www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 16(3), May-Aug 2025, 391-404. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2025.v16i3.843

Task Scheduling Algorithm to Reduce response Time Over Fog Computing Environment

Ayalew Belay Habtie [1], Temesgen Teshome [2]
[1] Addis Ababa University. Department of Computer Science.
[2] Addis Ababa Science and Technology University. Department of Software Engineering

E-mails: ayalew.belay@aau.edu.et, temesgenswe@gmail.com

Abstract. Fog computing is a new thrilling extension of cloud computing that provides computational, storage, and communication resources nearer to the network edge where end devices reside. This is especially important now with the huge growth of Internet of Things (IoT) devices that generate a huge range of task requests. Reliance only on cloud infrastructure to perform such tasks will lead to system overload and deterioration of the response required for time-critical applications. To prevent such issues, Cisco introduced fog computing in 2012 as a supplement to cloud computing, not a substitute. One of the biggest challenges of this model is the efficient distribution of tasks to corresponding resources such that latency is reduced and system throughput is increased. To address this problem, we developed a four-layered fog computing model after reflecting on findings from an exhaustive review of literature. Priority-based task scheduling algorithm that optimizes response time by dividing tasks into three categories—simple, medium, and large/complex forms the basis of our suggested model. Tasks are then dispatched depending on their priority: low and medium tasks are processed at the nearest fog node, and larger tasks are sent to the aggregate fog layer. The performance of this scheduling algorithm was evaluated with a simulation tool selected through a systematic review of available simulators. The algorithm's performance was tested on ten independent sets of tasks with varying sizes, which were distributed to various layers in the fog infrastructure. Shortest Job First (SJF), one of the most renowned scheduling algorithms found in the literature, was implemented for the comparison. Results indicate that smaller tasks below 4,280 MB ought to be assigned to the Nearest Fog Node (NFN) for enhanced performance and larger tasks are preferable to process in the Aggregate Fog Node (AFN). In addition, a minimum task size of 50 MB is determined, suggesting smaller tasks may not require special scheduling. The outcome indicates that the suggested algorithm significantly reduces the response time of tasks by effectively dealing with the distribution of simple and medium tasks to the nearest fog node and big tasks to the aggregate node, outperforming SJF.
Keywords: Fog Computing, Task Scheduling, Priority-Based Algorithm, Response Time

Article Info
Received January 31, 2025
Accepted April 14, 2025

1 Introduction

The distributed computing scenario has evolved along different paradigms during past decades. The 1960s witnessed utility computing as the initial appearance of distributed computing, which was then followed by cluster computing in a later stage. In the early 1990s, grid computing became entrenched as a new paradigm in distributed systems. Its place was later taken by cloud computing emerging on the scene around the early 2000s. In recent times, the shortcomings of the central model have necessitated the evolution of mainframe and cloud-based fog computing (Nath, S. B., et al., 2018).

Cloud computing brings a major benefit to service providers and consumers. Consumers have less expense, administrative burden removed, increased flexibility, increased reliability, and the capacity to access from almost anywhere (Firdhous, M., et al., 2014). However, it has some drawbacks. These include reliance on constant connectivity, multihoming problems, high

latency, unspecified security measures, lack of location-awareness, centralized architecture, increased network traffic, and large bandwidth consumption (Dastjerdi, A. V., et al., 2016).

To address these issues, fog computing has been proposed as a middle layer between cloud data centers and end-user devices. Cisco initially proposed fog computing, which aims to reduce latency, enable location-awareness, support mobility, and alleviate traffic jam (Liu, Y., et al., 2017). Fog computing accomplishes this by bringing resources closer to end users through geographically distributed nodes. It supports mobility, enhances quality of service, and is particularly well-suited for real-time analytics and local processing. Unlike cloud data centers, fog nodes are likely to be heterogeneous in capability and execute in resource-constrained environments.

As the adoption of IoT solutions is getting increasingly faster, scheduling tasks in the fog domain is now a pressing research topic (Gao, L., & Moh, M., 2018). Proper scheduling plays a crucial role in providing fast response and optimizing overall quality of service. In spite of that, current schedulers fall behind and do not adequately handle and schedule tasks from IoT devices onto suitable resources between different levels of the fog infrastructure (Sarddar, D., et al., 2018; Choudhari et al., 2018). Tasks are mapped in accordance with VM's in terms of not considering task type and task priority.

For proper scheduling, the tasks need to be classified for examination because of task complexity and capacity of the resource in different layer of the fog computing architecture. Selecting or determining which type of task is scheduled in the closest fog node or aggregate fog node is the problem to be addressed in this research. Because tasks are transferable across different physical equipment, task scheduling becomes considerably complex. In fog computing systems, resource management must dynamically consider the optimal location for task execution, i.e., whether tasks should be executed at the aggregate fog node or the cloud, with the goal of minimizing response times and maximizing throughput (Gupta, M., et al., 2017).

We suggest a priority-based task scheduling scheme where three categories of tasks are identified: simple, medium, and large. Then, these tasks are scheduled based on size along with priority. Larger tasks are sent to the aggregate fog layer, and simple and medium tasks are forwarded to the closest fog node according to their corresponding priority.

2 Related Work

Different researchers have studied various approaches to enhancing the performance of fog computing architectures, algorithms, and resource management methods. For instance, Diab Abdulgalil (2018) proposed a multitiered fog computing architecture with the purpose of processing EEG and ECG signals. This architecture consists of three layers: the first layer is responsible for data generation and communication, the second layer uses fog computing to carry out light-weight ECG signal processing, and the third layer performs intensive EEG signal processing and data storage in the cloud.

Abdelaziz (2018) proposed a collaborative fog computing structure, proposing a referential model that facilitates the design and implementation of fog platforms. The model invites abstraction, which allows development and deployment on fog nodes to be more efficient, and incorporates mechanisms to define and abstract objects, data analytics, and services. Santos et al. (2018) designed a three-layered fog computing framework, placed at the edge, near the end device, and in the cloud. They assessed the download performance of video in these independent geographical layers within a multimedia service.

Yousefpour et al. (2017) introduced an IoT-fog-cloud framework that aimed at reducing service delays of IoT applications. Their framework is based on an analytical model that has been proven to be effective in reducing IoT service delays. Hao et al. (2017) introduced a proof-of-concept system that combines IoT device communication with fog nodes in fog computing systems. Their WM-FOG framework provides a new software architecture for fog computing, and the authors tested its performance.

In order to provide the IoT service solution, Varghese, B., et al (2020) suggest a hybrid fog cloud structure for low-response time applications such as firefighting. The passage identifies different research contributions aimed at performance and efficiency improvement in fog and edge computing, particularly in IoT, 5G networks, and AI applications. Gao and Moh (2018) explored an offloading of tasks technique from mobile devices with the aim to minimize energy. They introduced an edge server-assisted priority task scheduling algorithm. Their performance metric, including execution time, execution cost, and energy consumption, was compared utilizing task data sizes and latency demand. The experiment results showed their algorithm reduced the task completion time and improved the Quality of Service (QoS).

Mohan and Kangasharru (2016) proposed a distributed task processing method in an Edge-Fog-Cloud network. They presented the Least Processing Cost First (LPCF) method which allocates the processing task to nodes in an optimal manner in order to minimize the processing time. Bittencourt et al. (2017) focused on hierarchical layer scheduling in fog and cloud computing based on three-layer architecture. In their work, various scheduling strategies for fog environments were presented where scheduling is performed in a manner that cloudlet utilization is maximized and network utilization and cloud costs are reduced. The discussed work is related to it, and it shows that the researches have developed some task scheduling algorithms but there still exist limitations like in Sarddar, D., et al., (2018) and Choudhari, et al.(2018).

In the previous work, everyone has employed three-tier architecture in the system instantiation. Also, the previously proposed solutions do not completely deal with the response time of tasks generated due to heterogeneous application including IoT devices. The majority of the scheduling issues being faced currently are directly linked to past architectures. In addition, the past works typically do not take into account task priority and size, leading to tasks being allocated randomly. The past works allocate tasks to the nearest fog node and handoff to a neighboring fog node and then employ algorithms such as First Come First Serve (FCFS) & Shortest Job First (SJF). But these methods didn´t effectively minimize the response time. That is why in this research we propose an algorithm that will minimize response time even more.

3 Fog Computing Architecture

Before the task scheduling algorithm is designed, it is crucial to understand the fog computing architecture and examine its various types in order to choose the most appropriate one for the proposed algorithm. The reason for the architecture is to illustrate how end devices communicate with fog devices and how tasks are meant to be allocated to the appropriate fog resources. Fog computing generally comprises a three-layer architecture: the bottom layer for IoT devices, the middle computing layer for fog devices, and the top layer for the cloud. In this research, however, we have employed a four-tier hierarchical fog computing architecture proposed by Joshi et al. (2018). The new architecture is more appropriate for precise and real-time monitoring of fog computing applications and offers quicker response times.

The four-layer architecture, as shown in Fig. 1, enhances the middle fog layer by dividing it into two different layers: the nearest fog layer (fog access layer) and the collective fog layer (fog control layer). This structuring enables the management and processing of tasks in different layers of the fog computing environment more efficiently. The above-stated architecture is a four-layer fog computing infrastructure with the purpose of making the processing of task requests released from IoT devices efficient. At the first level of the design, the layers encompass IoT devices producing a request for any task from its surroundings. A sample of SSNs in such IoT devices comprises different sensors along with certain actuators where the sensors always sense the changing environment and prepare the requests to be sent to the processing layer. The second and third levels are composed of fog nodes as an intermediate computing model between SSNs and cloud data centers. As shown in Fig. 1, the fog nodes are divided into two categories: Nearest Fog Node (NFN) and Aggregate Fog Node (AFN).

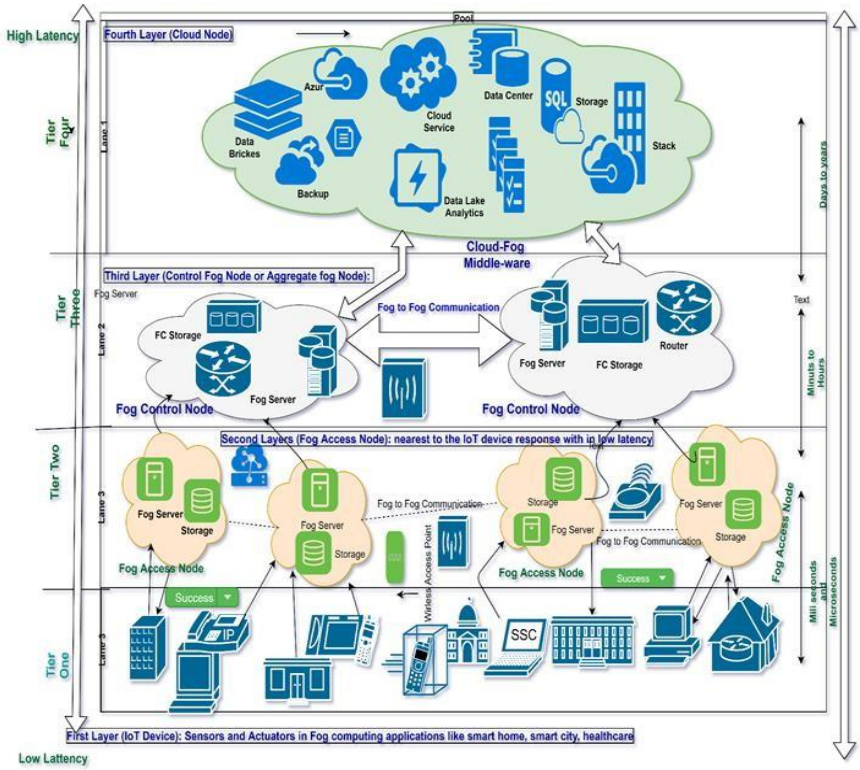

Fig .1. Four-tire architecture for fog computing adapted from Joshi, G et al (2018)

The simplest fog nodes, or Nearest Fog Nodes (NFN), are located closest to the end devices. The nodes are made up of small computing devices or edge devices. Each NFN is associated with and is responsible for an IoT device, usually a neighborhood or small group of people, and carries out task analysis on time. The output of this node is a two-part response: the latency to the terminal device after sending the analysis and reporting data to an aggregate node at the next higher layer to further analyze it. This level has limited compute and storage capacities.

The top-level nodes of fog are known as Aggregate Fog Nodes (AFNs). They constitute the third level in the hierarchy of fog. The AFNs provide resources enabling applications to perform high-speed computation and analysis. The AFNs connect directly to hundreds or tens of lower-level NFNs controlling community-level sensors. AFNs also oversee the NFNs through stable, albeit expensive, backhaul connections. The storage capacity and computing capability of AFNs are far greater than NFNs.

The fourth tier, or Cloud Computing Nodes, is the top tier in the architecture of the fog. The layer pools information and data from the middle computing nodes (levels two and three). It is characterized by tremendous processing power, massive storage, and big data computing. This four-tier fog computing architecture is designed to enable fast response with layers of nodes at various distances from the end devices: nearest, medium-range, and national levels. This architecture offers high computing power and intelligence for future fog computing applications. The fog nodes at the network edge perform data ingestion from IoT devices, which do not handle data processing themselves due to resource and energy constraints. As a result, there is minimal processing in the first tier. Computation begins in the second tier, which consists of nearest fog nodes located closest to the end devices.

The most time-sensitive data is processed at the second level, known as the Nearest Fog Node (NFN), which is located closest to the devices generating the data. Processing for less sensitive data is done in the second level at the Fog Control Node (FCN), as shown in Fig. 2. The primary function of the NFN is to enable the correct functioning of control loops and protective schemes for emergency operations. The NFN can monitor for possible issues and react by transmitting control signals to actuators. Every NFN has the necessary processing capabilities to manage the task it was given, with lower latency and faster reaction time to severe, time-critical problems.

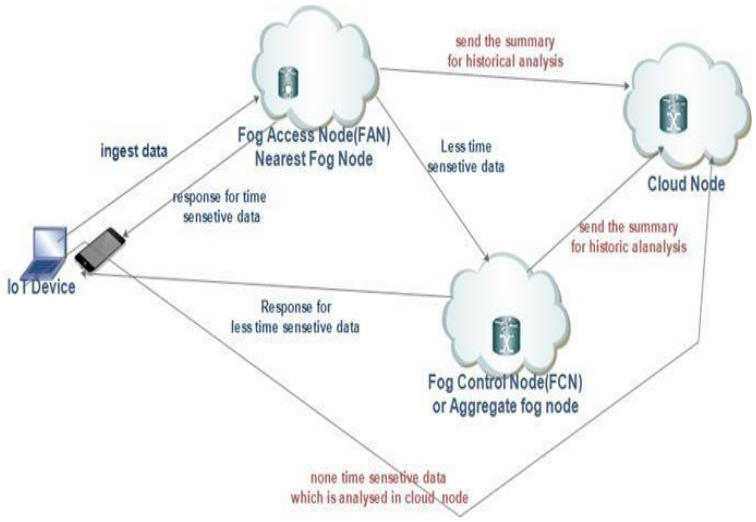

Fig. 2. Second tier processing of time sensitive and less time sensitive data

4 The proposed Task Scheduling Algorithm

For architecting the proposed algorithm, one should first know how each tier processes the task requests. When a request is initiated by an IoT device, it is first directed to the Nearest Fog Node (NFN), which acts as the intermediary between the IoT devices and the Aggregate Fog Node (AFN) of the second tier. NFN employs a data analysis algorithm in order to determine whether the task should be processed locally or be forwarded to the AFN. A filtration model in the NFN helps determine this so that the task is routed to the most appropriate environment based on the expected time duration required for it to be processed.

If the task is very short to process, i.e., microseconds or seconds, the NFN processes the request and returns the response to the IoT device. For tasks that are longer, i.e., minutes to hours, the task is forwarded to the AFN. If the processing time is greater than these, the request is forwarded to the cloud for processing. Whenever a request for data is forwarded to the Aggregate Fog Node (AFN), it receives the request from the NFN and processes the same. After processing the request by the AFN, it responds to the IoT device directly. The AFN also updates its content table, which is data about the IoT application. On processing, the AFN forwards a summary of the data to the cloud and updates its content table for maintaining proper records of the job.

When there is a request for data from an edge device or a fog device to the cloud, the cloud receives the data, processes it, and sends a response back to the concerned device. The cloud also analyzes the requests from the fog regarding updating periodically. If any updates or new rules need to be implemented at the fog end, these are sent from the cloud to the fog so that the system is synchronous and effective. As the number of edge devices grows and with their increasing functionality, they generate a fluctuating number of task requests whose durations vary from short to long. In this paper, the priority scheduling algorithm begins with a task categorization process that categorizes tasks based on duration. The categorization assists the algorithm in efficiently handling and prioritizing tasks so that resources can be allocated efficiently to meet the various demands of tasks. The task classifier separates tasks into three categories: simple, medium, and long tasks. Fig. 3 shows the model of the proposed algorithm.

The model consists of several components, and IoT devices issue a variety of task requests ranging from straightforward to intricate for execution within the fog environment. A task classifier (Algorithm 1) takes these incoming tasks and categorizes them as simple, medium, and large based on the task length. Simple and medium tasks are routed to the Nearest Fog Node (NFN) and are run based on the task priority. The scheduler ensures that the tasks are handled in an optimum way, with high-priority tasks having dominance over them. These tasks are placed in two separate queues: one for high-priority (short-length) tasks and the other for lower-priority (medium-length) tasks (Algorithm 2 & 3). However, the AFN receives big tasks, and these

are processed using the First Come, First Serve (FCFS) method. The system is made to be resource-effective and the tasks to be run based on categorization in order to promote the general performance of the fog environment.

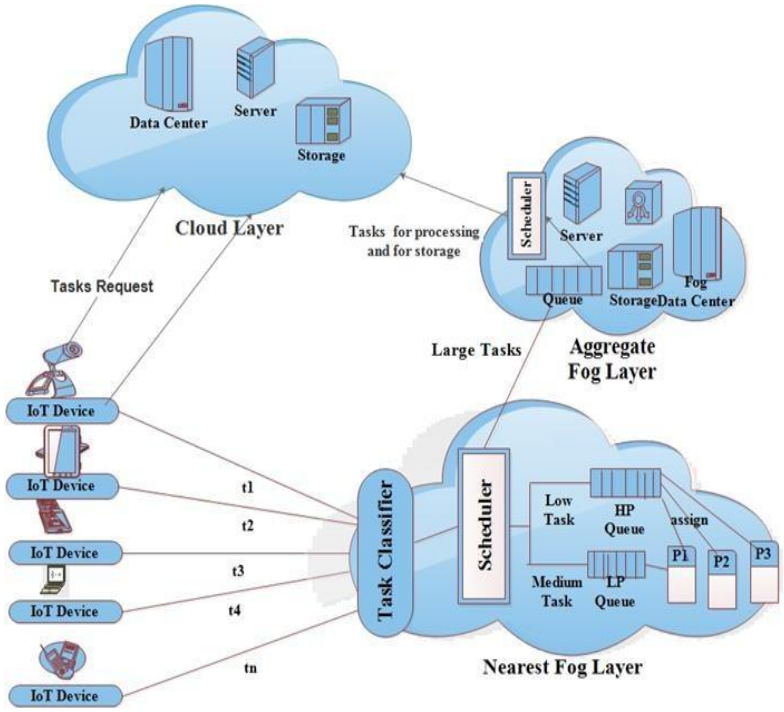

Fig. 3. Model representing the proposed task scheduling algorithm

Algorithm1. Task classifier Algorithm

Input:
- Task list with task length attribute
- Number of processors, processor speed

Output:
- Task type or task categorization

(Simple, Medium, Complex)

1. Read Incoming tasks
2. Set lowest threshold value (T1)
3. Set highest threshold value (T2)
4. For each task in task list:
   5. If (taskLength < T1)
      6. Categorize task as Simple task
   7. Else if (T1 < taskLength < T2)
      8. Categorize task as Medium task
   9. Else
      10. Categorize task as Large task
   11. End Else
12. End if else
13. End if

Algorithm2. Priority based Scheduling Algorithm

```
Input:
   Task list with task length attribute
   Number of processors
   Processor speed
Output:
   Task Priority (High Priority and Low Priority)

1. Read incoming Task
2. Task Classifier()
3. While (task is not large)
   4. For each task in task list:
      5. If (task is simple)
         6. Put task to high priority (Phigh)
      7. Else
         8. Put task to low priority (Plow)
      9. End Else
   10. End if
   11. End for
12. End while
```

Algorithm3. Task Assignment Algorithm

```
Input:
   Task list with task length attribute
   Number of processors
   Processor speed
Output:
   Task Priority (High Priority and Low Priority)
1. Read incoming Task
2. Task Classifier ()
3. If (taskLength > T2)
   4. Assign tasks to aggregate fog layer (AFN)
5. Else
   6. For each incoming task:
      7. While (! free resource available)
         8. Apply Priority-based scheduling()
         9. Put the task in the queue
      10. End While
      11. Assign the task to the available resource
   12. End For
13. End Else
```

5 Experimentation and Evaluation

The simulation to examine the proposed task scheduling algorithm, which is designed to reduce response time in a fog environment based on task length, priority, and execution, was carried out in Java and the Eclipse editor on the Cloud Analyst simulator. In the setup, primary parameters such as request data size, task length in millions of instructions (MI), task execution time, number of fog nodes, number of processors, and available storage capacity were customized. The experiment focused on

response time, execution time, and throughput as primary metrics for analysis. Additionally, the user base configuration, which represents IoT devices generating traffic, was established based on parameters such as transmission time of task requests and bandwidth, which defines the maximum amount of data transmission between areas or from IoT devices to fog nodes. These configurations enabled proper stimulation of realistic traffic patterns within the fog computing environment.

Due to the differences in the processing capacity, CPU usage, and memory size of Nearest Fog Nodes (NFNs) and Aggregate Fog Nodes (AFNs), the experiment was carried out on three various scenarios. In the first scenario of the Nearest Fog Node, the IoT devices and fog nodes were set up in the same region. The task request volumes, virtual machines, and storage sizes were described in Table 1, and the variable task lengths used throughout the experiment were presented in Table 2. The scenario aimed to examine the performance when IoT devices and fog nodes are located in the same geographical location, with a focus on task priority, scheduling performance, and response times under this specific setup.

The setting of the Nearest Fog Node (NFN) in Scenario 1 is presented in Table 1. It includes the number of virtual machines (VMs) per NFN, processor speed in MIPS, memory size, number of processors, and region location of each NFN. Below is the table with the setup:

Table 1: Nearest Fog Node Specification for Scenario 1

Name of NFN	No. of VMs in each NFN	Processor Speed (MIPS)	Memory (MB)	No. of Processors	Region Located
NFN1	3	100	1024	2	Region 2
NFN2	3	100	1024	2	Region 2
NFN3	3	100	1024	2	Region 2

This configuration is used to evaluate how the Nearest Fog Nodes perform in handling incoming task requests in the specified region. The parameters help determine the fog nodes' ability to process tasks efficiently based on available resources. The tasks with varying task lengths for Scenario 1 in Region 2 are summarized in Table 2. This table provides the task set number, the region location, and the corresponding task length in kilobytes (KB). Below is the table:

Table 2: Tasks with Different Task Length in Region 2 for Scenario 1

Task Set	Region Located	Task Length (KB)
Task Set 1	Region 2	0.01
Task Set 2	Region 2	0.1
Task Set 3	Region 2	0.3
Task Set 4	Region 2	0.5
Task Set 5	Region 2	0.8
Task Set 6	Region 2	1
Task Set 7	Region 2	3
Task Set 8	Region 2	5
Task Set 9	Region 2	8
Task Set 10	Region 2	10

This table is crucial for analyzing the performance of the Nearest Fog Node (NFN) based on task lengths and the region-specific configuration of the fog nodes in Scenario 1.

The second scenario considers the aggregate fog node, where the IoT devices and fog nodes are deployed within the same region. The number of task requests, virtual machines, and storage capacities are detailed in Table 3, while Table 4 presents tasks with varying lengths.

Table 3. Aggregate Fog Node specification for scenario 2

Name of Fog Node (FN)	Number of VMs per Node	Processor Speed (MIPS)	Memory Storage (MB)	Number of Processors	Region
NFN1	5	100	1024	2	Region 1
NFN2	3	100	1024	2	Region 1
AFN1	5	10000	204800	4	Region 0

Table 4. Tasks with different task length in region 1 for scenario 2

Task Set	Nearest Fog Node Region	Task Length (KB)
Task Set 1	Region 1	0.01
Task Set 2	Region 1	0.1
Task Set 3	Region 1	0.3
Task Set 4	Region 1	0.5
Task Set 5	Region 1	0.8
Task Set 6	Region 1	1
Task Set 7	Region 1	3
Task Set 8	Region 1	5
Task Set 9	Region 1	8
Task Set 10	Region 1	10

The third case is performed in two stages to evaluate the impact of task placement strategies. In the first stage, large tasks are assigned to the Nearest Fog Node (NFN) in the same area. In the second stage, the same large tasks are assigned to the Aggregate Fog Node (AFN), which is located in a different area. For consistency, the same task sizes—classified as large tasks—are used in both stages. During the first phase, three heterogeneous tasks are executed on a nearest fog node in Region 1, as detailed in Table 5 (node specifications) and Table 6 (task details). During the second phase, the same tasks execute in Region 1 but are scheduled to execute on an aggregated fog node that exists in Region 0, as specified in specifications in Table 5.

Table 5. Fog node specification for scenario 3 phase 1 (NFN) and phase 2 (AFN)

Name of Fog Node (FN)	Number of VMs per Node	Processor Speed (MIPS)	Memory Storage (MB)	Number of Processors	Region
NFN1	5	100	1024	1	Region 1
AFN1	5	10000	204800	4	Region 0

Table 6. Sample large task specification with different task length running on Region1 for scenario 3 phase 1

Tasks set Nearest fog node	Region	Task length in (mb)
Task Set 1	Region 1	0.5
Task Set 2	Region 1	1
Task Set 3	Region 1	5

6 Experimentation Result

The test was conducted several times under varying situations to analyze the performance of the algorithm that had been developed comprehensively. For the purpose of analyzing its effectiveness, experimentation involved comparisons against two widely used algorithms: First-Come, First-Serve (FCFS) and Shortest Job First (SJF) (Hameed, K., et al., 2016). Comparison of the results of these comparisons was made in order to analyze and compare the performance of the developed algorithm, particularly in terms of response time, execution time, and usage of resources, so as to provide a complete review.

In the first case, three algorithms of which the proposed algorithm is one were utilized to schedule the tasks on the closest fog node. Ten tasks were executed in this layer. Average results were considered, and response time was the primary performance metric. The experiment validated that the proposed algorithm significantly improved response time (111.97ms) compared to the other algorithms (115.01ms for FCFS and 113.88ms for SJF), as shown in Table 7. The observation to be made here is that this improvement in response time was achieved without affecting other performance factors such as resource utilization, throughput, and execution time, reflecting the effectiveness of the algorithm in reducing response time without lowering overall system performance.

Table 7. Comparison of FCFS, SJF and proposed task scheduling algorithm

Task Set	Task Length in (kb)	Average Re- sponse time for FCFS in (ms)	Average Re- sponse time for SJF in (ms)	Average Response time for proposed Algorithm in (ms)	Perfor- mance difference b/n SJF and Pro- posed (ms)
Task 1	0.01	112.76	116.25	102.33	13.92
Task 2	0.1	111.36	113.47	100.88	12.59
Task 3	0.3	113.3	112.43	101.93	10.5
Task 4	0.5	115.74	114.58	105.08	9.53
Task 5	0.8	115.7	118.39	109.53	8.81
Task 6	1	120.59	115.35	110.78	4.5
Task 7	3	120.45	117.98	113.86	4.15
Task 8	5	110.14	107.66	119.44	-11.78
Task 9	8	118.67	112.28	115.37	-3.07
Task 10	10	111.18	110.21	121.39	-11.18
Average Response time for all task in (ms)		115.01	113.88	111.97	1.91

As indicated from Table 7, the response time for all the tasks scheduled through the proposed algorithm is less than that of both the SJF and FCFS algorithms. When the response time is taken individually for tasks and their allocation, it is clear that for larger-size tasks allocated to the Nearest Fog Node (NFN), the response time is relatively greater than if the tasks are scheduled through the SJF algorithm. This implies that despite the proposed algorithm being generally better than the others when

responding to minimize total response time, there are some cases—such as massive tasks scheduled at the NFN—where SJF can guarantee better performance.

As illustrated in the line graph in Figure 4, the response time of the proposed algorithm is considerably lower for smaller tasks but increases with larger task sizes. From this experiment, we can conclude that tasks of size 8 KB or greater need to be scheduled on the Aggregate Fog Node (AFN) to achieve enhanced response times. Tasks of size less than 8 KB, on the other hand, need to be scheduled on the Nearest Fog Node (NFN) using a priority-based scheduling approach to achieve maximum performance, particularly in response time.

In the other cases of the experiment, different tasks were assigned to both the Nearest Fog Node (NFN) and the Aggregate Fog Node (AFN) and were executed with the assistance of the proposed algorithm. As can be seen in Table 8, the response time of the task (categorized as simple, medium, and large) is enhanced while assigning the task to the NFN with the rising size of the task. However, a significant improvement in response time is observed when these larger tasks are offloaded to the AFN. This shows that while the NFN efficiently handles smaller tasks, offloading larger tasks to the AFN results in improved overall performance, particularly in response time.

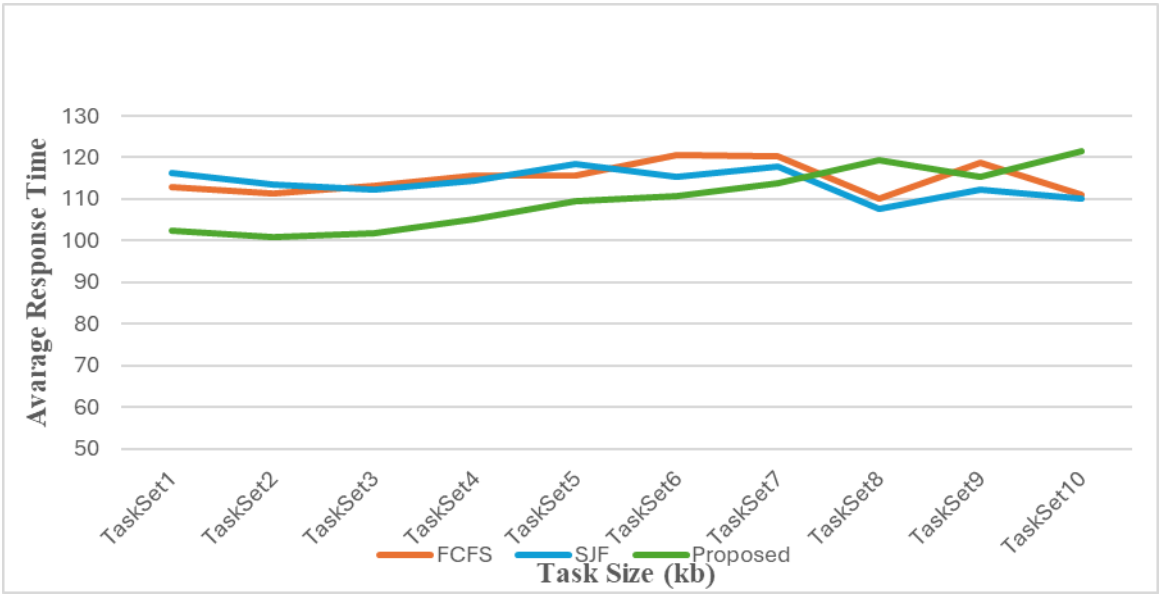

Fig . 4. Comparison of FCFS, SJF and proposed algorithms executing tasks on NFN

Table **8.** comparison of the proposed algorithm on NFN and AFN

Task Set Name	Task Length in (kb)	Average Response time NF	Average time AFN	Response	Thresh hold Indica-tor
Task 1	1	103.27	199.78		-96.51
Task 2	5	104.42	200.98		-96.56
Task 3	10	104.09	201.68		-97.59
Task 4	50	115.22	212.09		-96.87
Task 5	100	118.22	220.98		-102.78
Task 6	500	176.78	297.12		-120.34
Task 7	1000	288.64	386.53		-97.89
Task 11	4280	912.91	916.01		-3.1
Task 8	5000	1131.08	1049.58		81.5
Task 9	10000	2088.76	1838.21		250.55
Task 10	50000	8873.19	7441.01		1432.18
Average response time		1319.24	1212.17		107.07

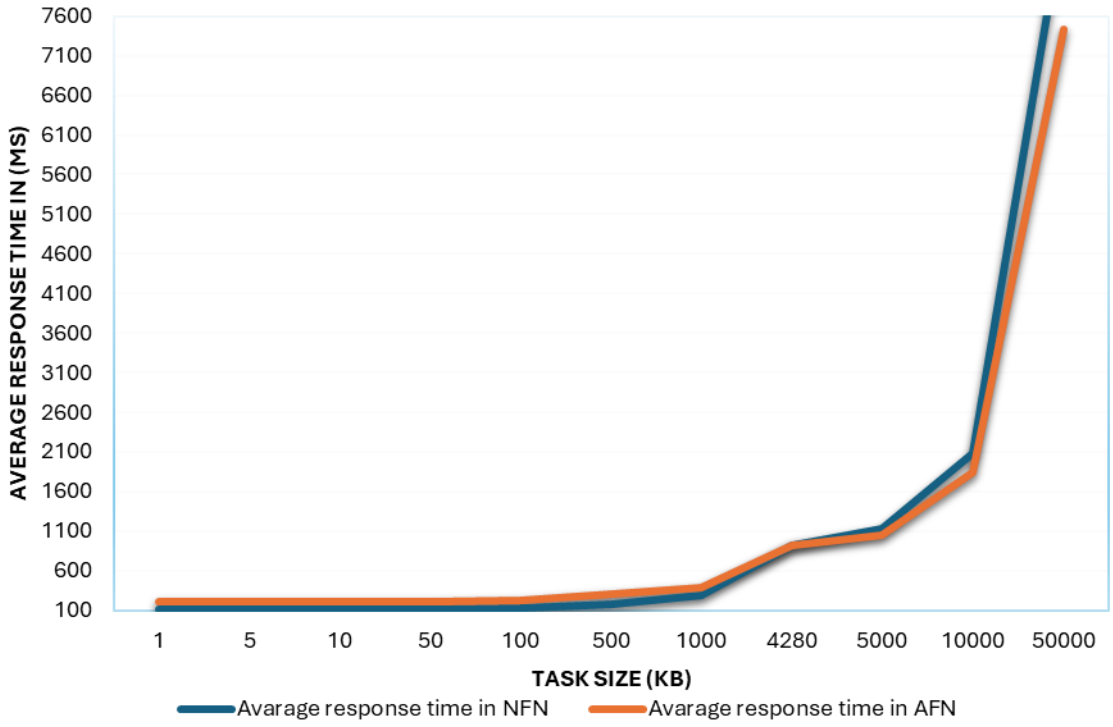

Fig. 5. Average Response time of the proposed algorithm executing tasks on NFN and AFN

Fig. 5 illustrates the response time of sample tasks, ranging from 1 MB to 50,000 MB, when processed in NFN and the AFN. According to the task classification assumption, tasks with smaller sizes exhibit better performance when assigned to the NFN, while larger tasks perform more efficiently when assigned to the AFN. Through multiple tests, an upper thresh- old value (T2) of 4,280 MB was consistently identified. Therefore, tasks smaller than 4,280 MB should be scheduled in the NFN, while tasks exceeding this size should be processed in the AFN for optimal response times. Additionally, the experiment determined that the minimum threshold task size is 50 MB. Tasks below this size may also benefit from being scheduled in the NFN.

7 Conclusion

This paper introduces a priority-based scheduling algorithm with the aim of enhancing task management in fog computing platforms to handle growing levels of complexity and diversity in task requests from IoT devices. The algorithm categorizes tasks into three categories—simple, medium, and large—based on their duration to enable enhanced scheduling and minimized response times. By undergoing extensive experimentation with a Cloud Analyst simulator, the study contrasts the novel algorithm against traditional scheduling algorithms like First-Come, First-Serve (FCFS) and Shortest Job First (SJF).

Smaller tasks of size less than 4,280 MB must be mapped to the Nearest Fog Node (NFN) for maximum performance, whereas bigger tasks are best served in the Aggregate Fog Node (AFN). Further, the task size of 50 MB is suggested as the lower bound, implying that tasks below this size may not require special scheduling. The findings present dramatic improvements in response times and demonstrate the potential of the algorithm for maximizing resource utilization and overall performance in fog computing-based applications.

For future developments, the priority-based scheduling algorithm can be further enhanced to account for task urgency and deadlines. Through the examination of the time sensitivity of the tasks being received, the algorithm can schedule tasks not just based on size but also on the basis of their urgency for timely completion.

Reference

Abdelaziz, J. (2018). *Architectural model for collaboration in the Internet of Things: A fog computing-based approach* (Doctoral dissertation, Université du Québec à Chicoutimi).

Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F., & Parashar, M. (2017). Mobility-aware application scheduling in fog computing. *IEEE Cloud Computing*, *4*(2), 26–35.

Choudhari, T., Moh, M., & Moh, T. S. (2018, March). Prioritized task scheduling in fog computing. In *Proceedings of the ACMSE 2018 Conference* (pp. 1–8). ACM. https://doi.org/10.1145/XXXXXX *(agregar si tienes DOI)*

Dastjerdi, A. V., Gupta, H., Calheiros, R. N., Ghosh, S. K., & Buyya, R. (2016). Fog computing: Principles, architectures, and applications. In *Internet of Things* (pp. 61–75). Morgan Kaufmann.

Diab Abdulgalil, H. (2018). *A multi-tier distributed fog-based architecture for early prediction of epileptic seizures* (Master's thesis, University of Waterloo).

Firdhous, M., Ghazali, O., & Hassan, S. (2014). Fog computing: Will it be the future of cloud computing? In *Proceedings of the Third International Conference on Informatics & Applications (ICIA2014)* (pp. xx–xx). *(Agregar páginas si las tienes)*

Gao, L., & Moh, M. (2018, July). Joint computation offloading and prioritized scheduling in mobile edge computing. In *2018 International Conference on High Performance Computing & Simulation (HPCS)* (pp. 1000–1007). IEEE. https://doi.org/10.1109/HPCS.2018.00159 *(revisar DOI)*

Gupta, M. (2017). Fog computing pushing intelligence to the edge. *International Journal of Science Technology & Engineering*, *3*(8), 42–46.

Hameed, K., Ali, A., Naqvi, M. H., Jabbar, M., Junaid, M., & Haider, A. (2016, August). Resource management in operating systems: A survey of scheduling algorithms. In *Proceedings of the International Conference on Innovative Computing (ICIC)* (pp. 2–5). Lanzhou, China.

Hao, Z., Novak, E., Yi, S., & Li, Q. (2017). Challenges and software architecture for fog computing. *IEEE Internet Computing*, *21*(2), 44–53.

Joshi, G., Deshpande, P., & Joshi, P. (2018). A layered architecture for fog computing for an effective real-time data processing. *International Journal of Computer Applications*, *180*(36), 23–25.

Liu, Y., Fieldsend, J. E., & Min, G. (2017). A framework of fog computing: Architecture, challenges, and optimization. *IEEE Access*, *5*, 25445–25454. https://doi.org/10.1109/ACCESS.2017.2764112 *(verificar si este DOI corresponde)*

Mohan, N., & Kangasharju, J. (2016, November). Edge–Fog cloud: A distributed cloud for Internet of Things computations. In *2016 Cloudification of the Internet of Things (CIoT)* (pp. 1–6). IEEE. https://doi.org/10.1109/CIOT.2016.7872910

Nath, S. B., Gupta, H., Chakraborty, S., & Ghosh, S. K. (2018). A survey of fog computing and communication: Current researches and future directions. *arXiv preprint* arXiv:1804.04365. https://arxiv.org/abs/1804.04365

Santos, H. L. M. D. (2018). *A multi-tier fog architecture for video on demand streaming*.

Sarddar, D., Barman, S., Sen, P., & Pandit, R. (2018). Refinement of resource management in fog computing aspect of QoS. *International Journal of Grid and Distributed Computing*, *11*(5), 29–44.

Varghese, B., Wang, N., Nikolopoulos, D. S., & Buyya, R. (2020). Feasibility of fog computing. In *Handbook of integration of cloud computing, cyber physical systems and Internet of Things* (pp. 127–146). Springer. https://doi.org/10.1007/978-3-030-40347-2_6 *(si aplica)*

Yousefpour, A., Ishigaki, G., & Jue, J. P. (2017, June). Fog computing: Towards minimising delay in the Internet of Things. In *2017 IEEE International Conference on Edge Computing (EDGE)* (pp. 17–24). IEEE. https://doi.org/10.1109/EDGE.2017.11