Bulletin of Electrical Engineering and Informatics
Vol. 11, No. 6, December 2022, pp. 3462~3470
ISSN: 2302-9285, DOI: 10.11591/eei.v11i6.4127

Hybrid load-balancing algorithm for distributed fog computing in internet of things environment

Abrar Saad Kadhim, Mehdi Ebady Manaa
Department of Information Networks, College of Information Technology, University of Babylon, Babylon, Iraq

Article Info

Article history:

Received May 24, 2022
Revised Aug 8, 2022
Accepted Aug 26, 2022

Keywords:

Distributed fog network
Hybrid load balancing
IoT sensors
Queuing time
Task scheduling

ABSTRACT

Fog computing is a novel idea created by Cisco that provides the same capabilities as cloud computing but close to objects to improve performance, such as by minimizing latency and reaction time. Packet failure can happen on a single fog server across a large number of messages from internet of things (IoT) sensors due to several variables, including inadequate bandwidth and server queue capacity. In this paper, a fog-to-server architecture based on the IoT is proposed to solve the problem of packet loss in fog and servers using hybrid load balancing and a distributed environment. The proposed methodology is based on hybrid load balancing with least connection and weighted round robin algorithms combined together in fog nodes to take into consideration the load and time to distribute requests to the active servers. The results show the proposed system improved network evaluation parameters such as total response time of 131.48 ms, total packet loss rate of 15.670%, average total channel idle of 99.55%, total channel utilization of 77.44%, average file transfer protocol (FTP) file transfer speed (256 KB to 15 MB files) of 260.77 KB/sec, and average time (256 KB to 15 MB) of 19.27 sec.

Corresponding Author:

Abrar Saad Kadhim
Departement of Information Networks, College of Information Technology, University of Babylon
51002, Babylon, Iraq
Email: abrar.kadhim@student.uobabylon.edu.iq

1. INTRODUCTION

Fog nodes are the distributed computer equipment that makes up a fog network. In the geographically distributed architecture of fog computing, heterogeneous internet of things (IoT) devices is connected to the network to provide compute and storage resources. A fog computing architecture provides a more adaptive, secure, and bandwidth-efficient method of data management. The IoT has steadily been integrated into human existence [1].

The IoT has shown its value and promise in several sectors, such as green infrastructure, smart home systems, and healthcare systems, among others [2]. Load balancing in a fog system facilitates the allocation of workload across resources equitably, with the goal of maintaining service availability if a service component fails. This is accomplished by ensuring optimal resource utilization through deployment, management, and sub-instances of applications [3]. Load balancing is a system that distributes the workload across many resources to prevent resource overload or underload. Resource allocation is a method for distributing the load among many resources. It may be performed using either hardware or software [4].

The objectives of load balancing are throughput improvement, response time reduction, and traffic optimization. The goal of the load balancing approach is to optimize how server-side resources are used, as well as to reduce the time it takes to process requests and improve scalability in a distributed setting [5]. To mitigate the effects of node load and disruption on the fog computing environments, each node should employ a dispersed structure. With regard to node load [6]. The service node utilizes distributed technologies to connect all of the network's machines. This produces better use of such devices and prevents a number of them from idling. If the number of service nodes is increased from one to many, the load will be divided and services will be accelerated [7].

In fog networks, task allocation strategies may be static, dynamic, or a combination of the two. In static techniques where fundamental system information is necessary, the rule should indeed be written further into the load balancer. This is because it is difficult to forecast user behavior and static load-balancing approaches are not always beneficial for the network [8]. The load balancer's pattern-based load distribution algorithm also makes the dynamic approaches better than the static ones [9].

To mitigate the effects of node load and disruption on the fog computing platform, each node should be using a dispersed structure. Regarding node load [10], the service node utilizes a distributed architecture to integrate the whole network's computer resources, therefore enhancing resource usage and avoiding a significant number of idle computing resources. If there are more service nodes instead of just one, the work will be spread out and the system will run faster [11].

The network's administration node takes on the most effort. When negotiating with the other network's management node, which is necessary because the services of the whole network need to be talked about, the distributed communication node may make the conversation go more smoothly [12]. Management units and registration nodes employ dispersed technologies to prevent the network from losing communication with other networks or failing to offer services to the users of this network in the event of a node failure [13]. The following is a discussion and summary of the most relevant literature on improving distributed fog computing capacity using effective allocation of resources (load balancing) techniques:

a. In order to decrease the load on the network and wait times, a fog-based monitoring system has been suggested [14]. Whenever the healthcare monitoring system is implemented on a large scale, they also provide a novel load balancing scheme (LBS) for dividing up the work among the fog nodes. Extensive simulations were run in the iFogSim toolkit to verify the efficacy of the suggested method, and the results were compared to those of the cloud-only solution, fog node placement algorithm (FNPA), and load balancing (LAB) scheme, in terms of delay and network utilization. When compared to cloud-only, FNPA, and LAB Scheme, the suggested deployment of the health monitoring system dramatically decreases latency and network use.
b. Hybrid load balancing algorithms for scientific processes (Tabu-GWO-ACO) are presented in [15] as part of the fog computing architecture of load balancing (FOCALB). These algorithms use elements from tabu search, grey wolf optimization (GWO), and ant colony optimization (ACO). Load balancing at the fog layer is used to improve resource consumption. For 20–200 fog nodes, iFogSim and eclipse are used to run simulations and get the resulting data. Simulations that compared FOCALB to other models in terms of processing time, costs, and energy use at fog nodes showed that all of these areas could save a lot of money.
c. Distributing software applications among fog devices and the cloud data centers has been suggested in [16] as a way to make better use of cloud-fog resources. The performance metrics of reaction time, latency, and energy consumption are all enhanced when application modules are placed on fog devices. They suggested two scheduling algorithms and compared their efficacy with that of a cloud-only solution using the iFogSim simulator. Almost any IoT application might make use of this method because of how general it is.
d. A hybrid approach that makes use of optimizing process time (OPT) techniques for load balancing in fog computing environments is presented in [17]. A comparison with different algorithms is done to investigate the suggested algorithm's performance. By using the suggested method for optimizing processing time in the load-balancing algorithm, the response times of both the data total cost center and the users are improved.
e. A cloud services system for precise load balancing has been proposed in [18]. For the purpose of load balancing, it is recommended to combine the whale optimization algorithm (WOA) with the bat algorithm (BAT). Comparisons of processing and reaction time are made between the model and state-of-the-art load balancing methods including throttled, round robin, whale, and particle swarm optimization algorithms. The results show that the suggested WOA-BAT outperforms the other three algorithms in terms of reaction time, with a 4.3% gain over RR and TH. Time-wise, it's at least 22.3% faster than any other algorithm they've tested.

The rest of our paper is organized as follows: i) introduction, ii) the proposed algorithm, iii) method, iv) results and discussion, and v) conclusion.

2. THE PROPOSED ALGORITHM

The proposed distributed technique is based on fog resource allocation, and its primary goal is to evenly divide work over several servers in order to maximize the efficiency with which each server puts its computing resources to use and to decrease the average response time for tasks. In other words, the throughput of the system will be maximized. For example, the weighted round robin method, the least connection technique, and the hybrid approach are all examples of scheduling algorithms, and the fog nodes that accept and distribute all assigned tasks to every server in the pool utilize one of these algorithms. In a distributed fog environment, the architecture has three layers: the first layer is wireless IoT sensors, access points, and gateway routers; the second layer is fog nodes; and the last layer is the server resources, as shown in Figure 1.

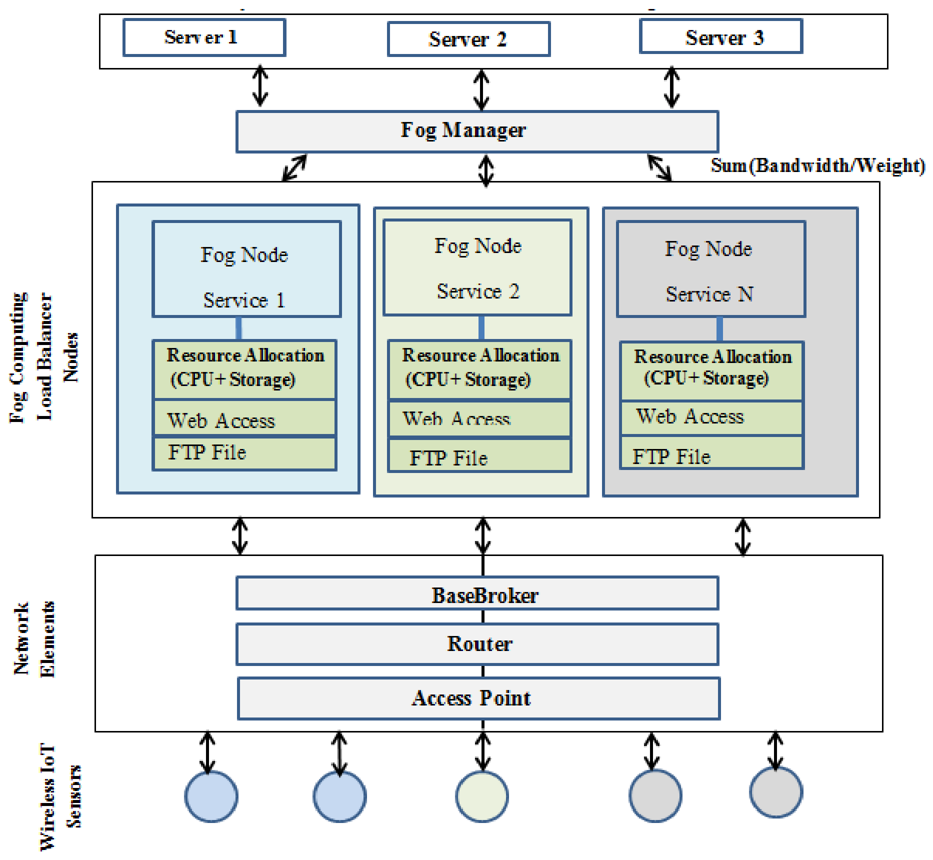

Figure 1. The proposed resource allocation system architecture in distributed fog computing environment

Since IoT sensor node connectivity can be very unstable due to mobility and connectivity issues, synchronization strategies made for distributed fog systems can't be used here [19]. In addition, the proposed distributed fog computing load balancing approach is based on the task scheduling among fog nodes and among servers. Each fog node can distribute tasks for multiple servers, so it provides enhancement in response time and equivalent overloading high traffic data rate system. It deals with huge amounts of requests and big data traffic in a distributed environment and decreases waiting processes in queue with adaptive data management, especially with fault tolerant state for centralized fog node.

3. METHOD

The hybrid approach to load balancing is based on both the algorithms implemented at the same time and the proposed value weight generated from both algorithms based on their characteristics, and then each fog node in a distributed environment evaluates the servers periodically to decide which is the optimal server to process the incoming request. The steps of the hybrid approach for explanation in this example: i) IoT sensors make requests to access a specific website as HTTP requests or file upload requests, ii) request passed to fog nodes to redirect it to a specific server, iii) fog nodes periodically send ICMP echo request packets to servers and it creates a weighed value (depending on the response time and number of connections) for each active server. In addition, fog nodes still send check packets periodically to be updated with the last server state and evaluate servers to be optimal for processing the next request. In addition, the proposed method is based on different evaluation metrics, and the main metrics as follow [20]:

a. A good example of this problem is the time it takes for a server in a fog scenario to response a request [21].
b. Channel resource utilization: it shows how the resources in a fog system are used to their fullest [22].
c. Latency: it is the amount of time between the load balancer receiving a request and returning a response [23].
d. The term "packet loss ratio" refers to the percentage of data packets that did not make it to their destination. The scheduler strives to limit the number of packets that are dropped because their deadlines have passed, and each message has a deadline by which it should be processed [24].
e. The load balancer's performance is measured in terms of its throughput, which is the rate at which requests are processed. Throughput is a useful indicator since it reflects the effectiveness of load balancers [25]. Table 1 gives more information about the environment in which the proposed method has been used. Also, C^{++} code for the proposed system was written with the help of the OMNET++ simulation environment.

Besides, Figure 2 showed the proposed fog node select optimal server to pass request from IoT sensors, in case of requested passed from different fog nodes at the same time, to redirect to different servers.

Table 1. Environment specifications for the proposed system

Operating systems	Windows 10 pro, 64-Bit
CPU	Core (TM) I5-4210U
RAM	8.00 GB
Implementation tools	OMNET++ 4.6, INET 3.3.0, and FogNetSim++

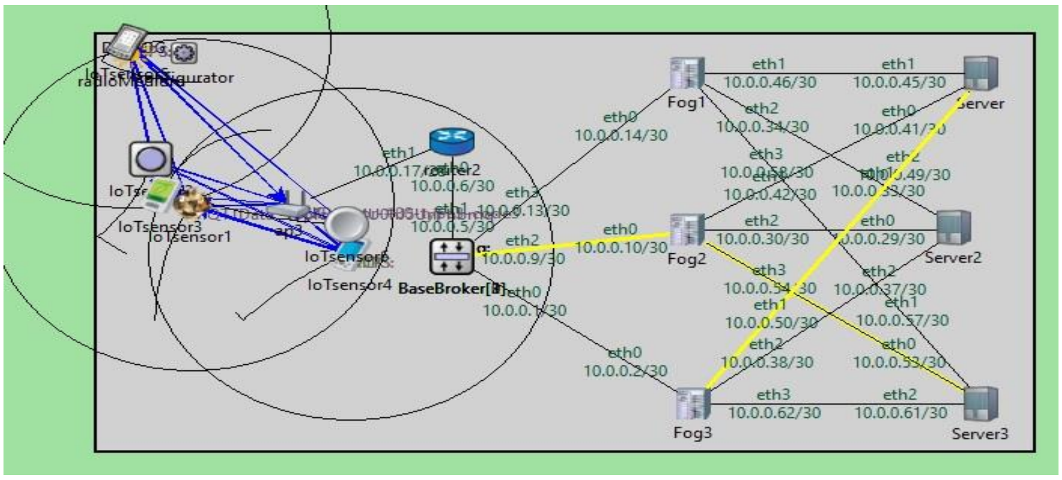

Figure 2. The proposed fog node redirect request to the optimal server

4. RESULTS AND DISCUSSION

The proposed system is based on four case studies, the 1st is represented without load balancer, while the 2nd case study is based on the load balancer with least connection (LC) algorithm, the 3rd case study based on the load balancing algorithms with weighted round Robin (WRR), and the 4th case study based on the hybrid approach combined LC and WRR implemented in each fog nodes.

4.1. The 1st case study of without load balance

The suggested system is based on four key findings. The first case study examines the condition of overload and high traffic load, necessitating a load balancing method to balance the load and reduce reaction time, which is exemplified by the second case study through the fourth case study of a load balancing strategy with increasing complexity (LC, WRR, and hybrid approach). Table 2 showed the HTTP web request by IoT sensors to the connected servers through fog node. Table 3 showed the channel resources allocation for IoT sensors and how sensors get benefits from the channel through channel utilization, and the amount of time required for each packet in queue before started to process by the servers.

Table 2. Principal evaluation criteria for the 1st case study

HTTP WEB request command	Throughput (Bps)	Latency (ms)	Response time (ms)	Packet loss rate (%)
Wireless-IoT-sensor 1	36.3977	39789.96	54.506	0.6877
Wireless-IoT-sensor 2	3.2	963.144	1.155	1.263
Wireless-IoT-sensor 3	39.591	20542.368	49.660	1.523
Wireless-IoT-sensor 4	42.832	14574.168	43.897	0.816
Wireless-IoT-sensor 5	35.099	13486.368	56.192	1.011
Wireless-IoT-sensor 6	4.954	1038.408	0.518	0.789
Server 1	2.498	1038.613	/	0.067
Server 2	95.824	29924.123	/	5.128
Server 3	61.801	19100.73	/	1.796
Fog 1	52.22	17008.123	/	2.018
Fog 2	56.535	16010.109	/	2.906
Fog 3	55.306	15905.056	/	0.898

Table 3. Channel resources allocation for without load balancer case study

Requests	Channel idle (%)	Channel utilization (%)	Queuing time (sec)
Wireless-IoT-sensor 1	69.431	0.098	27.027
Wireless-IoT-sensor 2	68.803	6.492	2.574
Wireless-IoT-sensor 3	69.969	0.737	1.287
Wireless-IoT-sensor 4	61.370	7.167	34.749
Wireless-IoT-sensor 5	70.032	0.0081	36.036
Wireless-IoT-sensor 6	70.042	7.265	15.444
Server 1	69.993	6.864	25.74
Server 2	69.974	0.0421	69.498
Server 3	67.233	6.793	2.574
Fog 1(avg all interfaces)	70.134	3.473	9.928
Fog 2(avg all interfaces)	72.236	5.919	10.919
Fog 3(avg all interfaces)	71.458	4.790	12.980

4.2. The 2nd case study of least connection load balance algorithm

The proposed system in the 2nd case study based on the least connection load balancing algorithm to balance the load within the three fog nodes by distribute the IoT sensors to the idle server from the active server pool through take into consideration the amount of connection overloading. Table 4 showed the HTTP request distributed among three servers. Table 5 showed the channel resources allocation for IoT sensors and it enhanced compared with the 1st case study through channel availability and utilization are increased and queuing time is decreased.

Table 4. The main evaluation parameters of the 2nd case study

HTTP WEB request command	Throughput (Bps)	Latency (ms)	Response time (ms)	Packet loss rate (%)
Wireless-IoT-sensor 1	44.517	32143.25	44.031	0.783
Wireless-IoT-sensor 2	3.989	778.05	0.933	1.370
Wireless-IoT-sensor 3	51.45	16594.6	40.117	1.56
Wireless-IoT-sensor 4	55.66	11773.35	35.461	0.989
Wireless-IoT-sensor 5	45.613	10894.6	45.393	1.211
Wireless-IoT-sensor 6	6.44	838.85	0.418	0.891
Server 1	3.248	839.016	/	0.090
Server 2	124.524	24173.39	/	6.07
Server 3	80.313	15430.01	/	2.204
Fog 1	69.714	13480.117	/	2.534
Fog 2	69.408	12480.287	/	1.101
Fog 3	75.9	15160.097	/	1.661

Table 5. Channel resources allocation for 2nd case study

Requests	Channel idle (%)	Channel utilization (%)	Queuing time (sec)
Wireless-IoT-sensor 1	95.112	0.134	20.58
Wireless-IoT-sensor 2	94.251	8.894	1.96
Wireless-IoT-sensor 3	95.850	1.01	0.97
Wireless-IoT-sensor 4	84.070	9.820	25.92
Wireless-IoT-sensor 5	95.934	0.0113	26.88
Wireless-IoT-sensor 6	95.948	9.953	11.52
Server 1	95.881	9.404	19.2
Server 2	95.857	0.057	51.84
Server 3	92.101	9.307	1.92
Fog 1(avg all interfaces)	95.937	4.996	4.68
Fog 2(avg all interfaces)	96.896	5.187	5.127
Fog 3(avg all interfaces)	97.855	5.615	5.068

4.3. The 3rd case study of weighted round robin load balance algorithm

In this case the proposed system is based on the weighted round robin load balance algorithm to distribute load among the connected distributed servers with distributed fog nodes depending on the weight value assigned to each server to decide which the optimal server for the incoming requests. Table 6 showed the main evaluation parameters for the 3rd case study. Table 7 showed the channel allocation for distributed fog computing environment with distributed three servers.

Table 6. The main evaluation parameters of the 3rd case study

HTTP WEB request command	Throughput (Bps)	Latency (ms)	Response time (ms)	Packet loss rate (%)
Wireless-IoT-sensor 1	46.741	30536.087	34.431	0.743
Wireless-IoT-sensor 2	4.188	739.147	0.793	1.301
Wireless-IoT-sensor 3	54.022	15764.87	34.099	1.489
Wireless-IoT-sensor 4	58.443	11184.682	30.141	0.939
Wireless-IoT-sensor 5	47.892	10349.87	38.584	1.150
Wireless-IoT-sensor 6	6.761	796.907	0.355	0.846
Server 1	3.409	797.065	/	0.085
Server 2	130.749	22964.729	/	5.772
Server 3	84.3278	14658.513	/	2.093
Fog 1	85.459	12132.105	/	0.734
Fog 2	82.32	11232.258	/	1.058
Fog 3	80.006	13644.087	/	0.288

Table 7. Channel utilization for 3rd case study

Requests	Channel idle (%)	Channel utilization (%)	Queuing time (sec)
Wireless-IoT-sensor 1	100	0.140	19.153
Wireless-IoT-sensor 2	99.212	9.338	1.824
Wireless-IoT-sensor 3	100	1.060	0.912
Wireless-IoT-sensor 4	93.153	10.311	24.624
Wireless-IoT-sensor 5	100	0.0118	25.536
Wireless-IoT-sensor 6	100	10.450	10.944
Server 1	98.908	9.873	17.28
Server 2	99.893	0.060	49.248
Server 3	95.939	9.771	1.824
Fog 1(avg all interfaces)	98.967	5.194	4.586
Fog 2(avg all interfaces)	98.125	5.243	5.024
Fog 3(avg all interfaces)	98.829	5.293	4.561

4.4. The 4th case study of hybrid load balance approach

The fourth case study based on the hybrid approach to distribute the load among servers through the weighted summation approach which assign weight value for each active servers and redirect traffic to the least weight value depending on the least connection and weighted round robin algorithms depending on the distributed fog nodes. Table 8 showed the used evaluation parameters for the 4th case study. Table 9 showed the channel utilization with queue time of the 4th case study.

The proposed system showed the distributed fog hybrid load balancer approach better in case HTTP and FTP file transfer with different file size compared with the least connection algorithm and weighted round robin algorithms. The least connection is better from weighed round robin in case of FTP file transfer, while the weighted round robin is better from least connection in case of HTTP requests. In addition, the hybrid approach was better in channel utilization from both standalone algorithms. Figure 3 and Figure 4 showed the system comparison. The proposed system compared with other related works and it showed better evaluation results as it showed in Table 10.

Table 8. The main evaluation parameters of the 4th case study

HTTP WEB request command	Throughput (Bps)	Latency (ms)	Response time (ms)	Packet loss rate (%)
Wireless-IoT-sensor 1	49.077	29009.282	32.709	0.705
Wireless-IoT-sensor 2	4.397	702.189	0.753	1.235
Wireless-IoT-sensor 3	56.722	14976.626	32.394	1.414
Wireless-IoT-sensor 4	61.364	10625.447	28.633	0.892
Wireless-IoT-sensor 5	50.286	9,832.376	36.654	1.092
Wireless-IoT-sensor 6	7.098	757.061	0.337	0.803
Server 1	3.578	757.211	/	0.080
Server 2	137.285	21816.492	/	5.483
Server 3	88.543	13925.587	/	1.988
Fog 1	89.731	10918.894	/	0.697
Fog 2	86.435	10109.032	/	1.005
Fog 3	84.005	12279.678	/	0.273

Table 9. Channel utilization with queuing time of the 4th case study

Requests	Channel idle (%)	Channel utilization (%)	Queuing time (sec)
Wireless-IoT-sensor 1	100	0.159	17.058
Wireless-IoT-sensor 2	99.708	10.678	1.624
Wireless-IoT-sensor 3	100	1.212	0.812
Wireless-IoT-sensor 4	96.015	11.790	21.930
Wireless-IoT-sensor 5	100	0.013	22.743
Wireless-IoT-sensor 6	100	11.950	9.747
Server 1	100	11.290	15.39
Server 2	100	0.069	43.861
Server 3	98.929	11.173	1.444
Fog 1(avg all interfaces)	100	5.609	3.668
Fog 2(avg all interfaces)	100	6.232	4.421
Fog 3(avg all interfaces)	99.99	7.271	4.104

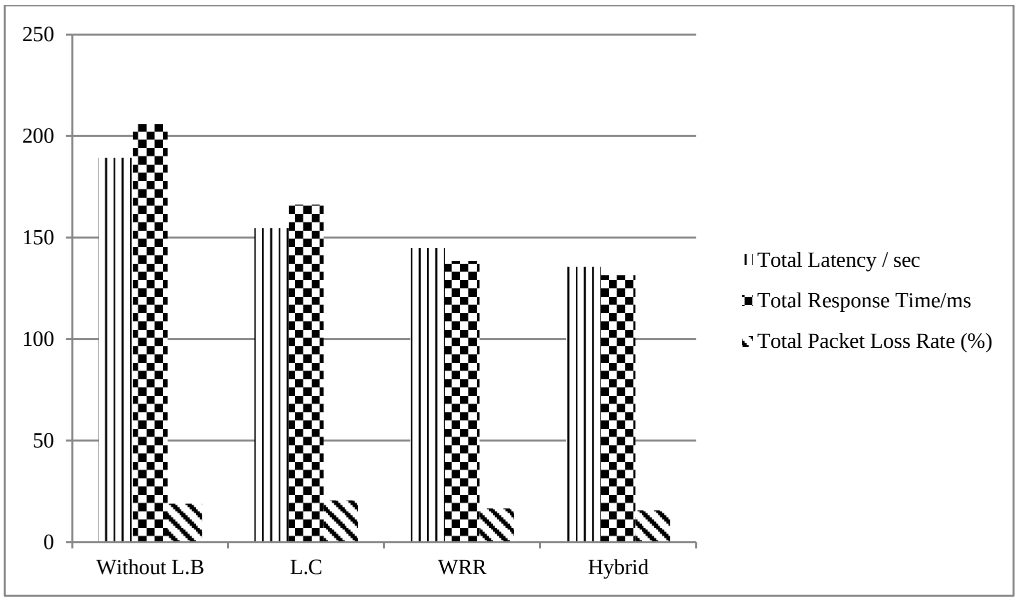

Figure 3. The total latency, response time and packet loss rate of the proposed system

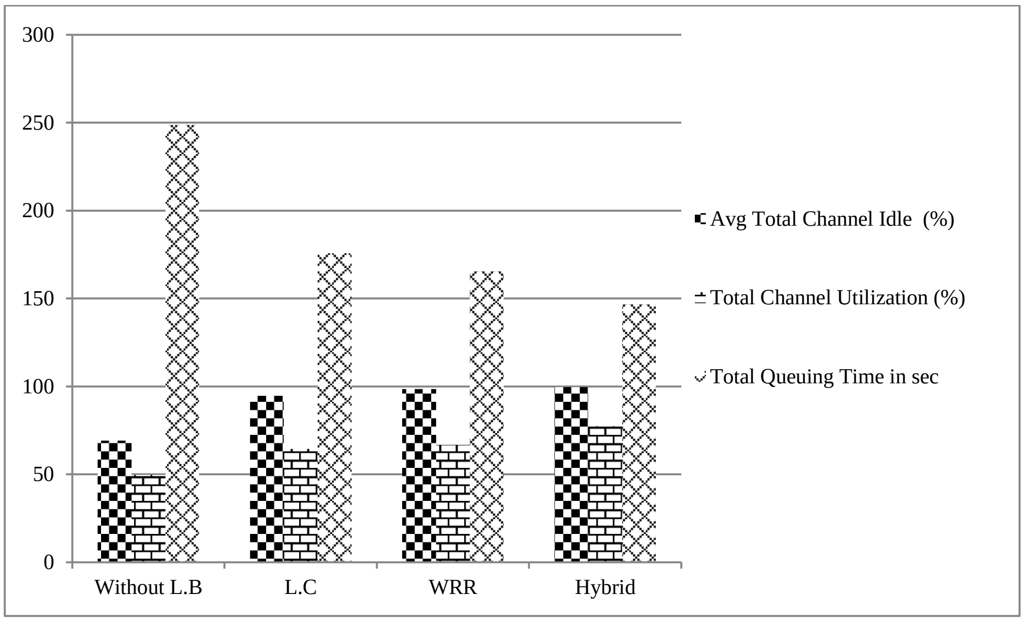

Figure 4. Channel resource allocation for the proposed system case studies

Table 10. The proposed system comparison with the other related works

Ref, year	Arch, tool	Algorithm	Total response time (ms)	Total time taken (sec)
[11], 2020	Cloud-Fog load balancing algorithms, iFogSim	Proximity algorithm	/	90
		Cluster algorithm	/	100
[17], 2022	Hybrid load-balancing-Fog, iFogSim	Optimizing Processing Time (OPT)	309.5	/
		First Come First Serve (FCFS)	326.7	/
		Priority algorithm	323.7	/
[18], 2022	cloud-fog, Java Netbeans and cloud analyst	Round Robin (RR)	270.14	/
		Whale optimization algorithm with bat algorithm (WOA-BAT)	256.59	/
The proposed system	3 Fog-3 Servers, Fognetsim^{++}, OMNET^{++}	Weighted Round Robin, least connection	131.48	57.838

5. CONCLUSION

The massive volume of data created by IoT devices must be optimally serviced. The data must be preprocessed before the network efficiency and service response time may be enhanced. The suggested multilayer architecture has a system for balancing the load that works well and lets access control be optimized. The main functionalities of the distributed fog layer in the proposed system is to efficient resource management, run complex jobs, data caching, computation offloading, support dynamic selection of fog nodes, improving collecting data from sensors, and scheduling tasks among active servers. The results show that the proposed system improved network performance by it ensures the efficient servicing of the IoT requests coming from the IoT layer using the services offered by distributed fog computing with hybrid load balancing algorithm.

REFERENCES

[1] K. S. Awaisi, S. Hussain, M. Ahmed, A. A. Khan, and G. Ahmed, “Leveraging IoT and Fog computing in healthcare systems,” *IEEE Internet Things Mag.*, vol. 3, no. 2, pp. 52–56, Jun. 2020, doi: 10.1109/iotm.0001.1900096.

[2] M. S, D. G, C. B, and D. D, “IoT based monitoring and control system using sensors,” *June 2021*, vol. 2, no. 2, pp. 111–120, Jun. 2021, doi: 10.36548/jismac.2021.2.004.

[3] M. Verma, N. Bhardwaj, and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *International Journal Information Technology and Computer Science*, vol. 8, no. 4, pp. 1-10, 2016, doi: 10.5815/ijitcs.2016.04.01.

[4] M. Kaur and R. Aron, “A systematic study of load balancing approaches in the fog computing environment,” *J. Supercomput.*, vol. 77, no. 8, pp. 9202–9247, Feb. 2021, doi: 10.1007/s11227-020-03600-8.

[5] S. S. Karthik and A. Kavithamani, “Fog computing-based deep learning model for optimization of microgrid-connected WSN with load balancing,” *Wirel. Netw.*, vol. 27, no. 4, pp. 2719–2727, Apr. 2021, doi: 10.1007/s11276-021-02613-2.

[6] 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,” *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 11, pp. 4951–4966, Feb. 2020, doi: 10.1007/s12652-020-01768-8.

[7] G. Sinha and D. Sinha, “Enhanced weighted round robin algorithm to balance the load for effective utilization of resource in cloud environment,” *EAI Endorsed Trans. Cloud Syst.*, vol. 6, no. 18, p. 166284, Sep. 2020, doi: 10.4108/eai.7-9-2020.166284.

[8] Z. Liu *et al.*, “DistCache: provable load balancing for large-scale storage systems with distributed caching.” *arXiv*, Feb. 14, 2019. doi: 10.48550/arXiv.1901.08200.

[9] M. Kumar and S. C. Sharma, "Load balancing algorithm to minimize the makespan time in cloud environment," *World Journal of Modelling and Simulation*, vol. 14, no. 4, pp. 276-288, 2018.

[10] C. H. Hong and B. Varghes, "Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1-37, 2019, doi: 10.1145/3326066.

[11] T. Aladwani, "Scheduling IoT healthcare tasks in fog computing based on their importance," *Procedia Computer Science*, 163, 560-569, 2019, doi: 10.1016/j.procs.2019.12.138.

[12] L. Li, Q. Guan, L. Jin and M. Guo, "Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system," in *IEEE Access*, vol. 7, pp. 9912-9925, 2019, doi: 10.1109/ACCESS.2019.2891130.

[13] J. Aguilar, M. B. Sanchez, M. Jerez, and M. Mendonca, "An extension of the MiSCi Middleware for smart cities based on fog computing," In *Smart Cities and Smart Spaces: Concepts, Methodologies, Tools, and Applications*. IGI Global, 2019, pp. 778-798, doi: 10.4018/jitr.2017100102.

[14] A. Asghar, A. Abbas, H. A. Khattak, and S. U. Khan, “Fog based architecture and load balancing methodology for health monitoring systems,” *IEEE Access*, vol. 9, pp. 96189–96200, 2021, doi: 10.1109/access.2021.3094033.

[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, Oct. 2021, doi: 10.1007/s10723-021-09584-w.

[16] M. I. Bala and M. A. Chishti, "Offloading in cloud and fog hybrid infrastructure using iFogSim," *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2020, pp. 421-426, doi: 10.1109/Confluence47617.2020.9057799.

[17] A. Abuhamdah and M. Al-Shabi, “Hybrid load balancing algorithm for fog computing environment,” *International Journal of Software Engineering and Computer Systems*, vol. 8, no. 1, pp. 11–21, Jan. 2022, doi: 10.15282/ijsecs.8.1.2022.2.0092.

[18] A. Saoud and A. Recioui, “Hybrid algorithm for cloud-fog system based load balancing in smart grids,” *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 1, pp. 477–487, Feb. 2022, doi: 10.11591/eei.v11i1.3450.

[19] H. Tran-Dang and D. -S. Kim, "FRATO: fog resource based adaptive task offloading for delay-minimizing IoT service provisioning," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2491-2508, 1 Oct. 2021, doi: 10.1109/TPDS.2021.3067654.
[20] T. Wang, Y. Liang, W. Jia, M. Arif, A. Liu, and M. Xie, "Coupling resource management based on fog computing in smart city systems," *J. Netw. Comput. Appl.*, vol. 135, pp. 11–19, Jun. 2019, doi: 10.1016/j.jnca.2019.02.021.
[21] N. X. Phi, C. T. Tin, L. N. K. Thu, and T. C. Hung, "Proposed load balancing algorithm to reduce response time and processing time on cloud computing," *International Journal of Computer Networks & Communications (IJCNC)*, vol. 10, no. 3, pp. 87–98, May 2018, doi: 10.5121/ijcnc.2018.10307.
[22] L. Campanile, M. Gribaudo, M. Iacono, and M. Mastroianni, "Performance evaluation of a fog WSN infrastructure for emergency management," *Simulation Modelling Practice and Theory*, vol. 104, p. 102120, Nov. 2020, doi: 10.1016/j.simpat.2020.102120.
[23] M. R. Alizadeh, V. Khajehvand, A. M. Rahmani, and E. Akbari, "Task scheduling approaches in fog computing: A systematic review," *International Journal of Communication Systems*, pp. 1-36, Aug. 2020, doi: 10.1002/dac.4583.
[24] M. Abbasi, E. Mohammadi-Pasand, and M. R. Khosravi, "Intelligent workload allocation in IoT–Fog–cloud architecture towards mobile edge computing," *Computer Communications*, vol. 169, pp. 71–80, Mar. 2021, doi: 10.1016/j.comcom.2021.01.022.
[25] E. de Matos *et al.*, "Context information sharing for the internet of things: a survey," *Computer Networks*, vol. 166, pp. 1-19, Jan. 2020, doi: 10.1016/j.comnet.2019.106988.

BIOGRAPHIES OF AUTHORS

Abrar Saad Kadhim is currently pursuing her Master of Science at University of Babylon. She will graduate in 2022. His Bachelor of Science is from University of Babylon in Iraq- College of IT, Department of Information Networks. Her main area of research is network load balancing in fog computing. She is currently focusing on the load-balancing algorithm for distributed fog computing in internet of things environment. She can be contacted at email: abrar.kadhim@student.uobabylon.edu.iq.

Mehdi Ebady Manaa is currently an assistant professor in the department of information network, College of Information Technology, University of Babylon. He received his bachelor's degree from the University of Babylon, college of science in 1999-2000. His Master of Science from University Utara Malaysia (UUM), Malaysia in 2012. He received his PhD in Computer Science and in the field Network Security, Data Mining, and Cloud Computing from the University of Babylon - College of Information Technology in 2016. He is currently focusing on the detection of the attacks using data mining techniques. The main interesting fields are network administration and distributed object-oriented topics, network protocols TCP/IP suite, mobile Ad-hoc network (MANet), data mining techniques (clustering and classification), communication software, network security, cloud computing, internet of things, and unstructured data. He can be contacted at email: it.mehdi.ebady@itnet.uobabylon.edu.iq.