Improving the performance of the fuzzy-internet of things joint system by using an efficient service deployment algorithm

Ali Ahmed Razzaq, Kunjam Nageswara Rao
Department of Computer Science and Systems Engineering, College of Engineering, Andhra University, Visakhapatnam, India

Article Info

Article history:

Received Nov 10, 2023
Revised Jul 1, 2024
Accepted Jul 26, 2024

Keywords:

Cloud computing
Energy consumption
Fog computing
Internet of things
Quality of service
Service localization

ABSTRACT

The aim of this paper to present research proposes a quality of service (QoS)-fog service placement algorithm, which is in the context of providing better performance when there are requests for high-priority (delay-sensitive) services. Our proposed algorithm attempts to maintain the implementation of delay-sensitive services within a fog environment. Its performance was evaluated by comparing it with two placement strategies (cloud only, edge-ward), using the MobFogSim simulator. The results showed that the proposal achieved better performance than the two mentioned strategies, from two perspectives (according to the scenarios). The first is at the service level, where the algorithm was able to achieve a lower average response time and the other is at the system level, as it was able to reduce the total energy consumption by adopting a mechanism to save energy when there is a low network load.

Corresponding Author:

Ali Ahmed Razzaq
Department of Computer Science and Systems Engineering, College of Engineering, Andhra University
Visakhapatnam 530003, India
Email: taifali607@gmail.com

1. INTRODUCTION

During the coming years, billions of things (such as home appliances, irrigation outlets, and lighting poles) will be connected to the internet, and this will lead to a real revolution that affects the amount of data that is collected and shared. Of course, what is presented is known as the internet of things (IoT). The term cloud computing refers to a service provided by data centers, and it plays an important role in processing application services generated by the IoT. The load on cloud servers increases with the increase in the number of IoT devices (network congestion occurs), and given that these servers are usually located spatially far from the IoT network, this will lead to a noticeable increase in transmission time and a decrease in the performance of time-sensitive applications. Other related work can found in [1]–[6]. Fog computing is a new technology for decentralized (distributed) computing, which aims to improve efficiency by trying to reduce the amount of data that must be transferred to the cloud for processing, analysis, and storage. In [7], [8] authors see that through Fog computing, application services are dealt with at the edge of the network, as it is the intermediary between the cloud and the end user, so its spatial proximity to the IoT network reduces network congestion.

Two approaches are followed in the context of implementing IoT application services. In the first approach, an IoT device processes its own tasks locally or chooses to offload them computationally to the nearest fog node or to the cloud, where it is considered to localize processing and maximize the total number of tasks served locally in the IoT devices and within the fog environment while meeting both scheduling requirements final and energy constraints. This is achieved according to different strategies, Tang *et al.* [9] considers that some of them rely on the battery power level of the IoT device and the availability of fog nodes (availability depends on the length of the task queue), to determine the processing location. While Zhu *et al.* [10] introduced an idea that relies on having two criteria for making offloading decisions, namely task execution time and the energy consumed during that process, in the cases of local and fog execution, and as a result of the comparison, the location of processing the task/service is determined.

There is another approach that excludes IoT devices from processing operations, where fog and cloud resources are exploited to accomplish end-user tasks so that all computational tasks of an IoT device are offloaded to fog nodes, to share the incoming data for processing, or may send part of this load goes to the cloud. Gupta *et al.* [11] propose a strategy to service placement in fog nodes close to IoT users. Considering that these nodes ultimately have limited resources and cannot handle all requests, the proposed strategy resorts to placement the remaining services on cloud devices in data centers. Other similar work can be found [12], [13].

One of the most promising solutions aimed at improving the utilization of available resources in a foggy environment studied by [14], [15]. Where they used the offloading of computational tasks (migration of virtual machines or containers), in other words, sharing the load between fog nodes, which reflects positively on the quality of service (QoS) provided to IoT applications by reducing delay. However, according to Gao *et al.* [16], the disadvantages of this improvement is increase in the energy consumption of the fog system. Based on this gap, several policies have been proposed to deploy IoT application services in the fog-cloud system, one of them [17] depend on the type of service, so that the goal is either to reduce the response time if the service is critical, or to try to reduce the fog system's use of the energy resource, if It was normal. While Alenizi and Rana [18] used dynamic task scheduling (DTS), with the aim of reducing the overall delay of critical (delay-sensitive) tasks, in addition to dynamic energy control (DEC) in order to reduce energy consumption in the fog layer while maximizing the use of resource-limited fog nodes. Additionally, Kayal [19] relied on publishing services without any central coordination, as their design was based on the Markov approximation method, seeking to reduce both fuzzy energy consumption and the communication cost of applications. Energy saving is an important area of research not only for fog computing, but also in cloud computing, where searching for idle data centers and putting them into energy saving mode or shutting them down has a significant impact on system energy consumption. Shutdown requires the migration of virtual machines to active centers that are not sufficiently loaded, and it is also possible to activate a closed center in order to ensure that the QoS requirements of the system. Other related studies on deploy IoT application services in the fog-cloud system can found in [20]–[24].

Based on the provided information, the research proposes the QoS-fog algorithm, which aims to improve QoS in the fog environment. This algorithm seeks to deliver enhanced performance when there are requests for high-priority services, especially those sensitive to latency. It manages the network load directed towards fog nodes, thereby reducing energy consumption in the fog environment.

This paper is structured as follows: the proposed algorithm presented in section 2. In section 3, discussion and performance evaluation. Finally, section 4 concludes our paper.

2. THE PROPOSED ALGORITHM

2.1. Proposed system architecture

This system consists of three components as follows:

- IoT devices (smart phones, smart home devices, and smart watches), which send their data in addition to service requests to the outside world through the fog gateway, where the latter sends the requests to the fog resource manager, who is responsible for offloading them to the nodes in the hierarchical structure that is supervised, according to the proposed algorithm QoS-Fog.
- The fog environment consists of nodes that perform both computational and relay tasks, and nodes that act as relays only, that is, they forward data (note that both types of fog environment devices use software-defined networks). Naturally, fog nodes can host a number of virtual machines as their resources allow, and since the fog resource manager is responsible for determining where to deploy services, in other words, (implementing/ executing) the end user's request. Resource manager will have several options according to the type of service requested (its priority) and the extent of Fog nodes availability at each level of the hierarchy (the selected hierarchical topology is only two levels deep (to prevent increased complexity and the subsequent delay that may affect the QoS provided to IoT applications).

Therefore, the places where services are hosted are one of three: the first level of the fog pyramid (high-priority services), the second level (low-priority services), and the cloud data center (of course, according to the proposed algorithm, low-priority services may be hosted by nodes of the first level of the hierarchy, Only if there are idle nodes, they enter energy saving mode, due to the decrease in computational load, in light of the congestion of the lower level nodes of course, in an attempt to avoid sending the mentioned service to the cloud).

– Cloud environment: End-user requests that the fog environment was unable to process are sent to the cloud servers, each of which has its own resources of processing power, storage space, and bandwidth.

2.2. The mathematical models used in the proposed algorithm

Our goal under this architecture is to achieve the QoS required for high-priority (time-sensitive) services, and manage energy at the fog level at the same time. Accordingly, the most prominent mathematical relationships upon which our work is based to achieve its goal are the following:

a) The energy consumption model adopted in the algorithm is based on the CPU utilization, modeled as a linear function of this utilization. Since the deployment/allocation of services excludes IoT devices, where there is no local processing, processing occurs at the fog or cloud data center level. Therefore, the total energy consumption of the system will depend on the sum of two factors: the energy consumption of fog nodes while executing the services allocated to them, plus the cloud energy consumption, given by (1):

$$E_{total} = \sum_{i=1}^{m} E_i + \sum_{p=1}^{k} E_L \tag{1}$$

The energy consumption of the fog nodes while executing the services assigned to them is represented according to the (2) [25], [26]:

$$E_i = P(u)_i \times \sum_{j=1}^{n} x_{ji} \cdot \frac{s_j^{size}}{s_j^{CPU}} \tag{2}$$

We find that this consumption depends on four factors:

– Electrical power consumption $P(u)_i$, which is modeled as a linear function that follows the amount of CPU usage $u_(i)$, and is given according to the (3) [25], [26]:

$$P(u)_i = Fn_i^{min} + (Fn_i^{max} - Fn_i^{min}) \cdot u_i \tag{3}$$

Where the symbols Fn_i^{max} and Fn_i^{min} indicate the upper (full utilization) and lower (idle state) limits of the fog node's power consumption.

– The logical variable $x_(ji)$through which we express whether the fog node Fn_i is hosting the service S_j or not. Naturally, this variable takes one of two values, either zero or one.

– The parameters S_j^{size} and S_J^{CPU} express the size of the task and the amount of processing capacity required to implement it, respectively.

b) With regard to the time factor, we are dealing with two important criteria in our research:

– Response time for the service Res_{sj} is considered a very important criterion, as to achieve QoS for high-priority services, this time must not exceed its deadline (time constraint).

The calculation of this time depends on where the service is hosted or placed for execution, and according to the architecture of the proposed system, the service will be located either in the two-level fog environment or in the cloud center, and therefore this time is given according to the (4):

$$\begin{aligned} Res_{sj} = (2T_{g,RM} + 2T_{RM,FN_{L1}} + T_{(W+PRO)_{L1}})x_{j,FNL1} + \\ (2T_{g,RM} + 2T_{RM,FN_{L2}} + T_{(W+PRO)_{L2}})x_{j,FNL2} \\ +(2T_{g,RM} + 2T_{RM,CM} + 2T_{CM,PM} + T_{(W+PRO)_{PM}})x_{j,cloud} \end{aligned} \tag{4}$$

The symbol $T_{g,RM}$ denotes the delay between the fog gateway and the resource manager in the fog system, while the symbols $T_{RM,FN_{L1}}$, $T_{RM,FN_{L2}}$, and $T_{CM,PM}$ express the delay between the fog resource manager and a service implementing node in the first and second hierarchical levels, and between the cloud resource manager and an implementing node in the data center, respectively. The waiting times experienced by service requests, including delays in processing and queuing, when executing occurs in first fog level, second fog level, in addition to cloud, are represented by the symbols $T_{(W+PRO)_{L1}})$, $T_{(W+PRO)_{L2}})$, and $T_{(W+PRO)_{PM}})$. As for the symbols $x_{(j,FNL2)}$, $x_{(j,FNL1)}$, and $x_{(j,cloud)}$, they represent logical variables that indicate the execution or non-execution of the service requested by the IoT device in the second level fog node, first level fog node, and in the cloud at straight.

– The length of the queue in which tasks are lined up for processing, is a factor that has a significant impact on determining where to host the high-priority service S_h within the second-level fog nodes, and this is what happens when the first level is congested. Be careful not to move the high-priority service to the cloud for processing, if there are fog nodes dedicated to the low-priority services S_l that are able to meet the requirements of the high-priority service in terms of resources and time constraints (deadline), or even in the case of time or resource unavailability, there may be a forced migration of low priority services to the cloud for processing (conditional forced migration). From the above, we find that the queues of the second level fog nodes handle both types of services, and therefore the queue length is given according to (5) [2]:

$$T_Q = \sum_{h=1}^{m} T_h^{process} + \sum_{l=1}^{k} T_l^{process} \tag{5}$$

The symbols $T_h^{process}$ and $T_l^{process}$ represent the expected execution time for high- and low-priority services, respectively. Of course, the queue waiting time must not exceed a specific threshold, in order for the fog node at the second level to be considered acceptable in terms of the possibility of hosting a high-priority service for execution.

2.3. The proposed quality of service-fog algorithm

Since IoT devices have limited resources, misuse of these resources can limit its usefulness, so offloading computational tasks between IoT devices and fog nodes or cloud servers improves the efficiency of resource management. Our proposed algorithm starts its work when the resource manager in the fog network receives service requests. A distinction is made between two types of requests, one with high priority and the other with low priority. Naturally, the process of classifying requests comes from the desire to take into account time-sensitive (critical) applications, and here the resource manager will take their deadlines as an important factor in determining the sequence of assigning them to the fog nodes for implementation (as this requires arranging them in ascending order).

Fog nodes periodically send an update on the status of their resources (processing power, memory) to the resource manager. This centralized view of resources gives the possibility of assigning services to appropriate nodes quickly, and here we take into account the proximity of the node to the resource manager if the service is high priority, i.e. sensitive to delay (considering that the architecture of the proposed system proposes a two-level structure of fog nodes, based on distance and computing capacity fog nodes belong to these levels), which means assigning them to one of the fog nodes of the first level, and this requires the manager to search for a node that meets the requirements for implementing the high-priority service. If the appropriate node is found, the algorithm before finally hosting the service, will check the processing capacity of the chosen node compared to a threshold (this condition is considered an additional safety factor on the readiness of the node, which basically meets the requirements, given that updating resources is periodic, and during the period between two updates, migrations may occur between fog nodes). The high-priority service is executed if the condition is met, otherwise the fog node is responsible for selecting a node from the same level (closest to the manager) in order to implement the service, out of the necessity of maintaining the QoS.

Fog nodes periodically send updates to the resource manager about their resource status (processing power and memory). This centralized view of resources allows for quick assignment of services to appropriate nodes. Here, we consider the proximity of the node to the resource manager if the service is high priority, meaning sensitive to delay (considering that the proposed system architecture suggests a two-level structure of fog nodes, where fog nodes belong to these levels based on distance and computing capacity). This implies assigning the service to one of the first-level fog nodes, requiring the manager to search for a node that meets the requirements for executing the high-priority service. If a suitable node is found, the algorithm will, before finally hosting the service, verify the processing capacity of the selected node against a threshold (this condition is considered an additional safety factor for the readiness of the node, which essentially meets the requirements, given that resource updates are periodic, and during the period between updates, migrations may occur between fog nodes). The high-priority service is executed if the condition is met; otherwise, the fog node is responsible for selecting a node from the same level (closest to the manager) to execute the service, driven by the necessity of maintaining the QoS.

The node then sends a hosting message to its neighbors. If it receives multiple responses, the crowded node, so to speak, selects the nearest node for hosting. However, if it does not receive a response, the crowded node sends feedback to the manager, who will schedule the incoming service to the nodes of the level furthest from the center in the hierarchical structure (second level). Here, the selection of a second-level node depends on the length of the waiting queue in it, as the time factor becomes very important, given that the service is sensitive to delay and its deadline may not be exceeded (intuitively, the nodes chosen for hosting by the manager meet the service requirements, but the trade-off between nodes depend on the time length of the waiting queue). If no server nodes are available, a low-priority service (or services) will be migrated to cloud data centers for processing, and the high-priority service will be hosted in its place for processing locally (fog).

Additionally, the low-priority services, as we mentioned, are hosted directly by the second-level nodes in the hierarchical structure, and are migrated for cloud processing in two cases, either because of a high-priority service that the fog first-level nodes did not meet its requirements, or because of the congestion of the second-level nodes. It may seem unfair to low-priority services, but our proposal, in the context of its attempt to manage the energy resource efficiently and put nodes into energy saving mode when the network load decreases, takes into account the congestion of the second fog level (that is, if there is a need to migrate a low-priority service to the cloud due to the density of fog-processed services, and if there is a level 1 fog node in energy saving mode (i.e. it has no services to execute, then the low-priority service is scheduled to that node for execution). Figure 1 show the flowchart of the proposed algorithm.

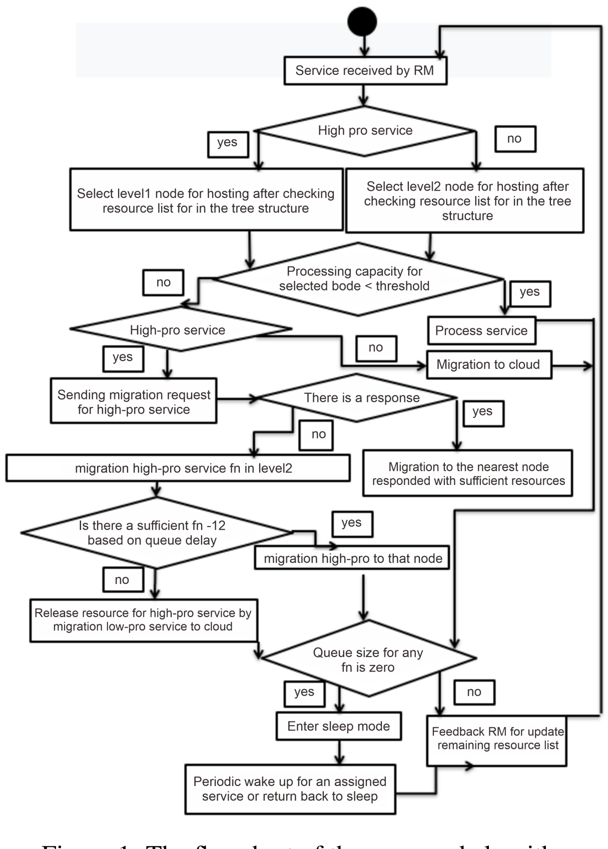
Service received by RM
High pro service
yes
no
Select level1 node for hosting after checking resource list for in the tree structure
Select level2 node for hosting after checking resource list for in the tree structure
Processing capacity for selected bode < threshold
no
yes
High-pro service
Process service
no
Migration to cloud
yes
Sending migration request for high-pro service
There is a response
yes
no
migration high-pro service fn in level2
Migration to the nearest node responded with sufficient resources
Is there a sufficient fn -12 based on queue delay
yes
migration high-pro to that node
no
Release resource for high-pro service by migration low-pro service to cloud
Queue size for any fn is zero
yes
no
Enter sleep mode
Feedback RM for update remaining resource list
Periodic wake up for an assigned service or return back to sleep

Figure 1. The flowchart of the proposed algorithm

3. DISCUSSION AND PERFORMANCE EVALUATION

The MobFogSim simulator [27] was adopted to evaluate the performance of the algorithms (QoS-fog (the proposed algorithm), Edge-ward [16], cloud only [16]). It is noted that this simulator is derived from MyiFogSim [28], an open-source fog computing simulator, developed as an extension to iFogSim [11] to enhance performance by modeling user mobility and the migration of virtual machines/containers between fog nodes, making it a comprehensive evaluation tool for fog computing networks [29]. The simulation was implemented on a personal computer with an Intel Core i5 processor running at 2.4 GHz and 4 GB of RAM. The infrastructure contained a fog resource manager (FRM, sometimes referred to as RM), which controls a variable number of fog nodes (processing and forwarding nodes), ranging from 50 to 1000 nodes depending on the studied scenarios. These nodes are classified into six types based on processing capacity and memory size, where the type plays a role in their placement within the intended hierarchical two-tier topological structure. Additionally, there is a cloud data center connected to the resource manager, and for simplicity, the PMs are considered homogeneous in terms of memory size and processing capacity, as the placement of services in our study occurs in the fog environment, with no focus on the cloud. These services are described by four attributes: size, deadline, processing requirements, and storage requirements, and varying numbers of these services, ranging from 50 to 1000, were tested. The IoT devices were distributed within the coverage range of the hierarchical network topology built based on two factors: Euclidean distance and the resources of the fog computing nodes. The aforementioned description is illustrated in Figure 2.

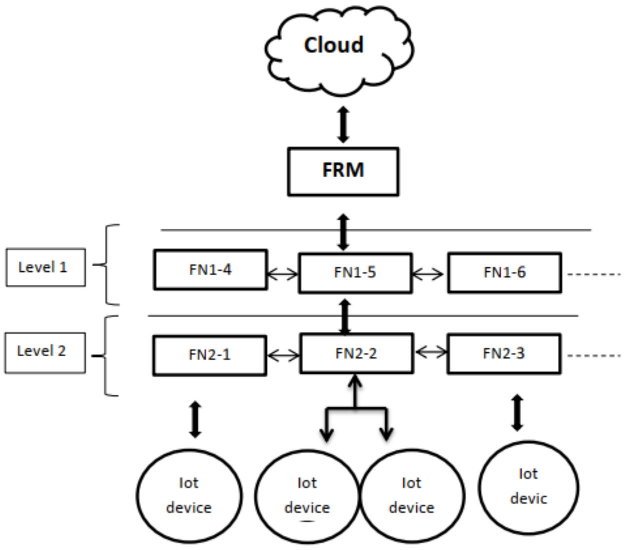

Figure 2. Illustration of the infrastructure under simulation

The evaluation process of the aforementioned algorithms took place by measuring two metrics:

- Total energy consumption: This metric is used to express the amount of energy consumption within the fog-cloud system resulting from the execution of services required by IoT devices. This metric was calculated under the influence of three parameters: the number of services, the number of fog nodes, and the percentage of high priority services to the total number of services.
- Average response time: This metric expresses the average time elapsed between sending a service request by IoT devices and receiving the response. Likewise, this metric was studied under the influence of two parameters: the number of services and the number of fog nodes.

Several parameters were configured to complete the simulation process, which was completed within a time of 500 ms for each scenario studied, and a threshold was set for the length of the waiting frame equal to 50 ms. Table 1 shows the resources of fog nodes and cloud servers (taking into account the heterogeneous nature of fog nodes - i.e. variability in resources - in addition to the characteristics of services, whether high or low priority, in terms of processing capacity and memory size. While variable simulation parameters across studied scenarios related to the service, such as size and deadline. Each one follows a random uniform distribution (RUD) as we illustrated in Table 2.

Table 1. The resources of fog nodes and cloud servers

Fog Node	CPU(MIPS)	MEM(MB)
Type fn_1	500	512
Type fn_2	1000	1024
Type fn_3	1500	2048
Type fn_4	3000	4096
Type fn_5	6000	4096
Type fn_6	8000	2048
Cloud server	15000	10240
high-pro/ low pro service	500-1000-1500-2000-2500	256-512-1024-2048

Table 2. Variable simulation parameters

Services	Size	Deadline
High-pro services	RUD[50,200]MI	RUD [100,400]ms
Low-proservices	RUD[500,1000]MI	RUD[800, 2000]ms

3.1. Study the total energy consumption

As shown in Figure 3, the proposed algorithm showed higher efficiency than its counterparts in terms of energy use (less energy consumption within the shared computing system), with the increase in the number of services sent by the IoT network towards the fog network. The result is explained by the ability of the proposed QoS-fog algorithm to manage the execution of the required services in a way that takes into account the network load, that is, the presence of a small number of service requests will allow idle fog nodes to be entered into energy saving mode, and this will reflect a decrease in the energy consumed by the system, noting that such a mechanism does not exist in edge-ward and cloud only, which leads to higher energy consumption.

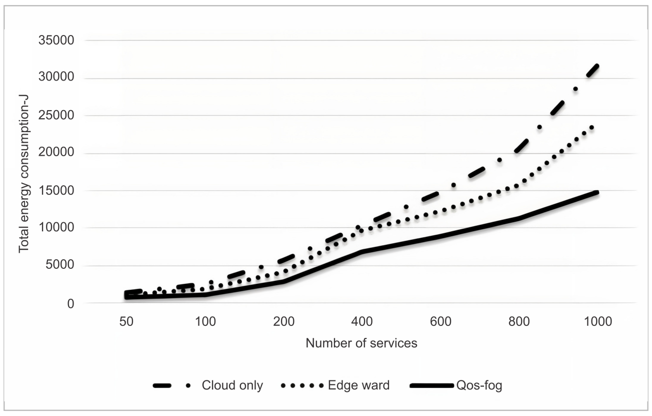

Figure 3. Total energy consumption as a function of increasing number of services

Figure 4 shows the effect of increasing the number of fog nodes on the total system energy consumption. Again, the proposed algorithm is able to achieve lower energy consumption compared to edge-ward and cloud only. We note that as the number increases, the fixed number of services, which is equal to 100 (in all the scenarios studied and expressed in the figure), are hosted on appropriate nodes in terms of resources and the ability to achieve the deadlines for the services. While the rest of the fog nodes enter the energy saving mode, knowing that increasing the number will reduce the probability of sending the required services to the cloud data center for processing, and this plays an important role in the noticeable energy decrease witnessed by the service placement algorithms under study (edge-ward, QoS-fog).

Figure 5 expresses the effect of the ratio of high-priority services to the total number of services required by the system to implement/ execute on the energy consumption metric when applying the three placement algorithms (strategies). As a result of its deployment/placement of high-priority services within the nodes of the first fog level, which has spatial proximity to the fog resource manager and high processing capabilities compared to the second level of nodes, the proposed algorithm was able to reduce the energy consumed, As a result of the reduced time required to implement/execute services, whether in terms of processing or network transmissions. Naturally, the results show that the higher the ratio, the lower the energy at the system level, and this is what we see with the three algorithms, and this behavior is due to the simulation settings that considered the sizes of high-priority services to be lower than low-priority services [14].

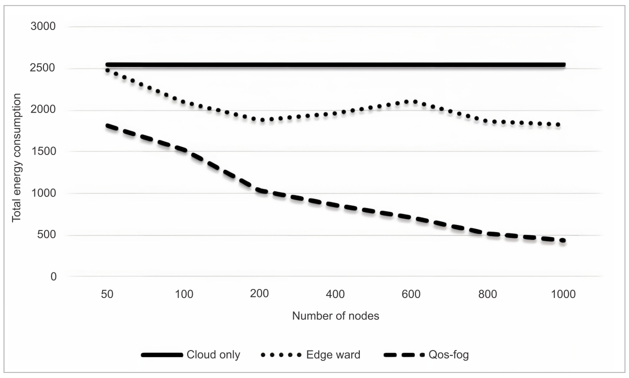

Figure 4. Total energy consumption as a function of increasing the number of fog nodes

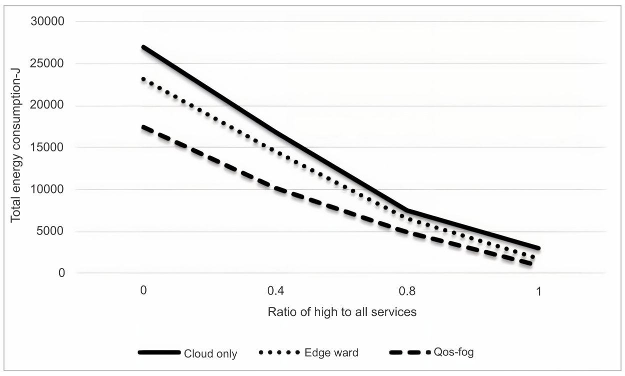

Figure 5. Total energy consumption as a function of the ratio of the number of high-priority services to the total number of services

3.2. Study the average response time

According to Figure 6, the proposed algorithm was able to achieve a lower average response time compared to similar placement algorithms under study, with an increase in the number of services required to be completed by the system. The reason for this superiority is due to our algorithm taking into account the time limits for each service. Note that this average increases with the increase in the number of services, while the proposal maintains the best performance.

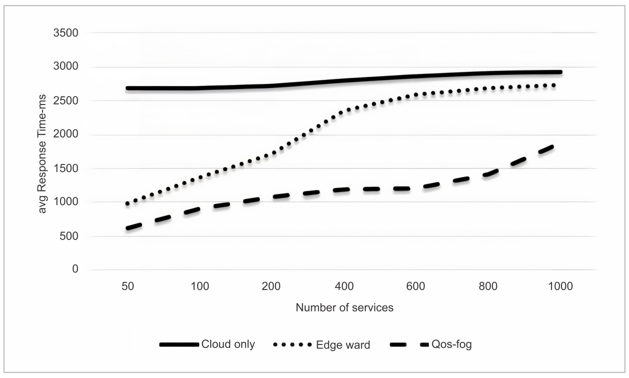

Figure 6. Average response time as a function of increasing the number of services

Increasing the number of fog nodes has a positive effect on the average response time (a noticeable decrease), when using placement algorithms that rely on fog as part of its architecture, i.e. QoS-fog and Edge-ward, with our algorithm superior in terms of performance, and this is what Figure 7 shows. While the number of fog nodes is not a parameter that has an impact on the cloud placement algorithm, as the processing takes place in data centers. The reason why the proposal achieves a shorter time is due to its exploitation of its hierarchical architecture, and the attempt to implement high-priority (time-sensitive) services in a fog subsystem by performing migrations of time-tolerant second-level services to the cloud (if necessary) in order to maintain the QoS represented by response time, which meets deadlines for those services.

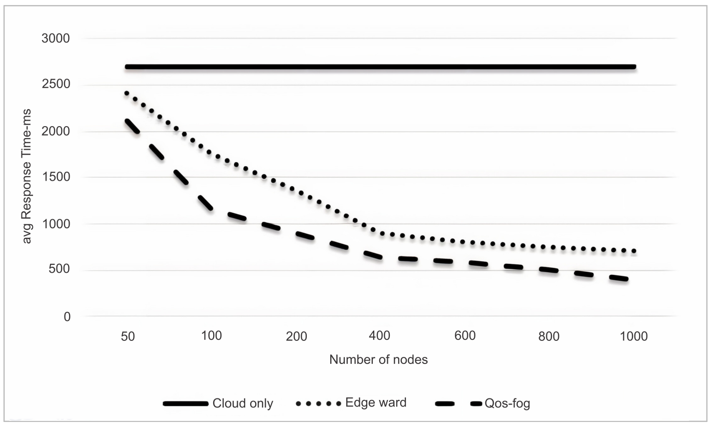

Figure 7. Average response time as a function of increasing the number of fog nodes

4. CONCLUSION

The proposed algorithm provided better performance than the edge-ward and cloud-only strategies, as we note its ability to achieve efficiency in system energy use under the influence of several parameters: the number of services, the number of fog nodes, and the percentage of high-priority services required by the IoT user, as a result of node feedback. The fog node is sent its load to the resource manager, which is used to manage power usage. In addition, the proposed algorithm was able to reduce the average response time for services, which means its ability to raise the level of service provided. In the future work, we will seek to evaluate the performance of the QoS-fog algorithm with other algorithms, dealing with different service priorities.

REFERENCES

[1] F. Alenizi and O. Rana, “Minimising delay and energy in online dynamic fog systems,” *Computer Science & Information Technology*, vol. 14, pp. 139-158, 2020.

[2] H. R. Arkian, A. Diyanat, and A. Pourkhalili, “MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications,” *Journal of Network and Computer Applications*, vol. 82, pp. 152–165, 2017.

[3] S. Pallewatta, V. Kostakos, and R. Buyya, “Placement of microservices-based IoT applications in fog computing: a taxonomy and future directions,” *ACM Computing Surveys*, vol. 55, no. 14, pp. 1–43, 2023, doi: 10.1145/3592598.

[4] R. Mahmud, R. Kotagiri, and R. Buyya, “Fog computing: a taxonomy, survey, and future directions,” in *Internet of Everything*, Singapore: Springer, 2018, pp. 1-15, doi: 10.1007/978-981-10-5861-5_5

[5] P. Hu, S. Dhelim, H. Ning, and T. Qiu, “Survey on fog computing: Architecture, key technologies, applications and open issues,” *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, 2017.

[6] C. Canali, C. Gazzotti, R. Lancellotti, and F. Schena, “Placement of IoT microservices in fog computing systems: A comparison of heuristics,” *Algorithms*, vol. 16, no. 9, 2023, doi: 10.3390/a16090441.

[7] R. Vilalta *et al.*, “Improving security in internet of things with software defined networking,” in *2016 IEEE Global Communications Conference (GLOBECOM), USA, 2016*, pp. 1-6, doi: 10.1109/GLOCOM.2016.7841889.

[8] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13-16, doi: 10.1145/2342509.2342513.

[9] Q. Tang, R. Xie, F. R. Yu, T. Huang, and Y. Liu, “Decentralized computation offloading in IoT fog computing system with energy harvesting: A Dec-POMDP approach,” in *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4898-4911, June 2020, doi: 10.1109/JIOT.2020.2971323.

[10] Q. Zhu, B. Si, F. Yang, and Y. Ma, “Task offloading decision in fog computing system,” *in China Communications*, vol. 14, no. 11, pp. 59-68, 2017, doi: 10.1109/CC.2017.8233651.

[11] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, “iFogSim: A toolkit for modeling and simulation of resource management

techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275-1296, 2017, doi: 10.1002/spe.2509.

[12] Y. Xiao and M. Krunz, "QoE and power efficiency tradeoff for fog computing networks with fog node cooperation," *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, USA, 2017, pp. 1-9, doi: 10.1109/INFOCOM.2017.8057196.

[13] B. Jamil *et al.*, "A job scheduling algorithm for delay and performance optimization in fog computing," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 7, 2019, doi: 10.1002/cpe.5581.

[14] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," in *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587-597, 2018, doi: 10.1109/JSAC.2018.2815360.

[15] M. Al-Khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, "Improving fog computing performance via Fog-2-Fog collaboration " *Future Generation Computer Systems*, vol. 100, pp. 266-280, 2019, doi: 10.1016/j.future.2019.05.015.

[16] X. Gao, X. Huang, S. Bian, Z. Shao, and Y. Yang, "PORA: Predictive offloading and resource allocation in dynamic fog computing systems," in *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 72-87, 2020, doi: 10.1109/JIOT.2019.2945066.

[17] H. H. O. Hassan, S. Azizi, and M. Shojafar, "Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments," *IET Communications*, vol. 14, pp. 2117-2129, 2020, doi: 10.1049/iet-com.2020.0007.

[18] F. Alenizi and O. Rana, "Dynamically controlling offloading thresholds in fog systems," *Sensors*, vol. 21, no. 7, 2021, doi: 10.3390/s21072512.

[19] P. Kayal, "Autonomic IoT application placement in edge/fog computing," *Journal of Mathematical Techniques and Computational Mathematics*, vol. 3, no. 1, pp. 1-8, 2024.

[20] A. Yousefpour *et al.*, "FOGPLAN: A lightweight QoS-Aware dynamic fog service provisioning framework," in *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5080-5096, 2019, doi: 10.1109/JIOT.2019.2896311.

[21] M. A. H. Monil, R. Qasim, and R. M. Rahman, "Energy-aware VM consolidation approach using combination of heuristics and migration control," *Ninth International Conference on Digital Information Management (ICDIM 2014)*, Phitsanulok, 2014, pp. 74-79, doi: 10.1109/ICDIM.2014.6991413.

[22] A. Mosa and N. W. Paton, "Optimizing virtual machine placement for energy and SLA in clouds using utility functions," *Journal of Cloud Computing*, vol. 5, 2016, doi: 10.1186/s13677-016-0067-7.

[23] M. A. H. Monil and R. M. Rahman, "Implementation of modified overload detection technique with VM selection strategies based on heuristics and migration control," *2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, USA, 2015, pp. 223-227, doi: 10.1109/ICIS.2015.7166597.

[24] M. Monil and R. M. Rahman, "VM consolidation approach based on heuristics, fuzzy logic, and migration control," *Journal of Cloud Computing*, vol. 5, 2016, doi: 10.1186/s13677-016-0059-7.

[25] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal Supercomputing*, vol. 60, pp. 268–280, 2012, doi: 10.1007/s11227-010-0421-3.

[26] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755-768, 2012, doi: 10.1016/j.future.2011.04.017.

[27] C. Puliafito *et al.*, "MobFogSim: Simulation of mobility and migration for fog computing," *Simulation Modelling Practice and Theory*, vol. 101, 2020, doi: 10.1016/j.simpat.2019.102062.

[28] M. M. Lopes, W. A. Higashino, M. A. M. Capretz, and L. F. Bittencourt, "MyiFogSim: A simulator for virtual machine migration in fog computing," in *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing (UCC '17 Companion)*, New York, USA, 2017, pp. 47-52, doi: 10.1145/3147234.3148101.

[29] D. Gonçalves, C. Puliafito, E. Mingozzi, O. Rana, L. Bittencourt, and E. Madeira, "Dynamic network slicing in fog computing for mobile users in MobFogSim," in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, UK, 2020, pp. 237-246, doi: 10.1109/UCC48980.2020.00042.

BIOGRAPHIES OF AUTHORS

Ali Ahmed Razzaq had a master's degree in computer network engineering from Andhra University. Now he is a research Scholar at Andhra University in the IoT specialty for the purpose of obtaining a Ph.D. He has several skills in the field of artificial intelligence and its programming in the field of networks and design websites by framework django in python. He currently work in the field of air navigation and its systems as a data entry for aviation transit (FDO) in the Iraqi Air Traffic Management Center in the Area Control Center (ACC). Now an air traffic controller at Baghdad International Airport (Iraq). He can be contacted at email: taifali607@gmail.com.

Prof. Kunjam Nageswara Rao is a Professor in Department of Computer Science & Systems Engineering at Andhra University College of Engineering. He has more than 24 years of teaching experience. He has published 3 patents and more than 50 research papers so far in various highly reputed international journals. His research interest includes - cloud computing, wireless networks, sensor networks, IoT, bioinformatics, medical image processing, network security, data mining, and data analyticss. He can be contacted at email: kunjamnag@gmail.com.