www.irbas.academyirmbr.com
International Review of Basic and Applied Sciences
April 2023
Vol. 11 Issue.2

Edge Node to Edge Node Offloading for Resource Constrained Devices using SciKit-Fuzzy Logic Methodology

MUHAMMAD ALTAF
Computer Science Department, COMSATS University Islamabad Vehari Campus, Vehari, Pakistan.

MUHAMMAD MUDASSAR
(Corresponding Author)
Computer Science Department, COMSATS University Islamabad Vehari Campus, Vehari, Pakistan.
Email: muhammad.mudassar@cuivehari.edu.pk

MUHAMMAD ASIM RAIZ
Computer Science Department, COMSATS University Islamabad Vehari Campus, Vehari, Pakistan.

Abstract

The Internet of Things (IoT) is a network of interconnected physical things/devices used to communicate with each other connected with the internet. Its major applications are home automation, smart wearable devices, smartphones, and industrial machinery that generate an enormous amount of data. This generated data is sent to edge computing for processing and later use. Cloud computing is a computing paradigm used widely to sell services over the internet on demand according to the need of the user to complete their tasks. Edge computing provides storage and processing capabilities, and it is much more useful in real-time applications like video games, and smart vehicles that need real-time decisions without any delay. Edge nodes are limited in terms of available resources like memory, processing, and battery. So, it requires a mechanism that can complete the execution of the tasks by offloading their tasks to another edge device connected in the network. The resources are identified by fuzzy logic that can reach their maximum level in case of CPU and memory utilization. In the case of battery life, the minimum level is identified and needs another node in the edge computing for the rest of the execution of the tasks. All the devices connected with a network in edge computing also share a server or a local memory that is used to save the current state of the running application or task before offloading so that the task can be resumed. The fuzzy logic system identifies the edge device resource that needs to be offloaded to another edge device to complete its execution. Since the results are generated by the fuzzy logic the overloaded node starts to find the nearby edge node and creates a connection by using a shared memory mechanism. When a connection is established, the sender edge node creates an image of the process and sends the receiver edge node, so that the receiver edge node can start execution of the process from where it is stopped by the sender edge node.

Keywords: *Edge Computing, Offloading, Fuzzy Logic, Internet of Things, Reliability.*

Introduction

The Internet of Things (IoT) is a network of interconnected physical things/devices used to communicate with each other connected with the internet. It has also made people's lives automated and saved precious time (Mu & Zhong, 2020). Edge computing provides services to the end users to reduce latency in terms of time and increase the performance of the system (Mudassar & Lejian, 2022). When a resource reaches its

maximum capacity, it needs to offload its computation-intensive tasks to nearby edge devices. Check-pointing is a method that saves a device resource's state to some storage, allowing it to be further restarted from where it is stopped on other devices connected with the same network (Mudassar et al., 2019). The edge node saves the state of the current running or executing process so that it can be reinstated on the other edge node. Cloud computing is a computing paradigm used widely to sell services over the internet on demand according to the need of the user to complete their tasks. It provides scalability, cost savings, data security and backup.

Edge computing provides services to the end users to reduce latency in terms of time and increase the performance of the system (Mudassar & Lejian, 2022). When a resource reaches its maximum capacity, it needs to offload its computation-intensive tasks to nearby edge devices. Checkpointing is a process that saves a resource's state to some storage, allowing it to be further processed in the form of scaling out a resource or restarting a failed resource from that stored state (Mudassar et al., 2019). The edge node saves the state of the current running or executing process so that it can be continued on the other edge node shown in Figure 1.

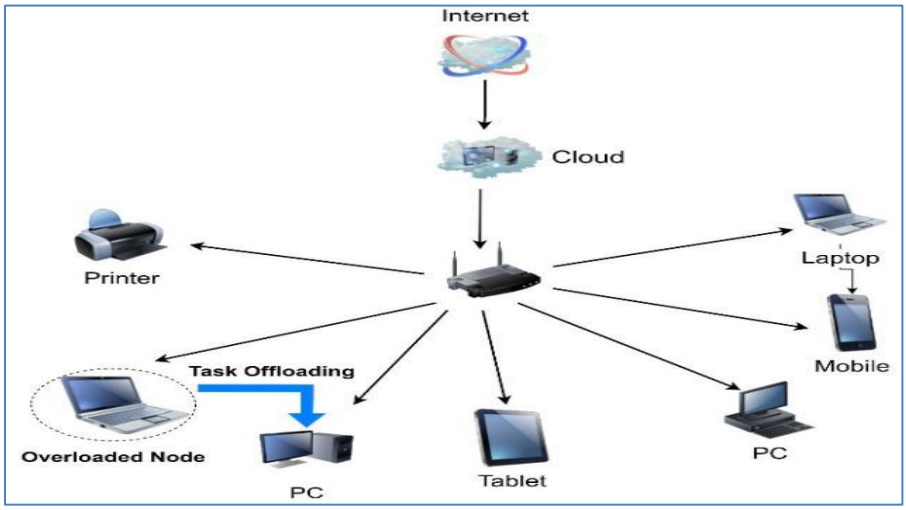

Figure 1: Task offloading from edge to edge

Fuzzy logic is a technique used to identify the overloaded node or resource that is reached its maximum capacity (Hossain et al., 2020). It has already been widely used for congestion control, handoff management, network, and workload-management issues. It is based on degrees of truth rather than the usual true or false Boolean logic. Its value can be 0, 1, or in between 0 or 1 i.e., 0.8 or 0.77. In Boolean, we can say a glass of cold water or a glass of hot water but in fuzzy logic, we can say a glass of warm water (neither cold nor hot). Fuzzy logic consists of Fuzzification, Inference Rules, and Defuzzification (Nguyen et al., 2020).

Task offloading is a process used to assign computation-intensive task execution from one edge node to another edge node at the edge of the network. The performance of the whole system increases by the task offloading (Feng et al., 2021). The fuzzy logic system identifies the edge device resource that needs to be offloaded to another edge device to complete its execution. Since the results are generated by the fuzzy logic the overloaded node starts to find the nearby edge node and creates a connection by using a shared memory mechanism. When a connection is established, the sender edge node creates an image of the process and sends the receiver edge node, so that the receiver edge node can start execution of the process from where it is stopped by the sender edge node.

In this research, a fuzzy logic-based offloading model is proposed to reduce latency in terms of time and increase performance in terms of efficient resource utilization. Smart devices are monitored by the fuzzy logic manager. When a device is identified as overloaded its current execution state is stored in a local storage and the tasks are offloaded to another edge node to complete their execution without any distraction. The fuzzy logic system identifies the edge device resource that needs to be offloaded to another edge device to complete its execution. Since the results are generated by the fuzzy logic the overloaded node starts to find the nearby edge node and creates a connection by using a shared memory mechanism. When a connection is established, the sender edge node creates an image of the process and sends the receiver edge node, so that the receiver edge node can start execution of the process from where it is stopped by the sender edge node. The key contributions are as following:

- Measure resource utilization of the edge nodes executing IoT tasks using SciKit-Fuzzy logic.
- Efficiently offloading a latency-sensitive task from one edge node to another.

Background and Basic Concepts

Nowadays, Edge Computing is a promising model based on heterogeneous computing resources located at the periphery of the network, close to the end user. It enhances edge network characteristics regarding efficient communication, data rates, storage capacity, and powerful computing. As a result, applications with latency-sensitive and computation-intensive tasks rely on the highest resources to unload and complete these missions. Basic et al. (2021) presented a fuzzy logic-based selection of optimal nodes to offload based on processing, bandwidth, and latency parameters. The results showed a significantly reduced application response time and the cost made in money, controlling the quantity switch between edge nodes. Mu and Zhong (2020) proposed a model in which the computing resources of IoT devices, opportunistic idle collaborators, and dedicated edge cloud are fully exploited. Calculation/ offloading assistance is provided by collaborators in idle/busy states, respectively. The problem is formulated as a finite horizon Markov decision problem to minimize the total expected energy consumption of IoT devices and collaborators, subject to strict compute deadlines limit. The offloading and resource allocation methodology to improve the system utility and computation time are designed. In their model Mobile Edge Computing (MEC) is insufficient to complete system demands. They have offloaded services into automobiles in Vehicular Networks (Zhao et al., 2019).

An approach is designed to offload streams of applications from one node to another fog node or cloud node (Liu et al., 2018). Furthermore, the non-orthogonal multiple access (NOMA) approach is used to reduce latency and many users can send data to the same fog node (FN) for necessary offloading processes in the same spectrum ways. Kim (2020) stated two methods Tempered Aspirations Bargaining Solution (TABS) and Gupta-Livne Bargaining Solution (GLBS) for task offloading in fog, and cloud computing. The first one is for time-sensitive offloading services and the second method is for computation-based offloading services. Performance evaluations are presented to demonstrate the feasibility of the proposed scheme and demonstrate superior performance over the existing model.

To save the state of the edge node Karhula et al. (2019) explained the technique named Checkpointing saves the logs and states of processes executing in the constraint devices. This technique is useful in IoT fault-tolerant systems and resource offloading. They have implemented checkpoint/restore in user space (CRIU) together with Docker to preserve the function states through checkpointing.

A fuzzy logic-based mobile edge orchestrator (MEO) offloading model to enhance the performance of IoT-based systems. It performs better in augmented reality, healthcare, compute-intensive, and infotainment applications in (Nguyen et al., 2020) and (Almutairi & Aldossary 2021). They have also presented that packet sizes affect the overall offloading performance of the system. The model of task offloading

technique that works on virtual machines based on location services in (Son & Lee 2017). They have implemented their offloading technique in fog computing to reduce latency and increase the performance of the virtual machines deployed on mobile devices. A fuzzy decision engine that considers both mobile and cloud variables. Cloud parameters and rules can be deployed to mobile devices using a notification service.

Table1: Comparison of offloading in the edge computing

Ref	Considering available resources	Device Heterogeneity	Latency Consideration	Offloading Task/Code/Services	Fuzzy Logic	Cloud/ Edge/ Fog	Remarks
Mu, S., & Zhong, Z. (2020)	Yes	No	Yes	Code Offloading	No	Edge	It works on limited IoT devices but needs to work on more general scenarios
H. Flores and S. Srirama (2013)	Yes	No	No	Code	Yes	Cloud	Code offloading may fail due to decision-based given information
Nguyen (2020)	Yes	No	Yes	Task	Yes	Edge/ Cloud	The solution is limited to these 4 domains, augmented reality, healthcare, compute-intensive, and infotainment applications
S. Deng, L. Huang (2014)	Yes	Yes	No	Services	No	Edge	Unknown locations create problems for this model.
J. Zhao, Q. Li, Y. Gong (2019)	Yes	Yes	Yes	Services	No	Mobile Edge	Mobile Edge Computing (MEC) is insufficient to complete system demands.
Y. Liu (2018)	Yes	Yes	Yes	Task	Yes	Fog	They have not considered energy consumption in an efficient way
S. Kim (2020)	No	Yes	Yes	Task	No	Fog	Privacy issues and device mobility is not considered
Y. Son and Y. Lee (2017)	No	No	No	Services	No	Fog	Latency is not considered
J. Yadav (2022)	Yes	Yes	Yes	Task	Yes	Fog/Cloud	Worked on whether to offload tasks or not in the fog computing
Y. Zhai and W. Xu (2017)	Yes	No	Yes	No	Yes	No	Calculates over the node's key performance parameters in stream processing systems
Our	**Yes**	**Yes**	**Yes**	**Task**	**Yes**	**Edge**	**As in related work, most research work is not considering latency and fuzzy logic so, our proposed work is on these parameters**

They have proposed a strategy for enhancing offloaded decision-making processes with evidence-based learning methods. Resources are also scaled out and in on the user's demand (Flores & Srirama 2013). A scaling technique that can be applied to the demand of the customer. It optimizes throughput after scaling, and it minimizes disruption of calculating while the scaling operation is running outside (Xu et al., 2016). An offloading model that decides dynamically whether to offload a task or not based on these variables'

residual energy, delay sensitivity, CPU usage, and bandwidth. Their proposed algorithm takes decisions by sending real-time applications to fog nodes and resource-intensive tasks to the cloud (Yadav & Sangwan 2022). Offloading decisions on mobile services based on a genetic algorithm (GA). Device mobility and the heterogeneous network, and unstable connectivity have a great impact on the offloading decisions. They have considered the user's location while executing processes (Deng et al., 2014). To overcome the issues of latency, an offloading methodology is proposed in multiple fog nodes to achieve low task delay and energy efficiency (Zhang et al., 2018).

The computation offloading model with the Dhrystone Millions of Instructions per Second (DMIPS) based workload measurement model. In addition, they have proposed the offloading decision model. They have evaluated that offloading from wearable smart devices to the cloud is energy efficient (Ko et al., 2021). To reduce the energy consumption of smart IoT devices Shan et al. (2018) presented the model which satisfies the delay constraints. They have considered the transmission problem while deciding on the offloading. A two-step approach that breaks down and builds up is used in this model. To improve the user experience and resource utilization in Zhang et al. (2021) have proposed the model of task offloading in multi-user and edge node resource control. This method has improved the user experience and resource utilization. It has reduced the complexity of the algorithm as compared to another proposed centralized model in a multiuser paradigm.

One of the offloading problems is dependent tasks in multi-user systems. A heuristic algorithm is used to tackle the problems of dependent tasks in task offloading (Fan et al., 2019). Their work is on task offloading from smartphones to the cloud. All the dependent tasks are added to the priority queue and tasks are reassigned based on the relative energy of the resources. This system reduces cost and energy consumption. The architecture of latency sensitivity is partitioned to offload by considering many parameters in the fuzzy inference system to make decisions more precise and accurate in the mobile edge paradigm. They have worked on latency-sensitive applications with huge sizes of applications (Shi et al., 2022). The process of offloading architecture and different intensity levels are explained in (Yousafzai et al., 2019). When the workload is increased, the time and energy intensities also become more efficient. It can fail when the mobile device is used to perform local execution. They worked on edge computing to reduce the latency. To take immediate decisions in the wireless traffic system, they have worked on device-to-device communication to direct communication with the end devices. Simulation results showed that their model has improved the efficiency of task offloading as compared to traditional models in (Wen et al., 2018), and (Li et al., 2020).

In Table 1, we discuss some of the most relevant research papers that focus on key aspects such as resource management, device heterogeneity, latency, types of offloading, fuzzy logic, and their relationship with networking paradigms like edge, fog, and cloud computing.

In our research, we present our proposed solution that utilizes fuzzy logic to address the task offloading problem. Our solution focuses on monitoring the available resources of constrained devices and deciding on whether to offload the task to another edge device or execute it locally. The proposed approach is designed to be adaptable to different environments, edge nodes, and latency-sensitive tasks.

Methodology

This section concentrates on methodology we used for this research. When an edge node is going to be overloaded, its tasks can be shifted from one node to another node connected in the edge computing network. Before shifting tasks to another node there should be checkpointing. Figure 2 shows the complete process of task offloading in edge computing in which all the edge nodes are monitored by fuzzy logic. IoT

applications like smart cameras and smart home stations are running low on resources monitored by fuzzy logic techniques, and they need to continue their execution.

So, they find the nearby edge node and offload their running tasks to complete their execution. In edge computing, nodes are locally connected to a network. In the edge computing network, smartphones, tablets, laptops, personal computers, and mobile is the edge node present locally near the device that needs to be offloaded. Before task offloading, the system identifies whether offloading is beneficial or not, if it decreases the performance of the system then it is not recommended to offload running tasks. They are useful in real-world applications due to their low latency.

IoT devices are running their applications and overloaded resources are identified by the fuzzy logic model. Fuzzy logic works on inference rules based on the parameters defined in the system. When an overloaded edge node reaches its maximum level, it needs to offload its task. So, the task offloading request is forwarded from the IoT device to the fuzzy logic handler. It ensures the availability of the edge node connected to the network. Checking the available nodes is a necessary part of the system as it yields the performance of the system. It checks if the available resources or edge nodes to offload the task are beneficial or not. If the offloading of the specific task increases the performance, then the system initializes the process of task offloading and provides resources for task execution.

Checkpointing saves all the data, contents, logs, and state of the process running before the task offloading process. By the checkpointing technique, the status of the application or process can be saved, and it can resume its execution from the current state rather than starting from the initial state. When a task needs to be offload, the local edge node saves the state of the task on the shared server that is directly connected to the edge node that resume the execution of the task from where it was stopped.

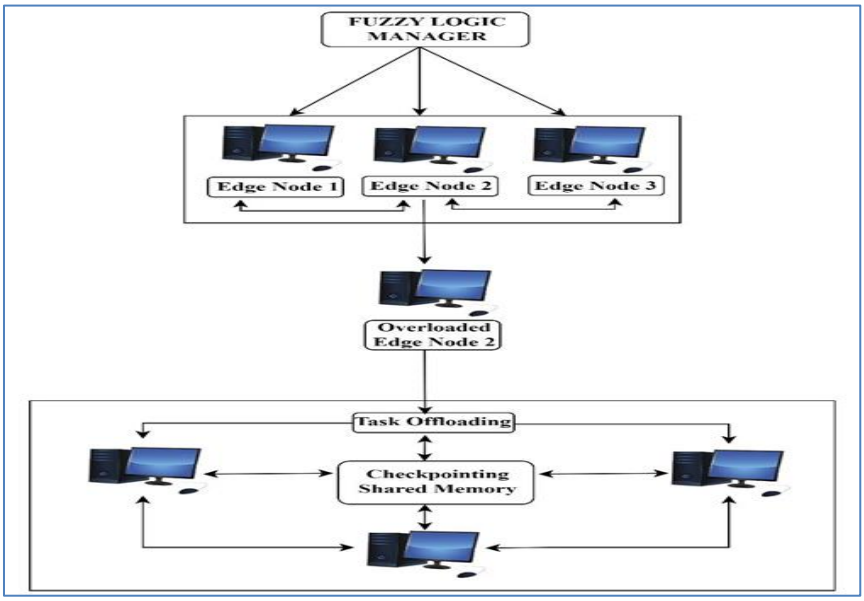

Figure 2: Architecture of the fuzzy logic-based task offloading

A. Components of the Proposed Model

In following different components are discussed of the proposed system.

1. Fuzzy logic manager

This is a component of the fuzzy logic module that manages the whole process of monitoring an edge node resource. This is responsible for declaring any edge node as overloaded and the task can be offloaded based on the results of this component called fuzzy logic manager.

2. Checkpointing mechanism

This is a local memory, or a server shared with all the edge nodes connected to a network. This component is responsible for resuming the task from where it is stopped. This increases the performance of the system as it saves the state or statistics of the running task that must be offloaded to another edge node.

3. Task offloading manager

It is responsible for task offloading from one edge node to another connected in a network. This component is directly connected to the shared server from where it can get the previous state of the task that should be offloaded from the sender edge node to the receiver edge node.

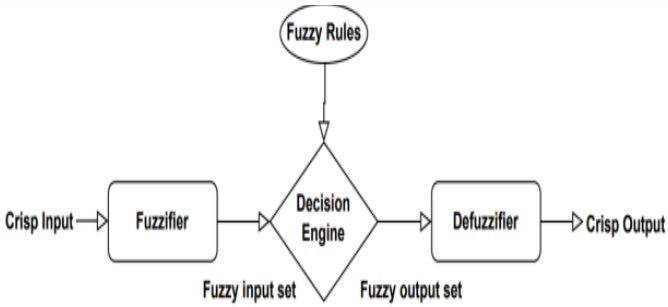

Figure 3: Fuzzy system used for the system

Fuzzy Logic System

Fuzzy system is the process of converting crisp input to a fuzzy membership function. Figure 3 shows the complete workflow of the fuzzy logic system. Fuzzifier converts crisp input to a fuzzy input set. Fuzzy input sets are supplied to the decision engine to make a conclusion that is based on a set of rules. The resultant fuzzy output sets are then mapped to crisp sets again (defuzzification), enabling the best result in crisp output to be obtained.

Organizing linguistic variables and membership function is a difficult task. It can be created from daily life activities. The membership function is the degree that defines the number of variables exactly. In this research, CPU utilization, memory, and battery have three values, which are given in Table 2.

Table 2: Fuzzy logic parameters

Parameters	**Linguistic Variables**
CPU	Low, Medium, and High
Memory	Low, Medium, and High
Battery	Low, Medium, and High

1. **Membership Function**

The membership function for CPU utilization have the values (0-35) show the Low linguistic variable, (30-70) is Medium, and (70 to 100) is the High linguistic variable. The membership function for memory utilization the values (0-35) show the Low linguistic variable, (30-70) is Medium, and (60 to 100) is the High linguistic variable. The membership function for battery life have values (0-40) show the Low linguistic variable, (30-60) is Medium, and (60 to 100) is the High linguistic variable. These are shown in figure 3.

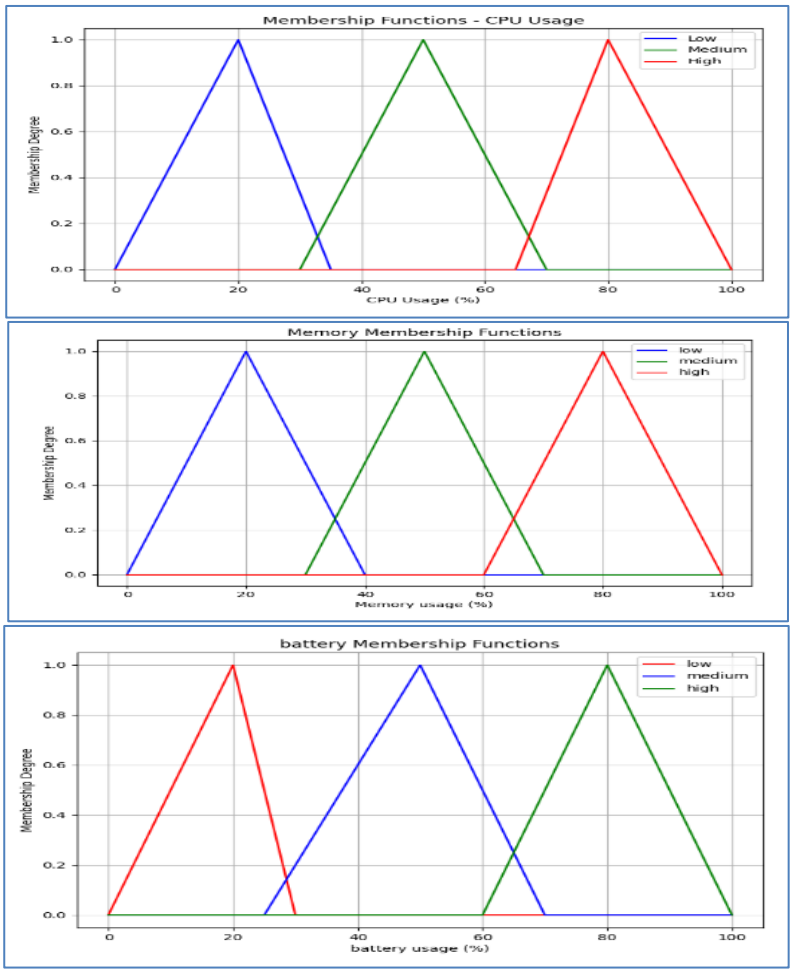

Figure 4: Membership functions for CPU, Memory and Battery

1. **Rules for battery**

There are two possibilities (Offload, Keep) present because of the inference rule. For example, If CPU utilization = High and battery = Low then Status = Offload. Similarly, If CPU utilization = Medium and battery = Medium then Status = Keep.

2. Rules for Memory

If CPU utilization = High and Memory = Low then Status = Offload. Similarly, the second rule is If CPU utilization = Medium and memory = Low then Status = Keep.

The set of all rules are given below:

Table 3: Fuzzy rules

CPU High	Battery is Low	Offload
CPU Medium	Battery is Low	Offload
CPU Low	Battery is Low	Offload
CPU High	Battery is Medium	Offload
CPU Medium	Battery is Medium	Keep
CPU Low	Battery is Medium	Keep
CPU High	Battery is High	Offload
CPU Medium	Battery is High	Keep
CPU Medium	Battery is High	Keep
CPU High	Memory is Low	Offload
CPU Medium	Memory is Low	Keep
CPU Low	Memory is Low	Keep
CPU High	Memory is Medium	Offload
CPU Medium	Memory is Medium	Keep
CPU Low	Memory is Medium	Keep
CPU High	Memory is High	Offload
CPU Medium	Memory is High	Offload
CPU Medium	Memory is High	Offload

3. Defuzzification

It is the process of converting fuzzy sets into crisp output. Its output value lies between 0 and 100. A final decision is based on these crisp values generated from the defuzzification process. After finding the overloaded resources that reach their maximum capacity, in the system, computationally intensive tasks that are necessary to be completed within a specific time frame are offloaded to nearby edge devices.

Results and Discussions

This section focuses is on presenting the outcomes of the conducted experiments or simulations to assess the performance and effectiveness of the proposed task offloading technique based on fuzzy logic.

A. Fuzzy System Results

When the dataset of CPU, Memory and Battery are passed to the fuzzy inference system, it gives the output of the system as to whether to offload the task from this running-out resource to another device or should continue the execution on this device. Some input values and their offload values that are tested in MATLAB SciKit-Fuzzy logic technique are presented in provided in the following table and figures as well.

Table 4: Input values for the fuzzy vriables to the fuzzy system

CPU Utilization (%)	Memory Utilization (%)	Battery Life ((%))	Offload Decision
84	09	07	86
13	83	60	86
13	15	07	86
16	70	76	79
20	20	20	30
…..	……	……	…….
N	n	n	N

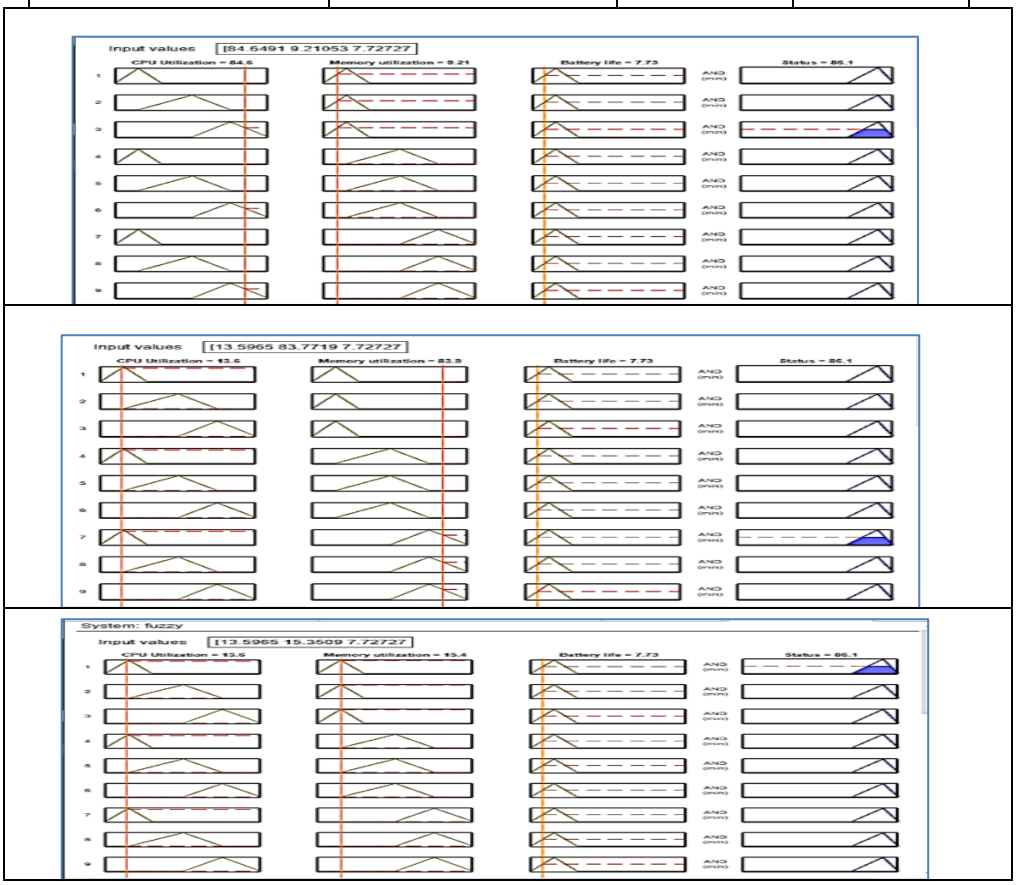

Figure 5: Matlab Results for fuzzy system decision for offloading

These outputs shows that whenever the processing resources like memory and CPU are approaching to the maximum and the battery is low the status suggested an offload. This is the decision making point where the resources of the edge node are not more capable to execute the task hence it is required to offload this task to some other node in the edge network.

B. Task Offloading and System Efficiency

The fuzzy logic system identifies the edge device resource that needs to be offloaded to another edge device to complete its execution. Since the results are generated by the fuzzy logic the overloaded node starts to find the nearby edge node and creates a connection by using a shared memory mechanism. When a

connection is established, the sender edge node creates an image of the process and sends the receiver edge node, so that the receiver edge node can start execution of the process from where it is stopped by the sender edge node.

A Word-Count application is executed on the edge node with huge data set and once the application executes the data the resources are consumed and at a stage the offloading is suggested. We have tested from 50K words to 400 million words. And the results are compared with and without offloading.

In Figure 6 a complete comparison is shown between task completion with offloading and without offloading with the increasing number of words in the text file. While we go through without offloading constraint resources can execute the task to 400 million words after that it cannot be further execution performed. In the case of offloading the number of words till 1B (1000M) is calculated and the CPU is not hanged as it was hanged when executing on the constrained edge node. This increases the efficiency and performance of the system.

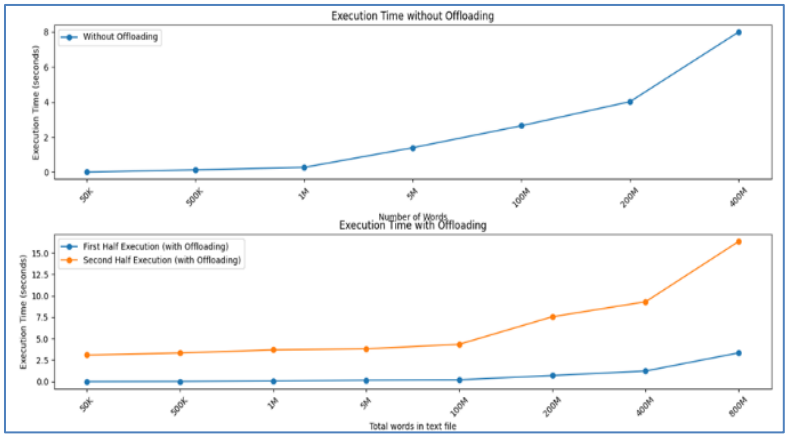

Figure 6: Comparison of task completion with offloading and without offloading

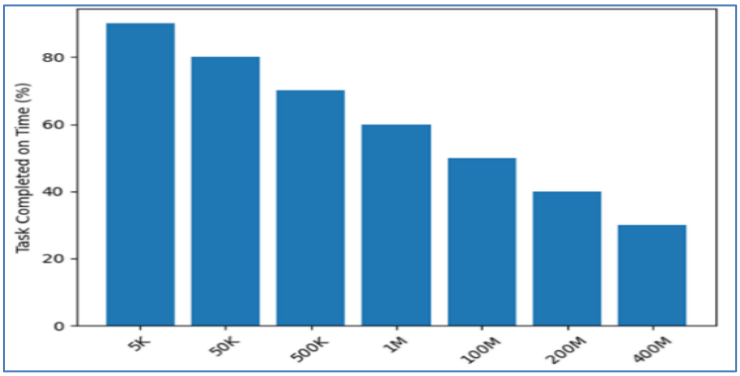

Figure 7: Task completion with size of data, more data less task completion and increasing the failure

The figure 7 shows the task completion when the number of words are increased. It is clear that with small size of file in terms of number of words the task is completed successfully, but when the file size increases by increasing the number of words the percentage of task completion decreases. This fact makes it clear that when the large data is processed on the smart devices these tend to fail before the completion. this matter can be resolved by providing the offloading methodology. And when the offloading methodology is

provided in the system the task completion increases as this is clear from the figure 7 that the application latency is decreased in presence of the offloading mechanism.

Latency is what when application is getting delayed from its due time. The figure 8 shows that when the offloading is provided in the edge network the application latency is decreased. This is because when the edge node goes overloaded in terms of the resources the as the data size is increased single edge node will fail and the latency will be increased as the task restarts again and again. But when the offloading is provided the tasks are offloaded to the other nodes hence completing the tasks well in time.

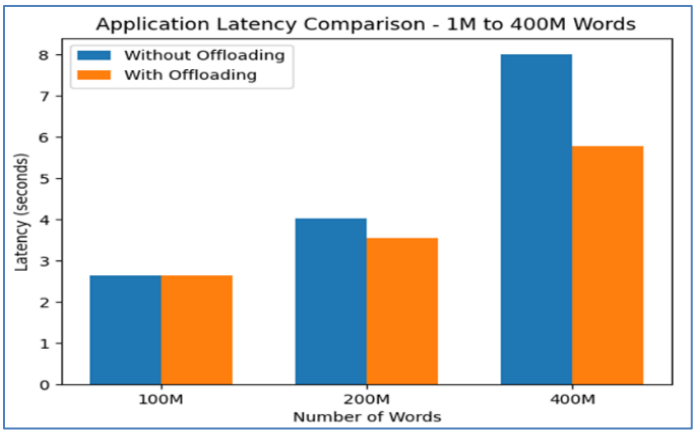

Figure 8: Latency comparison

The results make it clear that the system is working and node resources utilization is successfully detected by the fuzzy logic system and once the offloading is supported in the system the application executes smoothly.

Conclusion

In this research, a fuzzy logic-based system is proposed that monitors the constraint resources of smart devices. It decides whether to offload a running task or not based on the fuzzy logic inference system. In this research, there are three input parameters of the fuzzy logic system, CPU utilization, memory utilization, and battery life. These values are taken from the real-time values of a device. When these input values are passed to the fuzzy logic system it generates a value of offload decision. When a device is identified as a bottleneck by the fuzzy logic system, the task executing on the edge node is offloaded to some other edge node. This will keep the task executing and hence results in successful completion of the application. The results make it clear that the proposed methodology is working and the overall latency of the application can be decreased. This fact will help the IoT applications to be executed in the edge network on the edge nods.

References

Almutairi, J., & Aldossary, M. (2021). A novel approach for IoT tasks offloading in edge-cloud environments. *Journal of cloud computing*, *10*(1), 28.

Basic, F., Aral, A., & Brandic, I. (2019, June). Fuzzy handoff control in edge offloading. In *2019 IEEE International Conference on Fog Computing (ICFC)* (pp. 87-96). IEEE.

Deng, S., Huang, L., Taheri, J., & Zomaya, A. Y. (2014). Computation offloading for service workflow in mobile cloud computing. *IEEE transactions on parallel and distributed systems*, *26*(12), 3317-3329..

Fan, Y., Zhai, L., & Wang, H. (2019). Cost-efficient dependent task offloading for multiusers. *IEEE Access*, *7*, 115843-115856.

Feng, S., Chen, Y., Zhai, Q., Huang, M., & Shu, F. (2021). Optimizing computation offloading strategy in mobile edge computing based on swarm intelligence algorithms. *EURASIP Journal on Advances in Signal Processing*, *2021*, 1-15.

Flores, H., & Srirama, S. (2013, June). Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning. In *Proceeding of the fourth ACM workshop on Mobile cloud computing and services* (pp. 9-16).

Hossain, M. D., Sultana, T., Nguyen, V., Rahman, W. U., Nguyen, T. D., Huynh, L. N., & Huh, E. N. (2020). Fuzzy based collaborative task offloading scheme in the densely deployed small-cell networks with multi-access edge computing. *Applied Sciences*, *10*(9), 3115.

Karhula, P., Janak, J., & Schulzrinne, H. (2019, March). Checkpointing and migration of IoT edge functions. In *Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking* (pp. 60-65).

Kim, S. (2020). New application task offloading algorithms for edge, fog, and cloud computing paradigms. *Wireless Communications and Mobile Computing*, *2020*(1), 8888074.

Ko, J., Choi, Y. J., & Paul, R. (2021). Computation offloading technique for energy efficiency of smart devices. *Journal of Cloud Computing*, *10*(1), 44.

Li, J., Dai, M., & Su, Z. (2020). Energy-aware task offloading in the Internet of Things. *IEEE Wireless Communications*, *27*(5), 112-117.

Liu, Y., Yu, F. R., Li, X., Ji, H., & Leung, V. C. (2018). Distributed resource allocation and computation offloading in fog and cloud networks with non-orthogonal multiple access. *IEEE Transactions on Vehicular Technology*, *67*(12), 12137-12151.

Mu, S., & Zhong, Z. (2020). Computation offloading to edge cloud and dynamically resource-sharing collaborators in Internet of Things. *EURASIP Journal on Wireless Communications and Networking*, *2020*, 1-21.

Mudassar, M., Zhai, Y., & Lejian, L. (2022). Adaptive fault-tolerant strategy for latency-aware IoT application executing in edge computing environment. *IEEE Internet of Things Journal*, *9*(15), 13250-13262..

Mudassar, M., Zhai, Y., & Liao, L. (2019). Efficient state management for scaling out stateful operators in stream processing systems. *Big data*, *7*(3), 192-206.

Nguyen, V., Khanh, T. T., Nguyen, T. D., Hong, C. S., & Huh, E. N. (2020). Flexible computation offloading in a fuzzy-based mobile edge orchestrator for IoT applications. *Journal of Cloud Computing*, *9*, 1-18.

Shan, F., Luo, J., Jin, J., & Wu, W. (2018). Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless IoT environment. *IEEE Internet of Things Journal*, *6*(3), 4411-4422.

Shi, Y., Chu, J., Ji, C., Li, J., & Ning, S. (2022). A fuzzy-based mobile edge architecture for latency-sensitive and heavy-task applications. *Symmetry*, *14*(8), 1667.

Son, Y., & Lee, Y. (2017). Offloading Method for Efficient Use of Local Computational Resources in Mobile Location-Based Services Using Clouds. *Mobile Information Systems*, *2017*(1), 1856329.

Wen, J., Ren, C., & Sangaiah, A. K. (2018). Energy-efficient device-to-device edge computing network: An approach offloading both traffic and computation. *IEEE communications magazine*, *56*(9), 96-102.

Xu, L., Peng, B., & Gupta, I. (2016, April). Stela: Enabling stream processing systems to scale-in and scale-out on-demand. In *2016 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 22-31). IEEE..

Yadav, J. & Sangwan, S. (2022) "Dynamic Offloading Framework in Fog Computing," International Journal of Engineering Trends and Technology. 70(7), 32–42.

Yousafzai, A., Yaqoob, I., Imran, M., Gani, A., & Noor, R. M. (2019). Process migration-based computational offloading framework for IoT-supported mobile edge/cloud computing. *IEEE internet of things journal*, *7*(5), 4171-4182.

Zhang, G., Shen, F., Yang, Y., Qian, H., & Yao, W. (2018, May). Fair task offloading among fog nodes in fog computing networks. In *2018 IEEE international conference on communications (ICC)* (pp. 1-6). IEEE.

Zhang, Z., Li, C., Peng, S., & Pei, X. (2021). A new task offloading algorithm in edge computing. *EURASIP Journal on Wireless Communications and Networking*, *2021*(1), 17.

Zhao, J., Li, Q., Gong, Y., & Zhang, K. (2019). Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks. *IEEE Transactions on Vehicular Technology*, *68*(8), 7944-7956.