Tasks Oriented Round Robin Scheduling Model for Mobile Cloud Computing

Akomolafe Oladeji Patrick and Babalola Olawale Olaniyi

Computer Science Department, University of Ibadan, Oyo, Nigeria
dlensplc@gmail.com

Abstract

Data offloading helps to send computation intensive part of a mobile application tasks to the cloud, a resource rich environment, for execution and after the processing, the result is sent back to the mobile devices thereby minimizing the execution time and computational cost. A lot of these tasks having different requirements and nature compete for resources in the cloud; therefore, effective and clever scheduling method is required.

There have been a lot of scheduling research works in Mobile Cloud Computing (MCC) but most have been to minimize execution time and energy consumption, little consideration has been given to how more sensitive and important are some tasks over others. This often leads to application failure of some critical mission and delay sensitive mobile applications; this endangers vital processes even lives. In this work, we developed a model to bridge this gap by giving cognizance to how more important and delay sensitive some applications' tasks are and at the same time, with fairness to other less important tasks using Adjustable Time Slice Round Robin(ATSRR) Scheduling method.

The performance evaluation of this work was done using a developed simulator and it was established that more important and delay sensitive jobs were churned out more than less important tasks with minimized turnaround time and fairness to all jobs with ATSRR over other existing works. As a result, this model can find its application in a cloud environment whereby critical missions and delay sensitive mobile apps are serviced by the Cloud Service Provider

Keywords: *mobile cloud computing, scheduling, adjustable time slice, round robin, turnaround, fairness, delay Sensitive and delay tolerant mobile application*

1. Introduction

Statistics shows that by 2020, 70% of the world population would be dominated by mobile users [6] and a similar report indicated that in February 2017, more than 2.7million mobile application were already available in the Android market [4] . This is a motivation that mobile devices such as smartphone, tablets and wearable would continue to be relevant in computing world and in our society at large.

However, the proliferation of these mobile applications, especially the resource hungry and computation-intensive applications with stringent delay requirements, is a big challenge to the resource-constrained mobile devices with low compute power and limited battery life. Delay sensitive and intensive resource demanding mobile applications suffer if allowed to be executed on the mobile devices **[1].** The solution to the problem is Mobile Cloud Computing (MCC) **[9].** The Mobile Cloud Computing Forum considers MCC as "an infrastructure where both the data storage and the data processing happen outside of the mobile device" [3].

1.1 Cloud Computing

An enabling IT paradigm which helps to bridge the gap with the aim of optimizing the energy consumption and application execution time is called Cloud computing **[11].** According to National Institute of Standards and Technology (2008) ,Cloud computing is defined as "a model for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [13]**.**

Some of the mobile apps offloaded to the cloud can withstand some delays, that is, the delay tolerant mobile apps such as urban tomography, social networking **[8]** without suffering while delay beyond 250ms could be critical to face recognition, video conferencing, vehicular communications, authentication, mobile health, m-gaming, conversational video etc which are delay sensitive. As a result, processing and rendering of delay sensitive applications has become an emerging area of interest [2]. In addition, investigating which task gets which resource first with fairness in all circumstance plays a significant role in cloud environment as the cloud users get upset when some critical mobile applications suffer a noticeable delay, reported Cloud Providers [19]**.**

Tasks scheduling are responsible for mapping jobs submitted to cloud environment onto available resources in such a way that the total response time and latency are minimized and the throughput and utilization of resources are maximized [5]**.** Traditional scheduling methods include First come first serve, Shortest Job First, Priority, Round Robin etc. but they are however seldom used in real time environment because of their various disadvantages except Round Robin [15] and different environments requires different scheduling method [19].

There has been a lot of MCC research works which address mobile application tasks scheduling using both traditional and improved scheduling algorithms. However, most of which are to minimize makespan, energy consumption and cost but little considerations were given to starved processes ; one of the major problems in cloud [14], throughput maximization and its balance with the nature of the applications being scheduled. Specifically, a QoS and mobile aware framework by [10] claimed to improve the waiting time and the throughput of mobile cloud apps using FCFS method, this method suffers from convoy effect [16] and it's also devoid of fairness which is fundamental for an interactive environment like MCC.

Consequently, some delay sensitive mobile apps suffer and jobs throughput is minimal with Mahinur's framework and these related works because they are not tasks oriented. This is one of the biggest issues in job scheduling in cloud environment [17]. Most works, even with improved scheduling algorithms, did not give much cognizance to how sensitive some applications are while scheduling the offloaded tasks; some mobile applications offloaded into the cloud are delay tolerant such that if not serviced within a giving period of time, no harm would be done to them [17] while mobile cloud apps such as m-health, conversational videos, m-gaming are delay sensitive such that they have to be serviced within a particular deadline to avoid havoc to the jobs and even to lives. Building on these works, a Tasks Oriented Scheduling model is therefore put forward which gives cognizance to how sensitive an application is and at the same time minimizes the turnaround time with improved fairness to all jobs using an improved Adjustable Time Slice Round Robin (ATSRR) scheduling algorithm.

This work aims at developing a tasks oriented scheduling model using an improved Adjustable Time Slice Round Robin (ATSRR) scheduling framework.

2. Review of Related Works

The multitenant capability of cloud environment through virtualization is an underlying factor that ultimately demands deciding how, when and which job gets which resource first, that is, the scheduling method. Thus, there has been a lot of MCC research works which address mobile application tasks scheduling using both traditional and improved scheduling algorithms. However, most of which are to minimize makespan, energy consumption and monetary cost but little considerations were given to starved processes which is one of the major problems in cloud [14]. In addition, some of these works are also devoid of fairness which is fundamental for an interactive environment like MCC [18].

To the best of our knowledge, very few works have investigated prioritizing the offloaded task even the few ones didn't consider the starvation suffered by the less important jobs.

2.1 The Overview of the Scheduling Algorithm by Muhammad et al (2017)

Their approach was to select an elastic time quantum that will allow a process to execute completely if its remaining execution time is less than or equal to 0.2th of its actual time, this condition is for all tasks. First, the maximum burst time was obtained from the available processes in the ready queue. Then, a proportion of this time was used to set the time quantum. As a rule of thumb, 0.8th fraction of the maximum burst time was selected as Time Quantum. Now the scheduler assigns the CPU to all the processes in the ready queue with burst times less than the time quantum while larger ones are kept on hold. As soon as all the smaller processes complete their execution, the time quantum is set equal to the maximum burst time. The Gantt chart in figure 4.5 and table 4.3 shows the result of applying the above algorithm on the dataset in table 4.1.

2.2 The Overview of the Scheduling Algorithm by Mahinur & Zohra (2017)

[10] suggested a QoS and mobility aware optimal resource allocation architecture, namely Q-MAC, for remote code execution in MCC that offers higher efficiency in timeliness and reliability domains with the aim of increasing the number of requests processes per a period of time (throughput) and minimizing the waiting time. When offloading requests come from clients to the cloud, it maintains the tasks requests in a queue according to first-come-first-serve (FCFS) method. Same clients queue handler with the same approach also work in every cloudlet.

2.3 The Overview of the Scheduling Algorithm by Goel & Garg (2016)

[7] Proposed an algorithm which combines the working principle of fundamental scheduling algorithms. Dynamically Time Slice (DTS) is calculated which allocates different time quantum for each process based on priority, shortest CPU burst time and context switch avoidance time.

Shuffle the processes in ascending order according to the factor of each process in the ready queue (RQ) such that the head of the ready queue contains the lowest factor process based on the burst time, arrival time & priority of the process.From all these reviewed works, very few could actually account for how more important a particular task is over other, our work is therefore put forward to service more of the delay sensitive tasks than the less time conscious tasks however with fairness in all circumstances using Adjustable Time Slice Scheduling Round Robin method. The next chapter describes the model to achieve these task oriented framework.

3. The Methodology of the Proposed Adjustable Time Slice Round Robin Scheduling (ATSRR) model

The solution is a modified round robin algorithm that took into consideration the priority of the jobs entered into the system as well as the burst time of the jobs to determine a time quantum for the round robin scheduler. The quantum time was obtained by taking 80% (leveraging on [12]) of the largest burst time of jobs in the job queue. This allowed for the efficient dispatch of jobs and also solved the problem of too small or too big quantum times selections which can lead to problems like too many context switches when the quantum is too small and a convoy effect when the quantum time is too large.

3.1 The Components of the Methodology

The solution consists of four major modules and they are:

1. The Priority determination module,
2. Burst time determination module,
3. Quantum determination module, and
4. The modified round robin scheduler module.

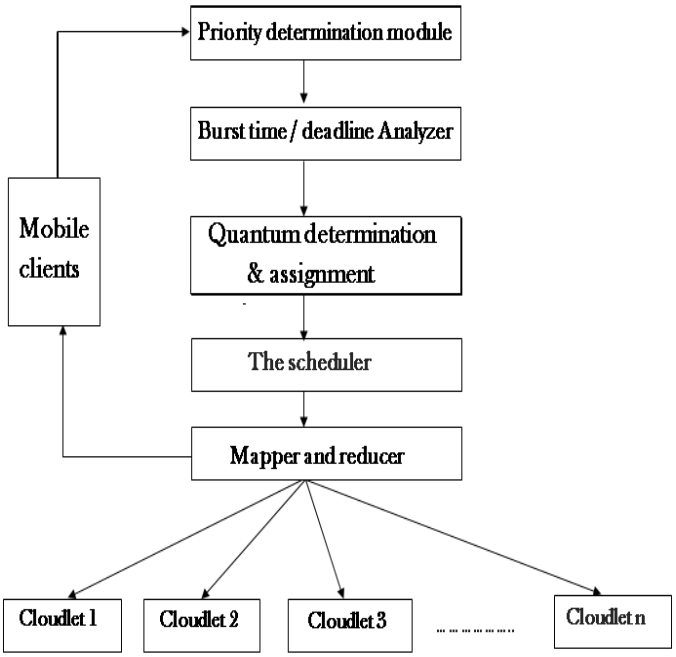

Figure 1: ATSRR Model

The first module is the priority determination module which determined the priority of the system. The determination of priority classified that job could be either delay sensitive or delay tolerant. The next module after the priority determination is the burst time determination module. This module made it possible to get the maximum burst time that was used to determine the quantum time of the system. The final part of the system was the scheduler which is actually a modified round robin scheduling algorithm.

The modified round robin algorithm used the dynamically determined quantum time to start scheduling. It interacted with the dispatcher to check if there were processes still in the ready queue. The algorithm checked the priority of the processes in the ready queue to determine selection of processes from the queue. The priority of the selected process

determined how the process was to be executed. When the priority was high, the process was assigned the CPU for the set quantum time, and then the scheduler checked the remaining burst time to decide its next move. When the remaining burst time was less than 20% of the actual time, the process was allowed to run to completion, else, if the remaining burst time was greater than 20% of the quantum time, the process was preempted and placed in the ready queue tail.

On the other hand, when the selected process was low priority process, the process was also assigned the CPU for the default quantum time, after which the scheduler checked the remaining burst time to decide its next move. When the remaining burst time was less than 5% of the actual time, the process was allowed to run to completion, else, when the remaining burst time was greater than 5% of the actual time, the process was preempted and placed in the ready queue tail.

Data insertion Algorithm for the Modified Priority Queue

```
Data = new data item (process)
Int I = 0;
If {! Is full ()}
IntArray[itemCount+1]= data;
Else
"start from the right end of the queue
For ( i=itemCount-1; I >= 0; i--)
"if data is larger, shift existing item to right end"
if(data> intArray[i])
Else
Break;
"insert data
intArray[i+1]=data;
itemCount+1;
```

Figure 2: Algorithm for insertion of data in the modified priority queue.

3.1.1 Burst Time/Deadline Analyzer

After determining the priority of a job and assigning the job to its designated position in the modified priority queue, the next task was to determine the burst time of the job. The burst time of a job is the amount of CPU time the job requires. This was "predicted" by using the following algorithm:

Burst Time Determination Algorithm

```
tn = Actual time;
p = wrished parameter;
T(n+1) = predicted burst time.
T(n+1) = tn(p) + (1-p)Tn
If( T(n) = T(0))
T(n+1) = initial burst time;
Else
T(n+1) = tn(p) + (1-p)Tn;
```

Figure 3: Burst Time Determination Algorithm.

Quantum Time Determination Algorithm

The algorithm for the determination of the quantum time for the round robin algorithm is shown in figure 3.4.

```
Q = Job priority queue;
I = 0;
Max = 0;
while (I < Q.Count)
if (max < Q[i])
Max = Q[i];
Else
Max = Max;
I = I + 1;
QT = 0.8 * max   ≠80% of the max burst time
```

Figure 4: Quantum Time Determination Algorithm

4. The Data Sets

The experiment was composed of dataset from Goel et al (2016). ATSRR algorithm was applied on this dataset and also the proposed methods by Mahnuir et al (2017), Muhammad et al (2017) and Goel et al (2016) were also applied on this data set for comparative analysis.

Dataset considered for the Turnaround Time and Waiting Times are presented for each method. The Gantt charts for each algorithm are also presented**.**

Table 1: Experimental Data set (Goel et al 2016)

PROCESS ID	BURST TIME	ARRIVAL TIME	PRIORITY
1	23	0	3
2	34	5	1
3	34	3	3
4	12	6	4
5	8	8	2
6	10	4	5
7	31	1	1
8	23	2	4
9	9	3	5
10	16	6	1

4.1 Performance Metrics

Below are the performance metrics considered in the experiment:

- **Turnaround time:** Total time taken from submission of the process till the completion. Turnaround time should minimize the time of users who wait for the output.
- **Waiting time:** Should be minimized as it is the total time spent in ready queue
- **Fairness:** CPU should be unbiased and every process should get its fair time to execute. But more attention should be given to delay sensitive jobs.

4.2 Simulation settings

A simulator was developed to run the work's algorithm. The simulator was designed to take advantages of the leveraging parts of the model and also to give a visual display of the algorithm at work.

4.2.1 The Simulator Implementation Tools

The simulator was designed using

1. The Microsoft C# programming language and
2. The Microsoft visual studio integrated development environment.

The simulator randomly generated new processes based on the priority determination module. During the creation of a new process, the priority determination module, burst time determination module and quantum time calculation modules served as input to the new process creation module. The arrival time of the processes were also randomized to create a real life arrival event of processes in the cloud environment..

4.3 Evaluation of ATSRR Model

The ATSRR model was compared with other three related works and the results are presented thereafter.

Table 2: Turnaround and Waiting Time of ATSRR

PROCESS ID	BURST TIME	ARRIVAL TIME	PRIORITY	TURNAROUND TIME	WAITING TIME
1	23	0	3	78	55
2	34	5	1	190	167
3	34	3	3	190	156
4	12	6	4	31	19
5	8	8	2	19	105
6	10	4	5	160	9
7	31	1	1	55	129
8	23	2	4	9	32
9	9	3	5	129	0
10	16	6	1	96.2	113

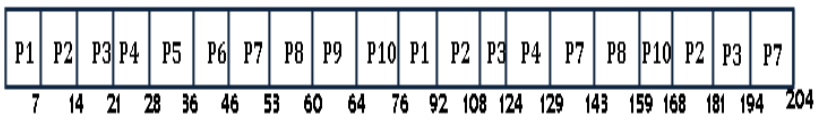

Figure 5: Gantt chart of the Goel et al (2016) algorithm

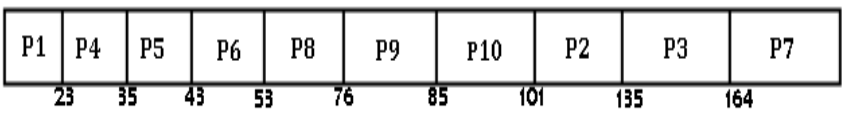

Figure 6: Gantt chart of the Muhammad et (2017) algorithm

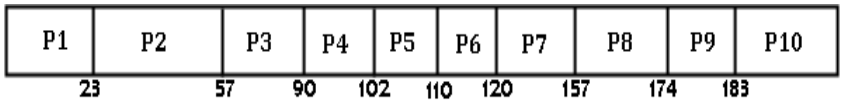

Figure 7: Gantt chart of the Mahnuir et al (2017) algorithm

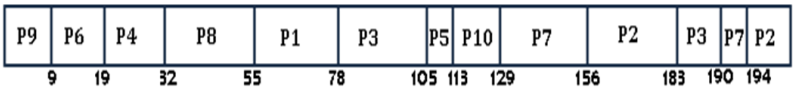

Figure 8: Gantt chart of the ATSRR algorithm

Table 3: Waiting Time Comparative Analysis of the Existing works and the ATSRR

PROCESS ID	PRIORITY	MAHINUR	MUHAMMAD	GOEL	ATSRR
1	3	0	0	76	55
2	1	23	101	160	167
3	3	57	154	158	156
4	**4**	**90**	**23**	**117**	**19**
5	2	102	35	28	105
6	**5**	**110**	**43**	**36**	**9**
7	1	120	164	185	129
8	**4**	**151**	**53**	**136**	**32**
9	**5**	**174**	**76**	**60**	**0**
10	1	183	85	152	113
Average Waiting Time		101.1	73.4	110	78.5

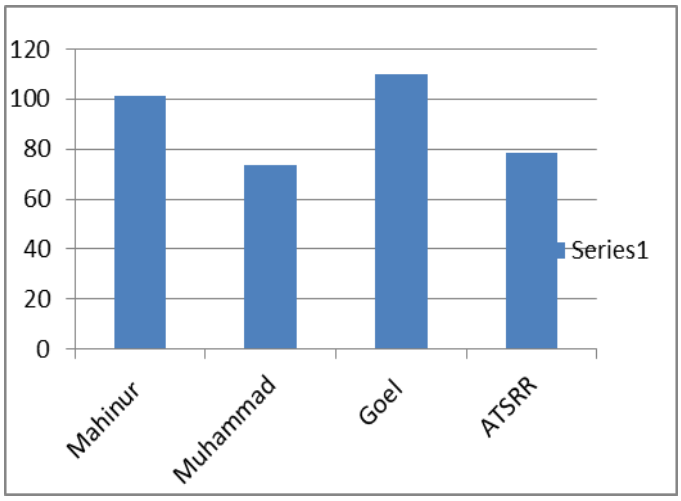

Figure 9: The Bar Chart for Waiting Time Comparative Analysis

Table 4: Turnaround Comparative Analysis of the Existing and the ATSRR

PROCESS ID	PRIORITY	MAHINUR	MUHAMMAD	GOEL	ATSRR
1	3	23	23	99	78
2	1	57	135	194	190
3	3	91	188	192	190
4	**4**	**102**	**35**	**139**	**31**
5	2	110	43	36	113
6	**5**	**120**	**53**	**46**	**19**
7	1	151	195	216	160
8	**4**	**174**	**76**	**159**	**55**
9	**5**	**183**	**85**	**69**	**9**
10	1	196	101	168	129
Average Turnaround Time		**120.7**	**93.4**	**131.8**	**96.2**

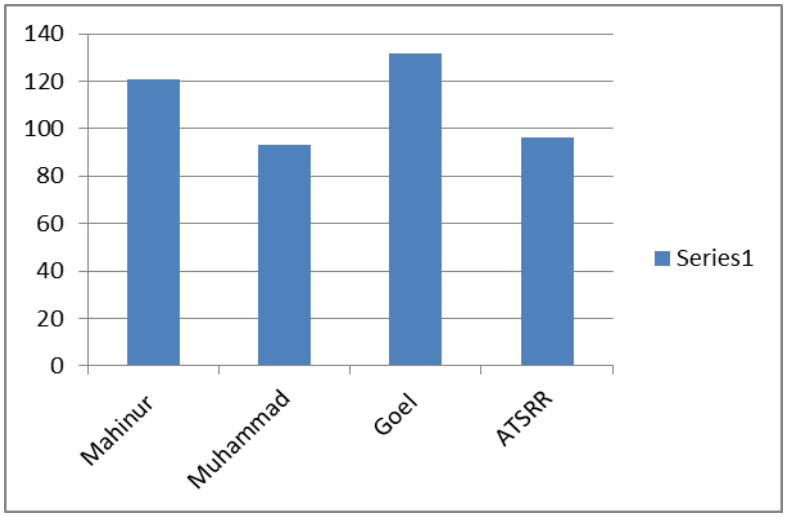

Figure 10: The Bar Chart for the Comparative Analysis of Turnaround

4.4 The Result Discussion and Quantitative Analysis of the Work

According to fig. 8, P4, P6, P8 and P9 which are the most critical tasks with higher priority of 4 and 5 are serviced within the first 55ms while in the other works as shown in the above Fig 5, 6 and 7, they were serviced at a longer time. The grant chats, Table 3 and table 4 are comparatively summarized in the below table 5, it gives a clear picture of the average Waiting Time and Turnarounds of these most critical tasks

Table 5: Quantitative Comparisons in terms of Turnaround and Waiting Time between ATSRR and other 3 Works

	Mahnuir	Muhammad	Goel	ATSRR
Ave. Waiting Time of the 4 Critical tasks P4, P6, P8 and P9	**131.25**	**48.75**	**87.25**	**15**
Ave. Waiting Time of the whole 10 processes	**101.1**	**73.4**	**110**	**78.5**
Ave. Turnaround of the 4 Critical tasks P4, P6, P8 and P9	**132.5**	**62.5**	**103.25**	**28.5**
Ave. turnaround of the whole 10 processes	**120.7**	**93.4**	**131.8**	**96.2**

From table 5, it is obvious that the most important jobs are serviced the fastest with average Turnaround Time of 28.5 when compared with other 3 works and this justifies the goal of this work. Even only the Muhammad et al's work is superior to our work in terms of the average turnaround of the 10 processes but still has the set back of not giving cognizance to how important a job is.

From this same table, we can easily deduce the significance of ATSRR in terms of the waiting time of the most critical tasks.

By far, the most critical tasks P4, P6, P8 and P9 have the least Average Waiting Time of 15ms with ATSRR when compared with other 3 works, this is a crystal evidence of the cognizance being given to these delay sensitive jobs and they churn out faster. Even in general consideration, after Muhammad's method, the ATSRR still has the next minimum average waiting time for the 10 tasks; this indicates that averagely, there is fairness to all other tasks despite that the more important jobs are given priority.

5. Conclusion

The ATSRR showed an improved fairness to all tasks during task execution when compared with the existing QOS and mobile aware framework and other related works . A task oriented scheduling round robin model with an adjustable time slice was designed which consists of three basic modules namely; the priority module, burst time determination module and the Adjustable Time Slice Scheduler. The priority module was designed to give cognizance to how more important was an application task was over others, the burst time module was to determine the amount of time a task would be using the cloud resource and the integral part of the research methodology took place in the Adjustable Time Slice Round Robin (ATSRR) scheduler. This scheduler helps to give fairness to all tasks.

Meanwhile, an improved turnaround time of average value of 28.5ms against the other 3 works with 132.5, 62.5 and 103.5 was achieved. This signifies that minimized response time was achieved for some jobs of high priority and the work has been able to achieve its objectives by servicing and churning out more important jobs (delay sensitive) tasks with higher priority faster than others existing related works.

6. Recommendation and Future Works

This work can bring about better cloud scheduler performance in terms of improved response time and priority to some more important tasks if deployed at the cloud

environment. Cloud service Providers that service much of critical mission and delay sensitive mobile apps like earthquake monitoring system and mobile health application would find this model interesting if integrated into their environment.

The developed simulator dynamically generates processes, determines burst time and assigns priority, for it to be more versed, future work can be extended to achieving these tasks statically.

References

[1]. A. Bourouis, A. Zerdazi, M. Feham, A. Bouchachia, M-health: Skin disease analysis system using smartphone's camera, Procedia Computer Science 19 (2013) 1116–1120.

[2]. ACM Transactions on Multimedia Computing, Communications, and Applications (ACM TOMM) Special Issue on Delay-Sensitive Video Computing in the Cloud. (2016).

[3]. Ali,M.: Green cloud on the horizon (2009) .In:Proceedingsofthe1stInternational Conference on Cloud Computing (CloudCom), Manila, pp. 451–459 (2009).

[4]. AppBrain. 2017.Android Applications. Retrieved from. https://www.appbrain.com/stats/number-of-androidapps

[5]. Demchenko Y, de Laat C (2011, March) Defining generic architecture for cloud infrastructure as a Service model, The International symposium on grids and clouds and the open grid forum Academia Sinica, pp 2–10.

[6]. Ericsson. 2016.A survey on Smartphones. Retrieved from https://www.ericsson.com/news/ 1925907", accessed on 02 February 2017.

[7]. Goel. N., & Garg,R.B., (2016). Performance Analysis of CPU Scheduling Algorithms with NovelOMDRRS Algorithm . IJACSA International Journal of Advanced Computer Science and Applications, Vol. 7, No. 1, 2016. Research Scholar Department of Computer Science, TMU, India Delhi Universty, India.

[8]. Huaming Wu and Katinka Wolter. (2017). Stochastic Analysis of Delayed Mobile Offloading in Heterogeneous Networks . IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. , NO. , 2017. IEEE International Conference on Communication, Computing and Digital Systems. University of Engineering and

[9]. Kurma, K., Lu, Lu Y-H., 2010. Cloud Computing for mobile Users: can mobile computation save energy?

[10]. Mahinur, A., Fatema, & Zohra. (2017). Q-MAC: QoS and Mobility Aware Optimal Resource Allocation for Dynamic Application Offloading in Mobile Cloud Computing .International Conference on Electrical, Computer and Communication Engineering (ECCE)/IEEE, February 16-18, 2017, Cox's Bazar, Bangladesh.

[11]. Muhammad H., ur R., Chee S., L, Ahsan I., Teh Ying W., (2017)Faculty of Computer Science and Information Technology University of Malaya 50603, Kuala Lumpur, Malaysia mhrehman@siswa.um.edu.my, csliew@um.edu.my, tehyw@um.edu.my. Opportunistic Computation Offloading in Mobile Edge Cloud Computing Environments.

[12]. Muhammad, U., Aamna, S.& Abu, B. 2017. An Efficient Dynamic Round Robin Algorithm for CPU scheduling. University of Engineering and Technology, Lahore mufarooq40@gmail.com. 2017 IEEE Inter national Conference on Communication, Computing and Digital Systems

[13]. Peter M., Timothy G., "The NIST Definition of Cloud Computing", Jan, 2011. http://docs.ismgcorp.com/files/external/Draft-SP-800-145 Cloud Definition.pdf.

[14]. Samir E.,, Shahenda S.,& Manar J. (2017). A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique. Journal of Cloud Computing: Advances, Systems and Applications.

[15]. Silberschatz, A., Peterson, J. L., and Galvin, B.,Operating System Concepts, Addison Wesley, 7th Edition, 2006.

[16]. Silberschatz, A. , Galvin, P. & Gagne, 2012. Operating System Concepts (9th Ed.) . Yale University, Wiley.

[17]. Swachil, J., Patel., Upendra, R., & Bhoi. (2014). Improved Priority based Job Scheduling. Algorithm in Cloud Computing using Iterative Method.Fourth International Conference on Advances in Computing and Communications.Computer Science & Engineering Dept.Vadodara, India .2014 Fourth International Conference on Advances in Computing and Communications.

[18]. Syed H., Muhammad S., Abd L., Yahaya C., Shafi'i M.. (2016). Resource Scheduling for Infrastructure as a Service (IaaS) in Cloud Computing: Challenges and Opportunities. Journal of Network and Computer Applications.

[19]. Tanenbaum,A . 2014. Modern Operating System (3rd ed.). The Netherland, Vrije Universiteit,.Amsterdam.