Hindawi
Scientific Programming
Volume 2022, Article ID 9208066, 15 pages
https://doi.org/10.1155/2022/9208066

Research Article

Effective Task Scheduling in Critical Fog Applications

Aimal Khan,[1] Assad Abbas,[1] Hasan Ali Khattak,[2] Faisal Rehman,[3] Ikram Ud Din,[4] and Sikandar Ali[4]

[1]*Department of Computer Science, COMSATS University at Islamabad, Islamabad 45500, Pakistan*
[2]*School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan*
[3]*Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus, Abbottabad, Pakistan*
[4]*Department of Information Technology, The University of Haripur, Haripur, Khyber Paktunkhwa 22620, Pakistan*

Correspondence should be addressed to Hasan Ali Khattak; hasan.alikhattak@seecs.edu.pk

Received 12 November 2021; Accepted 7 March 2022; Published 31 March 2022

Academic Editor: Jamil Hussain

Information and technology have witnessed significant improvement with the introduction of Internet of things (IoT) applications, and most of the IoT applications are dependent on the cloud. Cloud computing is assisting IoT applications by providing storage, analysis, and processing services on the cloud. However, Fog computing is the new paradigm that supports the cloud by providing scheduling, resources optimization, and energy optimization services. Scheduling tasks based on MIPs size and prioritizing the tasks with smaller MIPs size first make critical tasks with larger MIPs wait, which ultimately increases the delay and may result in some serious problems. This paper proposes a methodology for critical tasks having large MIPs size by scheduling and prioritizing the tasks based on the nature of the task. The proposed methodology for latency-critical applications reduces latency, energy consumption, and network utilization. This paper proposed a scheduler "Critical task First Scheduler" (CTFS), which schedules tasks depending on the nature of the requests, which are classified as either critical or noncritical. The proposed methodology is implemented in a healthcare scenario, and the simulations are performed in iFogSim simulator. Critical requests, such as emergency notifications, are prioritized and designated as critical, requiring immediate processing. The environment was kept the same for all the approaches that are implemented to demonstrate the effectiveness of the proposed approach. The results of the proposed approach were compared with First Come First Served (FCFS), Shortest Job First (SJF), and cloud-only approaches to demonstrate the effectiveness of the proposed approach in terms of latency, energy consumption, and network utilization. Simulation results show that the proposed CTFS approach outperformed the compared techniques for all three comparison parameters.

1. Introduction

The Internet of Things (IoT) is a new paradigm that aims to provide Internet access and power to all devices for communication, enabling them to join the Internet. As the amount of such devices grows, so does the amount of data they generate and their demand for computational capacity to process them. Because IoT has made objects "smart" by allowing them to detect, process, and communicate data efficiently over a network to conduct valuable tasks without the need for end user input [1]. Cloud computing has appeared to be a perfect model for providing the required services. As the amount of users in any network grows, the amount of generated data also rises, and hence, the need to efficiently process the information also increases manifold. By 2023, the cloud computing market, worth $272 billion in 2018, is expected to have grown by 230% to $623.3 billion [2]. The load on the cloud continues to increase because of the swift up growth in the amount of smart devices, moving cloud resources to their limits. By 2025, there are estimated to be $100 billion Internet-connected IoT devices, impacting the economy of more than $11 trillion [3]. The IoT has shown to be helpful in various fields, including smart grids, smart agriculture, and smart health. The IoT has had a major impact on the healthcare sector, as it has in other industries. The aim of using IoT technology is to increase efficiency in the health sector by automating human-led processes. Healthcare aims to provide efficient and quick results of the patient's condition, for which several healthcare architectures proposed are created by combining IoT and cloud computing. Cloud computing has many benefits, but it still confronts numerous problems. There is connection latency due to the placement of the cloud server and the end-user at a distance from each other.

Furthermore, the transmission to the cloud of all data generated by fog devices, given the current network infrastructures and limited data carrying capability. It is essential to mention that processing volumes of data by the IoT devices in the cloud may result in significant delays and overheads, particularly for time-critical applications. The reason for this is that cloud servers are usually placed far away to IoT devices, resulting in (i) increased transmission delay, (ii) reduced performance of delay critical applications, and (iii) network congestion [4]. To these challenges, a new computing paradigm was found necessary to support cloud infrastructure efficiently. Fog computing, as a result, has emerged as a new paradigm.

Fog computing, a novel architecture, has been proposed to bring cloud resources nearer to IoT devices. Fog computing is still in its early phases but has started to gain significance. It is considered a cloud computing extension, invented by Cisco [5], but there is no agreed-upon definition of Fog; it can be described as a distributed source of computational power that allows cloud resources to be extended to the network's edge [2]. Furthermore, because of the considerably significant network latency, shifting IoT tasks to the cloud increases the delay in data analysis reaction time.

The goal of healthcare systems is to highlight that, in many countries, there is no universal healthcare system and facing healthcare problems such as ageing population [6], continuous rise in prolonged sickness in many areas, and limited availability of medical professionals [7]. The global health sector will face a shortfall for health workers in the coming times by 12.9 million as per a recent study by World Health Organization (WHO) [8]. Moreover, around 60% of deaths globally and 85% of deaths in only China account for incurable diseases [9]. Chronic disease costs account for over 75% of the total healthcare costs of the United States [8], while people with inadequate and inaccessible health facilities struggle to survive in challenging conditions, with an increasing amount of patients being a challenge, rural areas, in particular [7]. With the significant rise in the number of patients as a result of the aforementioned challenges, it is become very important and critical to find a suitable solution to these problems, particularly in the field of healthcare. The technological advancement in wireless communication and the IoT will contribute significantly to improving performance, as well as a reduction in healthcare costs. The IoT helps these patients be followed up by providing very low-cost homecare systems that can detect early signs of illness and allow doctors to respond and treat patients more rapidly. Therefore, we propose a fog computing-based architecture for healthcare systems.

By processing the data received from the sensors, the proposed architecture for healthcare systems based on fog computing delivers information about patients' health state. The suggested architecture has three layers: a cloud layer, a fog layer, and an Internet of Things layer. The IoT layer is the initial layer, and it comprises sensors that sense and generate data in the healthcare environment for various purposes. The second layer is known as the fog layer, which contains the number of fog nodes that receive the data transmitted from the sensors being attached to the patients and process it to diagnose whether it is critical data or noncritical and using a proxy server on the topmost layer to save the results in the cloud server. The proposed healthcare system architecture is designed to give patients uninterrupted real-time medical support, and fog computing has validated its efficacy in time-critical applications. Fog computing brings resources nearer to the end-users to quickly handle the massive amount of data generated from the end-users. Therefore, fog computing is the most appropriate method for real-time data processing.

Fog server has minimal resources in comparison with cloud server, which has unrestricted storage and computing capacities. The load on the fog server grows with an increase in the number of requests in the massive systems [10]. The increasing request to a particular fog node in a healthcare system will overload that fog node. In this scenario, where the specific fog node is being overloaded, while the other fog nodes will remain inactive, this will lead to increased latency, energy consumption, and network utilization. Fog applications have limited resources, and time sensitive nature makes it more challenging to manage its resources [1]. Task scheduling and allocation of resources hold great importance for application, which requires fast response. For example, in healthcare systems, quick responses are needed from the doctors to save patient's life [11]. As a result, an efficient job scheduling technique is required to maximize the usage of these heterogeneous and resource-constrained fog devices [12]. Although there are already proposed task scheduling algorithms such as delay priority, Round Robin, Shortest job first (SJF), and first come first served (FCFS), they are still in their early stages. Algorithms such FCFS, SJF, and round Robin execute the tasks based on their arrival sequence. Hence, they cannot lower latency sensitive applications' response times. For time-sensitive applications, a scheduling algorithm is required, which reduces average response time, network usage, and energy consumption. We proposed a Critical Task First Scheduler (CTFS), which schedules the tasks to the best suitable fog node to decrease latency, energy consumption, and network utilization for the requests. As from [13], we assume that IoT data might experience increased traffic delay, energy consumption, and network utilization. The simulations were carried out using the iFogSim tool to assess the proposed approach's performance and compare it to strategies such as First Come First Served (FCFS), Shortest Job First (SJF), and cloud-only. In a cloud-only approach, all data is sent to the cloud server without the fog layer being implemented. The effectiveness of the proposed technique is proved by the results of simulations.

The contributions of the proposed CTFS approach are as follows.

(i) An approach is proposed to serve and improve Fog computing in multiple prospects. We propose a Fog scheduler that will handle resources-constrained Fog devices.

(ii) The study aims to minimize the delay in task execution and energy consumption and maximize the use of a fog device available.

(iii) In Fog computing, the essential feature is a real-time response without any delay. The Critical Tasks First Scheduler (CTFS) will schedule the tasks to minimize the response time for task execution based on their criticality.

(iv) Since all network components use energy, and Fog devices have limited resources, the proposed method aims to reduce the number of resources (RAM, Energy, and Processors) expended by Fog devices.

(v) The number of IoT devices is constantly rising network workload and results in network congestion. We aim for the reduction of network utilization in our work.

(vi) To demonstrate the effectiveness of the CTFS approach against FCFS, SJF, and cloud-only implementation, considerable simulations are performed in the iFogSim toolkit.

The paper is organized as follows. The related work is presented in Section 2, whereas the fog computing architecture for healthcare is described in Section 3. The parameters for performance evaluation are highlighted in Section 4, while Section 5 presents a case study from the healthcare domain that is used to explain our work. Section 6 explains the complexity analysis of the proposed algorithm. In Section 7, the performance evaluation of each parameter is explained and discussed, with findings compared to alternative techniques. Finally, Section 8 summarizes the findings of our study.

2. Related Work

Fog computing is the new paradigm for healthcare to efficiently handle the data from the sensors attached to the patients for timely actions. It handles the data received from the sensors at the fog node to reduce latency, energy consumption, and network utilization by connecting the fog node to the IoT device.

Low-Cost Health Management (LCHM) is proposed in [14], which is gathering data of heart patients. Moreover, sensors attached to patients collect and analyze data in order to process it more quickly; however, LCHM has a longer response time, reducing performance. FogCepCare is an IoT-based healthcare management system proposed in [15]. It combines the cloud server with the sensor devices to evaluate cardiac patients' health status and reduces task processing time at run time. FogCepCare uses a segmentation and clustering approach, as well as a communication and distributed processing policy, to reduce execution time. In a cloud environment, the FogCepCare is simulated to evaluate the performance with the existing models, and it reduces execution time. However, this work lacks an evaluation of the results in terms of crucial QoS factors like power consumption, latency, and accuracy. An IoT healthcare service is proposed in [16], which is based on a Software Defined Network (SDN) application that gathers data via smartphone voice control and determines patients' health status. Furthermore, an IoT healthcare service uses a smartphone application conceptual design to determine the type of cardiac arrest; however, the suggested application's performance is not tested in cloud environments.

It is proposed in [17] that a security solution for IoT-enabled healthcare mobility that employs the Datagram Transport Layer Security (DTLS) handshake protocol without requiring any device layer modifications can provide secure communication between numerous interconnected smart gateways. Furthermore, the suggested technique is applied using a simulation environment (Cooja), demonstrating that it effectively lowers communication overhead by 26% and latency by 16%. HealthFog intends to build on this work by deploying healthcare applications on real-world systems and fog nodes, resulting in a more effective alternative.

A fog-based healthcare architecture is proposed [18], in which several fog nodes are utilized to control patients' requests across various cities. At the fog nodes, the patient's status is watched, and if the patient's condition is found to be critical, the request is sent to a cloud server immediately; otherwise, the request is processed by fog node itself. While simulating a number of topology configurations for performance evaluation, the results are not comparable to any cloud or fog based healthcare system implementation.

A task is offloaded, and a management method for embedded devices and fog nodes is proposed in [19]. The goal is to reduce the time it takes for software-defined embedded systems to compute. The suggested technique combines job scheduling and storage allocation optimization using combined nonlinear programming. It is suggested in [20] to limit end-to-end delay and to disseminate tasks between fog nodes by restricting the numerous offloading functions required to relegate a task to an appropriate fog node. The author in [21] considered the energy utilization issue with the imperative that the latency of the Internet of things (IoT) more than a specified edge is required in applications, and energy reduction was finished by ideally allotting assignments to middle fog nodes. The primary objective is to increment the preparing proficiency of the undertakings considering the restricted assets and correspondence impediments.

The First-Come-First-Served (FCFS) scheduling method was employed to schedule tasks (EEG tractor game) on Fog nodes, and they use the FCFS algorithm to calculate network utilization, energy utilization, and loop delay. A new architecture is presented in [22] to adhere to the cloud and wireless network for real-time applications in the healthcare domain for timely actions. According to [21], there are two sorts of delays that might occur when data flows from IoT devices: network delay induced by the volume of service requests and computed delay produced by resource allocation for requests.

The authors presented in [23] the significance of fog computing with the conclusion that placing a fog layer between the cloud and end-users can reduce network traffic by 90 factors for offloading tasks have been identified: improved performance and energy savings [24]. Another study in [13] presented a Shortest Job First (SJF) for offloading of tasks by processing the smaller MIPS first to the cloud layer using a proxy server to reduce factors such as latency, energy consumption, and network utilization.

In previous studies, the simulation results of fog-based methodologies were compared to cloud-only implementation [25]. In the fog, we give a modelling of these two metrics. Then, using Evolutionary Algorithms, we represent the problem as restricted optimization and solve it quickly (EA). Our strategy stands out as an energy efficient option [26]. The authors of [27] concentrated on concerns such as remaining battery lifetime and energy-characteristics of fog devices. Unlike earlier studies, our model energy usage and delay are lower and are unlike others who solely focus on one factor.

A different strategy is used in [28], but in essence, they suggest an algorithm that tries to arrange components in the specified fog colony first and then tries to send excess components to other fog colonies. Components that cannot be assigned to a fog colony are sent to the cloud, which shows its inefficiency that ultimately increases the resource usage. According to Authors [29], fog nodes should be arranged in a rooted tree, with computation requests arriving at the leaves and being completed or directed to the root. To distribute requests among the fog nodes on the path from leaf to root, a branch-and-bound technique is proposed. This method increases the delay, which is its drawback. The goal of this work is to investigate the energy consumption tradeoff in a Fog-loT system between terminal nodes (TN) and fog nodes (FN). To that aim, we suggest a new protocol in which TNs broadcast their data at a predetermined transmission rate to possibly all FNs without first deciding which FN to associate with. FNs decide whether to process data locally or send it to the cloud center after receiving it, based on whether they are overloaded or not. This work only improves the energy consumption but ignores the other important factors such as delay and network usage.

There were comparison and resource optimization for the system to work efficiently and in a better way. In order to evaluate their proposed architectures, [30] did not include performance characteristics metrics such as latency and network use. Some previous studies [30] only considered one parameter but did not discuss its impact on latency and energy consumption. Another study in [31] only considered delay and network usage, while it did not consider energy consumption. The importance of delay for any task is addressed in [32], but other parameters were not addressed such as energy consumption and network usage. Scheduling of tasks is discussed in [33, 34] while decreasing network usage, but the other two important factors, that is, energy consumption and delay, were not addressed. Only execution time and energy consumption are reduced in [35] for scheduling of tasks, whereas network usage is not included. [36, 37] introduced tasks scheduling technique but only considered two parameters such as delay and energy consumption, while it ignored network usage, which is an important factor, while another study in [38] only improved energy consumption of the system but did not provide any information on its impact on the other two factors delay and network usage. We set up the environment, performed simulations for five different topologies setups, and compared the results to the FCFS and SJF schemes and cloud-only implementation. These simulations results show that CTFS outperformed cloud-only, FCFS, and SJF schemes for latency, network utilization, and energy consumption parameters.

Several studies addressed multiobjective task scheduling optimization in fog-based architectures that outplay the cloud-only implementation. In the study presented previously, none of the proposed approaches is compared with a fog-based architecture for healthcare systems. Thus, it is critical to build a fog-based technique that is more efficient and propose solutions in fog computing suffering from a lack of job allocation efficiency while also being inefficient in terms of energy-saving and latency. For healthcare, many scholars have proposed fog-based architecture and proposed algorithms to efficiently offload tasks. A fog-based architecture was not proposed by any of the researchers that efficiently optimize resources such as time delay, energy, and network utilization. Thus, it is imperative to propose and develop a fog-based architecture that is efficient and better than earlier suggested fog-based approaches. In healthcare systems, the most critical metrics are latency, energy consumption, and network utilization, while the majority of the studies merely evaluated their proposed methods against latency; however, energy consumption and network utilization were not included as performance parameters. As a result, comparing our proposed solution in terms of energy consumption and network utilization is critical.

3. Proposed System Architecture

Fog computing architecture consists of three layers, that is, the Internet of things (IoT) layer, Fog layer, and cloud layer. The middle layer is known as the fog layer, which works between the cloud and IoT layers. The fog layer is a distributed layer consisting of numerous fog devices and is responsible for the intermediation between cloud and IoT layer presented in Figure 1.

3.1. First Layer: IoT.

Sensors and actuators make up the IoT layer. Temperature sensors, heartbeat sensors, humidity sensors, cameras, GPS sensors, and other sensors take data from the outside world; it is converted into signals and conveys that to Fog nodes for further processing. Following analysis, the outputs of fog nodes are communicated to

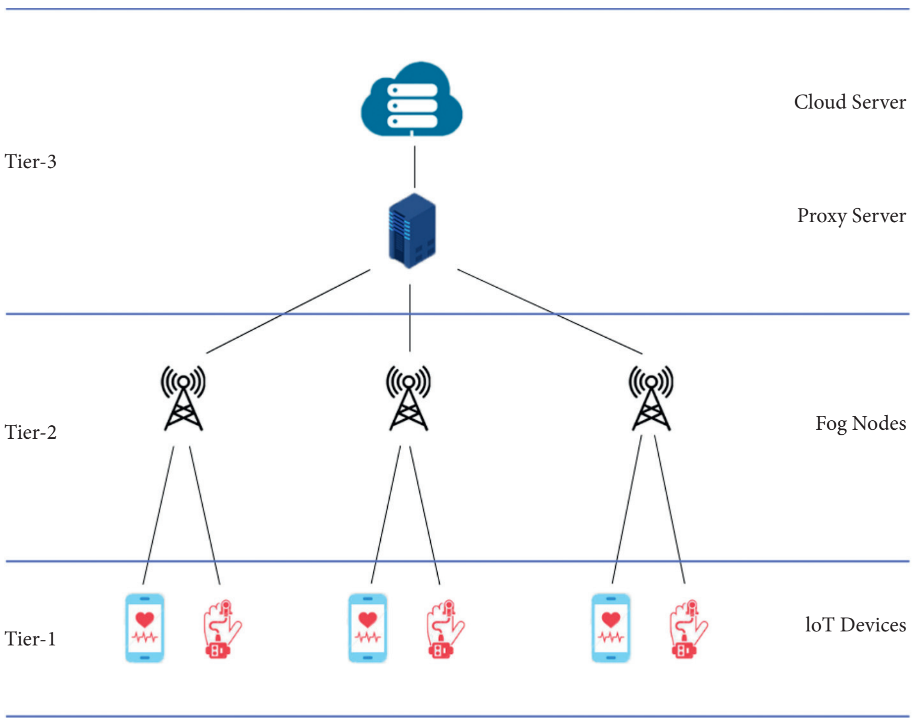

Figure 1: Fog-based architecture for healthcare.

actuators, which operate as controllers and take necessary action. The transmission between IoT and fog layer can be made possible in two ways, i.e., Device to-Fog (D2F) and Device-to-Device (D2D). The communication between devices to device (D2D) that are near to each other can be made possible through different ways such as Bluetooth, Wi-Fi, or Zig-bee. To communicate with the layer upper to it would be made possible via Device-to-Fog (D2F) layer.

3.2. Second Layer: Fog. Fog exists in the middle of cloud and IoT layers, the fog serves as an intermediary, and it weakly combines the Fog's Cloud and IoT layers, allowing for autonomous development and high levels of interaction of various layers. While leveraging the Cloud services above, the Fog layer serves as a foundation to the base IoT ecosystem. Devices such as proxy servers, mobile base stations, routers, and switches all together form the Fog layer. The fog layer is comprised of both low and high levels of devices. Both these low and high level devices are differentiated by their storage, processing capabilities, and power. Low-level devices have low RAM, storage, and power, while high-level devices are fog servers with rich capabilities such as RAM, storage, and power and are connected to the cloud servers. The Fog nodes will be communicating with each other using Fog-to-Fog (F2F) link. The Fog node with IoT will be communicating using Device-to-Fog (D2F) link. The architecture can be seen in Figure 1.

3.3. Third Layer: Cloud. In the cloud layer, the data collected by fog devices is stored in the cloud layer, which is on top. The information is retained, so that data analysis may be performed, and the results can be sent back to the devices for further processing.

Sense-Process-Actuate-Model (SPAM) is a model which states that sensors are the sources that collect data from the outside world and transfer data to the fog nodes for processing the signals to the actuators for further action. Peer-to-Peer (P2P), cluster techniques, or client-server is used by these applications for nodal collaboration. Directed Acyclic Graph (DAG) is used for developed applications for different modules following Distributed Data Flow model. It takes input data, processes it, and forwards it to the next modules.

3.4. Case Study. Fog computing is a paradigm shift for efficiently processing healthcare data from IoT devices at the nearer fog nodes. Fog computing is efficient in handling large amounts of data. The data of patients with cardiac problems can be handled by Fog computing at edge devices with sufficient computational power to reduce latency, reaction time, and delay, while cloud-based solutions in e-healthcare result in greater latency, which is undesirable in emergencies. A substantial majority of healthcare computing activities may be done by adjacent fog nodes leveraging fog computing, resulting in shorter latency and increased availability [39].

In healthcare systems application, there are two types of use cases: critical and noncritical. The patient's data, for example, is gathered and saved for the doctor to study later. This type of data storage and retrieval is not extremely time-sensitive and is tolerant of delays. Quick data analysis is required in some cases, for example, when a patient is in a critical state, to generate emergency warnings. Such jobs are time-sensitive and necessitate a fast response to avoid an emergency.

The following three types of application use cases are included in this smart healthcare case study.

3.4.1. Emergency Response System. In case of emergency, an emergency response system processes important data from different smart sensors such as blood glucose, heartbeat, blood pressure, and body temperature. This data comprises important information about the patient's health that must be processed in an emergency, such as blood glucose above 400 mg/dl or blood pressure levels above 140/90 mmHg.

3.4.2. Patient's Appointment Management System. Patient's Appointment Management System is a healthcare system that allows booking appointments and management of appointments while minimizing the chances of duplication of same time slot for distinct patients, and therefore, it is considered less critical.

3.4.3. Patient's Record Management System. The Patient Record Management System stores information in the database that includes patient personal details, doctor details, treatment, lab results, and visits. The moment a patient gets entry into the hospital, the receptionist generates a single entry for each patient to keep in the cloud-based Patient Record Database. We implemented three application modules to create this smart healthcare case study, which are shown below.

3.4.4. DCPB. DCPB is the lowermost layer of fog devices that collects the information from each of the three application components. It collects the critical data from the fog devices, processes it, and forwards it to be displayed on the notification screen. It similarly gathers data about a patient's appointment and records and sends it to the organizers' module.

3.4.5. Organizer. The organizer module is placed on the uppermost level fog device that collects data from the DCPB. It assigns time slots to patients in response to appointment requests. Patients are informed of the appointment schedule. It sends the patient's record, as well as certain important data, to the Patients Record Database.

3.4.6. Patients Record Database. This module is cloud-based. It gets data from the organizer to store and long-term analysis. It creates patterns based on a patient's health, hospital visits, and records and delivers them to the organizer.

4. Critical Task First Scheduler

In our proposed architecture, we have considered a healthcare scenario where the transmission of data is very important and critical as shown in Figure 2. The regular transmission of data overloaded the system and consumed extra resources, which led to delay and overutilization of resources. Therefore, it is imperative to solve the problem of task overloading by offloading the tasks to other neighboring fog nodes having less load and available resources to minimize the delay and resources. Here, to minimize the latency, energy consumption, and network utilization, a load balancing technique is required to efficiently offload the tasks among fog nodes. Therefore, we propose a Critical task First Scheduler (CTFS), to minimize the latency, energy consumption, and network utilization. A healthcare scenario for the proposed approach is explained by making different streams of data that will receive data smart sensors, while "CRITICAL" tuples include patient condition as shown in Figure 2. The following is an example of a case study and its components. Data must be processed and analyzed immediately. The "SCHEDULE" tuples are generated by the Patient's Appointment System, which are requests of appointments and appointment time is presented after processing. Thus, it is less critical. Patient data to be saved on the cloud server for later use is contained in the "RECORD" tuple from the Patient's Record Management System. Thus, it is delay-tolerant. We need the quickest alert since "CRITICAL" tuples are the most critical. The delays are computed for four end-to-end modules since "SCHEDULE" tuples are less critical, and "RECORD" tuples are delay-tolerant.

Some recent studies have proposed load balancing techniques, one of which is First Come First Served (FCFS) [13], which states that the nature of the received tuple is not important, but the arrival time is important; thus, the one arriving first will be served first, and so on. Another approach, Shortest Job First (SJF) [13], states that the received tuples MIPS should be checked in order to execute. The tuples with smaller MIPS will be executed on priority, while the other will wait. We are using the same procedure as in [13] to calculate the latency, energy consumption, and network utilization. Figure 3 shows the proposed CTFS algorithm where as the Table 1 defines the various terminologies of the algorithm.

For modelling resource management strategies in IoT and edge computing environments, iFogSim is extremely effective. We changed a few iFogSim classes and created a few new ones to implement the scheduler. The following is a quick overview of the classes that are utilized as shown in Figure 4.

4.1. Sensor. The data from sensors or fog devices are received at fog nodes in the form of tuples. Tuples are changed into variable-sized in this class. This class functions to simulate IoT sensors.

4.2. Fog Device. Different fog devices are simulated in this class. Processor, memory, capacity, storage, and uplink and downlink bandwidths are all defined for each Fog device. Fog nodes can be multilevel, and every Fog node has the ability to communicate with the other higher-level Fog nodes in the IoT layer via tuples.

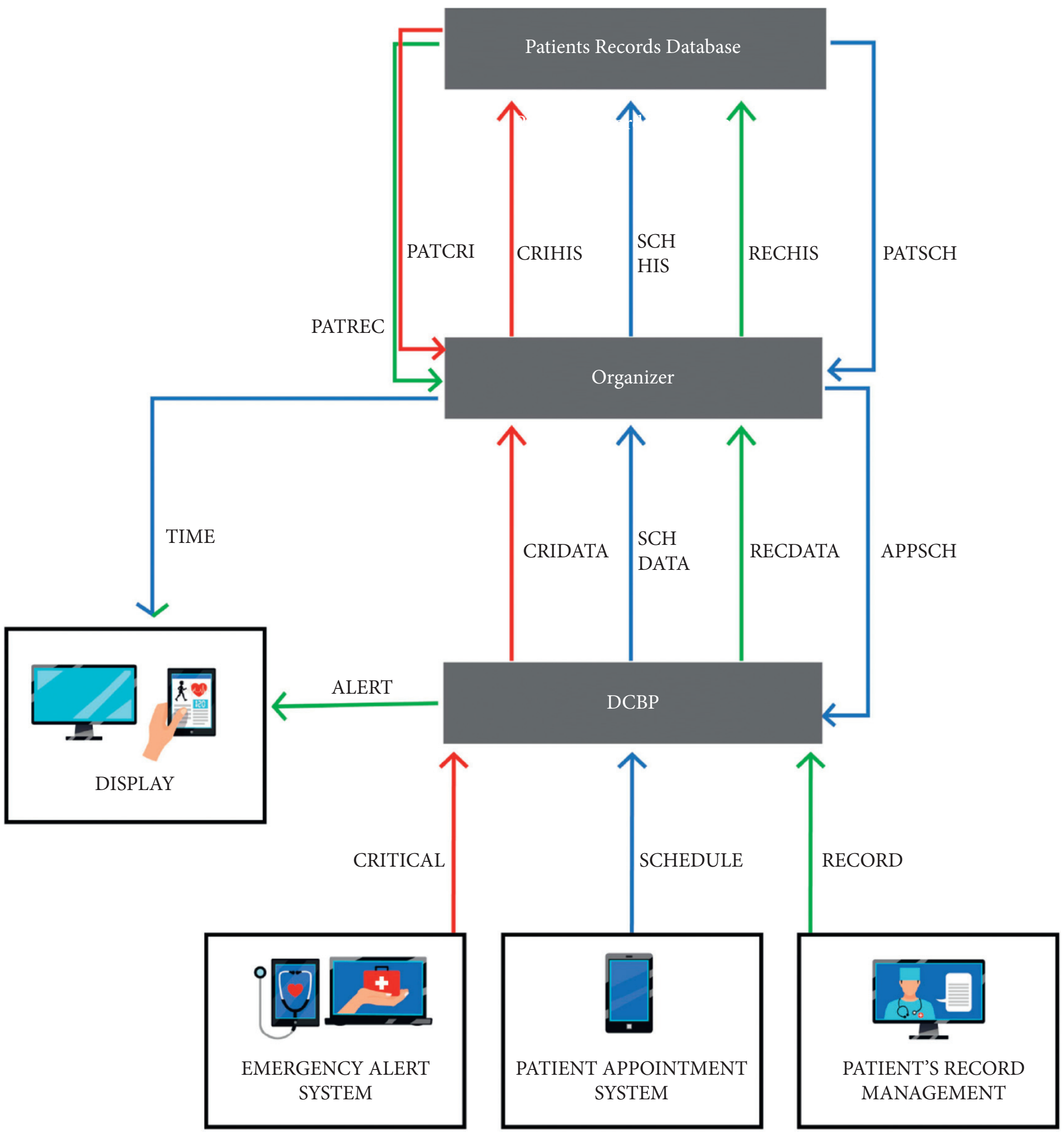

Figure 2: An example of a case study and its components.

4.3. Tuple. The tuple class is used to communicate amongst all of the entities in Fog. As MIPS, each tuple gathers data about the source, destination, and processing requirements. Tuple: the tuple class is used to communicate amongst all of the entities in Fog. As MIPS, each tuple gathers data about the source, destination, and processing requirements as shown in Figure 5.

4.4. Proposed Scheduler. This class extends CloudletScheduler, which keeps a queue W and a completed tuples list F. The queue is made up of all the tuples that are awaiting execution, whereas all of the tuples that have completed their execution are included in the completed tuples list. When a tuple is completed, the next tuple in the queue is chosen.

A tuple is received from the sensor using Transmit() method and forwarded to the Send(tuple) method for a low-level linked Fog device. A callback function is used to receive a tuple when it arrives via ProcessTupleArrival() method. This function determines whether the tuple should be handled by the Fog device or forwarded to a higher-level Fog device. In case the fog devices will process the tuples arrived, so it will send it to the Proposed Scheduler using the SubmitTuple() method. The tuples will be placed on the queue W using SubmitTuple() method. SchedulenextTuple() method is used to select the next tuple for execution to fog devices. Once the execution of the current tuples finishes, so the scheduler requests for the upcoming tuple to be processed via CloudletFinish() method; after successful execution of the tuples, it is placed in the completed tuples list F. After that, the next tuple will be selected from the queue W for execution as shown in Figure 6.

5. Experimental Setup

This section explains the environmental setup for simulation that was used to evaluate the suggested approach's performance.

```
INPUT: Tuple List (N1, N2, ..., NN)
OUTPUT: Fineshed Tuple FT
1:  do
2:    if new tuple arrived then
3:    Nt <----received tuple
4:        if Nt=CR then
5:           Allocate VM to Nt
6:           execute Nt
7:           FT <---- Nt
8:           F <---- NT
9:        else
10:          W<---- Nt
11:          queue in W
12:       end if
13:   end if
14:   if FT then
15:       Tmin <---- select Nt with Appropriate MIPs form W return allocate VM to MINT
16:       execute MINT
17:       FT<----MINT
18:       F<----FT
19:   end if
20: While Not end of tuple OR W=0
21: return tuple
```

Figure 3: Proposed algorithm.

Table 1: The proposed CTFS algorithm's symbols.

Symbols	Meaning
W	Queue of tuples
F	Finish list of tuples
M_t	Coming tuple
F_t	Completed tuples
MIN_R	Appropriate tuple
C_R	Critical tuples

In healthcare, the smart IoT sensors are attached to the body of the patient to sense and generate the data for transmission to the fog nodes. The data is processed and analyzed at the fog node to see and take the necessary actions. After processing, the data is forwarded to the cloud for storage using a proxy server. To evaluate the performance of the proposed approach, we have used the iFogSim toolkit. It is used by many researchers [40, 41] for simulations of their proposed architecture.

For evaluation of the proposed architecture, we need to define the fog devices in iFogSim and also set the values for different parameters. Table 2 shows parameter values that are equivalent in magnitude to those typically used in the associated literature [13, 25]. The computation of the device has different levels starting from zero to 3. The cloud server is represented by level 0, whereas level 1 represents the proxy server, which makes the connection between the fog nodes and the cloud servers. Level 2 shows the fog nodes that are located in various locations to aid IoT sensors inefficient calculation, while level 3 consists of sensors attached to end-users for data generation. The topology of the proposed healthcare architecture is given in Figure 7. The simulations for the proposed architecture are carried out on Dell 3521 (Intel Core i3, 3.3 GHz Processor, 520 Hard Drive) with the windows 10 operating system.

The proposed architecture is simulated for different topologies while only increasing the node of fog devices to see the results of the performance as given in Figure 7. We set the first configuration for 100 fog devices to see the effectiveness of the proposed architecture on how it distributes the increasing load among fog nodes. The simulations are conducted five times, with each configuration increasing the number of fog devices by 100. The performance parameters are latency, energy consumption, and network utilization.

6. Performance Evaluation

In this section, we compare the outcomes of the Critical Tasks First Scheduler (CTFS) to the First Come First Served (FCFS), Shortest Job First (SJF), and cloud-only approaches for various configurations. The algorithms used for comparison in this work is selected based on the work done in [13]. The loop delay, energy consumption, and network utilization are the three metrics that we must choose. Table 3 shows the notations used in various equations.

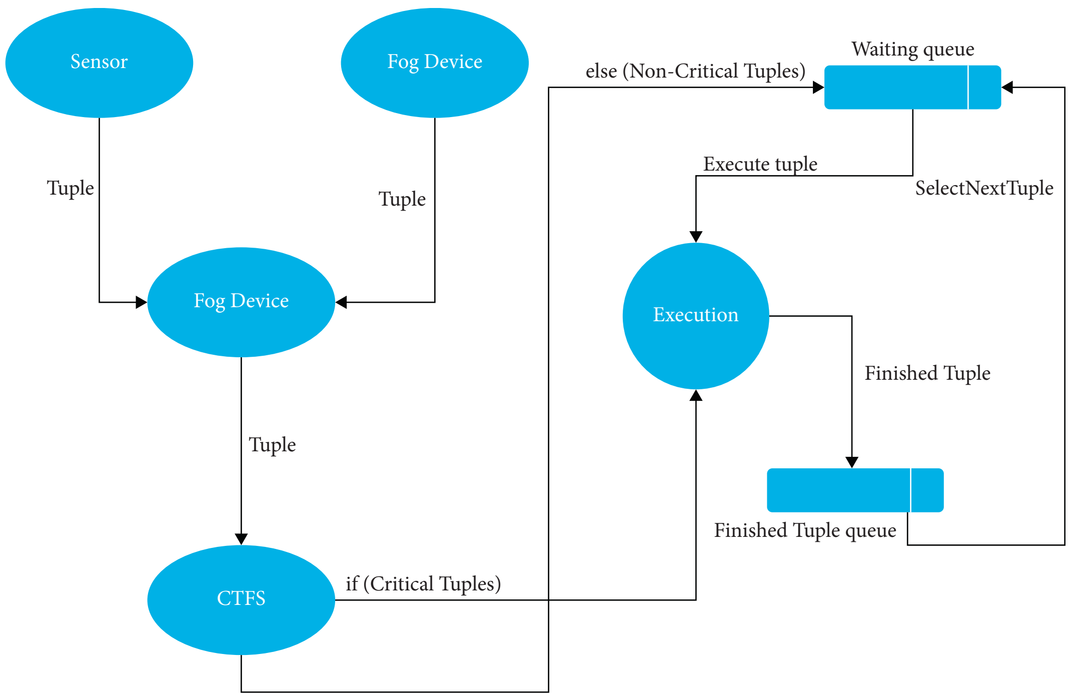

Figure 4: Block diagram of the proposed CTFS approach.

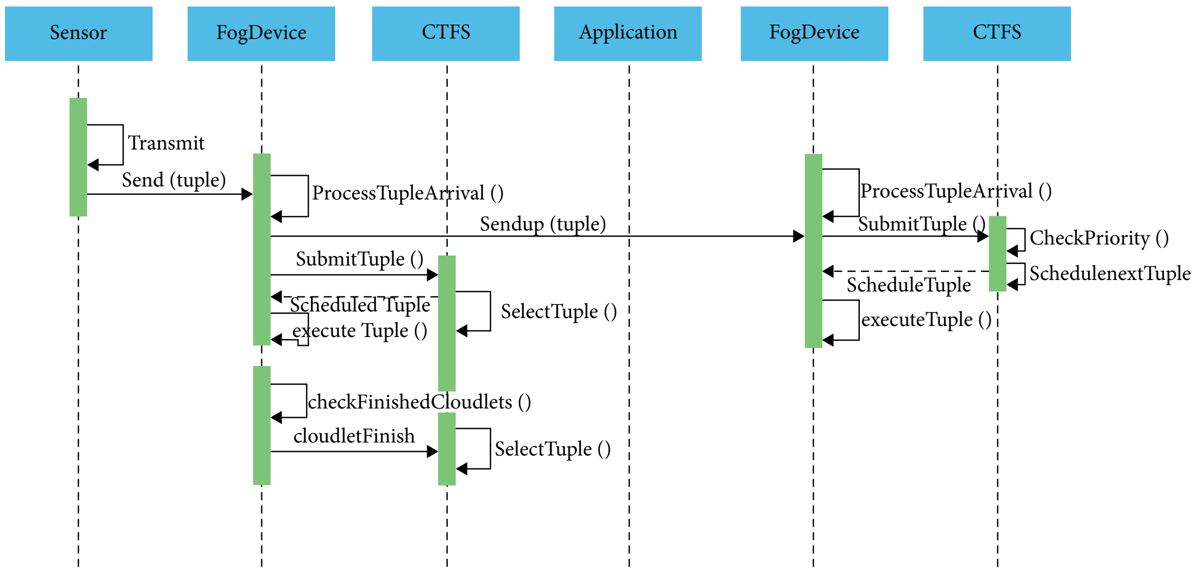

Figure 5: Tuple scheduling and execution sequence diagram.

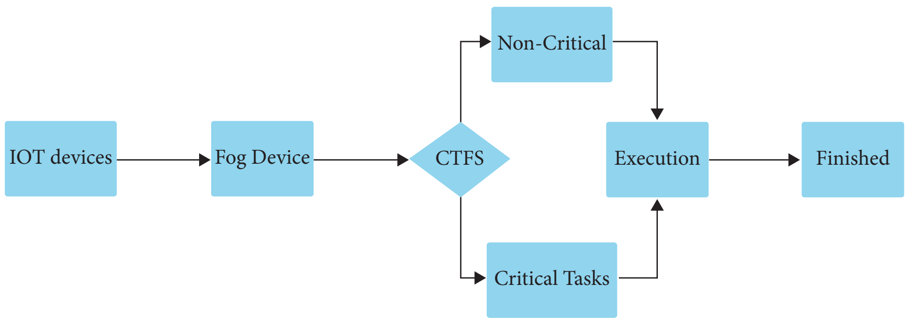

Figure 6: Flow diagram of the proposed CTFS approach.

Table 2: Configuration of Fog nodes.

Name	RAM	MIPs	Level	UpBw	DnBw	Rate per MIPs	Busy power	Idle power
Cloud	40000	44800	0	100	10000	0.01	1648	1332
Proxy server	4000	5600	1	10000	10000	0.0	107.339	83.4333
2nd level fog nodes	4000	5600	2	10000	10000	0.0	107.339	83.4333
1st level fog nodes	1000	800	3	10000	270	0	87.53	82.44

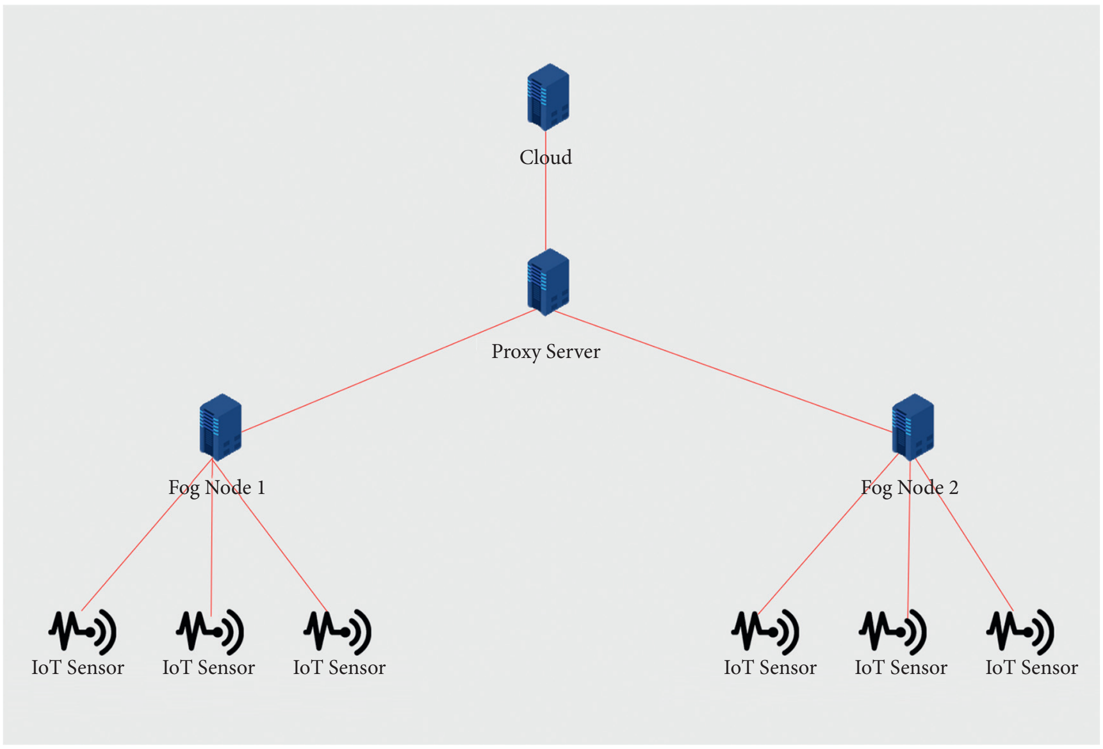

Figure 7: Topology of the proposed fog-based approach in iFogSim.

6.1. Average Loop Delay. The complete latency of all modules in the loop is measured using a control loop. The average CPU time is used to calculate the loop delay, Avg CPU all tuples of a specific type of tuple use it. Equation (1) is used to calculate the average CPU time in iFogSim as in [13].

$$\mathrm{AvgCPU}_T = \begin{cases} \dfrac{T_s \times \mathrm{NT} + T_E}{\mathrm{NT} + 1}, \\ \\ T_E + 1. \end{cases} \qquad (1)$$

T_S denotes the start of execution, T_E denotes the end of execution, and NT is the total number of performed tuples of a certain type. Equation 2 is used to calculate the execution latency of each tuple.

$$\mathrm{Latency}\,(i) = T_{S(i)} - T_{E(i)}, \quad \forall i \in T, \qquad (2)$$

T is the tuple set that already exists.

The results produced from our scheduler for application loop delays are compared with FCFS, SJF, and cloud-only approaches as these three are the only implemented approaches in a given healthcare scenario in iFogSim. The number of nodes is shown on the x-axis, whereas the average application loop delays are shown on the y-axis. Four color bars are used to distinguish between CTFS, FCFS, SJF, and Cloud-only approaches.

Table 3: Simulation setup notations.

Symbols	Meaning
T_S	Start time of execution
T_E	End time of execution
$E_{\mathrm{CONSUMPTION}}$	Energy consumption
T_{CURRENT}	Current time
T_R	Update time (last utilization)
HP_R	Host power (last utilization)
L_i	Latency of ith tuple
C_i	Network size of ith tuple
C_{CURRENT}	Current cost
T	Last utilization

The emergency alert's average loop delay is shown in Figure 5. In this situation, the Critical Tasks First Scheduler (CTFS) delays are significantly smaller than those of FCFS, SJF, and cloud-only. When using the CTFS, the growing number of nodes does not arise the latency. Figure 6 demonstrates a noncritical loop delay for the patient's appointment. Figure 8 shows that the loop delay of the CTFS is very small as compared to FCFS, SJF, and cloud-only.

Figure 9 illustrates the loop delay for Patients Record Management, showing that, for FCFS, SJF, and cloud-only, the delay grows as the number of nodes grows, but the CTFS performs better with lesser delays.

6.2. Energy Consumption. Reducing fog node power utilization lowers the total cost of electricity consumption as well as the environmental impact. There are two power states for fog nodes: idle and busy. Whenever the fog node is not processing tasks, it is said to be in idle mode, and when it is executing tasks, it is said to be in busy mode. The Fog device's energy consumption using equation (3) in iFogSim is calculated as in [13]:

$$E_{FN} = E_{\text{CONSUMPTION}} + \left(T_{\text{CURRENT}} - T_R\right) \times \text{HP}_R, \tag{3}$$

where $E_{\text{CONSUMPTION}}$ is the current energy consumption, T_{CURRENT} is the current time, T_R is the update time of last utilization, and HP_R is the host power in the last utilization. The energy may be calculated using the total of all the hosts' power during the set length of time for any of the Fog devices.

Figure 10 illustrates the average energy utilization of fog nodes. On the x-axis, the number of nodes is shown, whereas the energy consumed is shown on the y-axis. For a limited number of participating IoT nodes (less than 300), the average energy utilization of the CTFS is lower than that of FCFS, SJF, and cloud-only. As the number of fog IoT grows, so does the value. As a result, we employ to offer adaptive load balancing to send workloads while lowering the energy consumption of the server executing the request. Thus, the consumed energy by fog nodes using CTFS is lower than FCFS and SJF but minor increased as compared to cloud-only, but since it is a small increase, so it does not impact the overall system. It is also stated in [42] that running a huge number of IoT devices may increase energy consumption. Therefore, the difference of results for the various number of fog nodes can be seen in Figure 10, which depicts that our CTFS works more efficiently than FCFS and SJF. Figure 11 shows the energy consumption of the approaches in the paper.

6.3. Network Utilization. The last evaluation metric is network utilization. Network congestion is caused by the regular increase in the numbers of the device. Such an increase in the number of devices leads to network congestion and poor performance of the overall system. By spreading the load over several fog devices, fog computing helps minimize network congestion. Equation (4) is used to calculate network use.

$$N = \sum_{i=1}^{N} L_i \times C_i, \tag{4}$$

where N is the total number of tuples, L_i is the latency, and C_i is the network size of the ith tuple. Figure 12 shows the results of CTFS against the FCFS, SJF, and cloud-only. The number of nodes is represented on the x-axis, while the

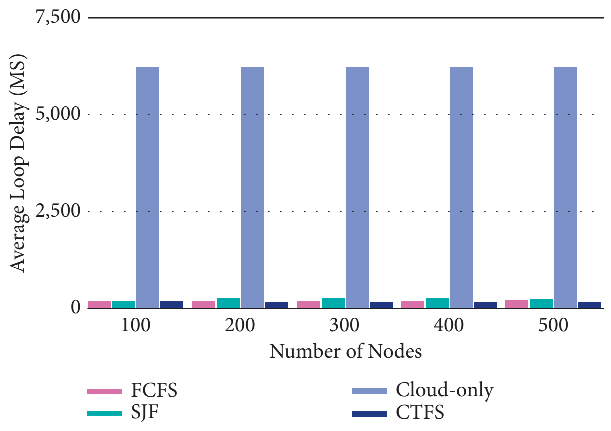

Figure 8: For an Average loop delay, the average loop delay is measured in milliseconds (ms).

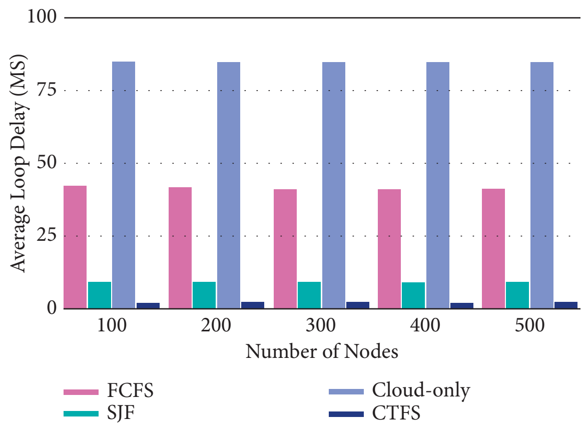

Figure 9: For an emergency alert, the average loop delay is measured in milliseconds (ms).

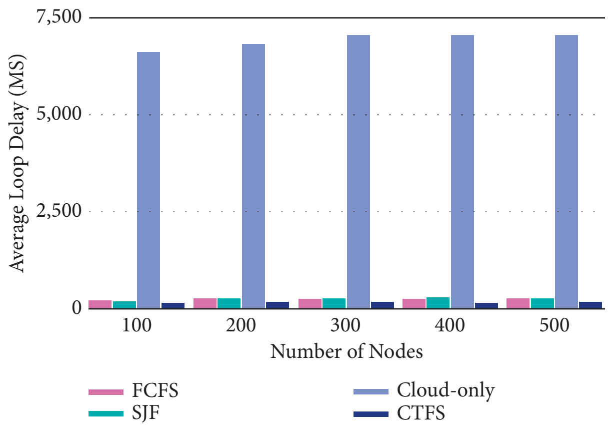

Figure 10: For the history of nodes for varying numbers of IoT nodes, average loop latency is measured in milliseconds (ms).

average network utilization is represented on the y-axis. The results show that the CTFS is decreasing network utilization efficiently as compared to FCFS and SJF but starts increasing at a certain point due to the growing number of IoT devices, which results in more energy consumption when compared to

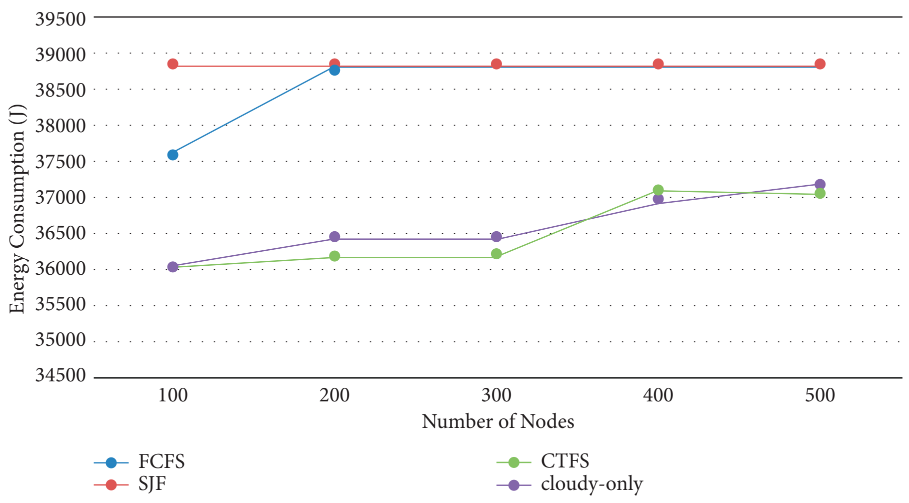

Figure 11: By changing the number of nodes of the Fog devices, the average used energy in (J) may be calculated.

the cloud-only approach. It is stated in [42, 43] that requiring high bandwidth sometimes increases the network utilization, but since network consumption of our proposed architecture and cloud-only in our healthcare scenario is still smaller, so there is no negative impact on the overall system. The consumption of the network appears to be growing as the number of fog nodes grows, which ultimately leads to more congestion more power utilization, while our proposed approach utilizes the unutilized fog nodes to quickly complete the tasks and save time and wastage of resources by assigning tasks. SJF approach is wasting the resources and making network congestion by processing the small tasks on fog nodes with high resources, which can be used for other tasks, which require heavy resources due to its inefficiency, which is solved in our proposed approach, which efficiently forwards the task to fog nodes according to their requirement for processing and completion. The empirical findings for network consumption performance measures demonstrating the Fog computing-based solution we propose are a better fit for time-critical systems. When used for a time-critical application like healthcare, a fog-based architecture provides rapid recovery of information about the patient's physiological data. The findings indicate that fog computing is useful in scenarios wherever fast data processing is critical. Figure 13 shows the energy consumption of the approaches in the paper.

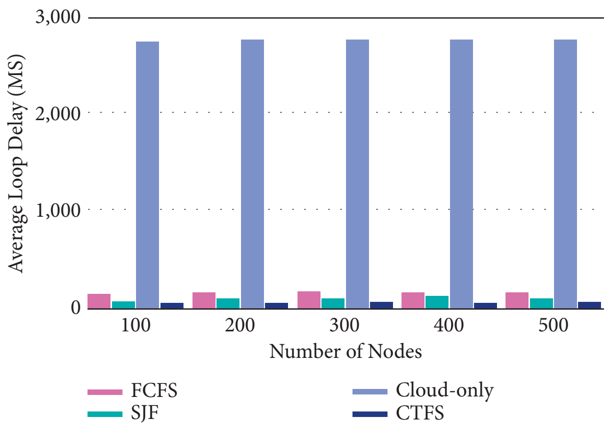

Figure 12: For record management, the average loop delay is measured in milliseconds (ms).

7. Discussion

The study presents three important findings: first, the healthcare system requires time-critical solutions, and fog computing is found to be the most suitable one for it, as it minimizes the latency and provides fast services. Second, the proposed CTFS approach is very efficient in healthcare compared to the FCFS, SJF, and cloud-only implementations in terms of the three performance parameters, i.e., latency, energy consumption, and network utilization. Third, energy consumption and network utilization are also reduced for the increasing number of devices on fog nodes, which shows the efficiency of the proposed approach. Cloud computing, due to its decentralized nature, used to be the best possible option for processing and storing data over cloud servers until the emergence of issues such as increased latency, energy consumption, and network utilization. Cloud computing was found to be making issues for time-critical applications such as healthcare where real-time data processing and execution are very imperative as human life might be at risk. Therefore, cloud-based approaches would not be the best solution for such time-critical applications. In contrast, a new layer is added to the cloud architecture, called the fog layer, which works between cloud servers and end-users. The fog layer brings the resources nearer to the IoT devices at fog nodes for required tasks. Thus, the load on cloud servers is minimized. The load distribution over fog nodes reduces latency, energy consumption, and network utilization. Therefore, it is noted from the results that fog computing is the best possible solution for time-critical applications such as healthcare. Our proposed fog-based CTFS approach reduces latency, energy consumption, and network utilization significantly compared to FCFS [13], SJF [13], and cloud-only implementation. The results clearly show that our proposed CTFS approach outperformed FCFS, SJF, and cloud-only implementation. The values of latency, energy

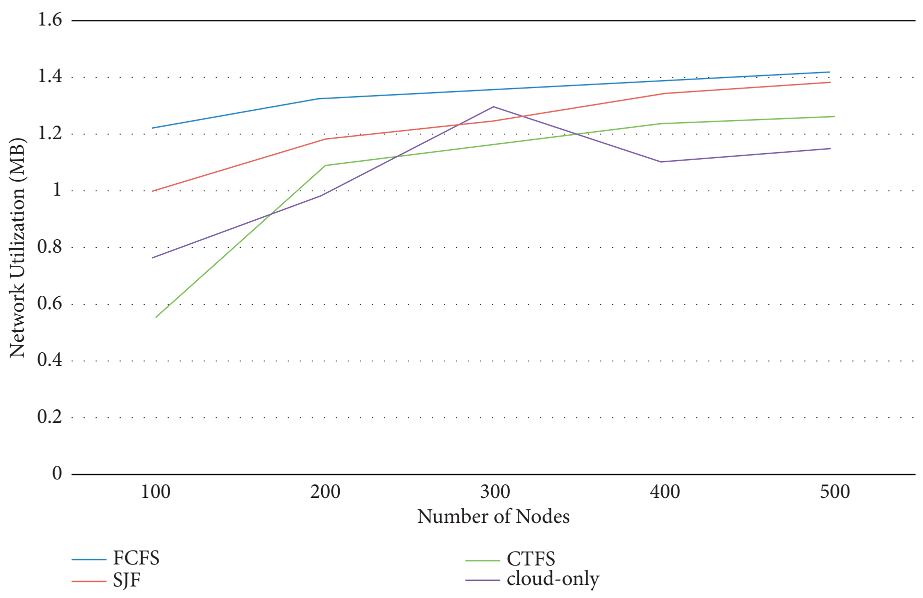

Figure 13: Network utilization of fog devices in megabytes (MB) versus the number of nodes (IoT devices).

consumption, and network utilization increase with the number of IoT devices with each configuration. Cloud-only implementation results for energy consumption and network utilization are also increasing but found to be decreasing at some points. The latency difference between the CTFS approach and cloud-only implementation is very high, but the difference in energy consumption and network utilization is small. Overall results state that CTFS is the most effective approach among the three for time-critical applications. The fog-based FCFS approach receives the task from the end-users at fog nodes and starts distributing load among different fog nodes but lacks prioritizing the critical one. Furthermore, it increases delay, energy consumption, and network utilization. In another fog-based approach, SJF processes the data at fog nodes and distributes the load among different fog nodes. The load distribution is prioritized by executing the smaller tasks first while keeping the larger ones awaiting, which ultimately increases the delay, energy consumption, and network utilization. Our proposed approach CTFS proved to be the most effective approach among FCFS, SJF, and cloud-only implementation for performance parameters latency, energy consumption, and network utilization. In summary, our proposed CTFS approach in comparison with FCFS, SJF, and cloud-only implementations performed very well in terms of latency, energy consumption, and network utilization. The proposed fog-based approach for healthcare will reduce latency to provide time-critical patients response and medical staff and others in their daily operation's instant services. The proposed approach will reduce the burden on healthcare workers and facilitate the patients as well.

8. Conclusion

Cloud computing is not an efficient option for real-time applications such as healthcare, where data is critical and must be transmitted quickly without delay. Cloud-based architecture is limited by high bandwidth utilization and increasing latency. Fog computing, a modern computing paradigm that supports cloud computing by assisting with real-time processing and analysis and other resources near the edge device, has evolved to address these limitations. Fog computing job scheduling is a significant challenge since edge devices have limited resources.

This study provides an improved task scheduling method for latency-critical applications that reduces latency, energy consumption, and network utilization. To demonstrate the time-critical and delay-tolerant jobs over fog nodes, we have used a healthcare case study. The proposed Critical Tasks First Scheduler (CTFS) is scheduling the task based on the nature of the requests, which are divided into two categories: critical and noncritical. The critical request such as emergency alerts is kept at the top priority and labelled as critical to be processed first without any delay as the human life would be at risk if it is not processed on time, while the other requests such as record management and patient's appointments are labelled as noncritical and delay-tolerant. To minimize the latency, energy consumption, and network utilization in a system requires the appropriate load distribution to make the system work efficiently. The proposed Critical Tasks First Scheduler (CTFS) outperforms the First Come First Served (FCFS), Shortest Job First (SJF), and cloud-only implementation in all three parameters, according to the results of the simulations.

Data Availability

Data are available upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, "Fog of everything: energy-efficient networked computing architectures, research challenges, and a case study," *IEEE access*, vol. 5, pp. 9882–9910, 2017.

[2] M. I. Bala and M. A. Chishti, "Offloading in cloud and fog hybrid infrastructure using ifogsim," in *Proceedings of the 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 421–426, Noida, India, January 2020.

[3] A. K. Majumdar, *Optical Wireless Communications for Broadband Global Internet Connectivity: Fundamentals and Potential Applications*, Elsevier, Amsterdam, Netherlands, 2018.

[4] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, 2017.

[5] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: principles, architectures, and applications," in *Internet of Things*Elsevier, Amsterdam, Netherlands, 2016.

[6] M. Haghi Kashani, M. Madanipour, M. Nikravan, P. Asghari, and E. Mahdipour, "A systematic review of iot in healthcare: applications, techniques, and trends," *Journal of Network and Computer Applications*, vol. 192, Article ID 103164, 2021.

[7] M. Islam, A. Rahaman, and M. R. Islam, "Development of smart healthcare monitoring system in iot environment," *SN computer science*, vol. 1, no. 3, pp. 1–11, 2020.

[8] R. Van de Pas, P. S. Hill, R. Hammonds et al., "Global health governance in the sustainable development goals: is it grounded in the right to health?" *Global Challenges*, vol. 1, no. 1, pp. 47–60, 2017.

[9] F. Meng, X. Zhang, X. Guo, K.-H. Lai, and X. Zhao, "How do patients with chronic diseases make usage decisions regarding mobile health monitoring service?" *Journal of Healthcare Engineering*, vol. 2019, Article ID 1351305, 2019.

[10] R. A. da Silva and N. L. da Fonseca, "On the location of fog nodes in fog-cloud infrastructures," *Sensors*, vol. 19, no. 11, Article ID 2445, 2019.

[11] S. Mishra and S. Jain, "Ontologies as a semantic model in iot," *International Journal of Computers and Applications*, vol. 42, no. 3, pp. 233–243, 2020.

[12] L. Liu, D. Qi, N. Zhou, and Y. Wu, "A task scheduling algorithm based on classification mining in fog computing environment," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 2102348, 11 pages, 2018.

[13] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 7, Article ID e5581, 2020.

[14] T. Nguyen Gia, I. B. Dhaou, M. Ali et al., "Energy efficient fog-assisted iot system for monitoring diabetic patients with cardiovascular disease," *Future Generation Computer Systems*, vol. 93, pp. 198–211, 2019.

[15] S. He, B. Cheng, H. Wang, Y. Huang, and J. Chen, "Proactive personalized services through fog-cloud computing in large-scale iot-based healthcare application," *China Communications*, vol. 14, no. 11, pp. 1–16, 2017.

[16] S. Ali and M. Ghazal, "Real-time heart attack mobile detection service (rhamds): an iot use case for software defined networks," in *Proceedings of the 2017 IEEE 30th Canadian conference on electrical and computer engineering (CCECE)*, pp. 1–6, Windsor, Canada, April 2017.

[17] G. Manogaran, R. Varatharajan, D. Lopez, P. M. Kumar, R. Sundarasekar, and C. Thota, "A new architecture of internet of things and big data ecosystem for secured smart healthcare monitoring and alerting system," *Future Generation Computer Systems*, vol. 82, pp. 375–387, 2018.

[18] H. A. Khattak, H. Arshad, S. ul Islam et al., "Utilization and load balancing in fog servers for health applications," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–12, 2019.

[19] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, 2016.

[20] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, "Container-as-a-service at the edge: trade-off between energy efficiency and service availability at fog nano data centers," *IEEE wireless communications*, vol. 24, no. 3, pp. 48–56, 2017.

[21] S. Aljanabi and A. Chalechale, "Improving iot services using a hybrid fog-cloud offloading," *IEEE Access*, vol. 9, pp. 13775–13788, 2021.

[22] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[23] B. Varghese, N. Wang, D. S. Nikolopoulos, and R. Buyya, "Feasibility of fog computing," in *Handbook of Integration of Cloud Computing, Cyber Physical Systems and Internet of Things*Springer, New York, NY, USA, 2020.

[24] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar, "A survey on iot security: application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82721–82743, 2019.

[25] S. Tuli, N. Basumatary, S. S. Gill et al., "Healthfog: an ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments," *Future Generation Computer Systems*, vol. 104, pp. 187–200, 2020.

[26] S. S. Chawathe, "Monitoring iot networks for botnet activity," in *Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pp. 1–8, Cambridge, MA, USA, November 2018.

[27] Y. Zhou, F. R. Yu, J. Chen, and Y. Kuo, "Virtual resource allocation for information-centric heterogeneous networks with mobile edge computing," in *Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 235–240, Atlanta, GA, USA, May 2017.

[28] M. M. Shurman and M. K. Aljarah, "Collaborative execution of distributed mobile and iot applications running at the edge," in *Proceedings of the 2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, pp. 1–5, Ras Al Khaimah, United Arab Emirates, November 2017.

[29] F. Liang, W. Yu, D. An, Q. Yang, X. Fu, and W. Zhao, "A survey on big data market: pricing, trading and protection," *Ieee Access*, vol. 6, pp. 15132–15154, 2018.

[30] F. S. Abkenar, Y. Zeng, and A. Jamalipour, "Energy consumption tradeoff for association-free fog-iot," in *Proceedings of the ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, Shanghai, China, May 2019.

[31] D. Hoang and T. D. Dang, “Fbrc: optimization of task scheduling in fog-based region and cloud,” in *Proceedings of the 2017 IEEE Trustcom/BigDataSE/ICESS*, pp. 1109–1114, Sydney, Australia, August 2017.

[32] W. Liang, J. Long, T.-H. Weng, X. Chen, K.-C. Li, and A. Y. Zomaya, “Tbrs: a trust based recommendation scheme for vehicular cps network,” *Future Generation Computer Systems*, vol. 92, pp. 383–398, 2019.

[33] K. Zhang, C. Shen, Q. Gao, and H. Wang, “Research on similarity metric distance algorithm for indoor and outdoor firefighting personnel precision wireless location system based on vague set on uwb,” in *Proceedings of the 2017 IEEE 17th International Conference on Communication Technology (ICCT)*, pp. 1162–1165, Chengdu, China, Octomber 2017.

[34] X.-Q. Pham and E.-N. Huh, “Towards task scheduling in a cloud-fog computing system,” in *Proceedings of the 2016 18th Asia-Pacific network operations and management symposium (APNOMS)*, pp. 1–4, Paris, France, May 2016.

[35] S. Kabirzadeh, D. Rahbari, and M. Nickray, “A hyper heuristic algorithm for scheduling of fog networks,” in *Proceedings of the 2017 21st Conference of Open Innovations Association (FRUCT)*, pp. 148–155, Helsinki, Finland, November 2017.

[36] D. Rahbari and M. Nickray, “Scheduling of fog networks with optimized knapsack by symbiotic organisms search,” in *Proceedings of the 2017 21st Conference of Open Innovations Association (FRUCT)*, pp. 278–283, Helsinki, Finland, November 2017.

[37] J. Wang and D. Li, “Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing,” *Sensors*, vol. 19, no. 5, p. 1023, 2019.

[38] K. Braiki and H. Youssef, “Resource management in cloud data centers: a survey,” in *Proceedings of the 2019 15th international wireless communications & mobile computing conference (IWCMC)*, pp. 1007–1012, Tangier, Morocco, June 2019.

[39] M. M. Hassan, K. Lin, X. Yue, and J. Wan, “A multimedia healthcare data sharing approach through cloud-based body area network,” *Future Generation Computer Systems*, vol. 66, pp. 48–58, 2017.

[40] S. R. Hassan, I. Ahmad, S. Ahmad, A. Alfaify, and M. Shafiq, “Remote pain monitoring using fog computing for e-healthcare: an efficient architecture,” *Sensors*, vol. 20, no. 22, Article ID 6574, 2020.

[41] M. Singh, B. B. Singh, R. Singh et al., “Quantifying covid-19 enforced global changes in atmospheric pollutants using cloud computing based remote sensing,” *Remote Sensing Applications: Society and Environment*, vol. 22, Article ID 100489, 2021.

[42] R. Kumar and R. Goyal, “Top threats to cloud: a three-dimensional model of cloud security assurance,” in *Computer Networks and Inventive Communication Technologies*-Springer, New York, NY, USA, 2021.

[43] Y. Hou, Q. Li, C. Zhang et al., “The state-of-the-art review on applications of intrusive sensing, image processing techniques, and machine learning methods in pavement monitoring and analysis,” *Engineering*, vol. 7, no. 6, pp. 845–856, 2021.