
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 3, pp. 294–301, DOI 10.12694:/scpe.v22i3.1891

FOG TASK SCHEDULING USING CLUSTERING BASED RANDOMIZED ROUND ROBIN

SHAHID SULTAN HAJAM* AND SHABIR AHMAD SOFI†

Abstract. Fog computing serves the delay-sensitive applications of the Internet of Things (IoT) in more efficient means than the cloud. The heterogeneity of the tasks and the limited fog resources make task scheduling a complicated job. This paper proposes a clustering based task scheduling algorithm. Specifically, the K-Means++ clustering algorithm is used for clustering the fog nodes. Randomized round robin, a task scheduling algorithm is applied to each cluster. The results show that the proposed algorithm reduces the system's average waiting time.

Key words: Task scheduling, K-Means++, clustering, round robin, random, Fog computing

AMS subject classifications. 68M14, 68T07

1. Introduction. Internet of things (IoT) are found everywhere around the globe, thus generating the data in bulk. Cloud provides its reliable services to IoT devices' enormous data, including storage, infrastructure, platform, and software as a service [10, 4]. Computing the data at the cloud needs massive network bandwidth and adds the burden of transmission delay. Therefore the delay-sensitive benefits of IoT cannot be responded to and processed quickly.

Fog computing, a computing paradigm brings cloud services closer to the edge of the network [6]. The fog benefits are available nearer the edge of the network, in between the IoT devices and the cloud servers. Fog reduces the cloud's burden by performing the filtering, pre-processing, and data analysis before sending the data to the cloud.

Although the cloud possesses enormous storage and processing capabilities, its large distance from the end devices incur delays that affect the delay-sensitive applications' quality of service. Hence the fog is a good solution for IoT. The fog layer comprises the numerous fog nodes or devices which are geographically distributed and dynamic. To efficiently utilize the resource and provide the quality of the service, resource and task scheduling is an urgent need. Heterogeneity, uncertainty, and dispersion of fog resources pose a challenge for resource and task scheduling.

K-means clustering algorithm, one of the most popular clustering algorithm, is an effective classification algorithm. It is an unsupervised learning algorithm used to find the groups in the unlabelled data [7]. Although K-means is popular for its speed, usability, effortlessness, it lags in accuracy mainly due to randomly choosing the centroids [3, 8]. K-means requires good initialization of the centroids; otherwise, it may end in a local minimum. K-means++ solves this problem of choosing the initial centroids. K-means++ chooses those datapoints as the initial centroid, which are farther from each other, hence avoiding the K-means clustering algorithm's poor clustering. In this paper, we choose the K-means++ clustering algorithm to cluster the fog nodes.

We begin by clustering the fog resources into clusters and then use the randomized round robin scheduling algorithm for task scheduling inside the clusters. At last, the proposed policy is compared with the random, round robin scheduling algorithms.

The remainder of the paper is organized as follows: Section 2 discusses the related work, the algorithms used in this work is discussed in section 3, the system model is described in section 4, Randomized Round Robin Scheduling Algorithm is discussed in section 5, section 6 comprises of experimental results, and finally there will be the conclusion in section 7.

*Department of Information Technology N.I.T Srinagar, Kashmir, India (shahid.nit002@gmail.com)
†Department of Information Technology N.I.T Srinagar, Kashmir, India

2. Related Work. The resource scheduling is NP-hard problem; various scheduling algorithms are proposed to solve this problem. A dynamic resource allocation policy is proposed for Industrial Internet of Things (IIoT), particularly for tactile applications in Fog computing [1]. Based on the average delay, jitter, latency, packet loss, blocking probability Fog nodes are ranked for resource allocation. Selection is performed based on required resources and ranking of Fog. The results show the resource is utilized efficiently and QoE is improved. Fog based IoT is also used in super markets in order to manage the resources smartly and in real time [16]. K-Means clustering algorithm along with the Principal Component Analysis are used to cluster the products as per their demand and the product distribution decisions are made based on the reinforcement learning model. Fog can also be helpful in medical data wherein it will reduce IoT data by filtering the data before sending to the cloud. In [11], the EEG data is reduced before sending to the cloud. Agglomerative hierarchical clustering is used to cluster the EEG data and Huffman encoding is applied to each cluster for data compression. K-means clustering along with Support Vector Regression is used to predict the drought [13]. The authors have also taken power consumption and accuracy of the system in consideration.

In [2] an architecture for resource allocation between cloud and Fog servers is proposed. In the Fog layer, a Fog server manager reviews the available resource and may agree to execute all or some of the tasks and postpone others. In terms of response time, loading time, and total cost compared with other algorithms, including optimize response time policy, re-configuring dynamically with load balancing, the proposed algorithm performs better.

The authors in [12] have compared the K-means, K-means++ and Fuzzy C-means clustering algorithms. The algorithms are fed with sorted and unsorted data and are analysed on the factors of elapsed time and number of iterations. Fuzzy clustering along with the particle swarm optimization is used for resource allocation in fog computing environment [14]. The fog nodes are clustered into three clusters which are made available to the tasks that require computing, bandwidth and storage resources respectively. In another work [17], the authors have used Agglomerative Hierarchical Clustering for the resource pooling and to diminish the delay. K-means clustering algorithm is used for analysis of speech data of patients suffering from Parkinson's diseases in fog architecture [7]. The authors in [5] have used K-means on River Ganga basin and diabetes data. The results shows fog computing performs well for medical and geospatial data. Unlike these studies, our paper uses K-means++ with randomized round-robin for task scheduling.

3. Algorithms. In this section we will discuss the algorithms used in this work.

3.1. K-means++ Clustering algorithm. k-means++ is an unsupervised clustering algorithm used to cluster the unlabelled data. Rather than randomly choosing the initial centroid points like the K-means clustering algorithm, the centroids are chosen iteratively based on the distance [3]. Initially, the first centroid is selected uniformly at random from the set of datapoints. The distance is calculated for each datapoints from the previously chosen centroid. The next cluster centroid is chosen with the probability proportional to the square of the distance ($d(x)^2$). After obtaining, the required number of cluster centroids datapoints are assigned to the clusters based on the minimum distance from the cluster centroids. The cluster centroids are updated by taking the mean of the datapoints of the cluster. The process is repeated until no datapoints are reassigned.

3.2. Round Robin Scheduling. Round Robin is the scheduling algorithm, which is mainly used and designed especially for time sharing systems. The processes which are ready to execute are kept in a ready queue (circular queue) in FIFO (first in first out) order. The scheduler assigns the CPU to the processes in a circular fashion for a period of time quantum. The process is given the CPU for time quantum; if the process is still executing, it is preempted and added to the end of the queue. The process will release the CPU if it completes before the time quantum expires. In both cases, the CPU is assigned to the next process in the ready queue. The value of time quantum will directly affect the performance of the algorithm. Shorter burst time will result in more context switches and hence adds extra burden to the processing time. If the burst time is very large, it will make the algorithm behave like FCFS. Various dynamic time quantum policies have been proposed [9, 15], where the time quantum changes for every iteration.

3.3. Random Scheduling. Random scheduling is a technique where the tasks are chosen randomly with equal probability. From the pool of the tasks, a task is chosen randomly and assigned to the processor and the

Algorithm 1 K-means++ algorithm

Input: Processing power of 'n' fog nodes, $C_F = \{c_{f1}, c_{f2}.....c_{fn}\}$ and number of clusters
Output: K clusters.

procedure K–MEANS++
  Select a cluster centroid c_{fi} uniformly at random from C_F
  for each c_{fi}
    compute distance from the nearest centroid chosen previously. **do**
  end for
  Select new centroid from c_{fi} with probability proportional to

$$\frac{D(c_{fi})^2}{\Sigma_{c_{fi}\epsilon C_F} D(c_{fi})^2}$$

  Repeat steps from 3 to 6 untill k centroids are obtained.
  for each $c_{fi's}$ **do**
    Assign c_{fi} to that cluster centroid whose distance from the cluster centroid is minimum.
  end for
  Recalculate the new cluster centroid by taking the mean of data points of cluster.
  Repeat steps from 8 to 11 until no data point (c_{fi}) is reassigned.
end procedure

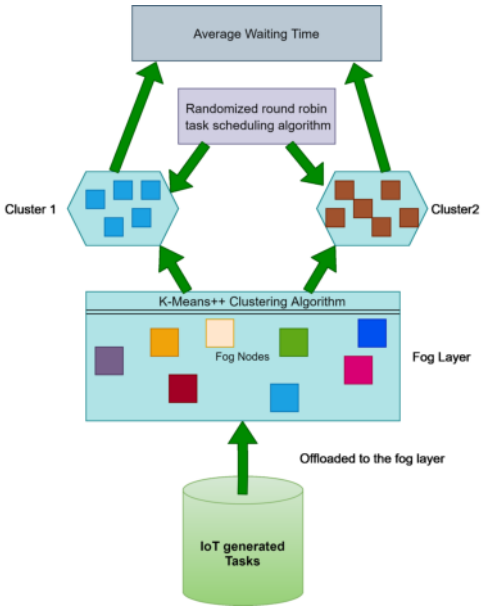


Fig. 4.1: System Model

process is repeated till the pool becomes empty.

4. System Model. The proposed system architecture is shown in figure 4.1. The IoT devices do no computation of their own; instead, tasks are offloaded to the fog layer for processing. The fog layer comprises multiple fog nodes; they are grouped into two clusters using the K-Means++ clustering algorithm. The IoT generated tasks are randomly distributed between the clusters, and also inside the clusters, the tasks are randomly assigned to the fog nodes. Randomized round robin scheduling algorithm is used to schedule the

Algorithm 2 Proposed Algorithm: Clustering based Randomized Round Robin Scheduling Algorithm

Input: Burst time, $S_T = \{s_{t1}, s_{t2}.....s_{tn}\}$ and time quantum (T_q)
Output: waiting time.

procedure RANDOMIZED ROUND ROBIN
  Cluster fog nodes into two clusters using K-means++ clustering algorithm.
  for each cluster **do**
    Place the processes (here $s_{ti's}$) in ready queue of each fog node randomly in FCFS order.
    while Burst time of the processes $S_T = \{s_{t1}, s_{t2}.....s_{tn}\} \neq 0$ **do**
      choose the process s_{ti} from $s_{ti's}$ randomly.
      if $s_{ti} < T_q$ **then**
        s_{ti} executed completely.
        goto step 5
      else
        s_{ti} is executed till T_q unit of time, interrupted and added to the queue.
      end if
      goto step 5
    end while
    for each process $s_{ti's}$ **do**
      Calculate Waiting Time
Waiting Time = Completion time - arrival time - burst time
    end for
  end for
end procedure

tasks present in the fog nodes' queue.

5. Proposed Algorithm. In this section we will discuss the clustering based randomized round robin scheduling algorithm in detail (Algorithm 2).

Clustering based Randomized Round Robin Scheduling Algorithm. Randomized round robin scheduling algorithm is an extension of the round robin algorithm. The algorithm begins by clustering the fog nodes by means of K-Means++ clustering algorithm. Based on the processing power, the fog nodes are clustered into two clusters. The IoT generated tasks are randomly distributed among the clusters. For each cluster, like the round robin algorithm, each task is executed for the time quantum, preempted, and added back to the queue. It differs in the selection of the tasks from the queue. From each fog node's ready queue, jobs are chosen randomly after another without repetition and executed for a quantum time. If the job's burst time is greater than the time quantum, the job is preempted and updated in the ready queue. The control of the processor is given to the next job chosen randomly from the ready queue. If the burst time is lower than the time quantum, then the job is completely executed, and the next process is randomly chosen from the queue without repetition. The process is repeated for the next iterations until the value of the processes' burst time becomes zero. By introducing the randomness in the round robin, the waiting time is reduced considerably compared to round robin. It is due to the selection of the tasks from the ready queue. The task does not have to wait for all other tasks to gain control of the processor back; instead, tasks can be chosen randomly, which reduces the waiting time.

6. Experimental Results. With the K-Means++ clustering algorithm, the fog nodes are classified into two clusters. The processing power of fog nodes is chosen as a criterion for clustering. The fog nodes having higher processing power form one cluster, and the rest will form another. The IoT tasks are randomly distributed among the cluster nodes. Each fog node has a scheduling queue in which the jobs are queued. Jobs are distributed randomly and scheduled independently on different processors. The tasks in the queue of the nodes are scheduled using a randomized round robin scheduling algorithm.

This section studies the performance of the proposed algorithm. The number of fog nodes and the tasks are

Table 6.1: Performance at different traffic

Type of algorithm	Number of tasks					
	1000	2000	3000	4000	5000	6000
	Average waiting time					
Random	6.200	12.728	19.541	26.143	32.414	39.079
Randomized Round Robin (RRR)	7.531	14.483	21.062	26.321	31.657	35.983
Round Robin	7.931	15.997	24.759	32.694	40.951	49.608
RRR without clustering	7.871	14.976	21.929	27.636	33.368	37.449

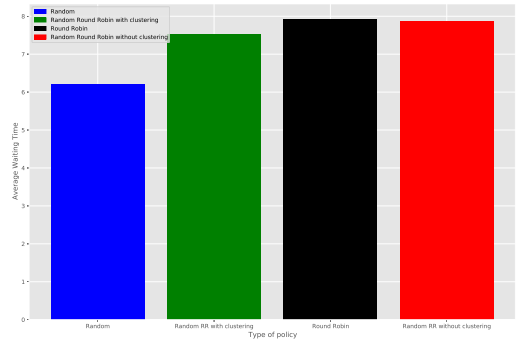
randomly set. The processing power of fog nodes is randomly set between 250-600 Millions of instructions per second(MIPS). Similarly, the tasks' size is randomly set between the range of 10-300 MI (million instructions). Fog nodes are clustered using the K-means++ algorithm based on the processing power. The randomized round robin algorithm is applied to each cluster and is analyzed for the waiting time. The time quantum is dynamically calculated for each fog node which is equal to the average of the size of the tasks present in the ready queue. The proposed algorithm is compared with the round robin, random and randomized round robin without clustering algorithms.

The random performs better than round robin in terms of waiting time because in round robin each process is given a time quantum after which the process is preempted and added back to the ready queue. This preemption adds an extra burden to the waiting time. The randomized round robin algorithm introduced the factor of randomness in the round robin, in which the jobs are chosen randomly from the ready queue. Choosing jobs randomly lower the waiting time of the processes as compared to the round robin.

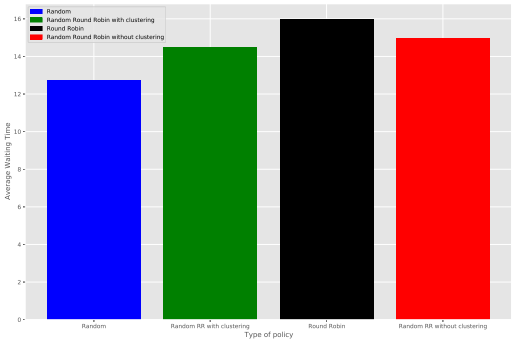
The proposed algorithm's performance is evaluated at different traffic as shown in table 6.1. The figures 6.7,6.14 shows the comparison of randomized round robin, round robin, random task scheduling, and randomized round robin without clustering algorithms.

Figure 6.7 shows the algorithms' performance with respect to the average waiting time at different numbers of tasks. To evaluate the algorithms at different traffic, the number of tasks is set to six different values that is 1000, 2000, 3000, 4000, 5000, 6000. The results in figures 6.1, 6.2, 6.3, 6.4 depicts that at lower traffic random algorithm performs better than all others. It is due to the preemption of task after every time quantum in round robin, which adds to the process's waiting time. In the randomized round robin, no doubt we choose the tasks randomly, but at the same time, we assign the processor to the processes or tasks circularly after every time quantum, which in turn will increase the waiting time. The randomized round robin algorithm performs better at high traffic load as shown in figures 6.5,6.6. The round robin performs worst at higher load because a task has to wait time quantum times the number of tasks in the ready queue minus one unit of time before attaining the processor. The random as such does not rely on time quantum so its waiting time is comparatively lower than round robin. A process has to wait for the entire burst time of the previous process before attaining the processor, which incurs the increase in that process's waiting time. In the randomized round robin, processes are given time quantum and are randomly chosen, which sometimes helps in choosing processes that can be completed in a few time quantum's which otherwise was blocked by the huge processes(huge burst time) as in random scheduling. The increased traffic increases the waiting time of the processes, but with the randomized round robin scheduling, the processes' waiting time decreases considerably.

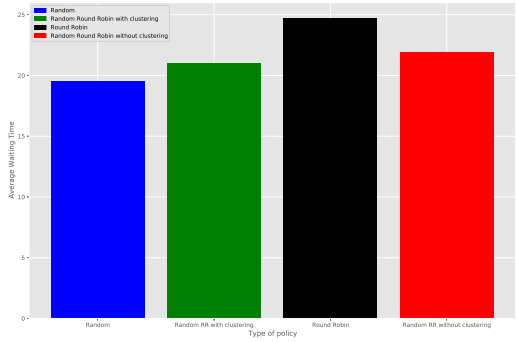
Figure 6.14 shows the average waiting of each fog node at different number of tasks. The figures 6.8, 6.9, 6.10, 6.11, 6.12, 6.13 shows the out performance of the randomized round robin algorithm with increase in the traffic. The results also show randomized round robin with clustering performs better than randomized round robin without clustering in terms of waiting time.



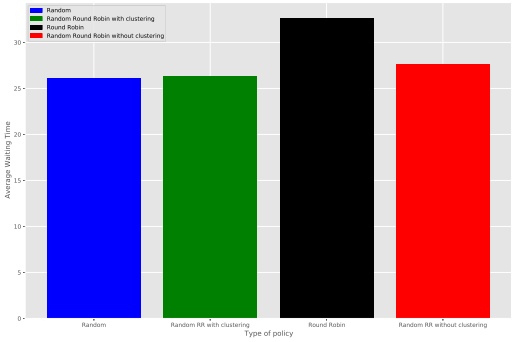
(a) Tasks = 1000



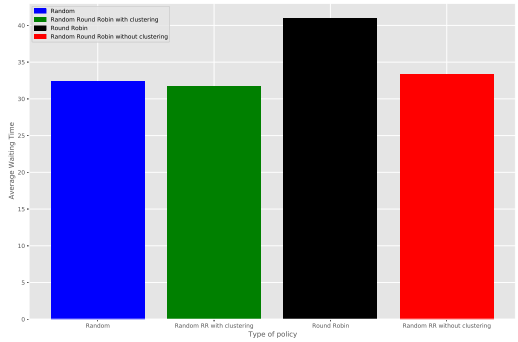
(b) Tasks = 2000



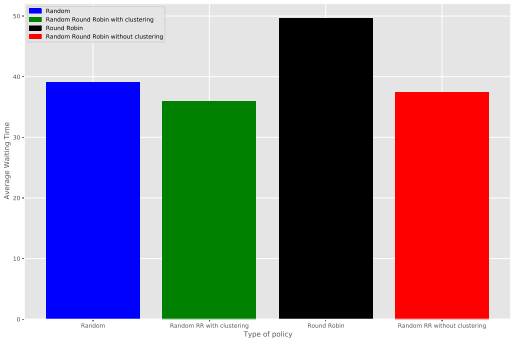
(c) Tasks = 3000



(d) Tasks = 4000

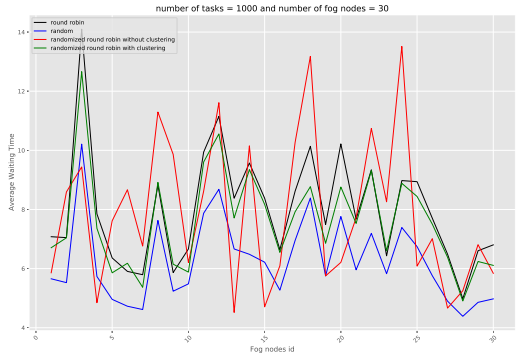


(e) Tasks = 5000

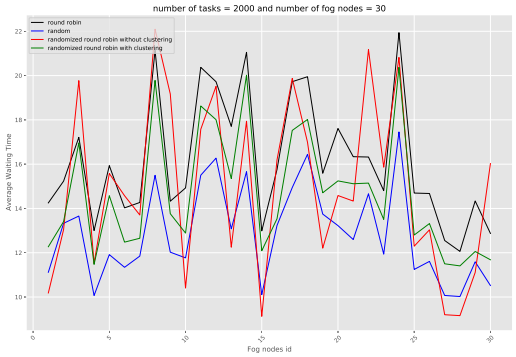


(f) Tasks = 6000

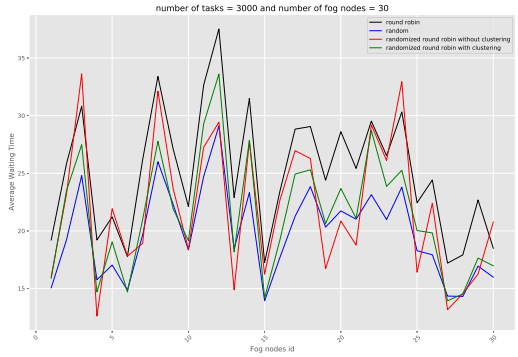
Fig. 6.1: Evaluation at different number of tasks



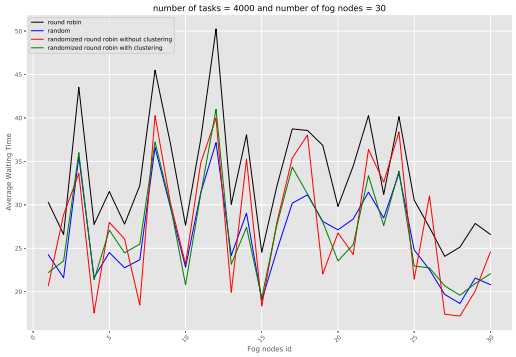
(a) Tasks = 1000



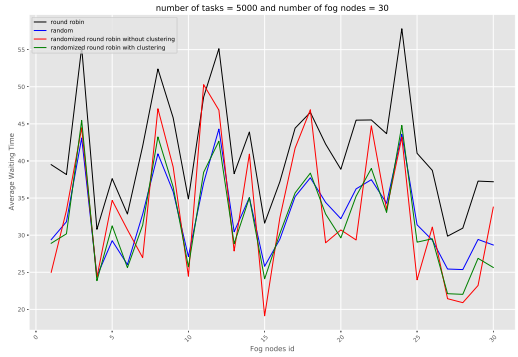
(b) Tasks = 2000



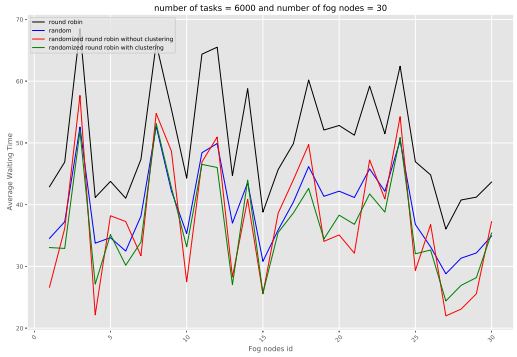
(c) Tasks = 3000



(d) Tasks = 4000



(e) Tasks = 5000



(f) Tasks = 6000

Fig. 6.2: Evaluation at different number of tasks

7. Conclusion. This paper discusses the task scheduling at the fog layer. K-Means++ clustering algorithm is applied to categorize the fog nodes into different clusters based on their processing power. The tasks generated by the IoT devices are distributed randomly among the clusters. The tasks are scheduled using a randomized round robin scheduling algorithm. The comparative analysis of different algorithms was done to see the effect on waiting time. The results shows that the proposed algorithm achieves the minimum waiting time with the increase in the number of tasks. When compared with the random, round robin, and randomized round robin without clustering scheduling algorithms, the proposed algorithm outperforms all.

The proposed algorithm shall be suitable for delay sensitive IoT applications where it will minimize the delay by minimizing the waiting time. The proposed policy can be used for smart transportation systems, smart parking, smart homes, and all those applications where delay of even microseconds matters.

REFERENCES

[1] M. Aazam, K. A. Harras, and S. Zeadally, *Fog Computing for 5G Tactile Industrial Internet of Things: QoE-Aware Resource Allocation Model*, IEEE Transactions on Industrial Informatics, 15 (2019), pp. 3085–3092.

[2] S. Agarwal, S. Yadav, and A. K. Yadav, *An efficient architecture and algorithm for resource provisioning in fog computing*, International Journal of Information Engineering and Electronic Business, 8 (2016), p. 48.

[3] D. Arthur and S. Vassilvitskii, *k-means++: The advantages of careful seeding*, tech. report, Stanford, 2006.

[4] A. A. Ateya, A. Muthanna, A. Vybornova, P. Darya, and A. Koucheryavy, *Energy-aware offloading algorithm for multi-level cloud based 5g system*, in Internet of Things, Smart Spaces, and Next Generation Networks and Systems, Springer, 2018, pp. 355–370.

[5] R. K. Barik, R. Priyadarshini, H. Dubey, V. Kumar, and K. Mankodiya, *Foglearn: leveraging fog-based machine learning for smart system big data analytics*, International Journal of Fog Computing (IJFC), 1 (2018), pp. 15–34.

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

[7] D. Borthakur, H. Dubey, N. Constant, L. Mahler, and K. Mankodiya, *Smart fog: Fog computing framework for unsupervised clustering analytics in wearable internet of things*, in 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), IEEE, 2017, pp. 472–476.

[8] A. Chadha and S. Kumar, *An improved k-means clustering algorithm: a step forward for removal of dependency on k*, in 2014 International Conference on Reliability Optimization and Information Technology (ICROIT), IEEE, 2014, pp. 136–140.

[9] B. Fataniya and M. Patel, *Dynamic time quantum approach to improve round robin scheduling algorithm in cloud environment*, IJSRSET, 4 (2018), pp. 963–969.

[10] N. Fernando, S. W. Loke, and W. Rahayu, *Mobile cloud computing: A survey*, Future generation computer systems, 29 (2013), pp. 84–106.

[11] S. K. Idrees and A. K. Idrees, *New fog computing enabled lossless eeg data compression scheme in iot networks*, Journal of Ambient Intelligence and Humanized Computing, (2021), pp. 1–14.

[12] A. Kapoor and A. Singhal, *A comparative study of k-means, k-means++ and fuzzy c-means clustering algorithms*, in 2017 3rd international conference on computational intelligence & communication technology (CICT), IEEE, 2017, pp. 1–6.

[13] A. Kaur and S. K. Sood, *Energy efficient cloud-assisted iot-enabled architectural paradigm for drought prediction*, Sustainable Computing: Informatics and Systems, 30 (2021), p. 100496.

[14] G. Li, Y. Liu, J. Wu, D. Lin, and S. Zhao, *Methods of resource scheduling based on optimized fuzzy clustering in fog computing*, Sensors, 19 (2019), p. 2122.

[15] S. M. Mostafa and H. Amano, *An adjustable round robin scheduling algorithm in interactive systems*, Inf. Eng. Express, 5 (2019), pp. 11–18.

[16] G. Neelakantam, D. D. Onthoni, and P. K. Sahoo, *Fog computing enabled locality based product demand prediction and decision making using reinforcement learning*, Electronics, 10 (2021), p. 227.

[17] L. Shooshtarian, D. Lan, and A. Taherkordi, *A clustering-based approach to efficient resource allocation in fog computing*, in International Symposium on Pervasive Systems, Algorithms and Networks, Springer, 2019, pp. 207–224.

Edited by: Dana Petcu
Received: May 24, 2021
Accepted: Sep 30, 2021