A Dual Ring Architecture Using Controllers for Better Load Balancing in a Fog Computing Environment

Birane Koundoul[1(✉)], Youssou Kasse[1], Fatoumata Balde[1], and Bamba Gueye[2]

[1] University of Bambey, Bambey, Senegal
{birane.koundoul,youssou.kasse,fatoumata.balde}@uadb.edu.sn
[2] University of Dakar, Dakar, Senegal
bamba.gueye@ucad.edu.sn

Abstract. Fog Computing is a paradigm that extends cloud computing by bringing network and cloud resources closer to the edge. This means that points of presence are placed close to end users for easy access and to enable delay-sensitive applications to have minimal response times. This Fog layer preprocesses data as close to the sensors as possible. However, with the increasing demand of IoT, even when close to sensors, fog nodes tend to be overloaded, compromising the response times of latency-restricted IoT (Internet of Things) applications, and therefore also compromising the quality of user experience. However, the limited storage and processing capacity leads us to ask the question: wouldn't load balancing be an asset in the Fog Computing environment to avoid overwhelming some fog nodes? It is in this sense that we will exploit load balancing in the Fog Computing environment. We will propose an algorithm for selecting the fog node with the best resource (BRFC) to handle the request. A controller node is placed at each zone to manage data access and placement. This will prevent some nodes from being overloaded or underloaded in order to improve response time, system performance, throughput, cost and even energy consumption.

Keywords: Fog computing · Load balancing · IoT · Quality of service · System performance

1 Introduction

Fog Computing is a geographically distributed paradigm, brings network power and computing to the edge of the network, closer to end users and IoT devices, as it is supported by widespread fog nodes [1]. This technology consists of a set of nodes that are heterogeneous in terms of storage, processing resources. However, these resources are very limited compared to the cloud. Cloud Computing with its almost unlimited storage and processing capacity requires load balancing of physical and virtual servers to better manage system performance. Since

A. D. Mambo et al. (Eds.): InterSol 2022, LNICST 449, pp. 144–154, 2022.
https://doi.org/10.1007/978-3-031-23116-2_11

the emergence of Fog Computing, the goal is to solve some of the problems that Cloud Computing faces. The latter with its almost unlimited processing and storage capacity can handle any type of task with any size. However, with the advent of the Internet of Things, some time-sensitive applications require a minimum response time. This is a hindrance for the traditional cloud. Fog computing was not born to replace cloud computing.

Most of the data requiring analysis, processing and storage is transmitted to the cloud. This can have a negative influence on latency, security, mobility, and reliability because with the existence of delay-sensitive applications, the reduction of response time is necessary. In [2,3], the authors showed that the proximity of the Fog layer to Internet of Things (IoT) devices can remarkably reduce latency and meet the requirements of extremely low latency.

However, as the volume of data sent by these connected objects continues to increase, this results in a reduction in system performance. As a result, load balancing in the Fog Computing environment is a necessity for a better quality of service.

In this paper, we will propose a solution somewhat similar to [4], but we will place a controller at each zone, responsible for managing the load balancing in each zone. This prevents underloading and overloading of some nodes. This controller node stores all the objects of the nodes in the zone.

The rest of the paper is structured as: in Sect 2 presents a summary of related work in the literature. In Sect. 3, we will discuss our model to better understand the load balancing in each zone managed by the controller node. In Sect. 4, we will represent our model as a graph to better understand the role of the controller node. In Sect. 5, we will show the experimental results obtained compared with those of Mostafa et al. Finally, we end the paper in Sect. 6 with a conclusion and propose some directions for future work.

2 Related Work

With the era of big data, the amount of data sent continues to increase exponentially. This is why researchers have tried to provide solutions by proposing algorithms, architectures, etc. to make systems more efficient. It is in this sense that Masip-Bruin et al. have proposed a new Fog to Cloud (F2C) architecture [5], to facilitate the communication between the Fog layer and the cloud layer. They also proposed a management system that allows them to discover the set of fog nodes and choose the best node among the others. This solution, the task can be divided into individual function and in turn also divided according to the fog layers. In addition, this solution, the authors did not explain how the task will be divided into individual functions or even specify the criteria for selecting the most optimal fog node to process the task.

In [4], Mostafa et al. proposed a solution for load balancing in Fog Computing. In their solution, they proposed an algorithm for fog resource selection (FResS) and also enables automatic fog selection and allocation for IoT systems. It is a solution that predicts the execution time using logs. A new module

is placed between the Fog layer and the IoT devices. This module consists of a task scheduler, a task manager, a resource selector and a history analyzer. These modules perform various tasks related to resource selection, predictions and history management. In this new module, if a task arrives, they check the logs if the task has been once executed to predict the execution time, waiting times in the queues and data transfer times. Otherwise the task scheduler sends the task description to the task manager to find similar tasks in order to predict the execution time and the resources needed to execute the task. This will lead to the task selector component until logging.

According to Xu et al. in [6], load balancing is one of the key factors to achieve resource efficiency and avoid bottlenecks, overloading or underloading of nodes and low load. In this paper, Xu et al. . proposed a corresponding resource allocation method in a fog environment through static resource allocation and dynamic service migration to achieve load balancing for Fog Computing systems. In this paper, the authors did not consider the negative impact of service migration, including traffic for different types of compute nodes, cost of service migration, performance degradation for service migration, and data transmission cost.

The paper [7], banerjee et al. proposed a distributed resource allocation protocol algorithm to achieve load balancing in a large-scale distributed network. With their algorithm, the response time and resource utilization could be significantly improved over FIFO. However, some preferred to use graph partitioning theory to balance the loads dynamically. This allows us to study the problem of trade-off between energy consumption and delay in F2C scenarios [8,9].

In [10], Souza et al. modeled the service allocation problem as a multidimensional knapsack problem (MKP) aiming at optimal service allocation considering delay, load balancing and energy consumption. It is an algorithm to solve the combinatorial optimization problem. Load balancing is an essential mechanism in the cloud as well as in Fog Computing. Load balancing helps to make networks more efficient. It distributes processing and traffic evenly across a network, ensuring that no device is overwhelmed. Several research works are interested in load balancing to make the system more efficient in terms of storage, processing, network traffic.

Mostafa et al. in [4] have addressed this same load balancing topic at the Fog Computing level. They introduced a new module to interact the fog layer and IoT devices. The role of this module is to predict the execution time of a task in order to reduce latency and manage balancing dynamically in the fog. The module consists of a task scheduler in charge of retrieving the task before sending the information about the task itself to the selector. However, several limitations are to be noted such as: 1) the fault tolerance is not fully ensured with the interconnection of zones. Because if a link between two zones fails, communication between some zones will be possible via the cloud. 2) In addition the bus topology applied in its architecture. This means that the performance decreases according to the number of zones and also if the central link fails, all its network will fail. 3) The location of FResS is not specified in relation to

the different zones, which means that it can be a bit far from some zones. This can lead to a long response time. 4) A lack of precision of the duration of the archived data.

Although work is progressing to solve load balancing in cloud computing or fog computing, our proposal allows to manage the load of tasks in the fog to avoid some nodes underloaded or overloaded. Our method is somewhat similar to that of [4], but we will place in each zone a controller node. The controller will schedule, select, and even store objects from the nodes storing the tasks. We will detail in Sect. 3 the placement and access of information according to the controller node.

3 Description of Our Approach with a Controller Node for Load Balancing

In our architecture [11], we proposed a model with three layers. At the Fog layer, interconnected zones in double ring mode are applied Fig. 1. This allows to manage fault tolerance. The controller node manages the placement and reading of data. If a new task arrives, the controller node consults its table to find the location of the task. However, if the task is not present in its zone, with inter-zone communication (A2A), the controller node transfers the task to a neighboring zone. This transfer does not happen randomly because the controller nodes communicate stored objects with each other via messages and an update message is sent after each new task stored or task location changed from one node to another.

In addition, we have represented our model as a graph with the controller node as its vertex. The controller node is connected to all other nodes either directly or indirectly to manage load balancing. All user tasks are redirected to the controller node, knowing the location of all objects stored by the nodes. However, it is still challenging to achieve load balancing for compute nodes in the fog environment when running IoT applications. Considering this challenge, a dynamic resource allocation method managed by a controller node for load balancing in a fog environment is proposed in this paper.

The controller node has some information about the other nodes, such as storage capacity, free memory space, CPU capacity, and RAM capacity. This allows it to select the best node for storing the task. With each new task that arrives, this controller node checks its table (DB) to determine the node with the necessary resources capable of storing the task. In addition to this, the controller node has the ability to transfer data from some nodes to other nodes to handle load balancing. This load balancing makes the system more efficient. According to Xu et al. in [11], due to the diversity of runtime and specifications of compute nodes in the fog, the resources could not be fully utilized. This makes it important to migrate data between nodes.

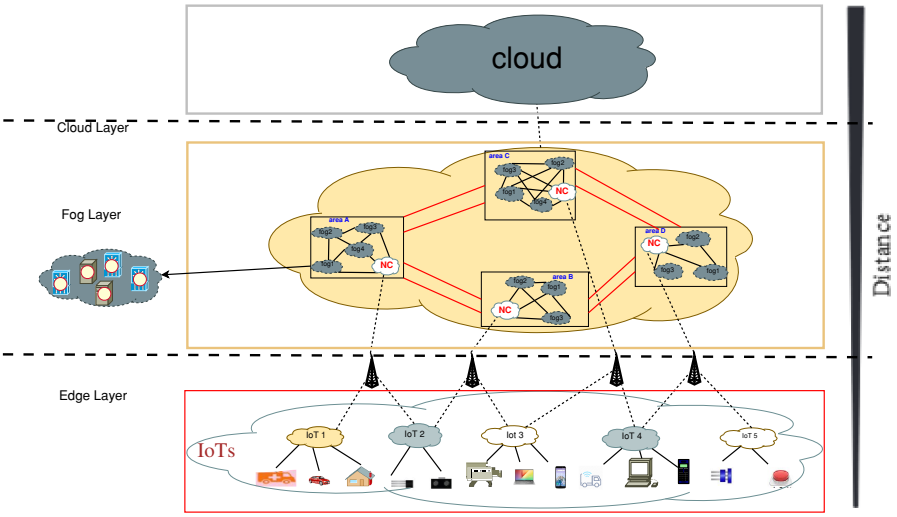

Fig. 1. Architecture for load balancing with the controller node.

3.1 Overview of the Load Balancing Method

Our method consists of four main steps, namely checking the table (database - DB) to select the node with more available resources (step 2) in order to transfer the task in this detected node (step 3). In this method, step 1 is to first check the DB containing all the information of the different nodes. To judge if a node is able to store the new task, it is necessary to detect the available space of all the compute nodes (see Fig. 2 of the sequence diagram). The returned information will allow the controller node to redirect the task to the selected node. Two or more nodes can be returned and in this case the controller node selects the best node among the others according to the storage capacity, the bandwidth but also the number of hops. Another step (step 4) is used for data migration between nodes. The controller node has the ability to move data from one node to another. In addition, some workloads from compute nodes with higher resource utilization are migrated to compute nodes with low resource utilization. Figure 3 illustrates the migration of data between the different areas performed by the controller node. Each area is connected to two other areas that they communicate through the controller nodes. Each task received by the controller node, a check will be made at the level of its table to select the best zone to transfer the task. After receiving and processing the task, a message is sent to the original controller node to update its table.

3.2 Determination of the Shortest Path

In this section, we will explain the method used at the controller node to determine the shortest path. All nodes are connected to the controller node. The

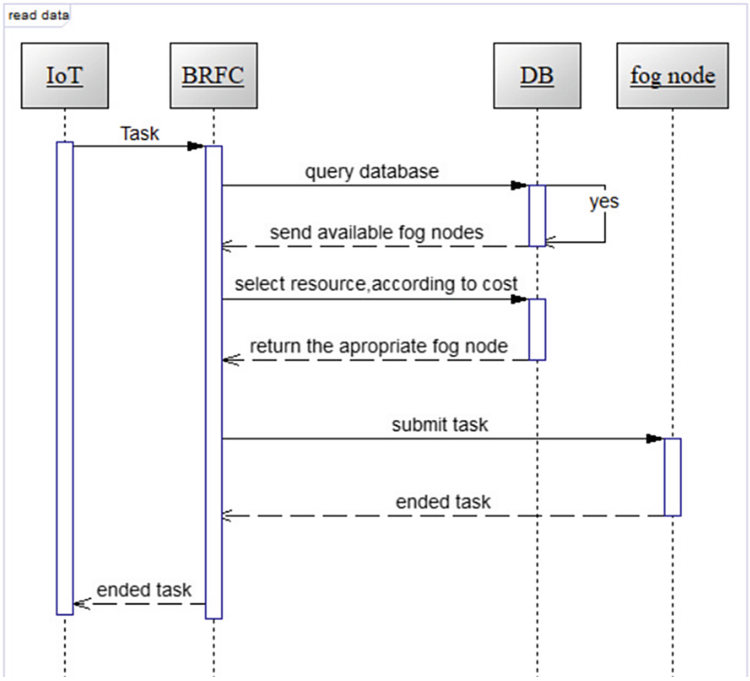

Fig. 2. Diagram for selecting the best fog to handle a task.

shortest path problem is one of the most classical problems in graph theory and the most important in applications especially for optimization problems. If we consider an area as a directed graph G (X, U, L), we associate to each arc u(i, j) a real number l(u) where lij, called the length or weight of the arc. This weight of the arc between two vertices s (source vertex or origin vertex) and d (destination vertex) of the graph allows to determine all the paths from s to d. A path is noted u* whose total length l(u*) is minimal.

Node a is considered the controller node. It is connected to all nodes directly or indirectly. The greedy algorithm is applied, which seeks to build a solution step by step by respecting certain conditions:

- never go back on a decision
- taking at each step the solution that seems to be the best locally
- hoping also to obtain an optimal solution.

Many computer techniques are likely to provide an exact or approximate solution to problems such as determining the minimum and maximum with respect to a function, determining the shortest path between two points, giving change from a vending machine, etc. The glutton algorithm is a possible method for solving these types of problems. It is an algorithm that allows for fast computation time and easy implementation but does not guarantee optimisation.

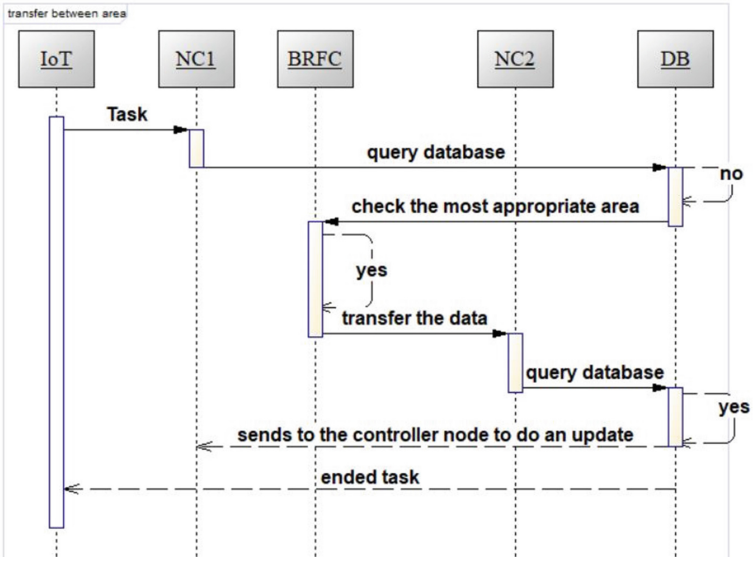

Fig. 3. Sequence diagram for data transfer between areas.

Let e_i be the set of nodes, nb_j = the weight of the arc and **s** the distance between two vertices (nodes). The algorithm will sort the arc weights in descending order, thus maximising the number of arc weights chosen.

4 Results and Experiments

After simulation with the GridSim tool, we have a better result compared to the proposal of Mostafa et al. in [4]. We used the same parameters to compare our two algorithms. One cloud storage site, three fog storage sites, and 150 IoT devices were defined by the simulator, 500 to 2000 task requests, and varying bandwidth connectivity from 500 MB/S to 2000 MB/S.

Depending on the number of tasks, the result in Fig. 6 shows that our algorithm is better in terms of response time. We see that our algorithm tries to minimize the request agreement time by redirecting some tasks to the neighboring zones. The choice of the zone always depends on the task because the controller nodes communicate the information they store with each other. Each time data is stored, the controller node records the location of the object and sends an update to the controller nodes. This facilitates the redirection of the request in case it is not present in the host area. Table 1 shows the set of cloud computing parameters and fog layer areas and Table 2 shows the characteristics of the tasks.

Table 1. Characteristics of the fog and cloud zones.

Settings	Fog	Cloud
Number of area	3	1
Number of nodes per area	6	12
Latency	[50–100] between area	≈200
CPU MIPS	[500–2000]	[3000–10000]
Bandwidth	500 Mo/s	2000 Mo/s

Table 2. Characteristics of the tasks

Properties	Values
Storage capacity (MB)	[1–10]
Bandwidth (MB/s)	[0,5–1]

The latency between fog nodes in a zone is 10 ms.
The results presented in Fig. 4 show that the adopted BRFC model outperforms the FResS model, so that the task response time was reduced by 19.10%. In Fig. 5, we have the cost according to the number of users. The *BRFC* model is even better. Of course, as the number of users increases, the cost also increases. However, with our model, we were able to reduce the cost by 28.18% compared to the FResS model.

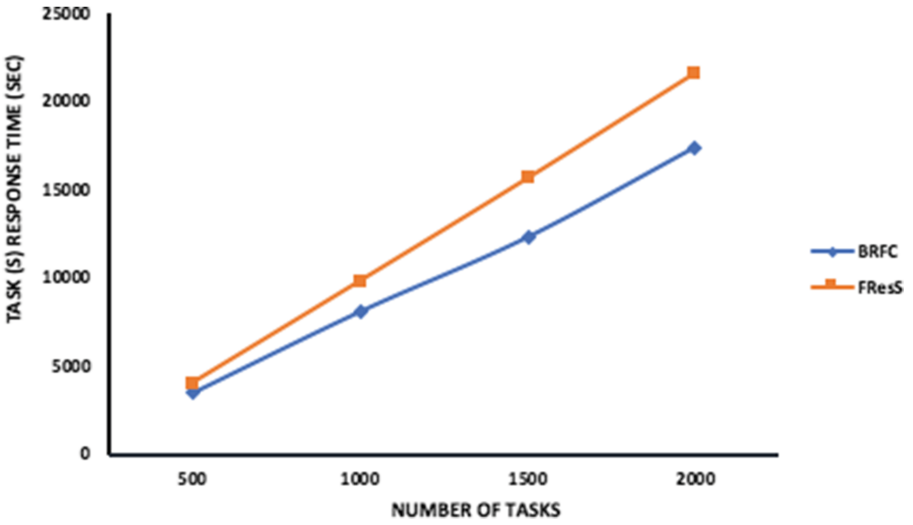

Fig. 4. Response time as a function of the number of tasks.

Finally in Fig. 6, we used the bandwidth to compare the two models. The bandwidth is a very important factor that influences the response time according to the number of users but also according to the number of tasks. The higher the bandwidth is, the more the execution time decreases. The results obtained in Fig. 6 show that we could reduce the total execution time of the tasks by 35.05% compared to the FResS model.

Table 3. Task turnaround time of our BRFC approach compared to the FResS approach.

Number of tasks	500	1000	1500	2000	Total
BRFC	3450	8070	12300	17400	41220
FResS	4000	9780	15600	21570	50950
Between	550	1710	3300	4170	9730

4.1 The Limits of My Model

After simulation with the GridSim tool, we found that our model is better compared to that of Mostafa et al. in [4]. However, some limitations are to be noted in our model such as the trade-off between the storage capacity and the number of objects to be stored. As the number of objects becomes larger, the capacity also increases. In addition, with only one controller node, if the latter fails, inter-zone communication will no longer be possible. This is why we considered

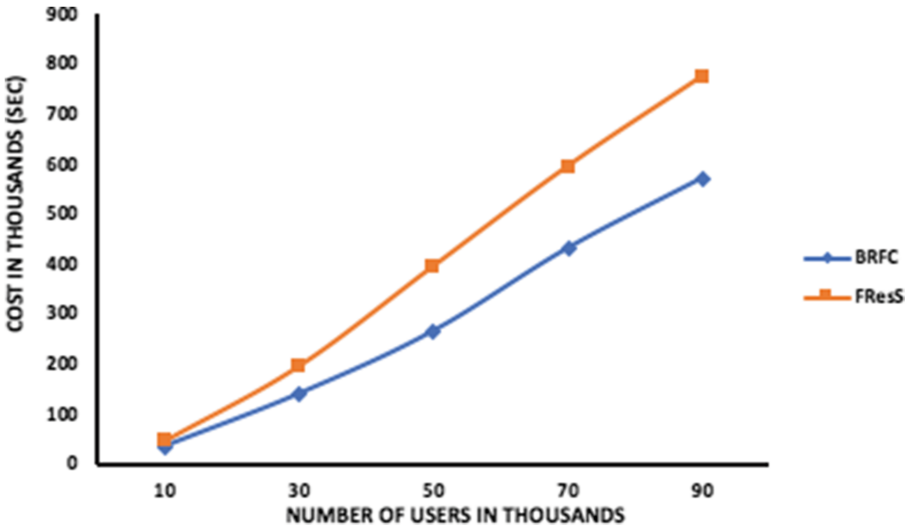

Fig. 5. The cost according to the number of users.

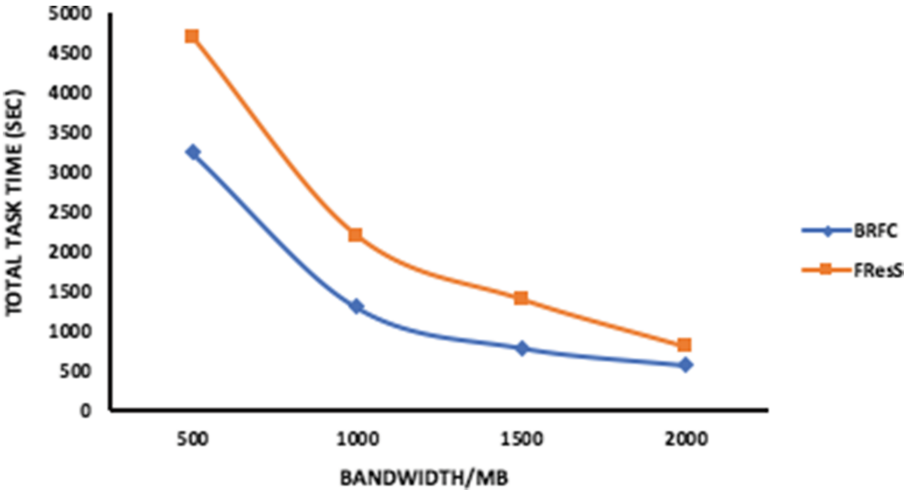

Fig. 6. Total working time depending on the variant bandwidth.

proposing a new model with two controller nodes (a primary node and a secondary node). The data will be redirected to the secondary node if the queue threshold has been reached. In addition to handle the optimization problems with the glutton algorithm, we will use a particular structure named matroid to design an optimal glutton algorithm.

5 Conclusion

Fog computing is an infrastructure that has emerged to solve some of the problems encountered by cloud computing. With the birth of the Internet of Things, load balancing is necessary to avoid overwhelming some fog nodes. In this article we have proposed a model with a controller node in each zone. The role of this controller node is to manage the placement and location of data. Unlike with the model of Mostafa et al. in [4] where a module is placed between the fog environment and the IoT devices to predict task execution times. However, after simulation with GridSim tool, we found our BRFC algorithm better compared to FResS. As the results show. We have reduced 19.10% of the task response time, 28.18% of the cost and 35.05% of the total task execution time. This means that we have a better model compared to FResS.

In addition, with the agreement times increasing with the number of requests. In future work, we plan to improve our model by adding two controller nodes (a primary controller node and a secondary controller node). The new secondary controller node will process new tasks if the threshold set for the agreement queue is reached at the primary controller node. In addition, we will also improve our glutton algorithm to handle the optimization. Finally, we also plan in future work to take into account the number of fog nodes at each zone in an incremental way.

References

1. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of thing. In: Proceedings of the first Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, août 2012, pp. 13–16 (2012). https://doi.org/10.1145/2342509.2342513
2. Yousefpour, A., et al.: All one needs to know about fog computing and related edge computing paradigms: a complete survey. arxiv180805283vr (2019) https://doi.org/10.1016/j.sysarc.2019.02.009
3. Auluck, N., Azim, A., Fizza, K.: Improving the schedulability of real-time tasks using fog computing. IEEE Trans. Serv. Comput. **15**, 372 –385 (2019). https://doi.org/10.1109/TSC.2019.2944360
4. Mostafa, N.: Cooperative fog communications using a multi-level load balancing. In: 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC), pp. 45–51, June 2019. https://doi.org/10.1109/FMEC.2019.8795325
5. Masip-Bruin, X., Marín-Tordera, E., Tashakor, G., Jukan, A., Ren, G.-J.: Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems. IEEE Wirel. Commun. **23**(5), 120–128 (2016). https://doi.org/10.1109/MWC.2016.7721750
6. Xu, X., et al.: Dynamic resource allocation for load balancing in fog environment. Wirel. Commun. Mob. Comput. **2018**, e6421607 (2018). https://doi.org/10.1155/2018/6421607
7. Banerjee, S., Hecker, J.P.: A multi-agent system approach to load-balancing and resource allocation for distributed computing. In: Parrend, P., Bourgine, P., Collet, P. (eds.) First Complex Systems Digital Campus World E-Conference 2015. SPC, pp. 41–54. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-45901-1_4
8. Ningning, S., Chao, G., Xingshuo, A., Qiang, Z.: Fog computing dynamic load balancing mechanism based on graph repartitioning. China Commun. **13**(3), 156–164 (2016). https://doi.org/10.1109/CC.2016.7445510
9. Deng, R., Lu, R., Lai, C., Luan, T.H., Liang, H.: Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet Things J. **3**(6) , 1171–1181 (2016). https://doi.org/10.1109/JIOT.2016.2565516
10. Souza, V.B., Masip-Bruin, X., Marin-Tordera, E., Ramirez, W., Sanchez, S.: Towards distributed service allocation in fog-to-Cloud (F2C) scenarios. In: 2016 IEEE Global Communications Conference (GLOBECOM), déc. 2016, pp. 1–6. https://doi.org/10.1109/GLOCOM.2016.7842341
11. Koundoul, B., Kasse, Y., Balde, F., Gueye, B.: Leveraging cloud inter-zone architecture for response time reduction. In: Faye, Y., Gueye, A., Gueye, B., Diongue, D., Nguer, E.H.M., Ba, M. (eds.) CNRIA 2021. LNICST, vol. 400, pp. 87–97. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90556-9_8
12. Xu, X., et al.: Dynamic resource allocation for load balancing in fog environment . Wirel. Commun. Mob. Comput. **2018**, e6421607 (2018). https://doi.org/10.1155/2018/6421607