ASEAN Engineering Journal

Full Paper

JOINT TASK SCHEDULING AND OFFLOADING IN FOG OPTIMIZED COMPUTING SYSTEM (FOCS) ALGORITHM FOR IOT BASED NETWORK APPLICATIONS

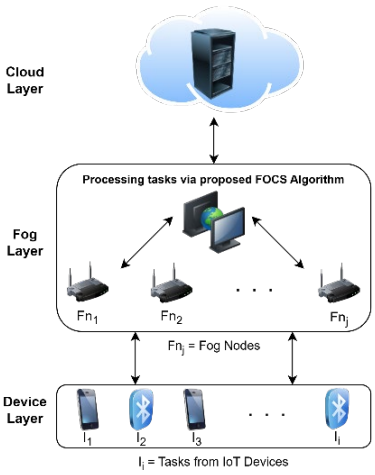
Majid Rafique[a], Nur Ilyana Anwar Apandi[a]*, Siti Nur Lyana Karmila Nor Azmi[a], Zamani Md. Sani[a], Nor Aishah Muhammad[b], Sanaullah[c]

[a]Faculty of Technology and Electrical Engineering (FTKE), Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia
[b]Telecommunication Software and System, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, UTM Johor Bahru,Johor, Malaysia
[c]Fakulti Teknologi Kejuruteraan Elektrik & Elektronik (FTKEK), Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia

Article history
Received
13 August 2024
Received in revised form
16 January 2025
Accepted
26 January 2025
Published online
31 August 2025

*Corresponding author
ilyana@utem.edu.my

Graphical abstract



Abstract

Fog computing, which serves as a middle layer between the cloud and Internet of Things (IoT) devices such as sensors, mobile devices, and smart infrastructures, is becoming more prevalent over cloud computing. Because of its proximity to edge devices, it can process data optimally, enabling real-time reaction demands and reducing latency, energy consumption, and communication costs. In this paper, the Fog Optimized Computing System (FOCS) algorithm is proposed to solve network overloading and processing difficulties that arise from the explosion of IoT devices. FOCS uses a task scheduling and offloading algorithm that classifies data into groups according to its size and routes it to the appropriate fog nodes. Larger data packets are simultaneously sent to fog nodes with higher capacity, while smaller packets are routed to fog devices with low data size in unit million instructions per second (MIPS). Load-balancing strategies ensure that data is delivered to the closest idle fog nodes when there is network congestion. FOCS provides faster response times and low latency while stabilizing the system in terms of energy consumption and utilization cost. The latency, energy consumption and utilization costs are minimized and become stable by the FOCS after a certain number of Inputs (tasks from IoT devices). The proposed FOCS algorithm with organized approach, optimizes the energy consumption by 71% and reduces latency by 35.8%.

Keywords: Fog Computing, Task Scheduling, Task Offloading, Latency, Energy Consumption

1.0 INTRODUCTION

In the digital internet of things era, intelligent technologies collect and gather big data, requiring supercomputers to manage the vast task. Cloud processing reduces latency in network overloading and improves responsiveness for IoT applications. Cloud computing services are typically based on large data centers, offering efficient task offloading but requiring quick IoT sensor response for the task scheduling [1]. Cloud computing services are usually based on large data centers with a lot of equipment that serve numerous users and host various applications. These data centers offer efficient task offloading although requiring quick response from IoT-based sensors. By facilitating data processing closer to the data source, Fog Computing significantly reduces latency, thereby improving response times and overall system efficiency. In Fog Computing, managing energy consumption involves processing data at fog nodes rather than fully depending on cloud infrastructure. Energy consumption represents the amount of power a system uses to process and transmit data. This approach reduces the system's overall power usage by optimizing local resource utilization, which

helps to enhance energy efficiency and minimize unnecessary power expenditure.

In Fog Computing, Utilization cost refers to the expenses associated with processing and transmitting data within a system. Managing utilization cost involves handling data at fog nodes rather than relying entirely on cloud infrastructure. This method helps reduce overall system costs by optimizing local resources, thus improving cost efficiency, and minimizing unnecessary expenditures.

Prolonged and irregular latency is vastly disregarded and objectionable in these kinds of situations. In the context of Fog Computing, Latency denotes the time delay encountered within a system during data transmission and processing. In the context of Fog Computing, it specifically refers to the duration taken for data to travel from its originating source, such as IoT devices, to nearby fog nodes and return.

Introduced a decade ago, fog computing immediately recognized itself as a potential candidate as solution for these challenges [2]. The architecture of Fog system is shown in Figure 1. Fog computing processes a small portion of workload and services using predefined algorithms and memory through local fog devices, creating a separate layer between IoT devices and the cloud. Fog computing reduces latency and improves system efficiency through total network usage by processing data closer to the source. This approach manages energy consumption by processing data at fog nodes, optimizing local resource utilization, rather than relying on cloud infrastructure, thereby reducing overall power usage and minimizing unnecessary power expenditure. Fog computing transfers power closer to devices, while cyclic queue scheduling ensures equal completion of tasks. It

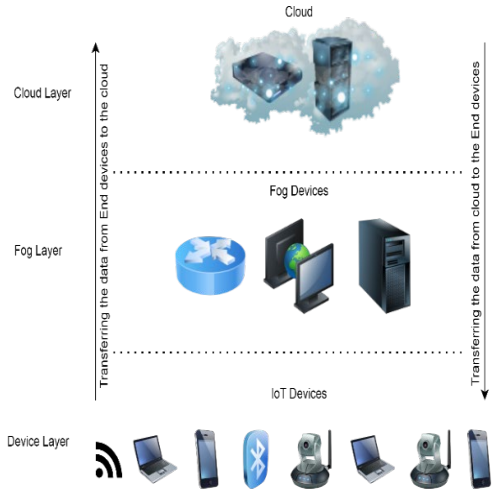


Figure 1: Fog Computing Architecture

manages utilization costs by handling data at fog nodes instead of relying on cloud infrastructure. This approach reduces overall system costs by optimizing local resources and minimizing unnecessary expenditures. This is crucial for IoT applications requiring low latency, high resilience, and broad geographic coverage. Fog computing also strengthens traditional cloud services by moving computational tasks closer to IoT devices, improving their ability to meet application demands.

1.1 Related Works

Task scheduling algorithms have been developed in earlier research to optimize computing technology factors, including latency, energy consumption, and processing cost [3], [4], [5], [6], [7]. A fundamental scheduling method is Round Robin, mostly applied to multitasking situations, where tasks from the device layer alternately receive an equal share of processing time [8]. To ensure that every task in the cyclic queue has an equal chance of finishing and avoiding starvation, each task is allocated a certain amount of time to complete; nevertheless, this method does not account for task priority.

On the other hand, task scheduling helps manage resources effectively in fog computing environments. Task offloading approaches are used in hierarchical cloud-fog-IoT computing environments to minimize energy consumption and fulfill Quality of Service obligations [4], [5], [9], [10], [11], [12], [13], [14]. However, techniques used in [3], [10], [14], [15] that prioritize latency or reduce energy consumption may put pressure on fog nodes near IoT devices. In [3], [5], [16], researchers have proposed various scheduling strategies to optimize performance indicators and resource constraints. Recent methods aim to strike a balance between quality of service and energy efficiency, often prioritizing user device energy consumption over fog node usage [4], [11], [15], [16] . However, there is a lack of research on latency minimization and energy-efficient task offloading in fog computing.

Numerous techniques for task offloading in fog computing have been found that, in contrast to cloud computing, increase energy efficiency and reduce latency in IoT devices. The author of [17] presents a management strategy that uses the Fog-2-Fog structure to enhance load balancing and lower latency in IoT applications. They ignore security concerns while presenting a mathematical model for fog nodes to distribute load. With an emphasis on task offloading decision-making, [18] offers a user-involved method for work offloading to cloud data centers and fog nodes. Algorithms for workload allocation optimization in cloud-fog computing systems are presented in works in [19], [20], with an emphasis on lowering transmission latency and energy usage, which offer a load-balancing system for automobiles that emphasizes adaptive learning techniques to optimize energy efficiency.

The authors in [3], addressed the problem of fog-based IoT network computational task offloading through fog node selection and task assignment optimization. This work proposed a strong optimization technique that improves end-users' execution speed and cost efficiency. Their proposed Ant Mating Optimization technique shows major improvements in makespan and energy consumption measures.

The literature review on task offloading and resource management in fog computing reveals limitations, including an emphasis on energy efficiency and latency reduction without considering scalability and security, which are summarized in Table 1. Due to difficulties with real-time flexibility and effective task distribution in dynamic contexts, real-world applicability is still unclear. Existing optimization methods are often evaluated in isolated scenarios without considering their performance in complex systems. Thus, this paper addresses the issue of task scheduling with size comparison allocation in fog nodes and task offloading for fog computing.

Table 1 Comparison of the existing Strategies for Task Scheduling and offloading.

Existing Work	*Task Scheduling*	*Task offloading*	*Low Complexity*	*Latency*	*Energy*	*Cost*
[3]	✓	✓	×	✓	✓	✓
[4]	✓	×	✓	✓	×	✓
[10]	✓	×	✓	✓	✓	×
[11]	✓	×	×	✓	✓	×
[15]	×	✓	✓	✓	✓	×
[16]	×	✓	✓	✓	×	×
[17]	✓	✓	×	✓	×	×
[18]	×	✓	✓	✓	×	×
[19]	×	✓	✓	✓	✓	×
[20]	✓	×	✓	✓	×	×
[21]	✓	✓	×	×	✓	×
Our work	✓	✓	✓	✓	✓	✓

In our paper, we proposed a Fog Optimized Computing System (FOCS) algorithm that is used to send the task to an appropriate computing device or fog node to address the above-mentioned challenges. We consider simultaneous optimization function between latency and energy consumption based on [10], [11], [17] To support the following work contributions, we compare the proposed FOCS algorithm with the Optimal Fog Algorithm (OFA) [17], First-Come-First-Served (FCFS) [10], [11], and the Fog Job scheduler algorithm (SJF) [10]. This study has the following important contributions:

1. **Extremely Low Latency:** The FOCS has an incredibly low latency which is better in terms of processing the tasks parallel via appropriate size for fog nodes module. Unlike in [10], [11], [17], that is done by using same size for fog node modules, our work solves the optimization by comparing and transferring the tasks to the appropriate size of fog nodes. This implies that the proposed system has a very quick response time, which could be useful for several applications requiring rapid processing.
2. **Low Energy Consumption:** The FOCS can use the least amount of energy possible during operation. Unlike in [16], [17], [18], [19], which only optimize the latency or response time separately, we consider simultaneous optimization of latency, energy consumption and processing, that includes communication cost. This factor matters for both cost-effectiveness and sustainability, especially in settings where energy efficiency is valued highly.
3. **Reduced Processing Costs:** The FOCS attempts to reduce processing costs by combining extremely low latency with low energy consumption because in case of less latency, the energy consumption by the resources will be low. This suggests that the system or technology developed is efficient in terms of total cost-effectiveness, speed, and energy.

This paper is organized into the following sections. An overview of relevant studies and research conducted in the field is given in Section 2. Section 3 outlines our system model's structure and its components. The experimental results are presented in Section 4, after which they are discussed. Finally, Section 5 concludes the findings of this investigation and suggests directions for further research.

2.0 METHODOLOGY

2.1 System Model

We consider a fog layer, which is the middle layer that is interconnected between IoT devices, and the cloud as shown in Figure 1. Network usage is assumed upon the upload size, processing size and the downloading size of the task but we will not much focused on it in this work. Table 2 presents important notations that are used in this system modeling.

Table 2: Important Notations in the Modelling.

Symbols	*Description*
I	Set of task/task from the end devices
F	Set of fog nodes
Fn_j	jth fog node
i	Total number of input device's tasks
j	Total number of fog nodes
T_{ij}^{up}	Uploading time of the task
T_{ij}^{proc}	Processing time of the task
T_{ij}^{down}	Downloading time of the task back to the end devices
D_i	Size of the task
R_{ij}^{up}	Uploading rate of the task
R_{ij}^{proc}	Processing rate of the task
R_{ij}^{down}	Downloading rate of the task
L_{ij}^{total}	Total latency
E_c	Current consumed energy
E_{ij}^{up}	Energy consumption at uploading a task
E_{ij}^{proc}	Energy consumption at processing a task
E_{ij}^{down}	Energy consumption at downloading a task
E_{ij}^{total}	Total Consumed Energy
P_j^{up}	Power required by the fog nodes during uploading
P_j^{proc}	Power required by the fog nodes during processing
P_j^{down}	Power required by the fog nodes during downloading
$MIPS$	Million instructions per sec
$MIPS_A$	Allocated MIPS in the fog nodes
$MIPS_T$	Total MIPS in the fog nodes
$MIPS_R$	Remaining MIPS in the fog nodes
C_c	Current Cost (past value)
C_{Total}	Total Utilization Cost
R_{MIPS}	Rate per MIPS
F_c	Fog node capacity
F_l	Task Load on the fog node
O_l	Overload

In the remaining subsections, we define the latency, energy consumption and utilization cost, also considering the total

network usage but it will not be considered as our primary objective. Latency is the total time in which the task from the end devices is uploaded, processed, and then sent back. Energy consumed is the total energy which is consumed by the fog devices such as routers, gateways etc. and by the cloud server during the entire process. Cost means the total cost w.r.t consumed energy during the whole time taken by the task cycle.

2.1.1 Latency

We assume that the set of task from the end devices is denoted by I, $I = \{I_1, I_2, I_3, \dots, I_i\}$, where i is the total number of input device's tasks and the set of fog nodes in the fog layer is denoted by F, $F = \{Fn_1, Fn_2, Fn_3, \dots, Fn_j\}$, where j is the total number of fog nodes.

For ensuring the task is to be assigned to computing devices as fog nodes [22]. Let $A(i,j)$ is function of assignment for task i to the fog node j, which can be expressed by

$$A(i,j) = \begin{cases} 1, & if\ task\_i\ is\ assigned\ to\ fog\ node\_j \\ 0, & otherwise \end{cases}$$

Let T_{ij}^{up} is the uploading time of the task, T_{ij}^{proc} is the processing time of the task, T_{ij}^{down} is the downloading time of the task back to the end devices, R_{ij}^{up} is the uploading rate of the task, R_{ij}^{proc} is the processing rate of the task, R_{ij}^{down} is the downloading rate of the task. All these rates are depending on the distances between end devices and fog nodes which are in process for the tasks, where $i \in I$ and $j \in F$. Time depends upon the size of task_D_i of task_i and transmission rate of task_i as R_{ij}^{up}, is given by

$$T_{ij}^{up} = \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{D_i}{R_{ij}^{up}} \quad (1)$$

$$T_{ij}^{proc} = \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{D_i}{R_{ij}^{proc}} \quad (2)$$

$$T_{ij}^{down} = \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{D_i}{R_{ij}^{down}} \quad (3)$$

The total time for completing a task is known as Latency. Based on (1), (2) and (3), the total latency for task i and fog node j denoted by L_{ij}^{total}, can be calculated by

$$L_{ij}^{total} = T_{ij}^{up} + T_{ij}^{proc} + T_{ij}^{down} \quad (4)$$

2.1.2 Energy Consumption

Let E is the energy consumed by devices, P is the power of devices and T is the time taken during whole cycle. Here energy consumption depends on fog nodes, is determined by the power and the time cycle of all devices within its loops, as.

$$E = P \times T \quad (5)$$

Let P_j^{up}, P_j^{proc} and P_j^{down} respectively, is the power required by the fog nodes during uploading, processing and downloading the task back to the end device, where $i \in I$ and $j \in F$. The energy consumption of the task during uploading, processing and downloading, respectively, denoted by E_{ij}^{up}, E_{ij}^{proc} and E_{ij}^{down}, can be expressed as

$$E_{ij}^{up} = P_j^{up} \times T_{ij}^{up} \quad (6)$$

$$E_{ij}^{proc} = P_j^{proc} \times T_{ij}^{proc} \quad (7)$$

$$E_{ij}^{down} = P_j^{down} \times T_{ij}^{down} \quad (8)$$

Based on the (6)-(8), the total energy consumption denoted by E_{ij}^{total} can be calculated by

$$E_{ij}^{total} = E_c + [\, E_{ij}^{up} + E_{ij}^{proc} + E_{ij}^{down} \,] \quad (9)$$

where E_c is the current consumed energy (updated after each task completion). Considering this system, after every completion of task, the consumed energy will be updated as $E_c = E_{ij}^{total}$.

2.1.3 Utilization Cost

The total cost of the whole cycle of the tasks in the system is dependent on time taken and allocated data size in unit million instructions per second (MIPS) along the rate per MIPS. Let $MIPS_A$ is the allocated MIPS by the fog nodes for tasks, $MIPS_T$ is the total MIPS of fog nodes and $MIPS_R$ is the remaining MIPS of the fog nodes, such that

$$MIPS_A = MIPS_T - MIPS_R \quad (10)$$

Based on (4) and (10), the total cost denoted by C_{Total}, can be estimated by:

$$C_{Total} = C_c + \{\, L_{ij}^{total} \times (R_{MIPS} \times MIPS_A)\} \quad (11)$$

where C_c is the current cost, which is updated after every process such that $C_c = C_{Total}$, and R_{MIPS} is the rate per MIPS.

2.2 Problem Formulation

The aim of this work is to optimize the above challenges i.e., Latency, Consumption of Energy, and the utilization cost. Therefore, we formulate the problem of optimization by task offloading and load balancing under the umbrella of Fog Computing. To get better optimization, the problem is formulated as follows:

(i) $\sum_{i=1}^{I} A(i,j) = 1$

(ii) Minimize L_{ij}^{total} based on (1), (2) & (3)

(iii) $R_{ij}^{up}, R_{ij}^{\text{proc}}, R_{ij}^{\text{down}} \geq 1$

(iv) Minimize E_{ij}^{total} based on (4)

(v) $MIPS_A \leq MIPS_T$ using (10)

Each task must be assigned to at least one fog node as expressed in (i). To optimize the latency, the uploading, processing, and downloading time from (1), (2) and (3) must be minimized from (ii). The rate in (iii) for the task must be higher than (ii). Total Energy consumption) for the system will be minimized by (ii) as expressed in (iv). Total number of MIPS must be greater than or equal to the allocated MIPS as expressed in (v).

2.3 Fog Optimized Computing System (FOCS) Algorithm

In this section, we propose the FOCS algorithm featuring two sub-algorithms, represents in Figure 2, that the optimization of the processing challenges for solving equation (4), (9) and (11) can be optimized more effectively than previous work done by existing algorithms Fog Job scheduler algorithm (SJF) [10], [11], [17], First-Come-First-Served (FCFS) [10], [11], [17] and Optimal Fog Algorithm (OFA) [10], [11], [17] respectively.

The first process in the FOCS algorithm, referred to as Algorithm 1 task scheduling in FOCS algorithm, is established to send the task to the appropriate fog node.

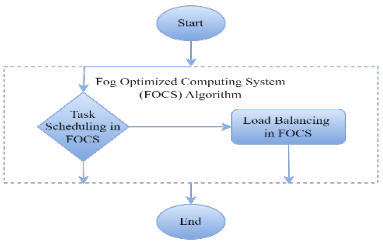


Figure 2 FOCS Algorithm

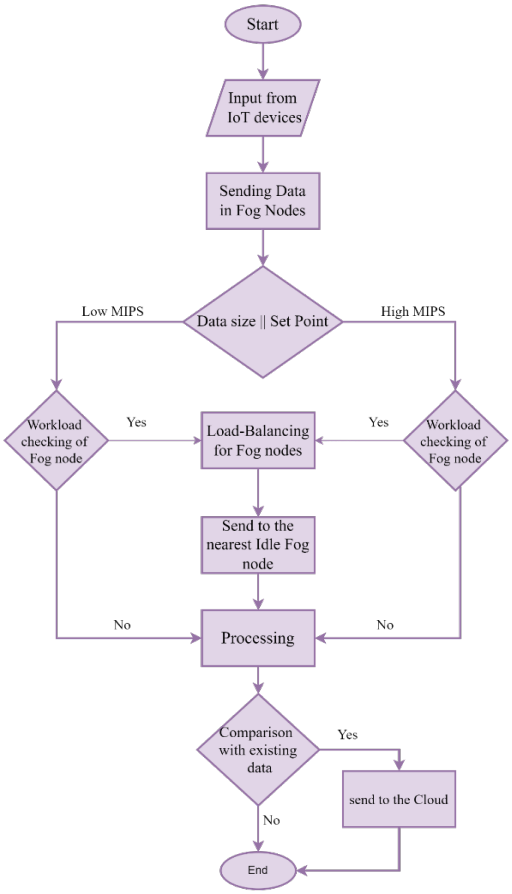


Figure 3 Flow chart of system by FOCS Algorithm

In Task Scheduling Algorithm, the input I_i relates the number of IoT devices' tasks and Fn_j relates to number of fog nodes.

In the main loops (line 3-27), data size is calculated and then updated in its function as per lines 7-10. Then, data size is compared with set size, if it is greater than or equal to set size, then this task will be assigned towards the high MIPS fog nodes as per line 11-21. If data size is smaller than set size, then it will be forward towards low MIPS fog nodes as per line 22-25. The function in lines 14-21 helps the task for idle fog nodes to allocate the task.

Algorithm 1: Task Scheduling in FOCS

```
1: Input: No. of IoT devices I, No. of Fog nodes F
2: Output: Allocation of Task i in appropriate fog
node Fnj
3: for (i = 1, i < I, i++)      where i = numbers of tasks
4: {
5:   for (j = 1, j < F, j++)   where j = numbers of fog nodes
6:         //calculate size of MIPS for Di
7:         void calculateSizeOfMIPS()
8:                 {Update Total MIPS;}
9:           if (Di >= Set MIPS) {
10:             Send the task to High MIPS module
11:             //Function to simulate workload
12:             void workload () {
13:               if (Module!=busy)
14:                     MIPS of Fnj = MIPSA ;
15:               else {
16:                   Check for next module}}}
17:         else {
18:               Send the task to Low MIPS module
19:               void workload ()}}}
20: end
```

Algorithm 2 is developed for determining the overload. In case of overloading, this algorithm is designed to balance the load of tasks according to the availability and capacity of fog nodes. F_l and F_c represent the allocated MIPS and Total MIPS, respectively. They can be calculated from (10) as per line 3. If the load is less than capacity, it will first check if fog nodes (capable of handling that load) are busy or not. The fog nodes with less capacity than the load, will be removed from the list and that task will be assigned to other fog nodes as per lines 5-20. In case of overloading of fog nodes (which are chosen for the task), the value of overloading is calculated and then the fog load is updated by overload (this cycle is continue from line 6) as per lines 21-27.

Algorithm 2: Load Balancing in FOCS

```
1: Input: Fl, Fc ;
2: Output: Ol ;
3: Calculate load on fog node using (10)
4: if (Fl < Fc){
5:    for (i = 1, i < I, i++){    where i = numbers of tasks
6:   for (j = 1, j < F, j++){ where j = numbers of fog nodes
7:           if (Fl >= Fcj) {where Fcj is capacity of fog node
8:                   void FN list() {
9:                   Remove Fnj from the list }}
10:          else {
11:                  void FN list() {
12:                  Add Fnj from the list}}}}
13: else {
14:        void calculate overload(){
15:        Ol = Fl – Fc ; }
16:        Fl = Ol;
17:        goto line 6;}
18: end
```

The flow of proposed FOCS Algorithm is shown in Figure 3. First, it entails arranging and categorizing fog nodes and end devices (such mobile phones and IoT sensors) according to their unique features and processing capacities. The basis for effective resource management and work distribution in the fog computing environment is this classification.

The FOCS gets the tasks or data generated by the IoT devices and processes them. These tasks or data are then sent

to the relevant fog nodes according to their size and processing needs. Comparing the data size to a predetermined set point is an essential part of this stage that decides whether fog nodes—low MIPS or high MIPS—should analyze the data.

The FOCS algorithm constantly evaluates the workload status of individual nodes during data transmission to the fog nodes. To ensure optimal resource use and minimize delay, a load-balancing mechanism is triggered in times of overload and redistributes workloads to neighboring fog nodes with available processing capability. After that, tasks are either processed locally within the fog nodes or sent to the cloud, based on several variables like the task's complexity and the availability of cache memory. This dynamic processing method improves the responsiveness and efficiency of the fog computing environment.

3.0 RESULTS AND DISCUSSION

3.1 Experimental Setup

For this work, the workstation is used with properties as CPU (Intel (R) Core (TM) i7-4790 @3.60 GHz), RAM (16 GB) and an Operating System (Microsoft Windows 10 Pro 64-bit). To simulate and evaluate the performance of the proposed FOCS algorithm, the simulation was developed by using Eclipse IDE 2024 with the toolkit Cloudsim [23], [24]. Simulation-based evaluation was selected to enable controlled manipulation of environmental factors and allow the repetition of simulations under different conditions and inputs.

Table 3 Configuration of the devices for Experimental Setup [10], [11].

PARAMETERS	*CLOUD*	*FOG-SCHEDULER*	*FOG*	*IOT DEVICES*
MIPS	40000	2000	10000	10000
RAM	20000	1000	5000	5000
upBW	100	5000	2000	1000
downBW	5000	5000	4000	5000
Level	0	1	2	3
RateperMIPS	0.01	0.0	0.1	0.1
BusyPower	5*100.45	100.45	90.45	87.52
IdlePower	5*80.25	80.25	80.25	82.44

The simulation was run at different numbers of IoT Devices and fog nodes for the comparison of results with existing algorithms in [10], [11], [17]. To have a fair comparison, we used the parameter settings as shown in Table 3 based on the simulation settings as used in [10], [11]. By using this configuration, the simulation results are obtained and evaluated at various numbers of IoT devices within 10 to 100, and number of fog nodes within 4 to 6.

3.2 Comparative Result Analysis

In this section, the simulation results of proposed work are discussed. The results are obtained from the experiments based on various numbers of IoT devices at multiple fog nodes where each IoT device represents as a task. To support this work contribution, first, we compare the proposed FOCS algorithm with existing algorithms as mentioned in Section 1.0.

Figure 4(a) shows the comparison of average latency of the proposed FOCS algorithm with OFA and FCFS under 10 and 100 input devices, respectively. OFA maintains a latency of 33.5 ms for all light tasks/packets, while FOCS optimizes latency to 21.5 ms for all input device values. This is achieved through parallel processing of task scheduling and load balancing, where tasks are scheduled and executed based on size comparison and load balancing. The proposed FOCS algorithm almost maintains the average latency, making it a superior choice for IoT system execution.

Figure 4(b) shows the energy consumption performance for the proposed FOCS algorithm with FCFS and SJF under various input devices. The proposed FOCS algorithm outperforms FCFS and SJF in terms of energy consumption for 100 to 200 IoT devices. The algorithm optimizes and stabilizes energy consumption at around 10.9 kJ for all input devices, ensuring task execution with lower energy consumption. This performance is consistent across different IoT device numbers due to parallel computation, which manages the appropriate usage of fog nodes and processing devices to optimize energy consumption. The proposed FOCS algorithm maintains the same performance in terms of energy consumption and average latency.

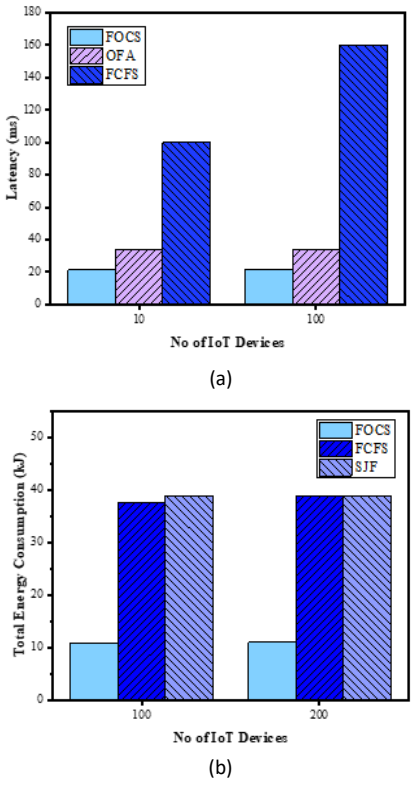


Figure 4 (a) Comparison of Latency, (b) Comparison of Energy Consumption

Now, the results in terms of latency, energy consumption and cost utilization of the proposed work at different fog nodes are to be discussed. In these experiments, the simulation was run by varying the number of IoT devices from 10 to 100 at different number of fog nodes i.e., 4, 5 and 6, respectively.

Figure 5 explains the behavior of latency for different IoT devices at different number of fog nodes. This figure shows that latency of the system varies as IoT devices varies. But the FOCS algorithm maintains it around 21.55 msec. The system remained stable at every change in fog nodes or IoT devices. This shows that the latency is not fluctuating by changing in IoT devices and fog nodes.

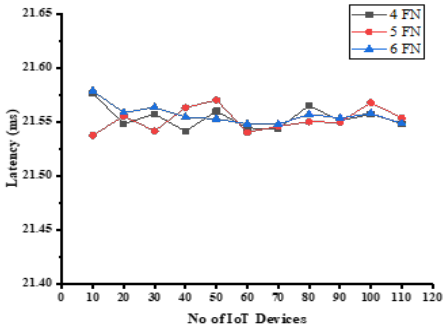


Figure 5 Latency at different fog nodes (FN)

In Figure 6, the performance of energy consumption is shown. Energy consumption of the system increases as the number of IoT devices increases. These values vary by varying the fog nodes i.e., 4, 5 and 6, respectively. It is observed that the proposed FOCS algorithm manages to stabilize energy consumption of the system at a certain number of IoT devices i.e., 90 and above. It maintains the total energy consumption around 11kJ for all fog nodes.

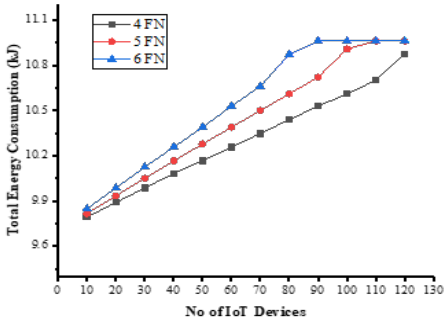


Figure 6: Energy Consumption at different fog nodes (FN)

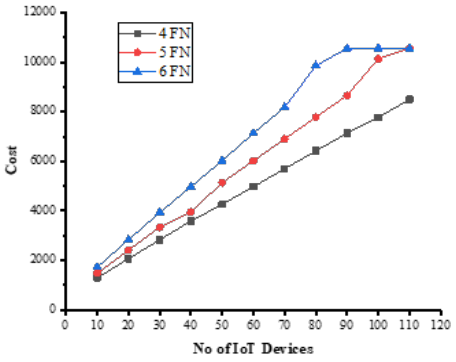


Figure 7: Utilization Cost at different fog nodes (FN)

Figure 7 illustrates the dynamic relationship between system utilization cost and the number of integrated IoT devices. As the number of IoT devices increases, the utilization cost exhibits a corresponding increase. The deployment of additional fog nodes (4, 5, and 6) introduces variability in costs. However, our proposed FOCS algorithm demonstrates remarkable efficacy in stabilizing system utilization costs beyond a threshold of 90 IoT devices at all fog node configurations.

4.0 CONCLUSION AND FUTURE WORK

In this paper, a joint task scheduling and offloading, which task scheduling with size comparison allocation in fog nodes and task offloading for fog computing has been addressed. The proposed FOCS algorithm sends the tasks from the end devices by comparing them with the data size and transfers them to the fog layer, where task offloading and load balancing occur. In addition, the FOCS system offers extremely low latency, allowing for quick task processing through appropriate fog node module size. It also has low energy consumption, considering latency, energy consumption, and processing, including communication cost. While comparing with the OFA [17], FCFS [10], [11], and the SJF [10], the proposed FOCS optimizes latency by 35.8% and energy consumption up to 71% as mentioned in the results. This

approach is beneficial for cost-effectiveness and sustainability, especially in energy-efficient settings. The system also reduces processing costs by combining low latency with low energy consumption, demonstrating its efficiency in terms of total cost-effectiveness, speed, and energy. The work can be expanded to IoT applications like smart homes and buildings, focusing on network usage and optimizing input devices with different IoT sensors. The parameters will be tailored to specific applications, and the focus will be on the distance function.

Acknowledgement

This research work is sponsored by the Ministry of Higher Education Malaysia (MOHE) through the Fundamental Research Grant Scheme (FRGS) with grant number FRGS/1/2022/TK07/UTEM/02/33, which is supervised and managed by the Centre for Research and Innovation Management (CRIM) at Universiti Teknikal Malaysia Melaka (UTeM), in accordance with the grant's terms and conditions.

Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper

References

[1] X. Yang and N. Rahmani, 2021. "Task scheduling mechanisms in fog computing: review, trends, and perspectives," *Kybernetes*, 50(1): 22–38. DOI: 10.1108/K-10-2019-0666.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, 2012. "Fog Computing and Its Role in the Internet of Things Characterization of Fog Computing," *MCC'12 Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 13–16,

[3] S. Ghanavati, J. Abawajy, and D. Izadi, 2022, "An Energy Aware Task Scheduling Model Using Ant-Mating Optimization in Fog Computing Environment, *IEEE Transactions on Services Computing*. 15(4): 2007–2017, DOI: 10.1109/TSC.2020.3028575.

[4] M. K. Pandit, R. N. Mir, and M. A. Chishti, 2020. "Adaptive task scheduling in IoT using reinforcement learning," *International Journal of Intelligent Computing and Cybernetics*, 13(3): 261–282, DOI: 10.1108/IJICC-03-2020-0021.

[5] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, 2020, "Joint Computation Offloading and Scheduling Optimization of IoT Applications in Fog Networks," *IEEE Transactions on Network Science and Engineering* 7(4): 3266–3278. DOI: 10.1109/TNSE.2020.3021792.

[6] S. Tong, Y. Liu, X. Chang, J. Misic, and Z. Zhang, 2023. "Joint Task Offloading and Resource Allocation: A Historical Cumulative Contribution Based Collaborative Fog Computing Model," *IEEE Transactions on Vehicular Technology*. 72(2): 2202–2215, DOI: 10.1109/TVT.2022.3213084.

[7] D. Nurcahya, S. A. Karimah, and S. A. Mugitama, 2023. "Performance Analysis of Scheduling Algorithms on Fog Computing using YAFS," *Sinkron*, 8(3): 1677–1686. DOI: 10.33395/sinkron.v8i3.12682.

[8] S. KumarPanda, D. Dash, and J. Kumar Rout, 2013. "A Group based Time Quantum Round Robin Algorithm using Min-Max Spread Measure," *International Journal of Computer Applications*. 64(10): 1–7, DOI: 10.5120/10667-5445.

[9] S. Aiswarya, K. Ramesh, B. Prabha, S. Sasikumar, and K. Vijayakumar, 2021, "A time optimization model for the Internet of Things-based Healthcare system using Fog computing," *Proceedings of the 2021 IEEE International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems, ICSES 2021*, 1–6, DOI: 10.1109/ICSES52305.2021.9633874.

[10] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, 2020, "A job scheduling algorithm for delay and performance optimization in fog computing," *Concurrency and Computation* 32(7): 1–13, DOI: 10.1002/cpe.5581.

[11] D. Rahbari and M. Nickray. 2019, "Low-latency and energy-efficient scheduling in fog-based IoT applications," *Turkish Journal of Electrical Engineering and Computer Sciences*, 27(2): 1406–1427. DOI: 10.3906/elk-1810-47.

[12] R. Bakhsh, N. Javaid, I. Fatima, M. I. Khan, and K. A. Almejalli, 2018. "Towards efficient resource utilization exploiting collaboration between HPF and 5G enabled energy management controllers in smart homes," *Sustainability (Switzerland)*, 10(10): 1–24. DOI: 10.3390/su10103592.

[13] I. Fatima, N. Javaid, M. N. Iqbal, I. Shafi, A. Anjum, and U. U. Memon, 2018. "Integration of Cloud and Fog based Environment for Effective Resource Distribution in Smart Buildings," *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*. 60–64, DOI: 10.1109/IWCMC.2018.8450422.

[14] B. R. Stojkoska and K. Trivodaliev, 2018. "Enabling internet of things for smart homes through fog computing," *2017 25th Telecommunications Forum, TELFOR 2017 - Proceedings*, 2017 :1–4, DOI: 10.1109/TELFOR.2017.8249316.

[15] W. N. W. Muhamad, A. C. Ribep, K. Dimyati, A. L. Yusof, and E. Abdullah, 2024. "Improvement of Energy Consumption in Fog Computing via Task Offloading," *Journal of Advanced Research in Applied Sciences and Engineering Technology*. 36(2): 199–212, DOI: 10.37934/araset.36.2.199212.

[16] M. K. Hussein and M. H. Mousa, 2020. "Efficient task offloading for IoT-Based applications in fog computing using ant colony optimization," *IEEE Access*, 8: 37191–37201. DOI: 10.1109/ACCESS.2020.2975741.

[17] M. Al-khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, 2019. "Improving fog computing performance via Fog-2-Fog collaboration," *Future Generation Computer Systems*, 100: 266–280, DOI: 10.1016/j.future.2019.05.015.

[18] R. Jindal, N. Kumar, and H. Nirwan, 2020. "Computing and Cloud Computing," *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 145–149.

[19] A. R. Hameed, K. Munir, S. U. Islam, and I. Ahmad, 2020. "Load-balancing of computing resources in vehicular fog computing," *Proceedings - 2020 3rd International Conference on Data Intelligence and Security, ICDIS 2020*, 101–108, DOI: 10.1109/ICDIS50059.2020.00020.

[20] F. A. Saif, R. Latip, Z. M. Hanapi, M. A. Alrshah, and S. Kamarudin, 2023. "Workload Allocation Toward Energy Consumption-Delay Trade-Off in Cloud-Fog Computing Using Multi-Objective NPSO Algorithm," *IEEE Access*, 11: 45393–45404 DOI: 10.1109/ACCESS.2023.3266822.

[21] R. Sing, S. K. Bhoi, and N. Panigrahi, 2023. "A Load Balancing Algorithm for Cloud-Fog-based IoT Networks using Bald Eagle Search Optimization," *2023 1st International Conference on Circuits, Power, and Intelligent Systems, CCPIS 2023*, 0–5. DOI: 10.1109/CCPIS59145.2023.10291583.

[22] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, 2020. "Joint Computation Offloading and Scheduling Optimization of IoT Applications in Fog Networks," *IEEE Transactions on Network Science and Engineering* 7(4): 3266–3278, DOI: 10.1109/TNSE.2020.3021792.

[23] K. S. Awaisi, A. Abbas, S. U. Khan, R. Mahmud, and R. Buyya, 2021. Simulating Fog computing applications using iFogSim toolkit," *Mobile Edge Computing*, 565–590, DOI: 10.1007/978-3-030-69893-5_22.

[24] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, 2017. "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Journal of Software, Practice and Experience*. 47(9): 1275–1296. DOI: 10.1002/spe.250