International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V70I7P204

Volume 70 Issue 7, 32-42, July 2022
© 2022 Seventh Sense Research Group®

Original Article

Dynamic Offloading Framework in Fog Computing

Jyoti Yadav[1], Suman Sangwan[2]

[1,2]*Deenbandhu Chhotu Ram University of Science and Technology, Haryana, India*

[1] jyoti89.cse@gmail.com

Received: 20 May 2022 Revised: 19 June 2022 Accepted: 23 June 2022 Published: 18 July 2022

Abstract *- To meet the needs of Internet of Things (IoT) devices, fog computing has emerged as a new paradigm. Offloading computation tasks is one of the most crucial issues in a fog environment. Computation offloading is the process by which devices send computation-intensive tasks to servers for processing. Because of network constraints, not all computation tasks can be delegated to servers. As a result, it is critical to determine how many tasks should be run on servers and how many should be run locally. Furthermore, due to the uncertainty in the requirements, finding the server to execute the offloaded task in a vibrant environment is difficult. To address this issue, a dynamic computation offloading framework has been proposed. Here clustering is used to locate the decision engine, and fuzzy logic is used for offloading decisions. The major objective of the paper is to determine whether to offload or not depending on CPU usage, delay sensitivity, residual energy, and task size and bandwidth. Our algorithm makes dynamic decisions by sending time-sensitive tasks to local devices or fog nodes for processing and resource-intensive tasks to the cloud. According to simulation results, the proposed algorithm is more efficient than the Energy-aware offloading clustering approach (EAOCA) and the Fuzzy-based offloading algorithm regarding task successful execution, CPU utilization, and average delay. It improves the rate of successfully executed tasks by 5.92% over EAOCA and 4.72% over Fuzzy based approach. It reduces delay by 11.73% over EAOCA and 8.74 % over the Fuzzy approach.*

Keywords *- Fog computing, Clustering, Offloading, Delay, Fuzzy logic.*

1. Introduction

The number of IoT appliances(e.g., smart home appliances, unmanned surface vehicles (USVs), intelligent automobiles, smart sensors) has increased significantly with the rapid development of various IoT fields, such as smart cities and intelligent transport systems, smart homes, and smart manufacturing.[1] According to Cisco, approximately 500 billion connected devices will be used by 2025.[2] IoT devices are often compact, battery-powered, and outfitted with sensors, and they have the limited battery, computational power, and storage capacity.[3] Moreover, these devices use wireless technology for communication.[4] Consequently, many latency-sensitive real-time computing activities, such as video surveillance, face recognition, and augmented reality (AR)[5], have difficulty delivering a real-time experience to customers when run on local devices. These computation-intensive tasks are typically sent to the remote cloud.[6] It is at a multi-hop distance from the client and centralized too, which incurs additional expenditures in terms of time and computing power to maintain the long-distance connection. However, because long-distance communication between IoT devices and distant clouds requires a lot of bandwidth, transferring all locally created tasks to remote clouds for processing would cause significant issues, including excessive latency and network congestion.

One of the successful techniques for addressing these issues nowadays is offloading all resource-intensive tasks from the local device to nearby fog servers.[7] The fog servers can be a gateway, router, access point, switches, hub, etc. [8], are closer to the local devices in terms of location, and have cheap communication costs and quick reaction times.[9] It ensures that latency-sensitive operations are completed in real-time and minimizes battery-powered devices' energy consumption. However, efficient offloading in a cloud-fog-end computing system remains difficult due to resource heterogeneity, variety of user needs, network complexity, and task interdependence. Offloading is an np hard optimization problem[10] because we must deal with many questions like what, where, how, and why offloading. Partially and completely offloaded tasks are the two types of offloading techniques that deal with what to offload. A partially offloading[11], [12] task allows users to offload parts of or all of their work at once, while a complete offloading task will enable users only to handle all of their tasks locally or in fog devices.[13] On the other hand, due to several uncertainties of the system and network, such as network traffic, limited bandwidth, and end device remaining battery, the decision engine should decide when the computation is offloaded to higher layers to meet the desired quality of service(QoS). Another dimension of the decision is where to offload.[14], [15] The decision engine should decide where to offload the computation based on several parameters such as latency, resource availability, and the computation capacity required for completing the task. Some researchers have attempted to overcome this issue by optimally offloading tasks to more competent computer resources.

Information about the current scenario is required to design a practical offloading framework, like remaining energy, remaining computation power, communication overhead, available bandwidth, etc. The data transmission speed between offloading decision engines and other devices should be carefully designed to avoid communication overhead. So the location of the decision engine has become an essential design issue. It can be located on any three-level, i.e., cloud, fog, and local devices. In this paper, the clustering technique is used to find a suitable location for the decision engine and avoid communication overhead compared to distributed and centralized techniques; then, fuzzy logic is used to find the best offloading destination based on multiple criteria.

So the main contribution of this work is summarized as follows :

- To enable the IoT devices to execute computation-intensive tasks that they can not accomplish due to limited resource availability.
- To find the best location for the decision engine deployment by clustering fog nodes based on weight parameters.
- To make the dynamic offloading decision using fuzzy logic to find the best destination for offloading.

The remaining paper is as follows: Section 2 describes the related work. Section 3 presented system architecture and proposed work. Section 4 presents simulation results, and Section 5 concludes the findings and the future research direction in this area.

2. Related Work

Several approaches have been proposed in the literature with the intent of efficient computation offloading in fog computing. Nowadays, fog computing has become the appropriate framework for IoT applications. Using computation offloading, we can improve the performance of IoT devices.[16], [17] However, computation offloading needs to consider various aspects, like network condition, server capability, task size, etc.[18] Various decision edges must be considered to recognize the benefit of computation offloading, like what, when, how, where, and why to offload.[19]

What to offload emphasizes finding the resource-intensive part of the problem. In full offloading, the whole application can be offloaded to a remote server, while in partial offloading, the application is divided for offloading. When to offload is about finding the right time for offloading a task to the cloud or fog node. Where to offload is about finding a destination for offloading. Authors in [16], [20] applied optimization techniques to find the best destination for offloading. Offload deals with application characteristics like energy consumption, response time, service time, etc. Computation offloading is a complex problem because we need to consider various aspects. Most current research focuses on static offloading choice algorithms, assuming that mobile fog-cloud environments remain constant.[21]–[23] During the application development phase, these algorithms build offloading methods via programme analysis, and the offloading strategies are fixed when the application development is completed. There is little research on dynamic adaptive offloading decision algorithms that generate offloading methods in real-time and often update them to adapt to dynamic mobile fog cloud settings. Many researchers focused on one or two parameters for offloading. The decision engine plays a significant role in offloading because it decides where to offload, as shown in Table 1. Moreover, transmission speed between remote servers and IoT devices should be efficiently allocated not to cause communication overhead. Therefore the location of the decision engine becomes an important design issue. Only a few authors have focused on the decision engine's location to take an offloading decision. It is therefore challenging to determine the location of the decision engine for offloading. The second most important concern in offloading is deciding to offload in a dynamic and uncertain environment. Fuzzy logic is one of the best techniques for rapidly changing uncertain scenarios. Few authors focused on fuzzy logic for solving such problems, as shown in Table 2. So a fuzzy-based decision engine is proposed in the fog computing simulator.

Table 1. Comparative analysis of different offloading algorithms based on the location of the decision engine

Reference	Location of decision engine	Objective	Tool	Remarks
[24]	Centralized	Workload orchestration	Open stack	Single point of failure
[25]	Centralized at cloud	Service orchestration	Matlab	Single point of failure
[26]	Distributed at fog nodes	Application placement	ifogsim	Fix devices used for application deployment
[27]	Clustering of fog nodes	Resource allocation	-	Clustering used for resource allocation
[28]	Clustering of the fog node	offloading	Matlab	Only the energy parameter is considered for clustering fog nodes
[29]	Clustering of cells under fog node	Prediction of fog node location	Numerically proved	Proved significance of clustering in fog computing
[30]	Clustering of fog nodes	Offloading	Matlab	Only the energy parameter is considered for clustering fog nodes

Table 2. Comparison of fuzzy-based decision engine

Reference	Algorithm	Parameters	Tool	Remarks
[31]	Fuzzy logic-based decision engine	Bandwidth, memory, data size	Real devices	The decision engine had been deployed centrally. Sending every request to a single device for decision-making can overload the device.
[32]	fuzzy multi-criteria decision making	RAM, uploading speed, downloading speed, task size	Raspberry pi (Real IoT devices)	The two-tier architecture is considered for offloading, i.e., IoT and cloud layer only.
[33]	A fuzzy evolutionary scheduler for multi-objective resource allocation in fog computing	Computation processing time and input/output overhead	-	Clustering and fuzzy algorithm are used for resource allocation

3. System Architecture

A generic fog computing architecture[34] consists of three main layers. Layer 1 is the cloud layer, composed of high computing resources located at a considerable distance. Layer 2 is called the fog layer, and this layer consists of distributed fog nodes that are deployed near the end-user. The fog nodes have less computing power as compared to the cloud. These can be deployed either by cloud data centres or service providers. The third layer is the user layer, consisting of mobile devices or IoT devices that may or may not have the processing power. IoT is the next step toward the pervasive and global interconnection of objects and human beings.[35] Our proposed framework has four main modules: task generator, decision engine, network manager, and resource manager, as shown in Figure1. The first module is the task generator which is accountable for task generation. Three types of applications: intelligent transport systems, smart buildings, and health care, are deployed on end devices to support scalability and heterogeneity. CPU utilization, delay and file size, and application characteristics are specified. The second or main module is the decision engine, responsible for task offloading decisions. It decides whether it is beneficial to offload the task or not if it is, and then finds the destination for offloading, either at the fog node or cloud.

The third module is network manager; it accumulates real-time network information. It is responsible for monitoring available bandwidth and network traffic. Allocated bandwidth also varies depending on congestion in the network. The fourth module is resource manager, which keeps track of available resources with virtual machines like energy, memory, and computing capacity.

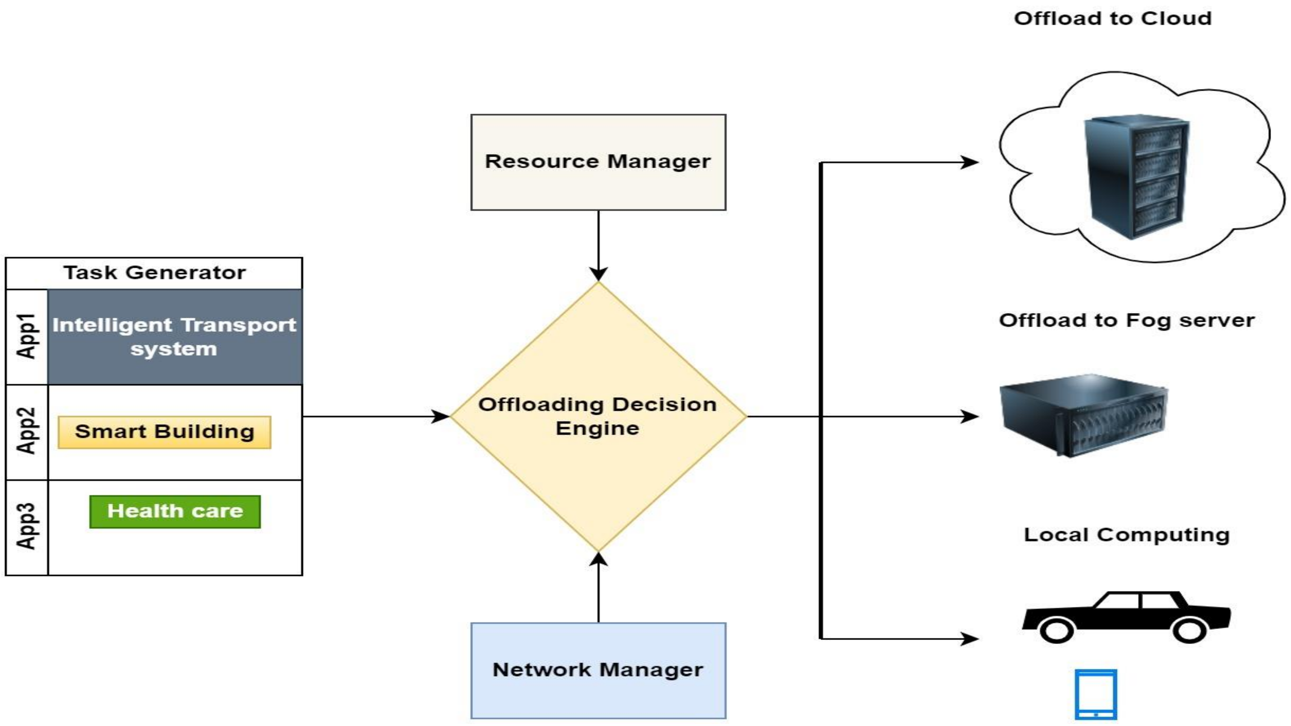

Fig. 1 System architecture

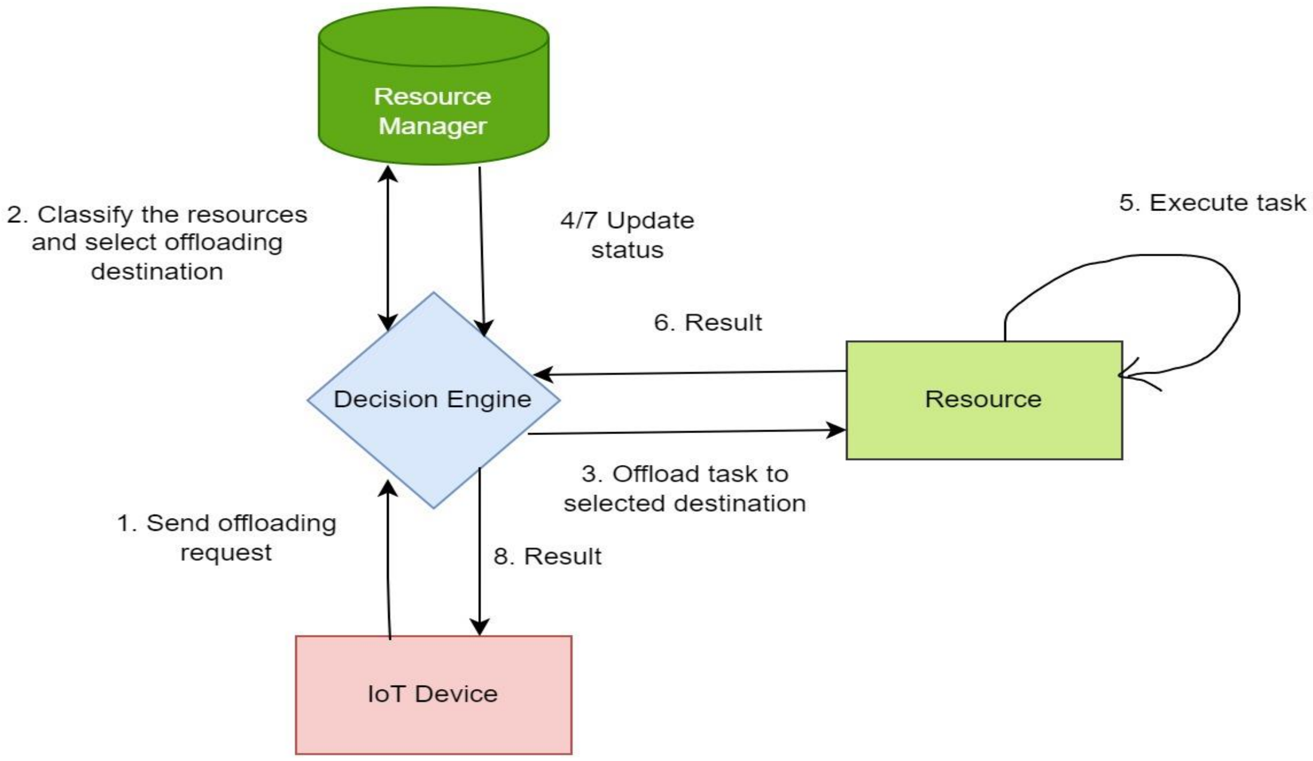
Resource Manager
2. Classify the resources and select offloading destination
4/7 Update status
5. Execute task
6. Result
Decision Engine
Resource
3. Offload task to selected destination
1. Send offloading request
8. Result
IoT Device

Fig. 2 Execution dataflow

It maintains a list of available resources in fog nodes and cloud virtual machines (VM). The decision engine selects the destination for the offloaded task from the available resources. Each device communicates its metadata to the resource manager when connecting with the network. IoT sensors have limited or negligible processing capacity. They must delegate the processing of their work to someone else.

For this reason, the decision engine will issue a computation offloading request. The request includes details about the device's condition and needs, such as the application ID (also known as the container ID), data to be processed, and task latency sensitivity and memory requirements, all of which are required to determine the appropriate offloading destination. The decision engine collects the information from the resource manager about available resources. Then decision engine decides where to offload the task. Finally, the device receives a decision indicating the location to execute the task. Then the task is offloaded to that particular resource, and results are collected and communicated to the end device. But the location of the decision engine plays a key role in this scenario. If we deploy it centrally on the cloud, it may result in bottlenecks, and the decision engine plays a role in the single source of failure.

Further, cloud deployment of decision engines causes excessive latencies and uses backhaul resources. We have proposed a cluster-based deployment strategy for the decision engine to overcome this issue. The following subsection will explain the cluster-based deployment strategy:

3.1. Cluster-Based Deployment Strategy

In every clustering algorithm, we need to follow two steps, i.e., cluster formation and cluster head selection. In fog computing, clusters can be made at end devices and the fog layer. This paper focuses on the fog layer, i.e., fog devices will make clusters, and one cluster head is selected. The cluster head is responsible for further communication with the end device layer or cloud. Here a weight parameter is associated with each device. The weight parameter consists of the computational capacity of fog nodes, residual energy, and distance between devices. The device with the highest weight is considered a cluster head. The fuzzy decision-making algorithm is deployed on the cluster head that decides the destination for offloading the data, i.e., at the fog or cloud layer. In this work, we have considered M fog nodes identified by the set $N = \{n_1, n_2, n_3, \dots\dots n_m\}$ randomly scattered in the (X, Y) dimension area. All fog nodes have computation capacity higher than end devices and lower than a cloud server.

$$W(t) = \alpha_1 . N_{CC} + \alpha_2 . E + \alpha_3 . D \qquad (1)$$

Where $W(t)$ represents the weight of the node. α_1, α_2, and α_3 are the weighted parameters, and the value of these parameters depends on the priority of the factors where E is the residual energy, N_{cc} represents the computation capacity of a node, and D represents distance.

$$W_t^c = W_t^p * b \text{ where } 0 < b < 1 \qquad (2)$$

Here W_t^c is the weight of the cluster member, and b is the hop cost from the cluster member to its cluster head. W_t^P is the weight of the cluster head. Here is the weight parameter to determine the role of each fog node. The fog nodes can be dynamically split according to their weight. A threshold value(θ_w) is employed to divide the fog nodes into two groups.

- N_w^h(t), the fog node having a higher weight than a given threshold selected as the cluster head
- N_w^l(t), the fog node having a lower weight than a given threshold selected as a cluster member

To be specific, we can define two sets as follows

$$N_w^h(\text{t}) = \{n_i \in N_w^h(\text{t}) | W(t) \geq \theta_w\} \qquad (3)$$

$$N_w^l(\text{t}) = \{n_i \in N_w^l(\text{t}) | W(t) \leq \theta_w\} \qquad (4)$$

At any time, each fog node belongs to a set S={0,1} in which 1 represents cluster head and 0 represents cluster member. Several cluster members and a cluster head make up each cluster. The cluster head is selected from the higher weight node-set and cluster members from the lower weight node-set.

Algorithm 1: Clustering Scheme

```
1.  Input: M fog nodes
2.  Output $C_i \forall i$
3.  Each node Calculate its weight factor $W$(t) using the equation (1)
4.  For all $n_i \in N$ do
5.     if $W(t) \geq \theta_w$ then
6.     $N_w^h$(t) ← $n_i$
7.      Else
8.     $N_w^l$(t) ← $n_i$
9.   End if
10. End for
11. While $N_w^h \neq \emptyset$ do
12.  Select $n_i \in N_w^h \mid max_{n_i}\{W(t)\}$
13. $C_i \leftarrow n_i$
14. $for\ all\ n_j \in N_w^l$(t) do
15.      $if\ d(n_i, n_j) \leq R$ then
16. $\tilde{C}_i \leftarrow n_j$
17. $end\ if$
18. End for
19. $while\ |C_i < M|\ do$
20. $C_i \leftarrow n_j | min_{n_j}\{W(t)\}\ \forall n_j \in \tilde{C}_i$
21. $remove\ n_j\ from\ N_w^l\ and\ \tilde{C}_i$
22. End while
23. Remove $n_i\ from\ N_w^h$
24. $if\ N_w^l \neq \emptyset$ then
25. $for\ each\ n_j \in N_w^l$ do
26. $C_j \leftarrow n_j$
27. $remove\ n_j\ from\ N_w^l$
28. $end\ for$
29.  End if
30. Then devices update weight using equation(2)
31. Update cluster head and cluster formation
32. end
```

Algorithm 1 depicts the clustering method's pseudo-code; initially, all nodes' weights are considered zero. Then the weight of each node is calculated(lines 1-5). Then, among the fog nodes, a threshold weight is considered so that fog nodes with a higher weight than the threshold weight are added in N_w^h(t) otherwise, they are added in N_w^l(t) (lines 6-12). Then starting from the ith fog node having the highest weight, the ith cluster C_i is created with the ith fog node as the cluster head(lines 13-15). All the fog nodes having a lower weight set are considered, and those with a distance for cluster head lower than the coverage range are selected and added to the temporary cluster $\tilde{C}_i$ Composed by the candidate's cluster members of the ith cluster head(lines 16-20). Among cluster members, the fog nodes with the lowest weight are added to the cluster C_i and removed from both N_w^l *and* $\tilde{C}_i$. The bound M has been introduced to limit the cluster size (lines 21-25). In the last, the remaining fog nodes in N_w^l are considered as isolated nodes(lines 26-31). Then the weight is updated to update the cluster.

After cluster formation, the decision engine is deployed over the cluster head. Then decision engine will decide where to offload data. Here fuzzy rules are used to specify the actions of the decision engine to determine where to execute the task based on different parameters. It determines the destination device for an offloaded request by considering CPU utilization, delay sensitivity, residual energy, task size, and bandwidth.

3.2. Offloading Algorithm

The IoT environment frequently changes with time. The offloading algorithm is expected to adopt such changes to meet the quality of service (QoS). To guarantee this, we have proposed offloading algorithm based on fuzzy rules to find the best offloading destination. Fuzzy logic deals with uncertain or decision-making problems based on specific criteria and helps reduce the problem's complexity.[36] Fuzzy-based decision engines act as a coordinator that collects info from the resource manager and decides the offloading destination based on fuzzy rules. Three linguistic variables are defined for each offloading objective. Fuzzifier converts each input to the linguistic variable based on the membership function.[37] Here membership functions are defined according to each criterion to represent the role of the parameter in decision making. In the knowledge base, we store knowledge about all input-output fuzzy relationships. Fuzzy inference determines the performance of the system by using if-then rules. These conditional statements approve the position of a particular variable.

The fuzzy logic-based decision engine operates on four variables

$$Fl = (p, q, r, s, t) \qquad (5)$$

Where p represents delay sensitivity, q represents task size, r represents CPU utilization, s represents residual energy, and t represents bandwidth.

3.3. Fuzzy Inference System

Here we have used three linguistic variables, i.e., low, medium, and high, for all parameters. In a Fuzzy logic system, membership functions quantify the linguistic variables. Here we have used the triangular membership function because of its simplicity. We associate linguistic terms with the crisp values using the membership function during fuzzification. The following equation represents fuzzy classes for each identifier.

$$F_p(x) = [\mu_p^L(x), \mu_p^M(x), \mu_p^H(x)]$$

$$F_q(x) = [\mu_q^L(x), \mu_q^M(x), \mu_q^H(x)]$$

$$F_r(x) = [\mu_r^L(x), \mu_r^M(x), \mu_r^H(x)]$$

$$F_s(x) = [\mu_s^L(x), \mu_s^M(x), \mu_s^H(x)]$$

$$F_t(x) = [\mu_t^L(x), \mu_t^M(x), \mu_t^H(x)]$$

In the inference step, we evaluate fuzzy rules from the knowledge base. The input and output in an inference system are fuzzy variables. These fuzzy variables are used for the defuzzification step. The fuzzy rule is an expression with condition and output. These are represented in if-then form.

Table 3. Fuzzy rules

Decision Index	p	q	r	s	t	Decision
D1	H	L	L	H	H	Fog
D2	H	L	L	L	L	Local
D3	M	M	L	M	M	Fog
D4	L	L	L	L	H	Cloud

Defining fuzzy rules is very complex and critical because the performance of FLS depends on fuzzy rules. In inference for aggregation step, maximum and minimum functions are used after that evaluated result of if part is applied to $then$ part using activation method. Finally, in the accumulation step, we used the maximum function to combine the results of multiple rules using Equation 6.

$$\mu_{fog} = \max\{\mu_{fog}^{D_1}, \mu_{fog}^{D_2} \dots \dots . \mu_{fog}^{D_n}\} \qquad (6)$$

The last step of FLS is defuzzification which converts the fuzzy output to real values. For defuzzification, various methods are used as the mean of maximum, modified height, maximum, centroid method, etc.

We have used a centroid defuzzifier in this step. The centre of gravity(CoG) value is evaluated using Equation 7.

$$X^* = \frac{\int x\mu(x)dx}{\int \mu(x)dx} \qquad (7)$$

After the centroid defuzzifier, output X^* becomes a crisp value between 0 to 100. The algorithm selects the local device, fog node, and cloud node based on the CoG value.

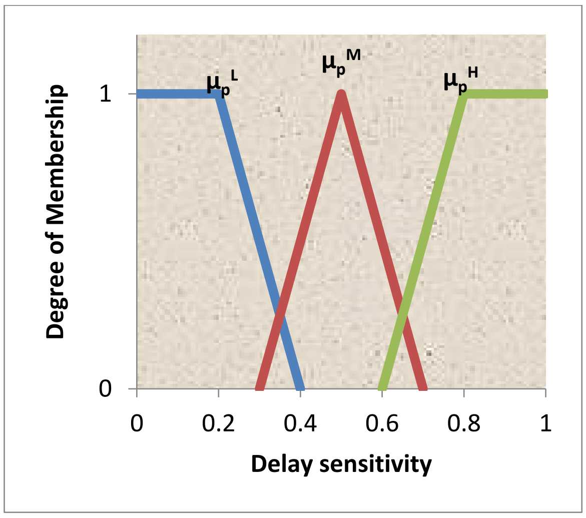

Fig. 3 Membership function for delay

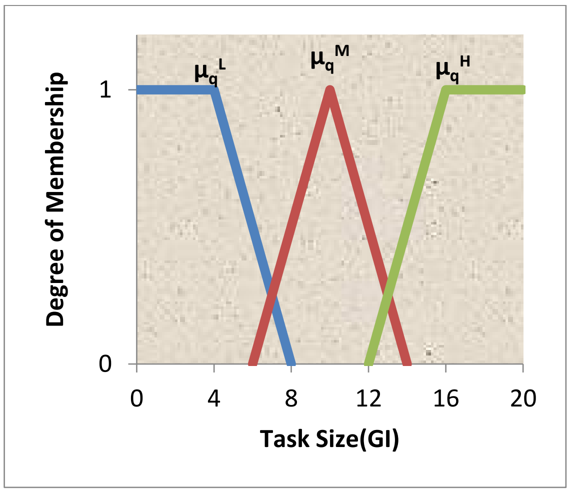

Fig. 4 Membership function for task size

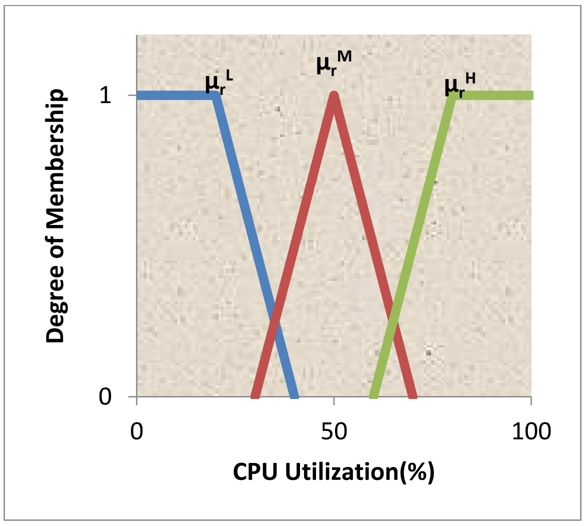

Fig. 5 Membership function for CPU utilization

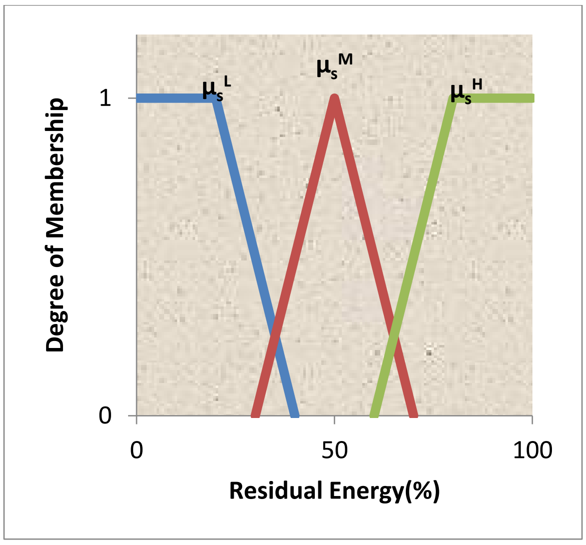

Fig. 6 Membership function for residual energy

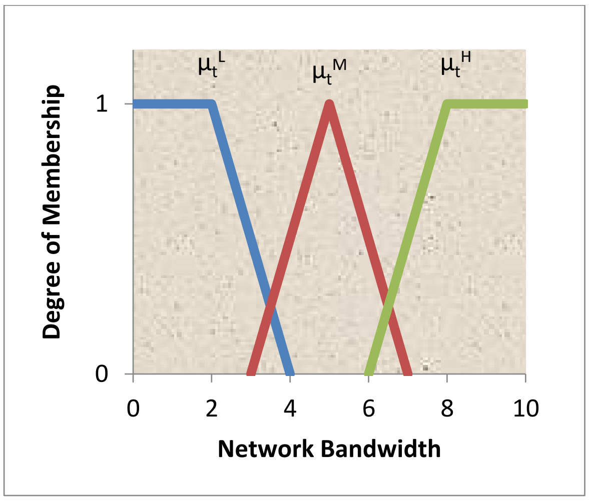

Fig. 7 Membership function for network bandwidth

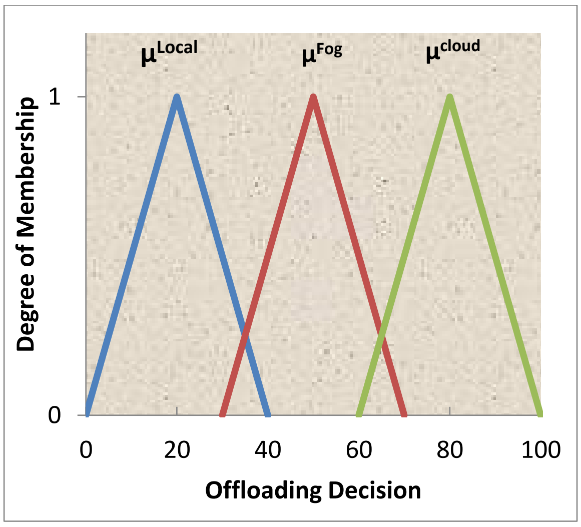

Fig. 8 Output membership function

Algorithm 2: Offloading Algorithm

OD: offloading decision

RC: Required capacity

EC: Existing Capacity

Input: Incoming task T, Read value of parameters(Delay sensitivity, task length. CPU utilization, residual energy, bandwidth)

Output: Offloading Decision

```
1.  Initialize $X^*$=0
2.  Read incoming task profile
3.  Calculate crisp value $X^*$ using equation (5)(After
    defuzzification value of $X^*$ lies between 0 to 100)
4.  {
5.  If(X*<=30) then
6.     If(RC<EC)
7.        OD=Local
8.    else
9.        OD=Remote
10.    end
11. else
12.    if (30<X*<=60)
13.       OD=Fog
14.    else
15.       OD=Cloud
16. }
17.  Return OD
```

4. Results and Discussion

By verifying the performance of the proposed algorithm, various scenarios are simulated using the PureEdgeSim simulator. This simulator expands the CloudSim toolkit by adding a mobility manager, network manager, simulation manager, and task generator to simulate the real physical fog environment. The simulator operates on a system with an Intel® coreTM i58250U CPU @ 1.60GHz processor, 8GB RAM, and a 64-bit operating system configuration. To justify the performance of the new framework, it is compared with the clustering technique EAOCA[28] and the fuzzy-based decision technique.[36] The task offloading request generates a different network and processing load on the remote server. For example, a high computational task needs a large amount of CPU processing capability; however, a data storage application needs low computational capability. However, data backup requires large network bandwidth, whereas computation-intensive tasks may require less bandwidth. Three applications were used to simulate a real-world scenario: an intelligent transport system, a smart building, and a health system to support heterogeneity. These applications were deployed on end devices. The number of end devices raise from 100 to 500 with the counter step of 100. In this study, the exponential growth of tasks is considered based on interval time. Delay sensitivity represents whether task is delay tolerant or not. If the value of delay sensitivity is low , the related application is delay tolerant. The tasks were generated randomly according to Table 4. Various parameters need to be set for the fog simulation environment. The major parameters of the simulation are shown in Table 5. Numerous metrics have been gathered from simulation results. Here, we will only pay attention to the number of tasks that were successfully completed, the average delay, and the average CPU utilisation at each level of the fog computing environment.

Table 4. Applications specification

Application	App1	App2	App3
Percentage of devices(%)	40	30	30
Task interval(sec)	2	8	4
Task Size(GI)	9	20	3
Delay Sensitivity	0.8	0.5	0.3
CPU Utilization (%)	5	15	2

Figure 9 shows the number of tasks executed at each level in fog computing architecture.

Table 5. Simulation parameters[38]

Parameters	Value
Minimum Devices(No.)	100
Maximum Devices(No.)	500
Simulation Duration	30(min)
Counter step	100
Number of the fog server	18
VM per fog server	2
VM in cloud	6
Simulation Area	500*500(meter)
Architecture	Fog and Cloud

It demonstrates that more tasks are being completed at the fog node than at the local device or cloud. Figure 10 shows the percentage of the number of tasks successfully executed. The simulation results show that most tasks can be completed successfully when the server has few tasks. However, the success rate decreased as the number of devices increased. The performance improvement rate of the proposed mechanism over the existing algorithm is calculated using Equation 8.[39] Suppose the proposed algorithm is ith and existing is the jth algorithm.

$$PIR(\%) = \left(\frac{(\Sigma_{ith}P - \Sigma_{jth}P}{\Sigma_{jth}P}\right) * 100 \qquad (8)$$

Where P represents the parameter value, PIR (%) is calculated based on the percentage of tasks successfully executed. The increase in the percentage of tasks successfully executed is 5.92% over EAOCA and 4.72% over fuzzy logic.

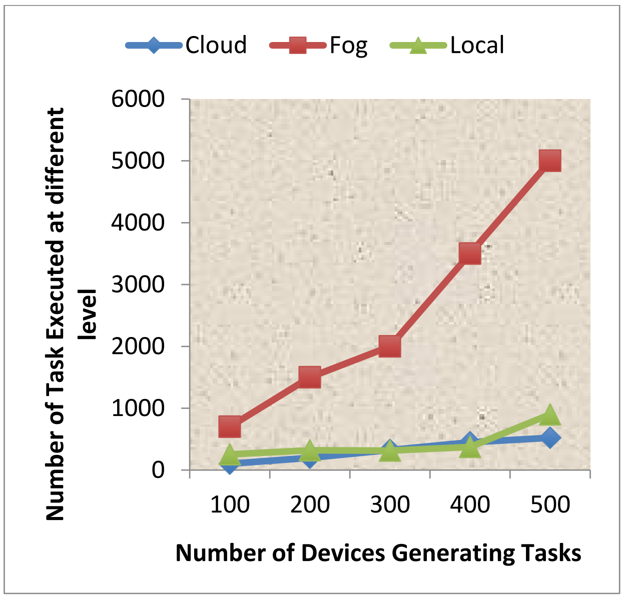

Fig. 9 Number of tasks executed at each level

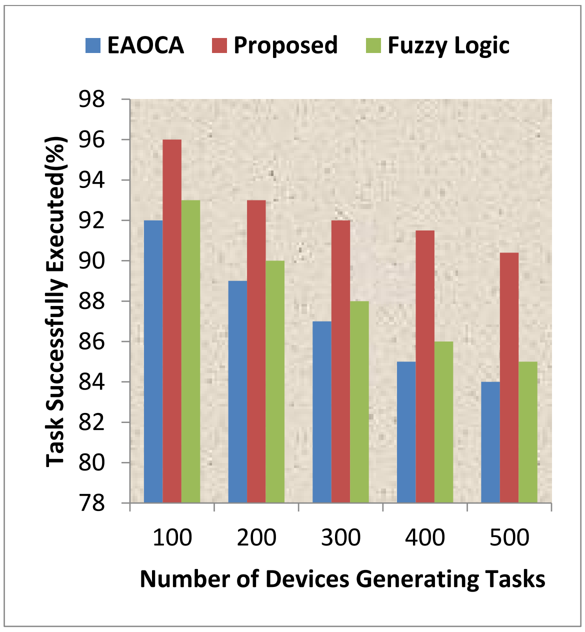

Fig. 10 Number of tasks executed successfully

The average CPU utilization of VMs executing on fog and cloud is depicted in Figure 11. Consistently low CPU utilization is not a preferred result compared to efficient utilization of resources. Because the CPU utilization is minimum with limited resources is much more efficient than minimum CPU utilization with high availability of resources. The CPU utilization is calculated using Equation 9.

$$\text{CPU Utilization}=\sum_{i=0}^{j-1} CPU_i \qquad (9)$$

Where j is the number of CPUs and CPU is the utilization of ith CPU. Our algorithm's average CPU utilization results represent that the dynamic fuzzy-clustering-based algorithm uses better resources than the EAOCA algorithm and fuzzy logic-based offloading framework.

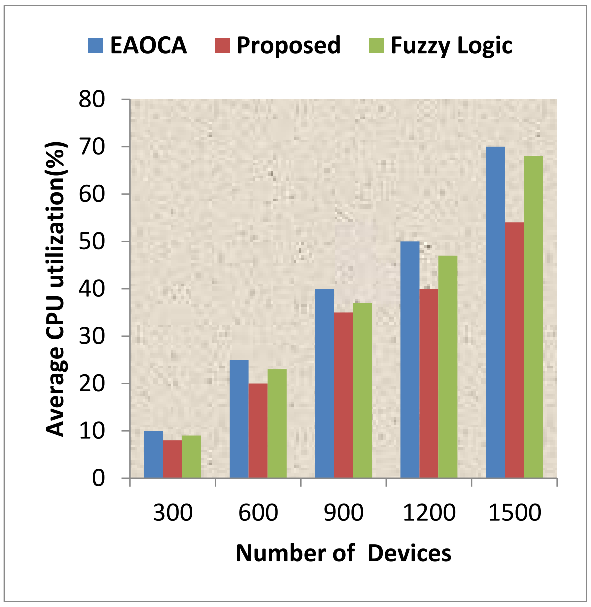

Fig. 11 The average CPU utilization

One of the key challenges in offloading is to minimize the average delay. It is the average time taken for transmitting, waiting, computing, and receiving back results. In Figure 12, a comparative analysis of the average delay is shown. The proposed framework has reduced delay by approximately 8.74% compared to fuzzy logic and 11.73% compared to the EAOCA algorithm because of more tasks executed at the fog layer than at the cloud layer. The above analysis shows that the proposed algorithm outperforms other algorithms regarding CPU utilization, delay, and percentage of successfully executed tasks.

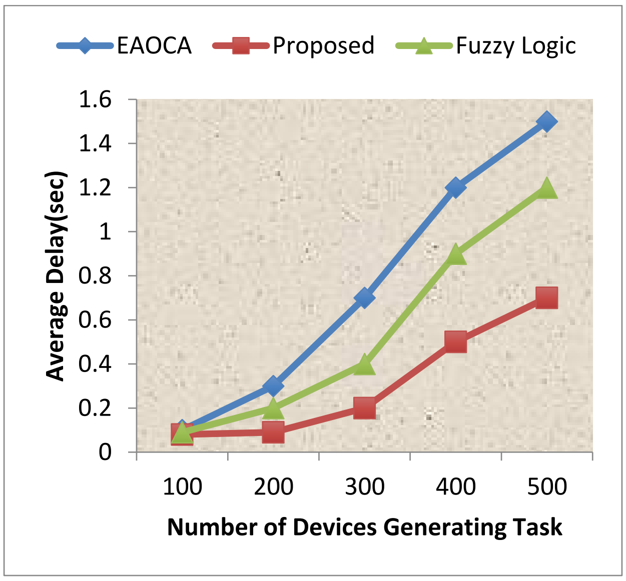

Fig. 12 The average delay

5. Conclusion

Nowadays, a significant number of IoT devices are used in different environments. These devices have limited resources. However, they generate a significant amount of data in a short duration. Cloud computing may be viable, but the inherent latency renders it unviable. Fog computing is a potentially useful solution located near the users. For resource-constrained IoT devices, it provides low latency, high bandwidth, sharp responsiveness, and dependability. As a result, we suggested a three-tier fog-cloud integration architecture in this article. We used the clustering technique to find offloading decision engine deployment location, and fuzzy logic was used to find the best offloading destination. The results show that the dynamic computation offloading framework outperforms the EAOCA and fuzzy logic-based offloading algorithms regarding the number of tasks successfully executed, CPU Utilization, and average delay. Optimization approaches could be employed to improve the decision engine's performance in the future.

References

[1] A. Y. Joshi and P. S. Khanvilkar, "An Energy Efficient Workload Offloading in Fog Computing," pp. 5640–5645, 2020.

[2] W. Paper, "Prepare to succeed with the Internet of Things," pp. 1–9, 2017.

[3] Anu and A. Singhrova, "Optimal Healthcare Resource Allocation in Covid Scenario Using Firefly Algorithm," *International Journal of Engineering Trends and Technology*, vol. 70, no. 5, pp. 240–250, 2022.

[4] J. de J. Rugeles Uribe, E. P. Guillen, and L. S. Cardoso, "A technical review of wireless security for the internet of things: Software defined radio perspective," *Journal of King Saud University - Computer and Information Sciences*, no. xxxx, 2021.

[5] S. Shahhosseini *et al.*, "Exploring computation offloading in IoT systems," *Information Systems*, no. xxxx, p. 101860, 2021.

[6] L. Zhang, Y. Liu, and S. Shen, "Construction of performance monitoring model for cloud computing service platform based on label technology," *International Journal of Information and Communication Technology*, vol. 17, no. 2, pp. 178–193, 2020.

[7] G. R. kumar, N. Saikiran, and A. Sathish, "FOG: A Novel Approach for Adapting IoT/IoE in Cloud Environment," *International Journal of Engineering Trends and Technology*, vol. 42, no. 4, pp. 189–192, 2016.

[8] D. Rahbari and M. Nickray, "Task offloading in mobile fog computing by classification and regression tree," *Peer-to-Peer Networking and Applications*, 2019.

[9] V. Meena, M. Gorripatti, and T. Suriya Praba, "Trust Enforced Computational Offloading for Health Care Applications in Fog Computing," *Wireless Personal Communications*, no. 0123456789, 2021.

[10] M. M. Hussain and M. M. S. Beg, "CODE-V: Multi-hop computation offloading in Vehicular Fog Computing," *Future Sciences*, no. 40, 2021.

[11] F. Yu, H. Chen, and J. Xu, "DMPO: Dynamic Mobility-Aware Partial Offloading in Mobile Edge Computing," *Future Generation Computer Systems*, vol. 89, pp. 722–735, 2018.

[12] Z. Ning, P. Dong, X. Kong, and F. Xia, "A Cooperative Partial Computation Offloading Scheme for Mobile Edge Computing Enabled Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2019.

[13] Z. Li and Q. Zhu, "Genetic Algorithm-Based Optimization of Offloading and Resource Allocation in Mobile-Edge Computing," *Information (Switzerland)*, vol. 11, no. 2, pp. 1–13, 2020.

[14] M. Babar, M. S. Khan, A. Din, F. Ali, U. Habib, and K. S. Kwak, "Intelligent Computation Offloading for IoT Applications in Scalable Edge Computing Using Artificial Bee Colony Optimization," *Complexity*, vol. 2021, pp. 1–12, 2021

[15] M. Keshavarznejad, "Delay-Aware Optimization of Energy Consumption for Task Offloading in Fog Environments Using Metaheuristic Algorithms," *Cluster Computing*, vol. 0123456789, 2021.

[16] S. Feng, Y. Chen, Q. Zhai, M. Huang, and F. Shu, "Optimizing Computation Offloading Strategy in Mobile Edge Computing Based on Swarm Intelligence Algorithms," *Eurasip Journal on Advances in Signal Processing*, vol. 2021, no. 1, 2021.

[17] H. Mahini and A. Masoud, "An Evolutionary Game Approach to Iot Task Offloading in Fog - Cloud Computing," *The Journal of Supercomputing*, no. 0123456789, 2020.

[18] M. Adhikari, S. N. Srirama, and T. Amgoth, "Application Offloading Strategy for Hierarchical Fog Environment Through Swarm Optimization," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4317–4328, 2020.

[19] F. Sufyan and A. Banerjee, "Computation Offloading for Smart Devices in Fog-Cloud Queuing System," *IETE Journal of Research*, 2021.

[20] M. Adhikari and H. Gianey, "Energy Efficient Offloading Strategy in Fog-Cloud Environment for Iot Applications," *Internet of Things*, vol. 6, pp. 100053, 2019.

[21] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On Reducing IoT Service Delay via Fog Offloading," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, 2018.

[22] M. G. R. Alam, M. M. Hassan, M. Zi. Uddin, A. Almogren, and G. Fortino, "Autonomic Computation Offloading in Mobile Edge for Iot Applications," *Future Generation Computer Systems*, vol. 90, pp. 149–157, 2019.

[23] N. Shan, Y. Li, and X. Cui, "A Multilevel Optimization Framework for Computation Offloading in Mobile Edge Computing,"*Mathematical Problems in Engineering*, vol. 2020, 2020.

[24] D. Santoro, D. Zozin, D. Pizzolli, F. De Pellegrini, and S. Cretti, "Foggy: A Platform for Workload Orchestration in a Fog Computing Environment," *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, vol. 2017, pp. 231–234, 2017.

[25] N. Morkevicius, A. Venčkauskas, N. Šatkauskas, and J. Toldinas, “Method for Dynamic Service Orchestration in Fog Computing,” *Electronics (Switzerland)*, vol. 10, no. 15, pp. 1–22, 2021.

[26] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of Experience (QoE)-Aware Placement of Applications in Fog Computing Environments,"*Journal of Parallel and Distributed Computing*, vol. 132, pp. 190–203, 2019.

[27] H. Cheng, W. Xia, F. Yan, and L. Shen, "Balanced Clustering and Joint Resources Allocation in Cooperative Fog Computing System," pp. 1–6, 2019.

[28] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "An Energy-Aware Offloading Clustering Approach (EAOCA) in fog computing," *Proceedings of the International Symposium on Wireless Communication Systems*, vol. 2017, no. 8, pp. 390–395, 2017.

[29] E. Balevi and R. D. Gitlin, "A Clustering Algorithm that Maximizes Throughput in 5G Heterogeneous F-RAN Networks," *IEEE International Conference on Communications*, vol. 2018, 2018.

[30] A. Bozorgchenani, S. Disabato, D. Tarchi, and M. Roveri, "An energy Harvesting Solution for Computation Offloading in Fog Computing," *Computer Communications*, vol. 160, no. 3, pp. 577–587, 2020.

[31] N. M. Dhanya, G. Kousalya, P. Balarksihnan, and P. Raj, "Fuzzy-Logic-Based Decision Engine for Offloading Iot Application Using Fog Computing," *Handbook of Research on Cloud and Fog Computing Infrastructures for Data Science*, no. 5, pp. 175–194, 2018.

[32] W. Wibisono, M. Widhi, P. Putu, T. Ahmad, and R. Anggoro, "An Adaptive Offloading Framework for Improving Performance of Applications in IoT Devices Using Fuzzy Multi Criteria Decision Making," vol. 7, pp. 31–36, 2018.

[33] C. ge Wu, W. Li, L. Wang, and A. Y. Zomaya, "An Evolutionary Fuzzy Scheduler for Multi-Objective Resource Allocation in Fog Computing," *Future Generation Computer Systems*, vol. 117, pp. 498–509, 2021.

[34] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, and A. Leon-Garcia, "Fog Computing: A Comprehensive Architectural Survey," *IEEE Access*, vol. 8, pp. 69105–69133, 2020.

[35] S. Trab, A. Zouinkhi, E. Bajic, M. N. Abdelkrim, and H. Chekir, "IoT-Based Risk Monitoring System for Safety Management in Warehouses," *International Journal of Information and Communication Technology*, vol. 13, no. 4, pp. 424–438, 2018.

[36] K. Khalid and E. N. Madi, "A Review of Computation Offloading for Mobile Cloud Computing Based on Fuzzy Set Theory," *International Journal of Engineering Trends and Technology*, no. 1, pp. 56–63, 2020.

[37] A. Kesarwani and P. M. Khilar, "Development of Trust Based Access Control Models Using Fuzzy Logic In Cloud Computing," *Journal of King Saud University - Computer and Information Sciences*, no. 40, 2019.

[38] C. Mechalikh, H. Taktak, and F. Moussa, "Towards a Scalable and QoS-Aware Load Balancing Platform for Edge Computing Environments," no. 7, 2019.

[39] M. Kumar and Suman, "Hybrid Cuckoo Search Algorithm for Scheduling in Cloud Computing," *Computers, Materials and Continua*, vol. 71, no. 1, pp. 1641–1660, 2022.