Efficient Data Replication Management in Fog Computing

BENAMAR Abed Zineeddine
Computer Science Department
University of Sciences and technologies Mohamed Boudiaf
Oran, Algeria
abedzineeddine.benamar@univ-usto.dz

REGUIEG Hicham
Computer Science Department
University of Sciences and technologies Mohamed Boudiaf
Oran , Algeria
hicham.reguieg@univ-usto.dz

2024 International Conference of the African Federation of Operational Research Societies (AFROS) | 979-8-3503-8644-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/AFROS62115.2024.11037083

***Abstract*— IoT devices are increasing in number by each year. These devices are generating a massive amount of data which needs to be processed and stored. For this purpose, cloud computing is used to provide a solution to this problem. Unfortunately, when it comes to time-sensitive applications that require a very short time of response, cloud computing can't meet this requirement. To bring computational power and storage closer to IoT end devices, Fog computing has been introduced in order to improve the systems overall performance and decrease latency.**

Processing and storing data in IoT mobile environments can be challenging because of the nature of mobile devices, when a node disconnects or powers off, some data can be lost. This work focuses on studying how manage fault tolerance in fog environments and how to ensure data consistency through all fog nodes using data replication mechanisms.

For that, we proposed a mobility-aware approach based on Density-Based Spatial Clustering of Applications with Noise (DBSCAN) with a cluster head selection, to replicate new data and achieve eventual consistency in replicas through mobile fog nodes. Experimental results show that our proposed approach was able to manage replica with efficiency, ensuring reliable and optimized performance across various scenarios.

Keywords*— *Data consistency, fault tolerance, fog computing, replication, clustering.

I. INTRODUCTION

In recent years, IoT devices have exponentially grown in number because of modern technologies (wearable devices, smart watches, mobile phones, wireless cameras, IoT sensors, smart vehicles, etc.). A global IoT forecast report published by Transforma Insights [1] showed that IoT devices growth is increasing by 10% each year. According to Fig.1 the number of IoT devices estimated by 2025 will be 19 billion devices.

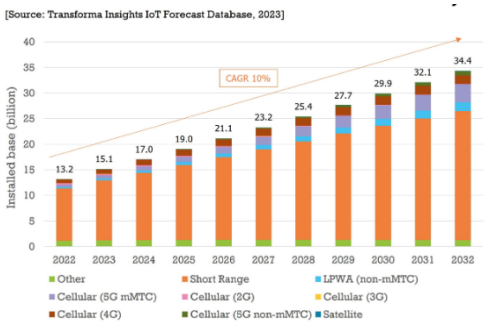


Fig.1. Global IoT forecast report by Transforma Insights [1]

This massive number of IoT devices has been generating a tremendous amount of data referred to as Big data. Big data is stored and processed by central servers located in distant sites called the cloud.

While cloud computing is very resourceful in terms of storage capacities and processing power, it has some disadvantages when it comes to time-sensitive applications such as real-time streaming or critical decision-making systems. The distance between the end users can lead to response delays and increased latency which may affect the system's overall performance. Time delays in milliseconds can have a significant impact such as in telemedicine applications or vehicular communications [2].

To cope with the aforementioned limitations of conventional cloud computing, especially for large-scale IoT applications, fog computing as a novel approach in this context has emerged. Cisco introduced fog computing in 2012, and there are a lot of definitions of fog computing. Yi et al. defines it as "Fog Computing is a geographically distributed computing architecture with a resource pool which consists of one or more ubiquitously connected heterogeneous devices (including edge devices) at the edge of network and not exclusively seamlessly backed by Cloud services, to collaboratively provide elastic computation, storage and communication (and many other new services and tasks) in isolated environments to a large scale of clients in proximity" [3].

Fog computing in the realm of IoT aims to enhance efficiency, boost performance, and minimize the volume of data transmitted to the cloud for processing, analysis, and storage. Consequently, the information gathered by sensors will be directed to edge devices within the network for temporary storage and processing, as opposed to being immediately dispatched to the cloud. This approach effectively trims down network traffic and latency.

Using fog computing in distributed mobile networks can be challenging, one of the challenges is to choose which fog nodes can be used for storage or processing data depending on various criteria:

- Which fog nodes are the best performing nodes in terms of processing power and storage capacity?
- Which fog nodes are more likely to stay connected for the longest period?
- Which fog nodes are closer to end-devices to minimize latency?

Due to the unpredictable nature of ad-hoc mobile networks, fog nodes can connect and disconnect frequently, a fog node

Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on August 11,2026 at 09:49:01 UTC from IEEE Xplore. Restrictions apply.
979-8-3503-8644-8/24/$31.00 ©2024 IEEE

disconnection during processing data can lead to data loss which may affect the data consistency in the system.
To prevent data loss in such systems, data replication techniques are used. Since data localization is crucial for processing, it's important to explore how data should be positioned within such an infrastructure to minimize latencies in the system [5]. In this article, we will propose a new approach for managing consistency of replicas in fog computing.
The problem addressed in this paper is where to replicate crucial and sensitive data and which data has more priority to be replicated in order to manage data consistency in a fog based IoT health application that has different consistency levels for different data consumers while taking in consideration mobility pattern and short response time.

The article is organized as follow: Section 2 addresses fog computing challenges. Section 3 presents the most import related works. In Section 4, our proposed approach is explained in details. Section 5 discuss the obtained results. Finally, Section 6 concludes this paper and presents the perspectives.

II. FOG COMPUTING SOLVING IOT CHALLENGES AND EXAMPLES

Fog computing provides new solutions to many IoT challenges such as latency constraints, network bandwidth, privacy and security…etc.
There are many significant areas where fog computing can play a vital role in different IoT applications. In what follow, we will provide an overview of various IoT applications that can benefit from fog computing.

- Smart Transportation System: Fog computing can be combined with Vehicular Ad-hoc Networks (VANETs) to enhance communication between vehicles, the idea is that road traffic lights send their current stat to smart vehicles in order that users wait a minimum of time and arrive to destination shortly. [6].
  Healthcare 4.0: using Fog computing in healthcare IoT application is beneficial, since it can reduce response time and monitor individuals in real-time more accurately than using the cloud, it can also help with the cost factor of storage. In aspect of security issues, fog computing can reduce sensitive information breaches. In Figure 3 we present an example of an IoT fog-based healthcare architecture [8].

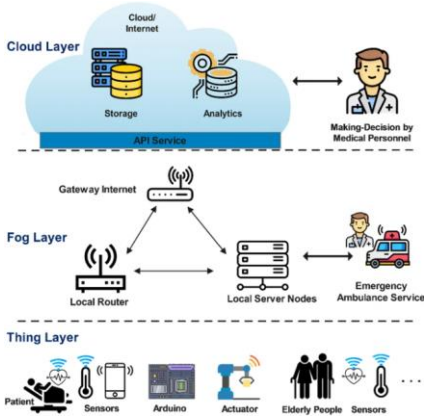


Fig.3. IoT fog-based healthcare architecture [8]

- Augmented and Virtual Reality: Augmented reality applications are extremely time-sensitive, a small delay can lead to serious errors in user experience. Thus, Fog computing is a promising solution in this domain. Fog can also augment processing power of user devices like smartphones [7].

III. RELATED WORKS

Many researchers have discussed data replication techniques in fog computing, each study has focused on one of these aspects: (i) data placement, (ii) data replication or (iii) data consistency to improve latency, availability and inconsistency in the fog computing environment. In what follow, we will present the most relevant works.
Torabi et al. [9] have classified data replication strategies into four categories: framework-based, graph-based, heuristic-based and metaheuristic-based.
Pfandzelter et al. [10] designed FReD, a prototype for a replication middleware that distributes data across different sites. The replication is based on a dynamic mapping of keygroups, it also uses version vectors to ensure eventual consistency and to solve conflicts among data replicas and also to keep track of each keygroup version, this middleware can be integrated with applications from different domains.
George et al. [11] addressed the problem of distributed computing in mobile clouds, they developed a k-out-of-n framework that is based on the Hadoop MapReduce framework, this framework splits the processing job into smaller tasks and runs these tasks on different nodes in parallel, this concept of parallelism can reduce the execution time when compared with a sequential execution on a single node.
Naas et al. [12] The authors address data replication issues in fog environments, highlighting the challenges of multiple copies, storage capacity, energy consumption, and network traffic. They propose two heuristics, iFogStorS and iFogStorP, to manage data replication and consistency while reducing access and replicas synchronization latencies, based on the number of replicas responding to a write/read request.
Saleh et al. [13] The study proposes iFogRep, an intelligent replication protocol for dynamic consistent IoT placement, using fog computing to reduce system latency, cost of replica access, and increase data availability. The protocol uses machine learning to predict optimal data copies and consistency levels, resulting in improved system performance.
The previous studies focused on placing one data replica on the proper fog nodes, which fueled another study made by **Huang et al. [14]** The authors propose a latency-aware model, iFogStorM, to support multiple replicas on fog nodes. This NP-hard problem requires a greedy algorithm, MultiCopyStorage, and heuristics to determine which consumer receives the data from. Previous studies have used different approaches to address data replication and consistency.**Table 1** shows a brief comparison between some studies.
For an efficient resource utilization in a highly dynamic environment like a fog computing environment, fog nodes need to be clustered. For clustering fog nodes, different approaches could be used, for example in [15] the authors used K-means and fuzzy logic, in [16] they used a metaheuristic approach based on Multi-Objective Particle Swarm Optimization algorithm to schedule fog devices to run IoT application effectively and achieve optimal resource allocation.
The K-means algorithm requires specifying the number of clusters (K) in advance, which may not be known or easily

Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on August 11,2026 at 09:49:01 UTC from IEEE Xplore. Restrictions apply.

determined in a mobile fog environment where there are thousands of fog nodes.

Another clustering algorithm called DBSCAN (Density-Based Spatial Clustering of Applications with Noise) developed by [17] could be used for clustering fog nodes, DBSCAN doesn't need to predefine cluster numbers, another advantage of using DBSCAN is its ability to identify clusters of different shapes and sized, and its capability to handle outliers.

DBSCAN algorithm uses two parameters:

- minPts: The minimum number of points clustered together for a region to be considered dense.
- eps (ε): A distance measure that will be used to locate the points in the neighborhood of any point.

DBSCAN algorithm is explained in these steps:

- Initialize Parameters: Set the parameters eps (maximum distance for neighborhood) and minPts (minimum number of points to form a dense region).
- Select an Unvisited Point: Choose an unvisited point in the dataset
- Neighborhood Query: Identify all points within eps distance from the selected point (this forms the epsilon-neighborhood).
- Core Point Check: If the number of points in the neighborhood is greater than or equal to minPts, the point is a core point and a new cluster is initiated.
  If not, mark the point as noise (it may later be reclassified as part of a cluster).
- Expand Cluster: From the core point, recursively add all density-reachable points (points reachable within eps from any other core point) to the current cluster.
- Repeat: Continue steps 2-6 until all points are visited.
- Finalize Clusters: Once all points are processed, finalize clusters, reclassifying any noise points that may have become part of a cluster.

IV. PROPOSED APPROACH

In order to address data consistency in a fog computing network, it is critical to find the best nodes to store data, then specify how to synchronize these replicas and maintain a high level of consistency that meets the IoT applications requirements, thus our work will follow these phases:

1- A clustering method will be used to cluster fog nodes based on their characteristics (Geographical position, Storage capacity, Processing power, Mobility pattern, Battery level).

For clustering fog nodes, we used K-means and DBSCAN algorithm to compare their results in our scenario.

2- Data producers (such as sensors) start sending their data to be replicated in fog nodes, selecting the best fog node to store the replicas is an important step, fog node can be selected by the shortest path algorithm or based on other criteria, for example if the data producer requests a real time processing and a short time response, logically the node with the best processing power is chosen, and if it requires a high level of consistency then the fog node with the highest bandwidth is selected.

TABLE 1. COMPARISON OF PREVIOUS STUDIES

Study	Consistency	Replication	Placement	Synchronization	Mobility	Algorithms	Tools	Fog nodes	Metrics
[10]	Yes	Yes	No	Yes	No	Not mentioned	Go# gRPC AWS DynamoDB	Fog nodes :10 1 ≤ concurrent clients ≤ 30	-Use case studies
[12]	Yes	Yes	Yes	No	No	-Floyd Marshall Algorithm - Median -P Median	iFogSim CPLEX-MILP	Small infrastructure :10 medium:50 large :1000	-Overall service latency - Computation time - Number of replicas -unsatisfied read requests
[13]	Yes	Yes	Yes	No	No	SVR model FP-MFIA model	iFogSim2	10 ≤ nodes ≤ 50	-average latency cost -Replica cost -availability of datums
[14]	No	Yes	Yes	No	No	Greedy algorithm	iFogSim	1000 GWs 50 LPOP 10 RPOP 5 data centers 1 ≤ clients ≤50	-overall latency

In [1] the authors proposed a mobility-aware hierarchical framework that selects mobile fog that volunteered their unused resources based on a requested minimum energy level, in second priority the node direction is checked, the node is rejected if moving away from the broker node. If these two criteria are verified then the node is added to the available resources list.

Since our focus is on healthcare fog computing applications, therefore selecting fog nodes is based on

Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on August 11,2026 at 09:49:01 UTC from IEEE Xplore. Restrictions apply.

short time response, because health applications are time sensitive. Additionally, mobility pattern of fog nodes will be used for node selection, to the best of our knowledge mobility has not been discussed in the related works.

3- Fog nodes must replicate important and sensitive data among each other. Hence clients will find requested data close to them.
Data replication mechanisms in literature are categorized into four categories: Full replication, partial replication, selective replication and dynamic replication

In a healthcare IoT fog-based application, we used a dynamic replication approach, Since DBSCAN algorithm does not assign cluster heads, we propose improving this algorithm by adding cluster head selection, each cluster head broadcast critical and most frequent accessed data to other nodes in the cluster. Therefore, data availability will be assured for emergency cases.
Figure 5 illustrates a proposed architecture for our approach.
The proposed architecture consists of three layers:

- IoT Layer: Also called the end-device layer or edge layer, this layer includes IoT devices such as sensors, controllers, and mobile devices. It collects all types of unsorted data and sends them to the upper layer.

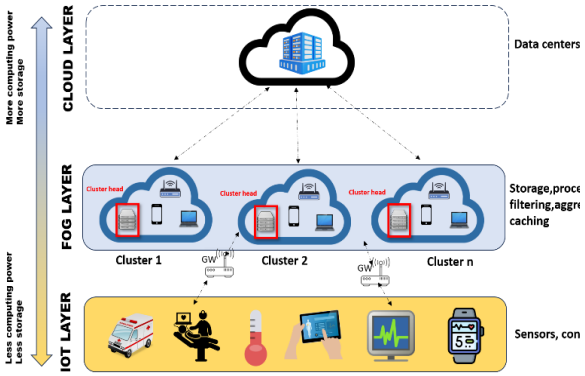


Fig.5. Proposed architecture

- Fog Layer: This layer comprises various fog nodes such as routers, servers, and high-end mobile devices. It pre-processes, filters, and aggregates data for proper usage.
- Cloud Layer: This layer uses data centers to process big data that cannot be managed by the fog layer.

4- An intelligent technique will be used to synchronize data among fog nodes, of course this operation would consume bandwidth and increase latency in the system. Hence using a classification for strong or regular consumers is suggested [9]. In our scenario strong consumers would be emergency rooms and operation rooms, because they need real-time responses. And regular consumers like daily consumers (home patients, routine doctor check-ups, etc.).
Eventual consistency will be achieved after executing this technique, Strong consumers will update their data first, then regular consumers and finally tolerant consumers. An example of tolerance priority is illustrated in Fig.6.

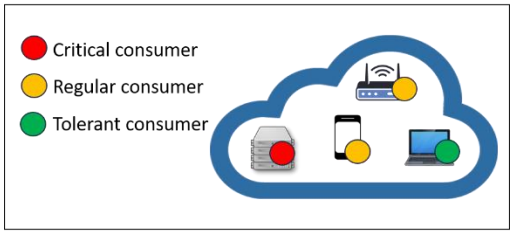


Fig.6. Data consumers classification

Finally, a time sensitive application should be put in place in order to test how fog computing could help in emergencies, For example in critical health care scenarios.

V. Experimentations

In this section we explain the implementation details of our proposed approach:

- Environment: we conducted our experiments on a machine with an Intel core i5 processor, 8GB RAM, running windows 10, the software environment included Python 3.12, Eclipse IDE, and iFogSim simulator developed by Cloudslab.
- First, we used synthetic data of 200 physical component with the following features: RAM, processing capability in million instructions per second (MIPS), cost per million instructions processed, uplink, downlink bandwidth, busy and idle power along with their hierarchical level, we also added the position coordinates of fog nodes.
Since this RAW data is collected from various data producers, a preprocessing phase is needed to convert the raw data into a clean dataset, this phase helps achieving better results and using multiple machine learning algorithms in one data set and best out of them is chosen.
We visualize the dataset based on two features (MIPS, RAM), we chose MIPS and RAM because they impact directly the time response of fog nodes.
Other features will be studied such as Uplink Bandwidth and Downlink Bandwidth, also node position.
- Secondly, to cluster fog nodes into similar groups, we used two clustering algorithms K-means and DBSCAN to study which one performs better in this scenario.

Clustering similar nodes together, can help in directing client requests to the appropriate fog node.
As aforementioned, to use K-means we must find the k value which is the number of clusters to create, after applying the Elbow method, the method showed that an optimal number of clusters for this dataset is 2.
We applied the K-means algorithm using two features (MIPS and RAM) with k=2, the results are shown in Figure 7.

Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on August 11,2026 at 09:49:01 UTC from IEEE Xplore. Restrictions apply.

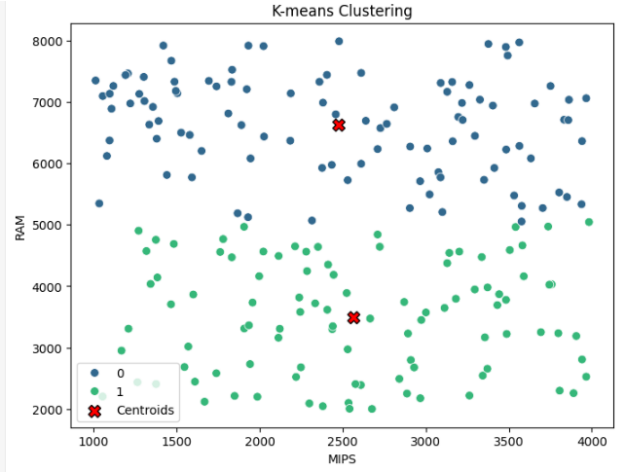


Fig.7. K-means clustering of MIPS vs RAM

To execute DBSCAN algorithm, **epsilon** and **minPts** must be predefined, in literature **minPts** is generally greater or equal do the dimension of the dataset, some studies chose minPts = dimension * 2, in our case we chose minPts= 4.

For determining **epsilon** one technique is used, we calculate the average distance between each point and its k nearest neighbors.

The k-distance graph in Fig.8 shows the distances to the nearest neighbors for each point in Fig.9, sorted in ascending order. The 'elbow' point in the graph suggests a suitable **epsilon** value for DBSCAN

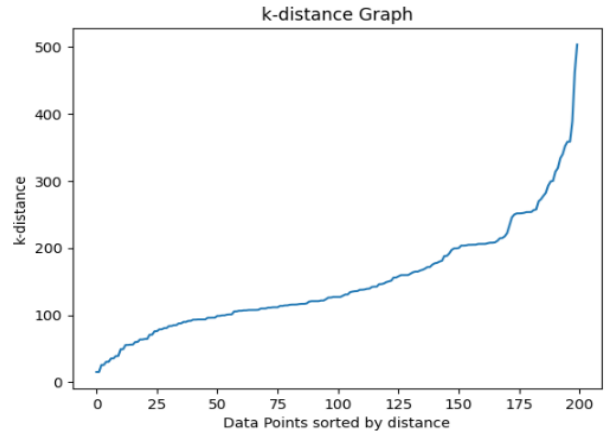


Fig.8. K-distance graph

From the k-distance graph, epsilon is selected starting from approximately 300.

After testing multiple values for epsilon, we chose 400 and it gave these results depicted in Fig.9:

We note that clusters are in different colors, and noise points (outliers) are in black.

We observe that DBSCAN outputted more clusters than K-means, and showed some anomalies (noise points) in the dataset.

- After clustering fog nodes, data producers such as sensors or actuators send their (Write) requests to fog nodes, if a client sent a complex read or write request, an appropriate fog node must be chosen to respond correctly to that request. In each cluster a fog node is assigned as a cluster head; to choose a cluster head we used a density-based selection method, this method involves selecting the point with the highest density within each cluster. In the context of DBSCAN, this can be interpreted as selecting the point with the highest number of neighbors within the **epsilon** radius.

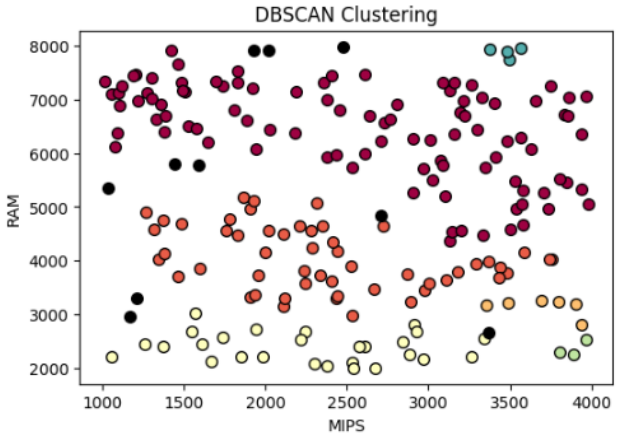


Fig.9. DBSCAN clustering results

- When clients send their requests, the cluster head will choose which fog node is appropriate to reply in terms of request complexity and time response sensitivity. For read requests fog nodes with higher Uplink and Downlink bandwidth are chosen to respond. In case of write requests fog nodes with higher resources (storage or processing) are chosen.

To estimate the load of fog nodes in read or write operations we used Idle power consumption parameter that can be calculated by iFogsim simulator.

We also used storage usage as another estimation metric. to check the amount of disk space being used and the read/write rates. High storage usage and frequent read/write operations can indicate heavy load Various techniques can be used to estimate node load such as real time monitoring tools, machine learning models or heuristic methods.

VI. CONCLUSION

Using fog computing in critical and time sensitive applications is a promising solution that extends cloud computing to make processing and storage closer to end users.

Our work focuses on keeping data replicas in fog nodes synchronized while maintaining a low latency in response time. Multiple studies were discussed in this paper, each study addressed the problem from a different perspective, yet there is still limitations and constraints to address in this field of study.

Data consistency is crucial for real-time decision-making applications; therefore, we proposed an eventual consistency model that prioritize critical data and give advantage to strong consumers to update their data.

As perspective, we plan to extend this clustering method with iFogsim, and proceed with the following steps of our approach to test data replication and consistency in a healthcare fog application.

REFERENCES

[1] Transofrma Insights (2023). 'Global IoT Forecast Report, 2022-2032'

[2] Atlam, H., Walters, R., & Wills, G. (2018). Fog Computing and the Internet of Things: A Review. Big Data and Cognitive Computing, 2(2), 10.

Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on August 11,2026 at 09:49:01 UTC from IEEE Xplore. Restrictions apply.

[3] Yi, S., Hao, Z., Qin, Z., & Li, Q. (2015). Fog Computing: Platform and Applications. 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), 73–78.
[4] Weng, C.-Y., Li, C.-T., Chen, C.-L., Lee, C.-C., & Deng, Y.-Y. (2021). A Lightweight Anonymous Authentication and Secure Communication Scheme for Fog Computing Services. IEEE Access, 9, 145522–145537.
[5] Naas, M. I., Parvedy, P. R., Boukhobza, J., & Lemarchand, L. (2017). iFogStor: An IoT Data Placement Strategy for Fog Infrastructure. 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), 97–104.
[6] Naha, R. K. (2021). Reliable scheduling and resource allocation for IoT applications in fog computing.
[7] Bastien Confais. (2018). Conception d'un système de partage de données adapté à un environnement de Fog Computing.
[8] Quy, V.K., Hau, N.V., Anh, D.V. et al. Smart healthcare IoT applications based on fog computing: architecture, applications and challenges. Complex Intell. Syst. 8, 3805–3815 (2022).
[9] Torabi, E., Ghobaei-Arani, M., & Shahidinejad, A. (2022). Data replica placement approaches in fog computing: a review. Cluster Computing, 25(5), 3561–3589.
[10] Pfandzelter, T., Japke, N., Schirmer, T., Hasenburg, J., & Bermbach, D. (2023). Managing data replication and distribution in the fog with FReD. Software: Practice and Experience,53(10),1958–1981.
[11] George, J., Chen, C. A., Stoleru, R., & Xie, G. G. (2019). Hadoop MapReduce for Mobile Clouds. IEEE Transactions on Cloud Computing, 7(1), 224–236.
[12] Naas, M. I., Lemarchand, L., Raipin, P., & Boukhobza, J. (2021). IoT Data Replication and Consistency Management in Fog Computing. Journal of Grid Computing, 19(3), 33.
[13] Saleh, S. S., Alansari, I., Hamiaz, M. K., Ead, W., Tarabishi, R. A., & Khater, H. (2023). iFogRep: An intelligent consistent approach for replication and placement of IoT based on fog computing. Egyptian Informatics Journal, 24(2), 327–339.
[14] Huang, T., Lin, W., Li, Y., He, L., & Peng, S. (2019). A Latency-Aware Multiple Data Replicas Placement Strategy for Fog Computing. Journal of Signal Processing Systems91(10),1191–1204.
[15] Sheikh, M. S., Enam, R. N., & Qureshi, R. I. (2023). Machine learning-driven task scheduling with dynamic K-means based clustering algorithm using fuzzy logic in FOG environment. Frontiers in Computer Science, 5.
[16] Sabireen, H., & Venkataraman, N. (2023). A Hybrid and Light Weight Metaheuristic Approach with Clustering for Multi-Objective Resource Scheduling and Application Placement in Fog Environment. Expert Systems with Applications, 223, 119895.
[17] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 226–231.

Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on August 11,2026 at 09:49:01 UTC from IEEE Xplore. Restrictions apply.