Fog Managed Data Model for IoT based Healthcare Systems

Benila S.[1*], Usha Bhanu N.[2]

[1] Department of Computer Science and Engineering, SRM Valliammai Engineering College, India
[2] Department of Electronics and Communication Engineering, SRM Valliammai Engineering College, India
sbenila@gmail.com, ushabhanu123@gmail.com

Abstract

In Internet of things enabled healthcare system, sensors create vast volumes of data that are analyzed in the cloud. Transferring data from the cloud to the application takes a long time. An effective infrastructure can reduce latency and costs by processing data in real-time and close to the user devices. Fog computing can solve this issue by reducing latency by storing, processing, and analyzing patient data at the network edge. Placing the resources at fog layer and scheduling tasks is quite challenging in Fog computing. This paper proposes a Fog Managed Data Model (FMDM) with three layers namely Sensor, Fog and cloud to solve the aforementioned issue. Sensors generate patient data and that are managed and processed by Fog and cloud layers. Tasks are scheduled using a Weighted Fog Priority Job Scheduling algorithm (WFPJS) and fog nodes are allocated based on Priority based Virtual Machine Classification Algorithm (PVCA). The performance of this model is validated with static scheduling techniques with variable patient counts and network configurations. The proposed FMDM with WFPJS reduces response time, total execution cost, network usage, network latency, computational latency and energy consumption.

Keywords: Cloud, Fog, Healthcare, Internet of things, Scheduling

1 Introduction

A recent International Data Corporation study predicts that the number of sensors and Internet-of-things (IoT) units would expand to 300 million and 500 million, respectively [1-2]. The IoT industry is expected to be worth $17 billion globally by 2022 [1, 3]. Approximately 30.7 percent of IoT devices will be used in healthcare. Bulk data processing is currently being used in healthcare via a large number of IoT devices. The Internet of Things in health care is mostly based on cloud computing [4]. To facilitate IoT, cloud computing may provide on-demand services such as networking, storage, processing, and high-performance computing. Cloud computing minimizes the complexity of healthcare IoT devices by reducing battery-draining processing tasks [5-6]. It does, however, have flaws such network congestion, insufficient bandwidth utilization and security issues. As the transmission rate increases in tandem with the vast volume of data, so does the latency in cloud computing. A massive amount of data is transmitted between sensors and the cloud in the healthcare IoT. An increase in the strength or quantity of data generated can increase the delay to destination. As a result, the probability of error is high in healthcare IoT with cloud due to packet loss and transmission delay, which is directly proportional to the volume of data transmitted. So, the destination's quality of service (QoS) is low.

Latency imposed by sending tasks to the cloud and subsequently returning the result is undesirable in certain time-critical internet of things applications, such as telemedicine and monitoring. Time-sensitive decisions will be made much closer to IoT devices. Healthcare infrastructure requires real-time data processing infrastructure all the time for time-sensitive applications. The most fundamental goal of healthcare IoT is to achieve the least feasible delay while also minimizing network bandwidth consumption [6-7]. Routers and gateways act as a point of contact between clouds and terminals. As the distance between clouds and healthcare IoTs increases, more routers are necessary to connect them, resulting in increased high network usage and significant bandwidth consumption.

To overcome the constraints of cloud computing, Cisco developed fog computing [8]. Fog computing, sometimes referred to as fog networking, is a decentralized computer architecture that resides between the cloud and data-generating devices [9]. It is comparable to cloud computing, but employs local clouds to process and execute activities in real-time, permitting IoT applications to run on network elements rather than cloud data centers. Fog computing has many benefits for IoT applications, including: increase the efficiency, transmit sensitive information to customers in real-time, solve data center disruptions, real-time interactive services, real-time mobility support, and automatic deployment. This adaptable framework lets users to optimize performance by placing resources, such as applications and the data they generate, in the user edge. The fog layer filters data, performs preprocessing, and reduces network usage and processing overhead in the cloud.

Fog computing and edge computing are variations of network designs that can be used to address cloud computing difficulties. Because of their low latency and modest bandwidth requirements, these network topologies can protect sensitive data. Edge Computing is a platform for extending cloud services to edge devices [10]. In order to reduce network latency and traffic, edge nodes and devices with computing power can undertake a vast range of computer functions such as data processing, storage of data, device management, decision-making, and privacy protection. The IoT network core oversees intelligent device connection and operation,

*Corresponding Author: Benila S.; E-mail: sbenila@gmail.com
DOI: 10.53106/160792642022032302003

while its edge computing provides sensing, interaction, and control between objects. Edge computing and fog computing have a common purpose, yet there are some fundamental distinctions between them. As a result of restricted resources in edge computing, resource conflict might arise, leading processing latency to increase. In healthcare systems, global analytics is required for prediction and decision making. Edge computing devices will not connect with cloud servers and integrated healthcare services are not provided. Fog computing can overcome these restrictions by incorporating cloud resources and edge devices.

Healthcare monitoring and applications that care for elderly patients, such as residential care, cardiovascular disease, chronic heart disease patients, hypertension, diabetic, and other chronic illnesses, all require real-time data analysis, real-time monitoring, and decision making. One of the most essential difficulties in fog based healthcare environments is how to sort out tasks and how to deal with sensitive data that is vital to the health of the patient. Until scheduling algorithms can be adjusted to work with healthcare applications, fog computing will not be able to meet these criteria.

.The following are the paper's main contributions:

- A three-module design is proposed that includes IoT, fog, and cloud modules to provide low latency and real time healthcare services.
- At the fog layer a Fog Managed Data model is integrated with Weighted Fog Priority Job Scheduling algorithm for efficient job allocation to fog nodes.
- Priority based virtual machine classification algorithm to allocate virtual machine at the cloud server for efficient processing of patient data.
- The proposed work is compared with existing cloud based algorithm and it reduces the total latency between healthcare IoT and cloud servers.

The rest of the paper is structured as follows: The associated literature reviews for healthcare IoT based on fog computing is described in Section 2. Section 3 describes the Fog Managed Data Model with Weighted Fog Priority Job Scheduling Algorithm and Priority based Virtual machine Classification Algorithm. Section 4 describes the experimental setup and performance evaluation. The results obtained through FMDM are summarized and discussed in section5. This paper comes to a conclusion in section 6.

2 Literature Review

A combined correlated sensor, fog and cloud based model is essential for offering efficient computed service to patients with real-time data, for delivering health-care and other delay-sensitive reports with shorter response time, tolerable energy usage, and better accuracy. Various techniques have been proposed to reduce energy, bandwidth consumption, latency reduction and network usage. Major contributions provided below are on the basis of load balancing, scheduling and offloading. A work was inspired by combining the power of accuracy of deep learning models with the reduced latency of edge computing nodes. IoT-based healthcare system connects preconfigured devices to process data from several patients in a deadline-oriented approach to make better decisions. Patients' healthcare data is collected using a variety of technologies, including IoT sensors and satellites [11-12]. In this approach, the health-care data is divided into two types: small data processed on a fog server and huge data handled on a centralized cloud repository.

A three-layer architecture was proposed as a combination of the terminal, edge, and core layers [13]. In the first level, resources are scheduled between fog groups, whereas in the second level, resources are scheduled between fog nodes inside the same fog group. To manage IoT resources, a random scheduling approach is implemented. The proposed model had a shorter service delay and a more stable task execution. However, scheduling across separate fog clusters, as well as the costs of resource distribution are not explored in this work. A priority based scheduling process with Optimize Response Time and Reconfigure Dynamic algorithms are implemented to minimize the cost of communication and latency [14]. The experimental results show that both methods cut expenses by considering priority in a cloud-only simulation scenario, but they also increase overall response time.

A hierarchical task allocation system based on the Smart Sensor-Fog-Cloud was proposed, in order to avoid job starvation and make the most of the available resources, it was able to dynamically modify the job priority [15]. By implementing a multi-core and multi-thread technique, the resource fragmentation and hunger issues are solved, optimize resource deployment, and lower the execution and reaction time of applications.

For energy consumption, resource utilization, and latency, multi-tier structure fog-based network architecture for internet of everything was proposed [16]. Load balancing, scheduling, security and privacy were not taken into account. The volume and velocity of IoT device-generated healthcare data is enormous. Both forms of data must be used to anticipate the present status of patients, which provides to the entire development of smart cities [17-18]. After collecting and aggregating data from smart devices in IoT networks, cloud servers store and analyze data. High latency is noted as an issue for time-critical IoT applications, and a solution termed iFogStor [19] is proposed to alleviate it. The Fog Computing concept was used to develop this technique. The data placement problem was characterized as a Generalized Assignment Problem (GAP) in iFogStor. For time-sensitive IoT applications, an accurate integer programming and heuristic technique is given, which needs a more exact model and architecture.

An algorithm for machine learning [20] is presented to conserve energy and bandwidth use, as well as network utilization, for mobile devices. In addition, cloudlet computation offloading and computational jobs are examined in the cloudlet environment. In smart cities, a mobile cloud computing-based method is proposed for delivering ubiquitous healthcare services that need acquiring and analyzing patient data at any time and from any location. Network slowness, high bandwidth utilization, and reliability are all obstacles to the adoption of cutting-edge healthcare applications. To solve these issues, the authors created UbeHealth [21], a healthcare system. To satisfy the potential and restrictions of a smart health monitoring system, an edge computing solution is proposed. Multi-access Edge Computing [22] is used in this method to detect, track, and monitor a patient's health history in a cost-effective manner. Furthermore, two separate features are proposed in order to construct an efficient, highly trustworthy, and low-distortion system. However, this method fell short of highlighting the data management challenges in the smart health industry. HealthFog, a smart healthcare system based on ensemble deep learning for the automatic diagnosis of cardiovascular diseases in an integrated IoT and fog computing environment is proposed [23]. It can be configured to a variety of operation modes to deliver the greatest Quality of Service or forecast accuracy depending on the circumstance and the needs of the user.

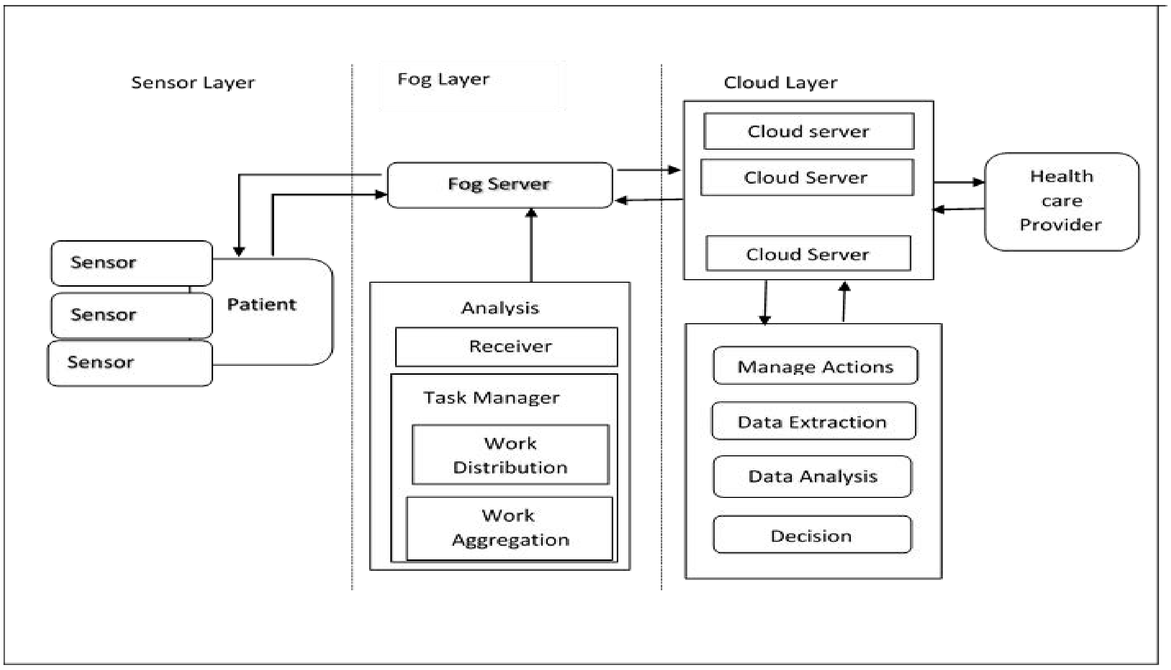

Figure 1. Fog managed data model

3 Proposed Methodology

To provide dependable healthcare services, fog, IoT, and cloud are linked into modified three-layer architecture. This architecture can be used to build an intelligent system for the user-closer edge network. These nodes serve as a connection point for numerous sensors or IoT devices. The critical components of the architecture are as follows:

- Sensors or Internet of Things devices
- Fog server
- Cloud Layer

3.1 Architecture

There are three layers in the proposed architecture. Figure 1 depicts the elements of the system. Sensor layer receives the data from the patients through the sensors. The fog server receives the information from the body area sensors. The cloud server's responsibilities include data analysis and report generating.

3.1.1 Senor Layer

There are three sorts of sensors in this component: activity sensors, medical sensors and environmental sensors. Medical sensors include electrocardiogram (ECG) sensors, electromyogram (EMG) sensors, electroencephalogram (EEG) sensors, temperature sensors, respiration rate sensors, oxygen level sensors and glucose level sensors. The sensor device collected two sorts of data: intrinsic and intrinsic. Extrinsic information includes things like temperature and location. Intrinsic data includes things like blood pressure, glucose levels, and heart rate. This component detects data from patients and sends it to the gateway devices that are connected. The data that has been submitted is then processed. The fog server receives all of the collected data.

3.1.2 Fog Layer

The real-time data from IoT sensors is received and analyzed by the real-time application which is shifted to the fog server. There are different types of Gateway devices such as mobile phones, laptop and tablets, which are acting as a fog server to collect sensed data from different sensors and forward this data to fog nodes for further processing. Through the use of multiple IoT sensors, the fog server is able to monitor a broad range of information. A Fog Managed Data Model (FMDM) is implemented in Fog Layer to efficiently manage and distribute data to fog clusters and cloud servers. A receiver for receiving sensor data and a task manager make up the FMDM. Work distribution and aggregation modules make up the task manager. All of the collected data are loaded into the fog server, which uses a job scheduling algorithm to determine how much work to assign to each cluster and when to offload the data to cloud server for processing. The major components of the Fog layer includes: Receiver and Task manager.

(1)Receiver
The receiver collects all the information from the body area network. The receiver includes a data storage which can store structured, unstructured and semi structured data received from medical sensor, activity sensor and environmental sensor. It will verify that the message is accurate and then send it to task manager for scheduling.

(2)Task Manager

A task manager is in charge of overseeing the workload. It is also known as workload manager, which is in charge of the workload generated by patient data. Dependent on the amount of work that needs to be done, the patient information are handled. There are two components in the task manager namely the work aggregation and work distribution.

• Work Aggregation
The task manager collects all the data from fog servers and the work is aggregated. The mapping is done in the work aggregation process in such a way that the work is converted in to a correct format for the data flow. Redundant data are removed from the collected data streams.

• Work Distribution
The smart gateway is used to distribute work with the help of a job scheduler. For this distribution process, the Weighted Fog Priority Algorithm is applied (Algorithm1). Each task from patient has a weight, current health condition and Disease severity. Based on these values priority value is calculated by the fog server and send to fog clusters for further processing. The cloud server time is determined by each fog server's processors and based on the priority.

3.2.1 Steps Involved in FMDM

The step-by-step approach to handle the sensor data for processing by the fog-managed data models (FMDM) is given below.
Step 1: Collect patient's medical information
Step 2: Let inP1, inP2, inP3, ... inPx are input derived from sensor data.
Step 3: Sensors send current reading of the inPx to the fog server.
Step 4: Aggregate data in Fog server. The works include

```
    4.1.1: Receive inPx, via the receiver, Rx.
    4.1.2: The task manager calculates weight and priority of the received data.
    4.1.3: Work distribution is done at the work distribution station.
    4.1.4, Weighted Fog Priority Job Scheduling algorithm is applied to prioritize the queue.
    4.1.5Work is dispatched to the queue.
    4.1.5 Calculate Computing demand for the task
            If the computing demand< computing resources
            in fog
               Assign task to fog cluster
                 else if computing demand > Computing
                 resources
                 Transfer to cloud servers
```

Step 5: Analyze, Process data, save report in memory for future.
Step 6: Send notifications to care providers.

The flow chart for FMDM is given in Figure 2.

3.2.2 Weighted Fog Priority Job Scheduling (WFPJS) Algorithm

The weight is calculated and assigned for job. The calculated weight used for prioritize the task in the queue. The task, which has the highest weight, is the first job in queue. The extracted medical record contains the details of the patients. The details of patients like name, age, gender, blood sugar level, blood pressure, pulse count, oxygen level, potassium, sodium etc. The WFPJS algorithm steps are given in Figure 3 and in Algorithm1.

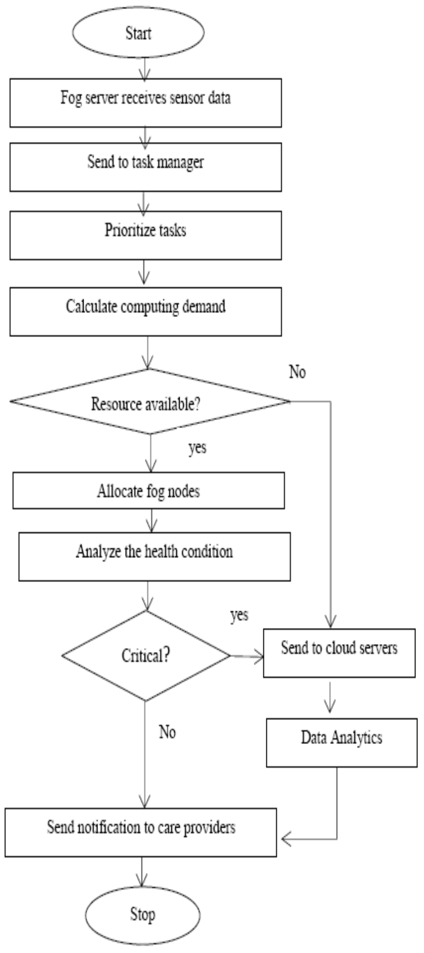

Figure 2. Flow chart for FMDM

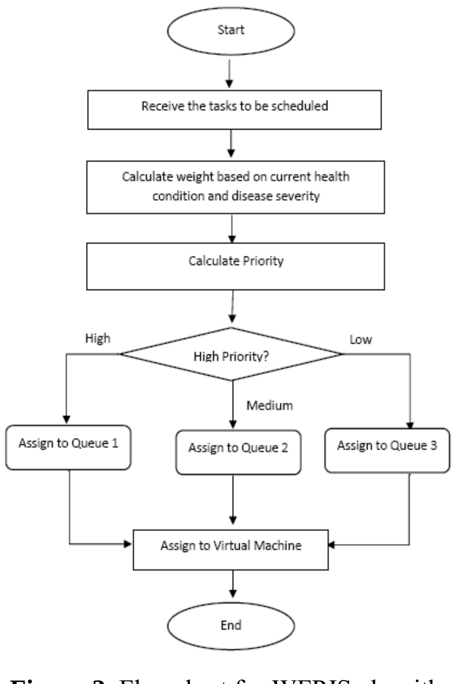
Start
Receive the tasks to be scheduled
Calculate weight based on current health condition and disease severity
Calculate Priority
High
High Priority?
Low
Medium
Assign to Queue 1
Assign to Queue 2
Assign to Queue 3
Assign to Virtual Machine
End

Figure 3. Flowchart for WFPJS algorithm

Algorithm 1. Weighted Fog Priority Job Scheduling (WFPJS) Algorithm

```
1.   input: information on the patient's medical
     condition
2.   output: Job send to fog/ cloud based on priority.
3.   start
4.   parameter initialization;
5.   data transmission to the fog layer;
6.   while
7.         if ((Current Health Condition&&
     High Disease Severity) then
8.         calculate weight  ;
9.               if (weight= high) then
10.              calculate priority;
11.             end
12.      end
13.  Ei(t)= Emergency Factor;
14.  apply priority function; ʒm
15.  if (High priority) then
16.    send to Priority queue q1
17.      else if(medium priority) then
18.           send to Priority queue q2
19.               else if( low priority)
20.               send to Priority queue q3
21.                    end
22.        end
23.      end
24.  send the data to fog/ cloud layer
25.  end while
26.  end
```

Priority queues q1, q2 and q3 are used to group the scheduled jobs according to their importance. The next responsibility for the fog server is to distribute the workload among the fog clusters/cloud servers according to their computational power and response time. For each task in the priority queue, the fog server must figure out how much computational power is required. The scheduled tasks are sent to the fog clusters via the fog server. This massive amount of data will be processed with the resources that have already been set aside. The data is split up into numerous fog clusters for processing in the background.

There are several different types of fog clusters, each with a different processing capacity and turnaround time. While distributing the tasks, the capacity and turnaround time of fog cluster virtual machines are determined. Virtual machines with high processing power and quick turnaround time are assigned to work on the most important tasks first. The virtual machines are classified using the Priority-based Virtual machine Classification Algorithm (PVCA).

The PVCA algorithm includes following steps

Phase1: Categorize virtual Machines based on cost and turnaround time (Fog clusters)

Phase2: Allocate Virtual Machines based on priority of tasks

In phase 1, the virtual machines are clustered based on the capacity and turnaround time. Set of virtual machines with high capability and low turnaround time are grouped to cluster1, set of virtual machines with medium capability and turnaround time are grouped to cluster 2 and cluster3 contains set of virtual machines with low capability and high turnaround time. These virtual machine sets are assigned with different levels of priority such as high, medium and low.

Phase 1: Clustering of Virtual Machines

Let T={ T1, T2……Tn} are tasks and X={ X1,X2….Xn} are virtual machines.

```
for all Ti ∈ L
  do
for all Xn ∈ N
    do
       Compute  turnaround time Ttat (Ti , Xn)
       if( capacity= high && Ttat = low)
            assign VM to cluster 1
            else if (capacity=medium  && Ttat = medium)
                 assign VM  to cluster 2
                 else if(capacity= low  && Ttat = low)
                      assign VM to cluster 3
```

In phase 2, the tasks scheduled in priority queue are assigned to the virtual machine clusters based on the priority assigned to the virtual machine grouping.

Phase 2: Allocation of Virtual Machines

```
Input: Priority assigned tasks in priority queue
Output: Allocate VM to priority queue
For each task Ti  in Q
  if ʒm = High
     invoke cluster 1 VM
    else if  ʒm = Medium then
      invoke cluster 2 VM
      else if  ʒm = Low then
           invoke cluster3 VM
```

Even with a large number of patients requesting services, fog clusters cannot handle the high computational demands and integrated data analytics required to handle them. These tasks are delivered to the cloud and processed there. Data must be offloaded from the fog server to cloud servers.

3.2.3 Cloud Layer

To extract insight from raw data, clustering and classification techniques are used. All of these operations are carried out based on the weighted priority. After all of the data processing has been completed, the data will be aggregated and a final choice will be made. Medical practitioners receive reports, which they use to investigate and treat patients. The networks are made and connected along with security setting. The patient data are transmitted without any data loss and also no data manipulation.

4 Experimental Setup and Evaluation of Performance

All of the scheduling and categorization algorithms are deployed in Java JDK 1.8 running on a Dell Inspiron 15 3000 Series Core i3 processor with 4GB RAM and 2.29 GHz clock. A real-world Internet of Things (IoT) testbed is difficult and expensive to build, and it doesn't provide a reproducible setting for laboratory studies. Because of this, iFogSim, an open-source simulator is used. Modeling and simulating fog computing systems with iFogSim allows for resource management and scheduling strategies across edge and cloud resources to be evaluated under various scenarios for latency, energy usage, network congestion, and operational expenses. For simulation, iFogSim functionalities have been integrated with CloudSim [24-25]. iFogSim communicates with CloudSim by sending a message signal in the middle of the datacenter. As a result, CloudSim is in charge of managing the various functions amongst the fog computing components. ISP Gateway, Smartphone and WiFi gateway can act as fog server. Fog server configuration details are listed in Table 1. Virtual machines are used to model patient data. The RAM capacity, CPU speed and processing power are listed in Table 2 for various fog device configurations. To make fog devices, we developed a cloud simulation environment that extended the datacenter class and VM is used to simulate patient data. Cloudlets have been extended so that user requests could be executed. The performance of this model is tested with different number of patients and different number of network configurations. The cost parameter per unit is listed in Table 3.

Table 1. Fog server configuration

Type of device	RAM(GB)	CPU(GHz)	Power (W)
ISP Gateway	6	4	117.445
Smartphone	2	2.6	88.64
WiFi Gateway	6	4	117.445

Table 2. Virtual machine configuration

Virtual Machine	Cluster1	Cluster 2	Cluster 3
RAM (GB)	5	4	2
Processing Power (MIPS)	22000	15000	11000
Bandwidth (Mbps)	1024	1024	512

Table 3. Cost parameters

Cost Parameters	Value
Communication cost of fog – cloud	0.5
Processing cost per time unit	0.2
Communication cost per data unit	0.3
Unit cost of memory used	0.1
Unit cost of storage used	0.2

Weight and priority of the patient data are calculated as a function of current Health condition, High disease severity and the emergency factor.

Weight is calculated as,

$$Wi(t)=CHCi(t)*HDSi(t) \tag{1}$$

Where, Wi is the weight assigned, CHCi is the current health condition of the patient and HDSi is the high disease severity.

The priority function is given as,

$$ʒm = Wi(t)/e^{-Wi(t))* Ei(t)} \tag{2}$$

Where, Ei(t) is the emergency factor. Highest the value of the ʒm, first enters in to the queue and process in the server. Level of priority varies as $ʒm_H$, $ʒm_M$, $ʒm_L$ are high, medium, low respectively. The ʒm is order descending way. The highest the CHC & HDS, the ʒm is also high. The highest the ʒm enters the fog server. If the same ʒm for different patients, the priority is given according to the time of arrival of patient data to the fog server. First come first server is applicable for the same ʒm patients within the priority level. The values of CHc, HDS, E, ʒm are given in Table 4.

Table 4. Values of CHc, HDS, E, ʒm

CHC	HDS	E	ʒm
0.2	0.2	0.1	0.02
0.9	0.2	0.1	0.12
0.9	0.9	1	0.57

Turnaround time of VM is calculated as,

$$Ttat= Wt + Ttet \tag{3}$$

Wt represents the waiting time and Ttet represents total execution time.

Ttet is calculated as,

$$Ttet = Tete- Tste \tag{4}$$

Where, Tete is the end time of task execution and Tste is start time of task execution.

Network latency of FMDM can be calculated based on the number of hops travelled per user request. If each hop assumes the same latency, the total number of packets transferred from an edge node (e_n) to a fog node (f_n) and from f_n to another f_l and f_l to e_n.

$$Fog_{NL} = \frac{D_u H_n(e_n+f_n+f_l)}{P_c} \tag{5}$$

Where, H_n is the number of hops, P_c is the total number of data packets sent, and D_u is the unit hop delay.

Computation latency can be calculated as a function of waiting time and service time. Assuming a queuing system, for the fog device f_i with the traffic arrival rate a_{ri}
and service rate s_{ri}, the computation latency is given in (6) that involves queuing delay while waiting for service, service time, and time taken for virtual machine installation.

$$fi_{CD} = \frac{1}{s_{ri}-a_{ri}} \tag{6}$$

The data passes through multistage fog computations involving inter-fog communication delay (fl_d).Total computation latency(fi_{TCD}) is given by,

$$fi_{TCD} = \frac{1}{s_{ri} - a_{ri}} + fl_d \qquad \textbf{(7)}$$

Cost of Computation (C_{fi}) in Fog layer is calculated as,

$$C_{fi} = U_{cfi} + R_d(l_{Ti}) \qquad \textbf{(8)}$$

Where, U_{cfi} is the unit cost of the VM and the R_d is the running duration of virtual machine for particular task. UC includes processing cost per time unit, Communication cost per unit cost, Unit cost of memory used, Unit cost of storage used. Running time varies based on task length (l_{Ti}) for FMDM.

Energy Consumption is calculated for FMDM and cloud based models. As the tasks are processed at user premises with fog nodes the energy consumption is low in FMDM compared with cloud based models.

5 Results and Discussion

5.1 Execution Time

The Execution time is the overall time taken to process the task at the fog server. As the tasks are preprocessed and scheduled by fog server and assigned to fog virtual machines directly, there is a reduction in execution time. As the number of requests coming to the cloud is reduced, the overall execution time of tasks is cloud also get reduced. The execution time of FMDM with WFPJS algorithm is compared with the existing cloud based FCFS scheduling. The experimental results show overall reduction in execution time. The values are listed in Table 5 and Figure 4.

Table 5. Execution time

No. of Cloudlets	Sequential Algorithm FCFS	FMDM with WFPJS
10	48.807	22.1565
20	97.614	44.313
30	146.421	66.4695
40	195.228	88.626
50	244.035	110.7825
60	292.842	132.939
70	341.649	155.0955
80	390.456	177.252
90	439.263	199.4085
100	488.07	221.565

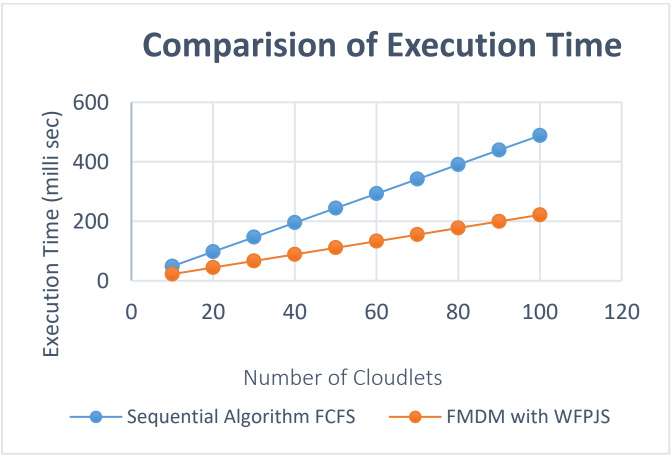

Figure 4. Comparision of execution time

5.2 Total Cost of FMDM

Most of the pre-processing are done in fog server itself and cloud server which is located near to fog servers are selected for data processing. So the U_C cost is low in FMDM. The communication cost is negligible when compared with cloud based IoT data Processing. The total cost of computation is low in FMDM even though there are large number of patients and large number of cloudlets. The values are listed in Table 6 and Figure 5.

Table 6. Cost of computation

No. of Cloudlets	Sequential Algorithm FCFS	FMDM with WFPJS
10	293.872	225.1
20	587.744	450.2
30	881.616	675.3
40	1175.488	900.4
50	1469.36	1125.5
60	1763.232	1350.6
70	2057.104	1575.7
80	2350.976	1800.8
90	2644.848	2025.9
100	2938.72	2251

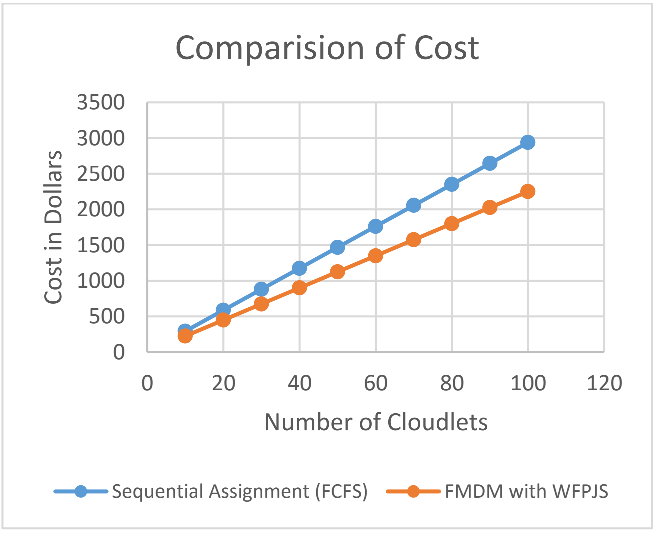

Figure 5. Cost of computation

5.3 Network Performance Metrics

The network parameters such as energy consumption, network latency, computational latency with respect to different configuration and with different number of patients has been measured for FMDM and cloud computing.

Figure 6 shows how FMDM and clouds change the amount of energy consumed. Reduced energy consumption can be seen with fog servers, whereas using more energy can be seen with cloud servers. The energy consumption of FMDM and clouds is depicted in Figure 3 as a function of the number of patients. The fog server consumes less energy when processing data, whereas the cloud server consumes more energy when processing data. When compared to a cloud server, even with a larger number of patients, the energy consumption is significantly lower. The FMDM can reduce the energy consumption with patient count.

Figure 7 depicts the fog and cloud computation latency as a function of the number of patients. During data analysis and management, the fog server minimizes calculation latency, but the cloud server requires some additional calculation latency during data processing. The computing latency is reduced as compared to a cloud server even when there are more patients.

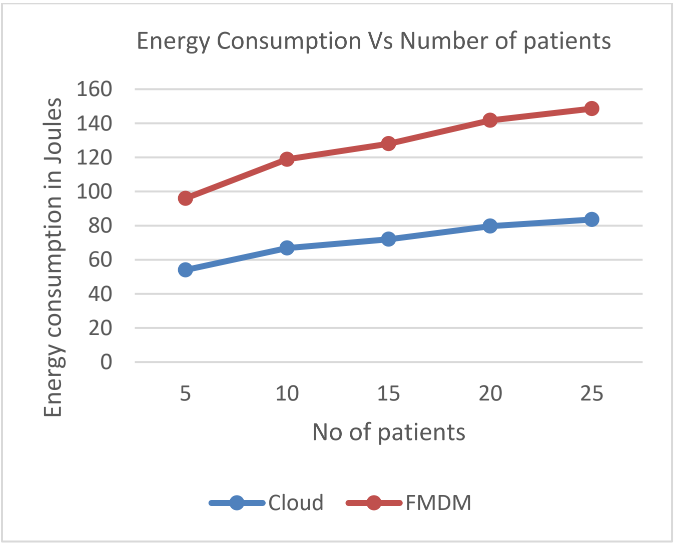

Figure 6. Energy consumption

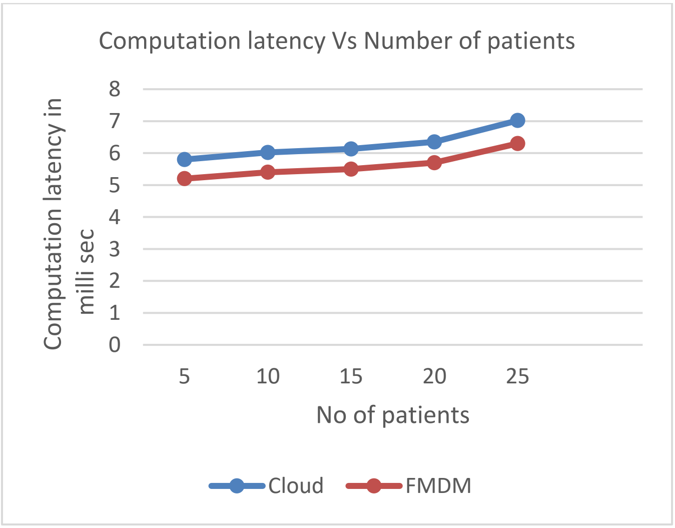

Figure 7. Computation latency

The network delay of FMDM and clouds is depicted in Figure 8 as a function of the number of patients. The fog server minimizes network latency during data processing, but the cloud server adds greater network delay during data processing. When compared to a cloud server, network latency is reduced even when the number of patients is doubled.

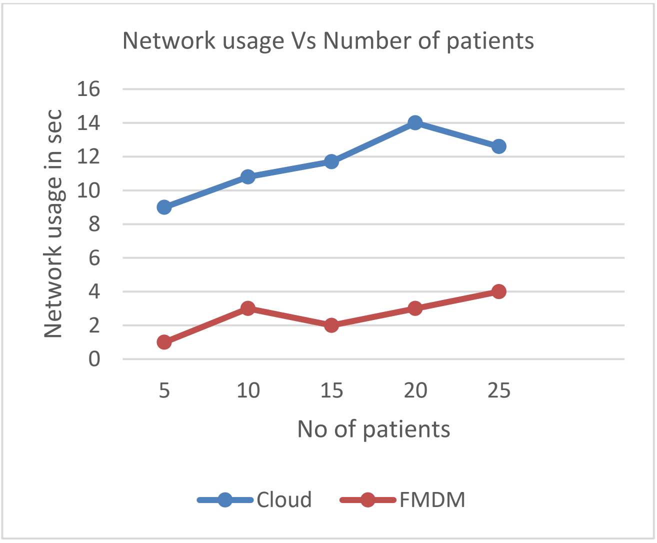

Figure 8. Network latency of fog and cloud

6 Conclusion

Real-time data processing and analysis are required for IoT-based health monitoring applications. Any time lost while transmitting data to the cloud and back to the application is intolerable. As a result, fog computing is introduced between sensors and cloud computing to collect and process data more efficiently. It reduces the amount of data transported between the sensors and the cloud servers and improves overall system efficiency. But this paradigm lags in task scheduling and task allocation. Traditional scheduling results in overpricing and processing delays. The proposed Fog Managed Data Model with Weighted Fog Priority Job Scheduling and virtual machine allocation scheme schedule task to fog virtual machines based on priority. The simulation results show that the proposed method improves response time while costing less than the existing static scheduling algorithms. The network parameter metrics for FMDM and cloud-based systems are compared. The simulation results show that the FMDM reduces network usage, computational latency, and network latency when the patient count and network configurations are varied. The system will be deployed with various edge devices in the future research, and the system's performance will be examined.

References

[1] S. S Gill, R. C. Arya, G. S. Wander, R. Buyya, Fog-Based Smart Healthcare as a Big Data and Cloud Service for Heart Patients Using IoT, *International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI)*, Coimbatore, India, 2018, pp. 1376-1383.

[2] M. M. Hassan, K. Lin, X. Yue, J. Wan, A multimedia healthcare data sharing approach through cloud-based body area network, *Future Generation Computer Systems*, Vol. 66, pp. 48-58, January, 2017.

[3] C. S. Nandyala, H. K. Kim, From cloud to fog and IoT-based real-time U-healthcare monitoring for smart homes and hospitals, *International Journal of Smart Home*, Vol. 10, No. 2, pp. 187-196, February, 2016.

[4] P. K. Sahoo, S. K. Mohapatra, S. L. Wu, SLA based healthcare big data analysis and computing in cloud network, *Journal of Parallel and Distributed Computing*, Vol. 119, pp. 121-135, September, 2018.

[5] X. Q. Pham, E. N. Huh, Towards task scheduling in a cloud-fog computing system, *18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Kanazawa, Japan, 2016, pp. 1-4.

[6] H. Khaloufi, K. Abouelmehdi, A. B. Hssane, Fog Computing for Smart Healthcare data Analytics: An Urgent Necessity, *NISS2020: Proceedings of the 3rd International Conference on Networking, Information Systems & Security* (*NISS2020)*, Marrakech, Morocco, 2020, pp. 1-5, April, 2020.

[7] S. Cirani, G. Ferrari, N. Iotti, M. Picone, The IoT hub: A fog node for seamless management of heterogeneous connected smart objects, *12th Annual IEEE International Conference on Sensing, Communication, and Networking Workshops (SECON*), Seattle, WA, USA, 2015, pp. 1-6.

[8] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, *Proceedings of the first edition of the MCC workshop on Mobile cloud computing, MCC'12*, Helsinki, Finland, 2012, pp. 13-16.

[9] S. S. Gill, R. Buyya, Resource provisioning based scheduling framework for execution of heterogeneous and clustered workloads in clouds: from fundamental to autonomic offering, *Journal of Grid Computing*, Vol. 17, No. 3, pp 385-417, September, 2019.

[10] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge Computing: Vision and Challenges, *IEEE internet of things Journal*, Vol. 3, No. 5, pp. 637-646, October, 2016.

[11] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, K. Kwak, The Internet of Things for health care: A Comprehensive Survey, *IEEE Access*, Vol. 3, pp. 678-708, June, 2015.

[12] Y. Sun, F. Lin, H. Xu, Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II, *Journal of Wireless Personal Communications*, Vol. 102, No. 2, pp. 1369-1385, September, 2018.

[13] A. V. Dastjerdi, R. Buyya, Fog Computing: Helping the Internet of Things Realize Its Potential, *IEEE Computer*, Vol. 49, No. 8, pp. 112-116, August, 2016.

[14] T. Choudhari, M. Moh, T. S. Moh, Prioritized task scheduling in fog computing, *Proceedings of the ACMSE Conference*, Richmond, Kenturcky, 2018, pp. 1-8.

[15] Z. Sun, C. Li, L. Wei, Z. Li, Min Z. Min, G. Zhao, Intelligent sensor-cloud in fog computer: a novel hierarchical data job scheduling strategy, *Sensors*, Vol. 19, No. 23, Article No. 5083, December, 2019.

[16] P. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, R. Buyya, FOCAN: A Fog-supported Smart City Network Architecture for Management of Applications in the Internet of Everything Environments, *Journal of Parallel and Distributed Computing*, Vol. 132, pp. 274-283, October, 2019.

[17] S. Tuli, R. Mahmud, S. Tuli, R. Buyya, FogBus: A Blockchain-based Lightweight Framework for Edge and Fog Computing, *Journal of Systems and Software*, Vol. 154, pp. 22-36. August, 2019.

[18] Cisco, Fog computing and the Internet of Things: Extend the cloud to where the things are, white paper 2015.

[19] M. I. Naas, P. R. Parvedy, J. Boukhobza, L. Lemarchand, iFogStor: An IoT data placement strategy for fog infrastructure, *IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, Madrid, Spain, 2017, pp. 97-104.

[20] H. Cao, J. Cai, Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach, *IEEE Transactions on Vehicular Technology*, Vol. 67, No. 1, pp. 752-764, January, 2018.

[21] T. Muhammed, R. Mehmood, A. Albeshri, I. Katib, UbeHealth: A personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities, *IEEE Access*, Vol. 6, pp. 32258-32285, June, 2018.

[22] A. A. Abdellatif, A. Mohamed, C. F. Chiasserini, M. Tlili, A. Erbad, Edge computing for smart health: context-aware approaches, opportunities, and challenges, *IEEE Networks*, Vol. 33. No. 3, pp. 196-203, May-June, 2019.

[23] S. Tuli, N. Basumatary, S. S. Gill, M. Kahanil, R. C. Arya, G. S. Wander, R. Buyya, 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, March, 2020.

[24] R. Buyya, S. N. Srirama, Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit, in: Fog and Edge Computing: Principles and Paradigms, Wiley STM, 2019, pp. 433-465.

[25] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. S. F. De Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Journal of Software Practice and Experience*, Vol. 41, No. 1, pp. 23-50, January, 2011.

Biographies

Benila S. received her B.Tech in Information Technology in 2006 and her M.E. in Network Engineering in 2008 from Anna University. She is an assistant Professor in the Department of Computer Science and Engineering at SRM Valliammai Engineering College, Tamilnadu, India. She is currently pursuing her Ph.D. degree at Anna University. Computer networks, the Internet of Things, and big data analytics are some of her research interests.

Usha Bhanu N. received her PH.D. in 2014, from the College of Engineering, Anna University, Chennai, India. She had completed her B.E. in Electronics and Communication Engineering in 1996 from Bharathiar University and M.E. in VLSI Design 2006 from Anna University. She is currently working as Professor in the Department of ECE in

SRM Valliammai Engineering College, Tamil Nadu, India. Her areas of research interest includes VLSI design, Signal and Image processing and Internet of Things (IoT).