**International Journal of Reliable and Quality E-Healthcare**
Volume 11 • Issue 4

# An Efficient Fog Layer Task Scheduling Algorithm for Multi-Tiered IoT Healthcare Systems

Ranjit Kumar Behera, National Institute of Science and Technology, Berhampur, India*
https://orcid.org/0000-0002-7408-4900

Amrut Patro, National Institute of Science and Technology, Berhampur, India

K. Hemant Kumar Reddy, National Institute of Science and Technology, Berhampur, India

Diptendu Sinha Roy, National Institute of Technology, Meghalaya, India

## ABSTRACT

IoT-based healthcare systems are becoming popular due to the extreme benefits patients, families, physicians, hospitals, and insurance companies are getting. Cloud is used traditionally for almost every IoT application, but cloud located far away from the devices resulted in an uncertain latency in providing services. At this point, fog computing emerged as the best alternative to provide such real-time services to delay-sensitive IoT applications. However, with the surge of patients, fog's limited resources may fail to handle the explosive growth in requests requiring advanced monitoring-based prioritization of tasks to meet the QoS requirements. To this end, in this paper, a level monitoring task scheduling (LMTS) algorithm is proposed for healthcare applications in fog to provide an immediate response to the delay-sensitive tasks with minimum delay and network usage. The proposed algorithm has been simulated using the Cloudsim simulator, and the results obtained demonstrated the efficacy of the proposed model.

## KEYWORDS

Cloud Computing, Delay Sensitive, Fog Computing, IoT, Level Monitoring, Smart Healthcare Systems, Task Scheduling

## INTRODUCTION

Recently, with the rising number of chronic diseases along with the rise in global population, researchers as well as healthcare industry are with a great hope of new smart healthcare solutions which can prove to be beneficial for both medical practitioners as well as patients in providing real-time services in case of medical emergencies. Despite of the fact that smart healthcare solutions can prove to be rewarding, yet, industries did not opt for such smart healthcare systems due to the exorbitant nature of IoT solutions. But latterly, in the age of pandemic like coronavirus, it has become a necessity for the industries to adopt such IoT based smart solutions for the betterment of humankind.

DOI: 10.4018/IJRQEH.308802

*Corresponding Author

The Internet of Things (IoT) is an environment of diverse objects having capabilities to inter-operate and communicate with each other over the network requiring minimum human interaction (Wortmann & Flüchter, 2015). In situations, when there is a need of processing large amount of data which requires high computing power, IoT devices alone cannot satisfy the requirements, hence uses Cloud to meet such requirements (Biswas & Giaffreda, 2014). Cloud computing (Mell & Grance, 2011) is a virtualization based powerful computing technology which provides on-demand services such as computing power, storage, applications etc over the internet. Notwithstanding the various benefits of Cloud, several inescapable issues and downside is also noticed. Cloud datacenters located far away from the users and the presence of high traffic in Cloud results in high latency, raising questions about execution of tasks requiring immediate responses. There comes the need of Fog.

Fog computing, is an intermediate layer of distributed network present in between the Cloud and IoT which is capable of providing Cloud services with a minimal delay to the real-time IoT applications, being close to the source of data (Bonomi et al., 2012). The need for Fog is better realized in healthcare systems, when large amount of requests arrives from delay-sensitive applications like wearable health monitoring devices, ventilators etc. In situations of pandemic or disasters, when there is an unpredictable surge in the number of patients, the requests may also increase vigorously. In such circumstances when there is lack of an efficient task scheduling model, fog computing circumscribed by resources may not be able to handle the high traffic and may fail to serve the delay-sensitive tasks which requires immediate response.

Noticing such issues and challenges in fog and healthcare applications, researchers have done a lot of studies to come out with an economical and competent model (Behera et al., 2020; Rahmani et al., 2018; Roy et al., 2018). The authors in (Tang et al., 2019), have proposed a fog based smart healthcare model, consisting of three layers, namely multisource data layer, heterogeneous Fog network, and healthcare service layer. A novel fog enabled data sharing strategy is also introduced where pre-processing and re-encryption of the transferred data is done on the employed nodes to reduce encryption burden for healthcare applications. However, the authors have not focused much on resource management. A better resource management methodology would have resulted in even less energy consumption and better performance. In (Mahmoud et al., 2018), the authors have proposed an energy-aware fog based model for healthcare, that allocates the tasks to the fog nodes based on remaining energy consumption and CPU capacity. The main goal of their proposed strategy is to minimize the overall energy consumption by allocating the tasks based on DVFS (Dynamic Voltage Frequency Scaling) and IRR (Improved Round Robin) algorithms. Although their model proved to be delivering promising results, but in case of healthcare scenarios providing immediate response to delay-sensitive tasks should be the primary goal. Searching for energy-efficient fog nodes to allocate tasks instead of allocating them to the nodes which can provide immediate response, would not prove to be a good strategy. Whereas in (Khattak & Arshad, 2019), authors have considered traffic overhead, introducing a load balancing strategy for health-care applications. Their model consists of an algorithm that is provided with heart rate and patient category according to which they categorize the request as normal or critical. The second algorithm checks whether a request sent to the nearest fog server is capable of processing or not. If not, it shifts the request to nearby servers until a server with enough resources is not encountered. In case the task is found to be critical, the employed fog server forwards it to the cloud for further processing. Forwarding critical tasks to Cloud for processing would result in disastrous situations due to high latency. Delays in health-care applications cannot be tolerated and their model would fail in providing immediate responses to the needy tasks. Proposal of a priority-based task scheduling algorithm is given by the authors in (Choudhari et al., 2018), where the arriving tasks are first assigned with one out of the three priority levels i.e., low, medium, and high priority. An arrived task after being assigned with its priority level is scheduled to the fog node present near to the client's location, the request is forwarded to the adjacent fog nodes until a node satisfying the task's deadline is found. In worst case, when no fog node is capable of executing the task within its deadline, even after splitting, it is ultimately sent to cloud. However, forwarding tasks to nearby fog nodes again and again would lead to an increase in service delay. It is also seen that, in situations of high traffic, tasks with high priority levels are rejected and sent to cloud which increases the probability of failure in executing tasks within their deadlines. While in (Jamil et al., 2020), a delay and performance-optimizing job scheduling algorithm for healthcare systems is proposed. Authors have considered three importance levels for tasks based on application category. Three application modules such as DCPB, Organizer and Patient's Record Database are considered in their proposed architecture. For every insertion, the waiting list is sorted in an ascending order based on the length of tasks. The Shortest Job First(SJF) scheduling policy is followed in their algorithm while scheduling tasks from the waiting list. Nevertheless, their algorithm performed better than the existing First Come First Service(FCFS)algorithm in terms of average loop delay, energy consumption and network usage still, a lot of unavoidable drawbacks can be seen. Though the authors have prioritized the tasks, yet the scheduling is not done on the basis of importance level. Critical or delay-sensitive tasks are not served with a higher importance while following SJF. Moreover, implementation of instruction set based SJF would be catastrophic for critical tasks with large instruction set. Allocating virtual machines with low computational power to a critical task would not be a good idea as they need to be executed with the least possible service time.

Hence, being mindful of such issues in fog, in this paper a level monitoring task scheduling algorithm in fog for healthcare is proposed to provide an immediate response to the delay-sensitive tasks with minimum energy consumption and network usage. Moreover, giving thought to the drawbacks in the aforementioned papers, a level assigning algorithm is also proposed and it is made sure that the tasks are served according to their respective importance levels while scheduling. Some of the major contributions of this paper are summarized as follows:

1. Initially a comprehensive case study on healthcare applications and the considerations of our model is provided. Followed by an overview of the proposed model.
2. A level monitoring task scheduling algorithm is designed for scheduling tasks to virtual machines where immediate response and minimum completion time can be achieved. The necessary mathematical equations are formulated for the proposed model.
3. Finally, using CloudSim, we evaluate the performance of our proposed algorithm in terms of average delay, energy consumption and network usage.

The remaining parts of the paper are organized as follows. Section II provides a comprehensive case study on healthcare applications and the considerations in our model. Section III scrupulously illustrates the proposed model. Moreover, the problem formulation and mathematical analysis is also done in this section. Section IV consists of a detailed discussion on the experimental setup followed by the result analysis of our model. Finally, the paper concludes showing the directions for future work in Section V.

## CASE STUDY ON HEALTHCARE APPLICATIONS

The ubiquitousness nature of fog makes it a well-qualified and an acceptable technology for various real-time IoT applications. Many real-time IoT applications require an instantaneous response. For example, in smart healthcare applications, seeking services from cloud would prove to be a disastrous decision due to unpredictable delays. Hence, fog being close to the source of data can be used as a solution for various healthcare computing tasks. Processing tasks in nearby fog nodes would provide immediate response and smaller delays to such real-time applications rather than sending them to cloud.

Albeit healthcare applications need to be served immediately, but there are many small small applications under the umbrella of healthcare that needs to be served with higher priorities. Some of them are delay-sensitive while others can tolerate some delay. For example, tasks from health monitoring applications like electrocardiogram (ECG) or ventilators are strictly intolerant to delays.

Whilst, tasks from hospital management systems can tolerate some delay. Therefore, such tasks also need to be prioritized based on the type of applications from which they have arrived. In this paper, three categories of applications are considered which are mentioned as follows:

1. **Critical Health Monitoring Applications:** This category consists of distinct health monitoring applications that require computation on health condition of patients containing data such as blood pressure, heartbeat, oxygen level, etc. Such tasks need to be processed immediately and hence are assigned with highest priority or task level 1.
2. **Appointment and Display Applications:** Tasks for appointment bookings, alerting users in case of appointment delays, display services, etc comes under this category. Applications akin to this category are comparatively less critical, hence can be assigned with medium priority i.e., task level 2.
3. **Management and Analysis Applications:** Applications containing patients information responsible for managing databases for long-term analysis are of this category. These tasks can tolerate delays, therefore, are assigned task level 3.

## PROPOSED MODEL

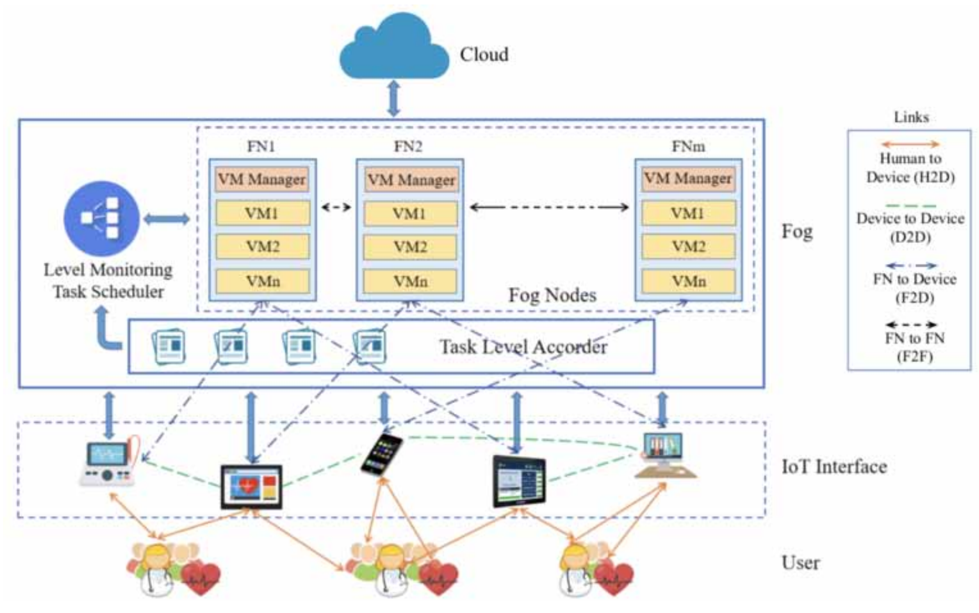
For a better realization of the proposed task scheduling mechanism, an overview of the model, sequence diagram, design as well as its implementation is scrupulously illustrated in the following subsections.

### Overview of the Model

The overall model is presented in this section with the help of Figure 1. It can be seen that the architecture consists of four layers namely User, IoT Interface, Fog, and Cloud. The user layer refers to the end-users of the services, for example, patients, doctors, managers, etc.

The second layer is the IoT Interface which comprises an ample mix of IoT applications such as smartphones, health monitoring devices, display devices, etc. The Fog layer consists of several

**Figure 1. Overview of the model**



fog nodes responsible for processing the tasks arriving from the end-users via the IoT interface. The uppermost layer is the Cloud having large centralized datacenters where patient's database records are maintained for long-term analysis. The Fog layer is equipped with additional components like Task Level Accorder (TLA), Level Monitoring Task Scheduler (LMTS), and Virtual Machine Manager (VMM). Whenever a task arrives from the IoT interface, it is first received by TLA which categorizes the task based on application type. The three task levels considered in our model are already discussed in Section II. TLA forwards the level assigned tasks to Level Monitoring Task Scheduler (LMTS) which appends them into a task list further scheduling them to a suitable virtual machine (vm) present in fog nodes, where minimum completion time can be achieved.

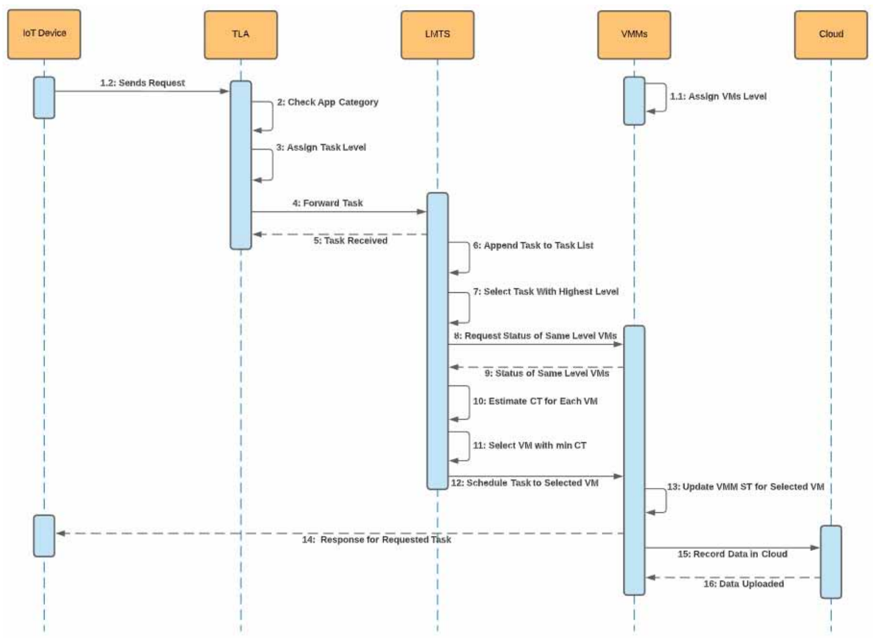
The virtual machines are assigned with three computational levels i.e. VM level 1 indicating most powerful, VM level 2 indicating moderately powerful and VM level 3 indicating least powerful. Every fog node incorporates several vms and a Virtual Machine Manager (VMM). The VMM maintains a VM Scheduling Table (ST). For 'n' number of virtual machines present in a fog node, the VMM maintains a ST with 'n' number of rows. Also, the VMM plays the role of completion time estimator which estimates the completion time for a particular task if it is scheduled to a vm present in the fog node with same level as that of the task.

## Sequence Diagram of the Model

The order in which various activities taking place in the model starting from the arrival of tasks till their scheduling is illustrated here with the help of a sequence diagram as shown in Figure 2.

In the first instance (Seq.No. 1.1) all the virtual machines present in various fog nodes are assigned with VM levels. The VM levels are assigned based on the computational power of the vms using Algorithm2 presented in the next subsection. At the start, it is also noticeable that the tasks arrive from various IoT (Seq.No. 1.2), and according to the category of application, TLA assigns task level to the arrived task (Seq.No. 2, 3). Algorithm1 presented in the next subsection depicts the behavior of TLA. Then the arrived task is forwarded to the LMTS (Seq.No. 4) which in return

**Figure 2. Sequence diagram of the model**


IoT Device
TLA
LMTS
VMMs
Cloud
1.2: Sends Request
1.1: Assign VMs Level
2: Check App Category
3: Assign Task Level
4: Forward Task
5: Task Received
6: Append Task to Task List
7: Select Task With Highest Level
8: Request Status of Same Level VMs
9: Status of Same Level VMs
10: Estimate CT for Each VM
11: Select VM with min CT
12: Schedule Task to Selected VM
13: Update VMM ST for Selected VM
14: Response for Requested Task
15: Record Data in Cloud
16: Data Uploaded

acknowledges TLA that the forwarded task is received (Seq.No. 5). The received task is further appended into an arrived task list maintained by the LMTS (Seq.No. 6). For each time a task is to be scheduled, the LMTS selects a task having the highest task level among tasks present in the arrived task list which means it first looks for the most delay sensitive ones, then it looks for tasks with task level 2, and 3 (Seq.No. 7). During (Seq.No. 8), the status of all vms such as VM level and estimated completion time is requested from the VMMs and hence they return the status of all vms from the maintained ST consisting of parameters like vm id, vm levels, scheduling queue and estimated completion time for each vm (Seq.No. 9). The estimation of completion time for scheduling the selected task on the complementary virtual machines with same level is calculated (Seq.No. 10) and a suitable vm with minimum completion time is selected for scheduling the task (Seq.No. 11, 12). The VMM for the scheduled vm updates its ST (Seq.No. 13). After execution of a task the responses are turned in to the users accordingly (Seq.No. 14). If necessitates, the information is uploaded to Cloud for analysis and backup (Seq.No. 15,16). The Algorithm3 present in the following subsection depicts the behavior of LMTS.

## Design and Implementation

The notations used in the design and implementation of our model is presented in Table 1.

The assumptions and derivations for equations used are presented further in this section. Let $\alpha_t^{wn}$ be a binary function which returns 1 if task level of $Task_t$ is same as the computational level of $VM_w^n$:

$$\alpha_t^{wn} = \begin{cases} 1, if\ Level_t\ is\,same\,as\,Level_w^n \\ 0, otherwise \end{cases} \tag{1}$$

Then the expected average completion time for $Task_t$ on all virtual machines with same level can be calculated as:

**Table 1. Notation table**

| Notation | Description |
|---|---|
| $FogN_n$ | nth number fog node |
| FogNList | List of all fog nodes |
| $VM_w^n$ | wth virtual machine present in $FogN_n$ |
| $VMList^n$ | List of virtual machines present in $FogN_n$ |
| $W^n$ | Total number of virtual machines in $FogN_n$ |
| $MIPS_w^n$ | MIPS of wth virtual machine in $FogN_n$ |
| $VM_w^n$.STQueue | Scheduled tasks queue of $VM_w^n$ |
| $VM_w^n$.TotalST | Total number of tasks present in $VM_w^n$.STQueue |
| $VM_w^n$.STQueue.CT | Estimated completion time for executing all the tasks present in $VM_w^n$.STQueue |
| $Task_t$ | tth number task |
| TList | List containing all the arrived tasks |
| $expCT_t^{wn}$ | Expected average completion time for $Task_t$ on $VM_w^n$ with same level |
| $Level_w^n$ | Computational level assigned to $VM_w^n$ |
| $Level_t$ | Priority level assigned to $Task_t$ |
| $VMM^n$.STable | Scheduling table of $VMM^n$ |

$$\exp CT_t^{wn} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{W^n}\left(\alpha_t^{wn} * VM_w^n \cdot STQueue \cdot CT\right)}{TotalVM} \tag{2}$$

The *VM*$_w^n$*.STQueue.CT* used in equation (1) can be calculated as:

$$VM_w^n \cdot STQueue \cdot CT = \sum_{i=1}^{VM_w^n \cdot TotalST} \frac{VM_w^n \cdot STQueue_i \cdot InstLength}{MIPS_w^n} \tag{3}$$

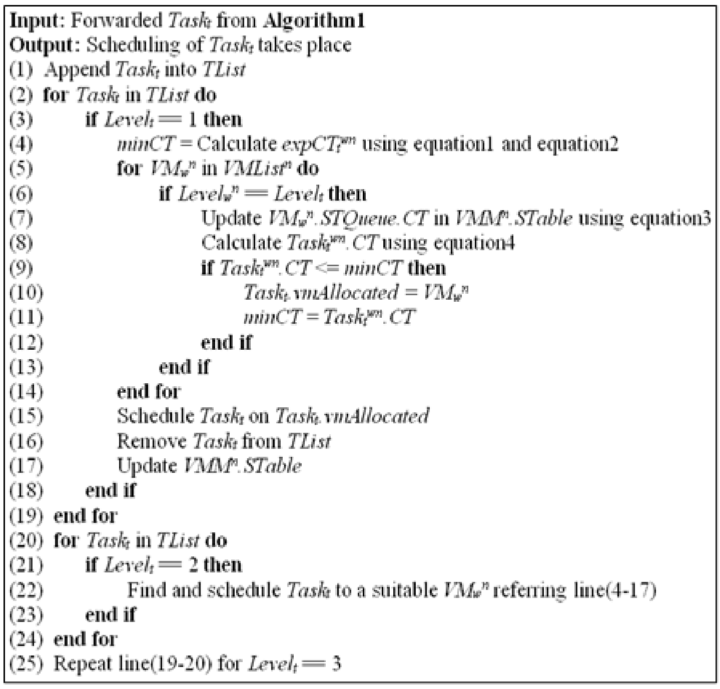
Now the estimated completion time of *Task*$_t$ if scheduled to *VM*$_w^n$ can be derived by adding estimated completion time for executing all the tasks present in *VM*$_w^n$*.STQueue* with the estimated execution time for *Task*$_t$ on the particular VM, as interpreted in equation (4):

$$Task_t^{wn} \cdot CT = VM_w^n \cdot STQueue \cdot CT + \rightleftarrows \left(\frac{Task_t \cdot InstLength}{MIPS_w^n}\right) \tag{4}$$

The network usage for all the arrived tasks can be calculated as:

$$NetUsage = \sum_{t=1}^{Tlist.length} Lat_t * NetUse_t \tag{5}$$

**Algorithm 1. Task Level Accorder**



```
Input: Forwarded $Task_t$ from Algorithm1
Output: Scheduling of $Task_t$ takes place
(1)  Append $Task_t$ into $TList$
(2)  for $Task_t$ in $TList$ do
(3)      if $Level_t == 1$ then
(4)          $minCT$ = Calculate $expCT_t^{wn}$ using equation1 and equation2
(5)          for $VM_w^n$ in $VMList^n$ do
(6)              if $Level_w^n == Level_t$ then
(7)                  Update $VM_w^n.STQueue.CT$ in $VMM^n.STable$ using equation3
(8)                  Calculate $Task_t^{wn}.CT$ using equation4
(9)                  if $Task_t^{wn}.CT <= minCT$ then
(10)                     $Task_t.vmAllocated = VM_w^n$
(11)                     $minCT = Task_t^{wn}.CT$
(12)                 end if
(13)             end if
(14)         end for
(15)         Schedule $Task_t$ on $Task_t.vmAllocated$
(16)         Remove $Task_t$ from $TList$
(17)         Update $VMM^n.STable$
(18)     end if
(19) end for
(20) for $Task_t$ in $TList$ do
(21)     if $Level_t == 2$ then
(22)         Find and schedule $Task_t$ to a suitable $VM_w^n$ referring line(4-17)
(23)     end if
(24) end for
(25) Repeat line(19-20) for $Level_t == 3$
```

**Algorithm 2. VM Task Level Assignment**

```
Input: $Task_t$
Output: Task level assignment takes place, Task forwarded to LMTS
(1) Receive $Task_t$
(2) if $Task_t.App$ in $CRITICAL$ category then
(3)     $Level_t = 1$
(4) else if $Task_t.App$ in $APPOINTMENT$ category then
(5)     $Level_t = 2$
(6) else
(7)     $Level_t = 3$
(8) end if
(9) Forward $Task_t$, $Level_t$ to LMTS
```

where, $Lat_i$ is the latency for $Task_t$, and $NetUse_t$ is the network usage of $Task_t$.

## EXPERIMENTAL SETUP AND RESULT ANALYSIS

In subsection 4.1, a detailed description on the experimental setup, simulation tool used and the conducted experiments is provided. Also at the end, an analysis of the results achieved are done in subsection B.

### Experimental Setup

The CloudSim simulator, a framework for modeling and simulation of cloud-fog environments was used for evaluating the efficacy of our model. Two datacenters DatacenterFog and DatacenetrCloud were instantiated. With adding functionalities to the submitCloudlets() method of Broker class, it was capable of monitoring task levels and finding a suitable virtual machine for a selected task, which means technically it was depicting the behavior of LMTS. The driver program of the simulator was set up to sporadically generate a bunch of tasks. The task levels were assigned by taskLevelAccorder() method defined inside the driver program. For evaluating the performance of the proposed model, three experiments were conducted with varying infrastructure i.e., 20, 50 and 100 fog nodes. In each experiment, 10 simulations were conducted with increasing the number of arriving tasks by 25 for each simulation. However, due to page limitation, result comparison for low and high fog infrastructure could not be presented in the following subsection.

**Algorithm 3. Level monitoring Task Scheduling**

```
Input: $FogNList$
Output: Computational level assignment for virtual machines take place
(1)  for $FogN_n$ in $FogNList$ do
(2)      for $VM_w^n$ in $VMList^n$ do
(3)          if $MIPS_w^n <= ThresVal_1$ then
(4)              $Level_w^n = 3$
(5)          else if $MIPS_w^n > ThresVal_1$ and $MIPS_w^n <= ThresVal_2$ then
(6)              $Level_w^n = 2$
(7)          else
(8)              $Level_w^n = 1$
(9)          end if
(10)     Add $Level_w^n$ to $VMM^n.STable$
(11)     end for
(12) end for
```

## Performance Evaluation

For evaluating the performance of our model and for comparing the results with other existing models like FCFS and SJF (Job scheduling algorithm proposed by (Jamil et al., 2020)), two main parameters were considered i.e., average delay and network usage.

### *Average Delay*

Figure 3 shows the result comparison for medium fog infrastructure with the existing models. It is clearly perceptible that in situations when large number of requests arrived, the proposed model shows supreme performance than the existing FCFS and SJF (Jamil et al., 2020) in terms of average delay.

However, it is seen that in the initial stages (25-50) the performance of our model is quite same as that of FCFS and SJF, but, later on with increasing no. of tasks the model proves itself as superior. This minimal increase in the average delay in the initial stages is due to the traversing of all the 50 scheduling tables maintained by the VMMs present in the fog nodes. Nonetheless, it is also comprehensive that with the increase in the number of tasks (125-250) the proposed model has undoubtedly shown an outstanding performance with minimizing the average delay up to 30% of SJF and about 50% of FCFS. Hence, with the increase in the number of tasks, this constant traversing delay seems to be trivial.

### *Network Usage*

An increase in the number of tasks and fog nodes leads to network congestion which results in latency and performance degradation. Hence, network usage is also considered as an evaluation parameter. The network usage can be computed using equation(5) derived in Section III. From Figure 4, it can be seen that with an increasing number of tasks, the gap of network usage between our model and the other existing model widens. The proposed model shows a much better performance by saving about 25% to 30% of network than SJF and 45% to 50% of network than FCFS.

**Figure 3. Average delay with varying number of tasks**

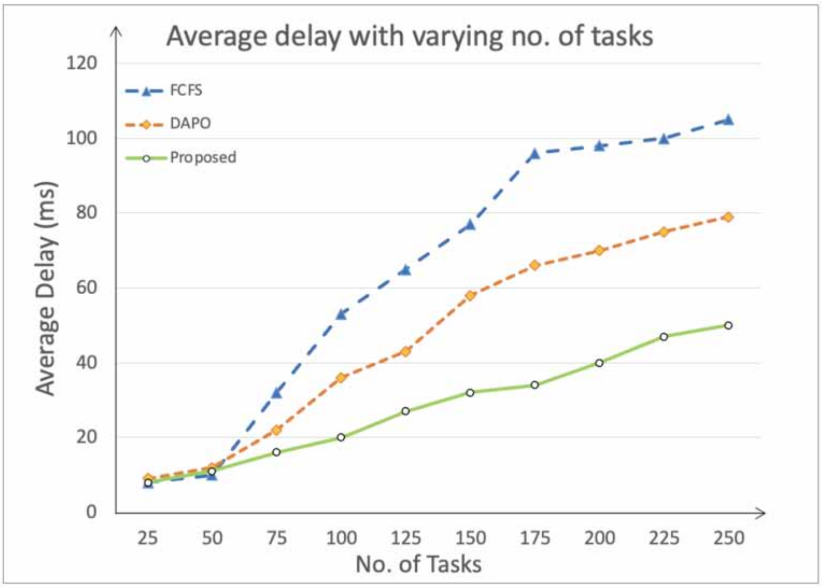
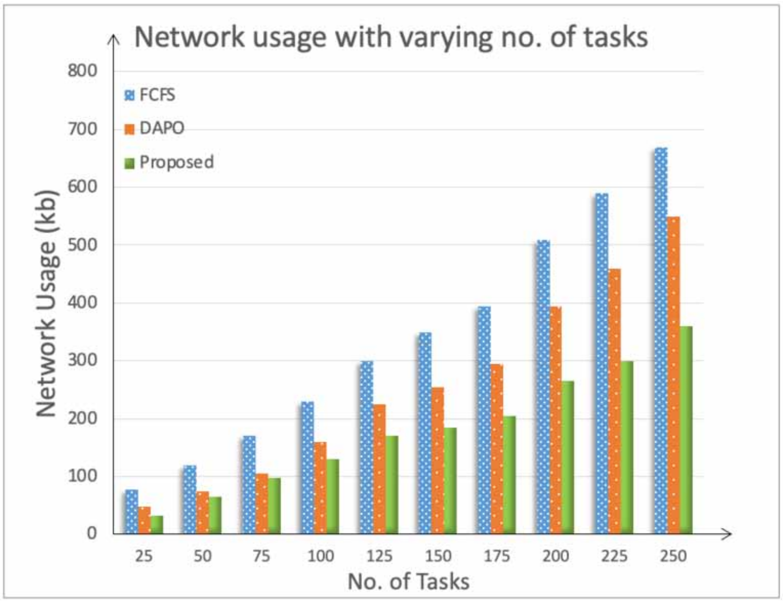


**Figure 4. Network usage with varying number of tasks**



## CONCLUSION AND FUTURE WORK

With the ever increasing needs for an efficient IoT based healthcare solution, the uses of Fog for any such solution has become a new normal. But Fog having limited resources may fail to handle such a huge traffic in absence of a proper task scheduling model. Henceforth, considering the challenges in fog, in this paper an efficient level monitoring task scheduling model for healthcare applications is proposed to provide immediate response to the delay-sensitive tasks with in healthcare applications along with minimum network usage. Through rigorous simulation and performance evaluation, it is clearly evident that our proposed model has outperformed the existing models like FCFS and SJF (proposed by Jamil et.al. (2020)) in terms of average delay and network usage. In the future, the authors shall be implementing reinforcement based prediction models for scheduling along with a focus on load balancing and maximum resource utilization of fog nodes.

## ACKNOWLEDGMENT

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

## REFERENCES

Behera, R. K., Reddy, K. H. K., & Roy, D. S. (2020). A novel context migration model for fog-enabled cross-vertical IoT applications. In *International Conference on Innovative Computing and Communications* (pp. 287-295). Springer. doi:10.1007/978-981-15-0324-5_25

Biswas, A. R., & Giaffreda, R. (2014, March). IoT and cloud convergence: Opportunities and challenges. In *2014 IEEE World Forum on Internet of Things (WF-IoT)* (pp. 375-376). IEEE.

Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012, August). Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* (pp. 13-16). doi:10.1145/2342509.2342513

Choudhari, T., Moh, M., & Moh, T. S. (2018, March). Prioritized task scheduling in fog computing. In *Proceedings of the ACMSE 2018 Conference* (pp. 1-8). Academic Press.

Jamil, B., Shojafar, M., Ahmed, I., Ullah, A., Munir, K., & Ijaz, H. (2020). A job scheduling algorithm for delay and performance optimization in fog computing. *Concurrency and Computation: Practice and Experience, 32*(7), e5581.

Khattak, H. A., & Arshad, H. (2019). Utilization and load balancing in fog servers for health applications. *EURASIP Journal on Wireless Communications and Networking*, (1), 1–12.

Mahmoud, M. M., Rodrigues, J. J., Saleem, K., Al-Muhtadi, J., Kumar, N., & Korotaev, V. (2018). Towards energy-aware fog-enabled cloud of things for healthcare. *Computers & Electrical Engineering*, *67*, 58–69. doi:10.1016/j.compeleceng.2018.02.047

Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing*. NIST.

Rahmani, A. M., Gia, T. N., Negash, B., Anzanpour, A., Azimi, I., Jiang, M., & Liljeberg, P. (2018). Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Generation Computer Systems*, *78*, 641–658. doi:10.1016/j.future.2017.02.014

Roy, D. S., Behera, R. K., Reddy, K. H. K., & Buyya, R. (2018). A context-aware fog enabled scheme for real-time cross-vertical IoT applications. *IEEE Internet of Things Journal*, *6*(2), 2400–2412.

Tang, W., Zhang, K., Zhang, D., Ren, J., Zhang, Y., & Shen, X. (2019). Fog-enabled smart health: Toward cooperative and secure healthcare service provision. *IEEE Communications Magazine*, *57*(5), 42–48. doi:10.1109/MCOM.2019.1800234

Wortmann, F., & Flüchter, K. (2015). Internet of things. *Business & Information Systems Engineering*, *57*(3), 221–224. doi:10.1007/s12599-015-0383-3

*Ranjit Kumar Behera was born in India and received his B.Tech and M.Tech in CSE from National Institute of Science & Technology, Berhampur in 2004 and 2010 respectively . Currently, he is pursuing his Ph.D in Computer Science and Engineering under Biju Patnaik University of Technology (BPUT), Rourkela and at the same time he is working as Assistant professor in CSE at NIST, Berhampur. His research interests include Internet of Things and Fog Computing.*

*Amrut Patro was born in India and received his B.Tech in CSE from National Institute of Science & Technology, Berhampur in 2021. His research interests include Internet of Things, Fog Computing and Machine learning.*

*K. Hemant Kumar Reddy was born in India and received his M.Tech, Ph.D from Berhampur University in 2008 and 2014 respectively. He is currently with the National Institute of Science and Technology, Berhampur, India as an Associate Professor. His research interests include distributed and grid computing, Cloud computing, Fog computing, service oriented architectures.*

*Diptendu Sinha Roy was born in India. He received his B. Tech from Kalyani University in 2003 and subsequently his M. Tech and Ph.D from Birla Institute of Technology, Mesra, India in 2005 and 2010 respectively. He is currently with the National Institute of Technology, Meghalaya, India. Dr. Sinha Roy's research interests include distributed, grid computing, Cloud computing, Fog computing, software reliability and optimization in engineering. He also works towards design and analysis of distributed infrastructure of power systems.*