to the fog node and the total time taken to Completion of Request (C_R). RT is evaluated by

milliseconds (ms). In Fig. 5 RT is derived by increasing the number of fog nodes to reduce the response time by sharing workload among the fog nodes.

$$RT = Rs + C_R \tag{7}$$

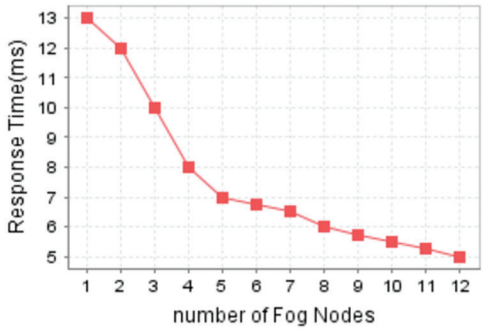

Figure 5: Response time

6.2 Delay (d)

Fig. 6 is estimated by the difference between the present time (Pt) of the request execution and starting time (St) of request execution and it is represented by seconds (s).

$$d = Pt - St \tag{8}$$

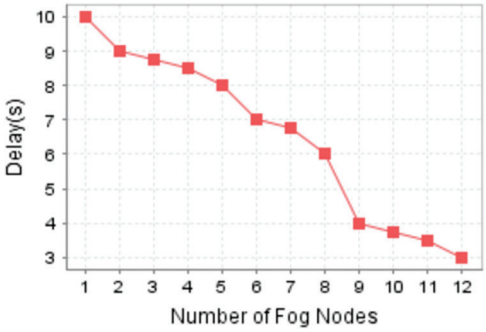

Figure 6: Delay

6.3 Energy Consumption (Ec)

Energy consumption is denoted by the total amount of energy required to complete the request by fog nodes. It is represented by the unit Joule (J) in Fig. 7

$$Ec = Pec + (Pt - LEcUT) \tag{9}$$

where Ec represents energy consumption, Pec is the present energy consumption, Pt is the present time of the request execution, and LEcUT is the last energy consumption updated time.

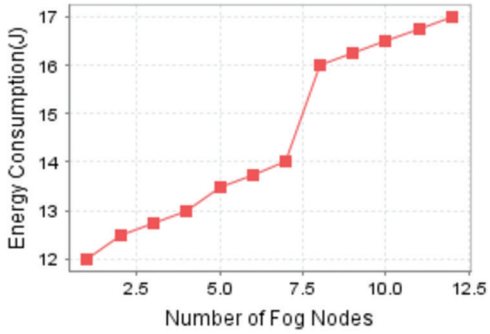

Figure 7: Energy consumption

6.4 Load Balancing Rate (LBR)

The load distribution rate is calculated to find the performance of the server, and how efficiently the task is distributed among the fog nodes. The load balancing percentage was evaluated by calculating the total number of requests (TNr) and a total number of available fog nodes (Tfn) are represented in Fig. 8.

$$LBR = TNr/Tfn \tag{10}$$

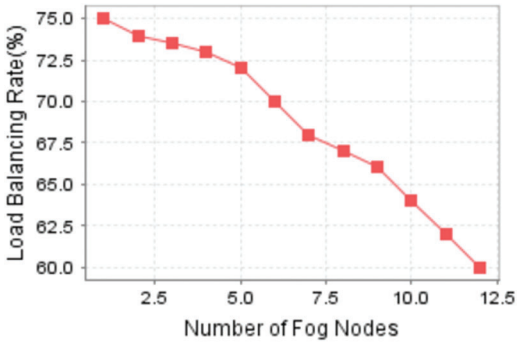

Figure 8: Load balancing rate

7 Comparison Analysis

The comparison is done with our proposed load balancing algorithm with the existing methods of all performance metrics. The response time is the amount of time taken to complete the request, it is represented by a millisecond (ms). The Time taken to complete the number of requests with several fog nodes is compared with the Genetic algorithm, graph partitioning, and load balancing with ANN are represented in Fig. 9. Load balancing has directly impacted the performance of the system and response time. CoR-based load balancing effectively decreased the response time by 30% by comparing with [28–30]. By improving the reliability and scalability of node performance. The main reason for the delayed performance is the number of fog nodes. If the number of nodes is increased the number is decrease the delayed performance and vice versa. The performance comparison is done with (Non-dominated Sorting Gentic Algorithm) NSGA-II [31], and load balancing with ANN. By this comparison, almost 60% of the delay is reduced because the number of fog nodes increased in the network is represented in Fig. 10. CoR-based classification significantly reduces the energy consumption of the task execution.

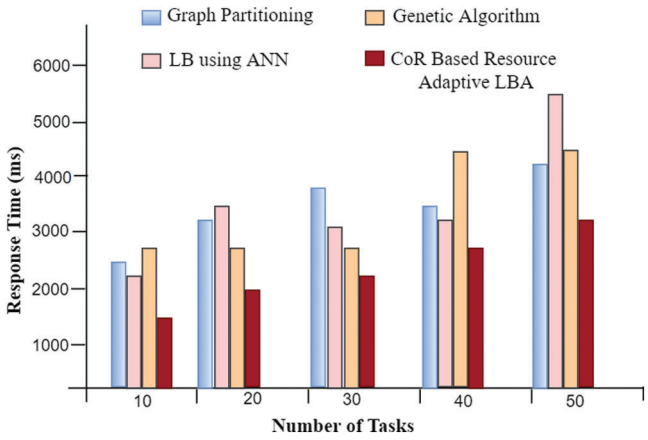

Figure 9: Response time

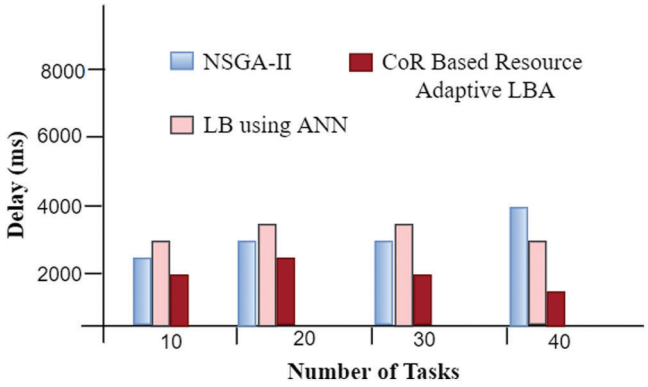

Figure 10: Delay comparison

The proposed work is compared with the advanced-cache method and DEER [32]. These existing methods take 12000000 KJ even for less number of tasks, if the number of tasks is increased in the nodes our method requires only 1400000 KJ to complete the execution as given in Fig. 11. CoR Resource-Based load balancing algorithm increases the user efficiency in terms of QoS and Quality of user Experience (QoE).

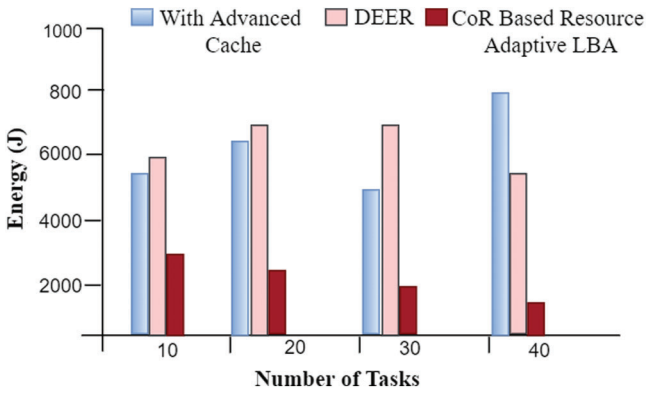

Figure 11: Comparison of energy consumption

8 Conclusion

This paper proposed CoR based Resource Adaptive load balancing algorithm in the distributed fog computing environment. In the previous approach node failure, resource shortage, and storage allocation problem reduces the performance of the LBA. Optimized K-means elbow clustering is used to group the number of distributed fog nodes. A Decision tree is used to classify the type of IoT requests. By this classification, the large size of data and delay-tolerant requests are moved to cloud computation. With this separation load balancing performance improves the Quality of Service. CoR-based classification of requests improves the better performance of server utilization to reduce the latency problem raised in cloud computing. The simulation results show a better load balancing rate in the mobility user environment. The parameters of response time, delay, latency, and load balancing rate can be evaluated in this work. Classification-based load balancing improves the overall system performance in the mobile fog computing environment.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Internet of things—number of connected devices worldwide 2015–2025. Accessed: Jul. 1, 2020 [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-woldwide/.

[2] M. Kaur and R. Aron, "A systematic study of load balancing approaches in the fog computing environment," *The Journal of Supercomputing*, vol. 77, no. 8, pp. 9202–9247, 2021.

[3] M. Rahman, F. Afsana, M. Mahmud, M. S. Kaiser and M. R. Ahmed, "Toward a heterogeneous mist, fog, and cloud-based framework for the internet of healthcare things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4049–4062, 2018.

[4] F. Banaie, M. H. Yaghmaee, S. A. Hosseini and F. Tashtarian, "Load-balancing algorithm for multiple gateways in fog-based internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7043–7053, 2020.

[5] X. Sun and N. Ansari, "Traffic load balancing among brokers at the IoT application layer," *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 489–502, 2017.

[6] R. Fayos-Jordan, S. Felici-Castell, J. Segura-Garcia, J. Lopez-Ballester and M. Cobos, "Performance comparison of container orchestration platforms with low cost devices in the fog, assisting internet of things applications," *Journal of Network and Computer Applications*, vol. 169, no. 1, pp. 102788, 2020.

[7] S. Sharma and H. Saini, "A novel four-tier architecture for delay aware scheduling and load balancing in fog environment," *Sustainable Computing: Informatics and Systems*, vol. 24, no. 1, pp. 100355, 2019.

[8] J. Y. Baek, G. Kaddoum, S. Garg, K. Kaur and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," in *Proc. IEEE Wireless Communications and Networking Conf.*, Marrakesh, Morocco, pp. 1–7, 2019.

[9] J. C. Guevara, R. D. S. Torres and N. L. da Fonseca, "On the classification of fog computing applications: A machine learning perspective," *Journal of Network and Computer Applications*, vol. 159, no. 1, pp. 102596, 2020.

[10] F. M. Talaat, M. S. Saraya, A. I. Saleh, H. A. Ali and S. 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, no. 11, pp. 4951–4966, 2020.

[11] A. U. Rehman, A. Z. Ahmad, A. I. Jehangiri, M. A. Ala'Anzy, M. Othman *et al.,* "Dynamic energy efficient resource allocation strategy for load balancing in fog environment," *IEEE Access*, vol. 8, no. 1, pp. 199829–199839, 2020.

[12] Y. Ping, "Load balancing algorithms for big data flow classification based on heterogeneous computing in software definition networks," *Journal of Grid Computing*, vol. 18, no. 2, pp. 275–291, 2020.

[13] C. Li, H. Zhuang, Q. Wang and X. Zhou, "SSLB: Self-similarity-based load balancing for large-scale fog computing," *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 7487–7498, 2018.

[14] F. M. Talaat, S. H. Ali, A. I. Saleh and H. A. Ali, "Effective load balancing strategy (ELBS) for real-time fog computing environment using fuzzy and probabilistic neural networks," *Journal of Network and Systems Management*, vol. 27, no. 4, pp. 883–929, 2019.

[15] M. Kaur and R. Aron, "FOCALB: Fog computing architecture of load balancing for scientific workflow applications," *Journal of Grid Computing*, vol. 19, no. 4, pp. 1–22, 2021.

[16] Q. Fan and N. Ansari, "Towards workload balancing in fog computing empowered IoT," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 253–262, 2018.

[17] S. P. Singh, R. Kumar, A. Sharma, J. H. Abawa and R. Kaur, "Energy efficient load balancing hybrid priority assigned laxity algorithm in fog computing," *Cluster Computing*, vol. 25, no. 1, pp. 1–18, 2022.

[18] S. P. Singh, R. Kumar, A. Sharma and A. Nayyar, "Leveraging energy-efficient load balancing algorithms in fog computing," *Concurrency and Computation: Practice and Experience*, vol. 13, no. 1, pp. 800, 2022.

[19] S. A. Darade, M. Akkalakshmi and D. N. Pagar, "SDN based load balancing technique in internet of vehicle using integrated whale optimization method," in *AIP Conf. Proc.*, New York, USA, vol. 2469, no. 1, pp. 20006, 2022.

[20] M. M. S. Maswood, M. R. Rahman, A. G. Alharbi and D. Medhi, "A novel strategy to achieve bandwidth cost reduction and load balancing in a cooperative three-layer fog-cloud computing environment," *IEEE Access*, vol. 8, no. 1, pp. 113737–113750, 2020.

[21] J. Wan, B. Chen, S. Wang, M. Xia and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4548–4556, 2018.

[22] H. A. Khattak, H. Arshad, G. Ahmed, S. Jabbar, A. M. Sharif *et al.,* "Utilization and load balancing in fog servers for health applications," *EURASIP Journal on Wireless Communications and Networking*, vol. 1, no. 1, pp. 1–12, 2019.

[23] P. Kansal, M. Kumar and O. P. Verma, "Classification of resource management approaches in fog/edge paradigm and future research prospects: A systematic review," *The Journal of Supercomputing*, vol. 1, no. 1, pp. 1–60, 2022.

[24] Z. Nezami, K. Zamanifar, K. Djemame and E. Pournaras, "Decentralized edge-to-cloud load balancing: Service for the internet of things," *IEEE Access*, vol. 9, no. 1, pp. 64983–65000, 2021.

[25] M. Keerthika, N. S. Ram, M. Rajkumar, M. V. Manivannan and S. Monish, "To optimize the multi accesses download time using scheduling approach in fog computing," *Materials Today: Proceedings*, vol. 37, no. 1, pp. 1475–1479, 2021.

[26] C. Puliafito, D. M. Gonçalves, M. M. Lopes, L. L. Martins, E. Madeira *et al.,* "MobFogSim: Simulation of mobility and migration for fog computing," *Simulation Modelling Practice and Theory*, vol. 101, no. 1, pp. 102062, 2020.

[27] T. Wang, Z. Liu, Y. Chen, Y. Xu and X. Dai, "Load balancing task scheduling based on genetic algorithm in cloud computing," in *Proc. IEEE 12th Int. Conf. on Dependable, Autonomic and Secure Computing*, Dalian, China, pp. 146–152, 2014.

[28] S. Ningning, G. Chao, A. Xingshuo and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Communications*, vol. 13, no. 3, pp. 156–164, 2016.

[29] Y. Sun, F. Lin and H. Xu, "Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II," *Wireless Personal Communications*, vol. 102, no. 2, pp. 1369–1385, 2018.

[30] M. H. Shahid, A. R. Hameed, S. Ul Islam, H. A. Khattak, I. U. Din *et al.,* "Energy and delay efficient fog computing using caching mechanism," *Computer Communications*, vol. 154, no. 1, pp. 534–541, 2020.

[31] N. Mazumdar, A. Nag and J. P. Singh, "Trust-based load-offloading protocol to reduce service delays in fog-computing-empowered IoT," *Computers & Electrical Engineering*, vol. 93, no. 1, pp. 107223, 2021.

[32] S. P. Singh, "Effective load balancing strategy using fuzzy golden eagle optimization in fog computing environment," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 1, pp. 100766, 2022.