International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering, 2015

# Implementation of an EDF Algorithm in a Cloud Computing Environment using the CloudSim Tool

Sayda Khidr Fadlalah Ali
Dept. Electrical and Electronics Engineering
University of Khartoum - UofK
Khartoum, Sudan
sayda_elshekh@hotmail.com

Mustafa B. Hamad
Dept. Computer Science, Faculty of Mathematical Science
University of Khartoum-UofK
Khartoum, Sudan
mbhamad@uofk.edu

***Abstract*—This study designed an implementation of a dynamic real time scheduling algorithm in cloud computing environment using the CloudSim tool, aims for exploring jobs and hosts assignment managed by the cloud broker which schedules cloudlets (tasks) to hosts according to EDF algorithm by ordering each of which by deadline which has been estimated for each cloudlet (task) proportionally to its execution time.**

**This broker implementation expected to reduce the cloudlet waiting time for a cloud resource, hence minimize the number of cloudlets that missed deadline, the wall time, the delay after deadline for each cloudlet compared to first come first served policy, but it does not affect the execution time of cloudlets on the processing elements.**

**In the simulated cloud environment the number of resources with their characteristics remained fixed except for the number of hosts which increased several times, three cases were taken and every time handled the same number of cloudlets, to analyze the performance results of the cloudlets (task) at each case, until identifying the appropriate number of hosts.**

**The findings of the results analysis showed that the cloudlets have spent less time in the cloud data center which resulted in better performance outcomes, also the deadline value and the number of hosts had a major impact on the cloudlets performance, And the number of the resources hosts, and virtual machines needed for any cloud infrastructure could be determined by this scheme in order for all cloudlets to meet the deadline while the wall time and the waiting time remain acceptable.**

***Index Terms*—Cloud Computing, Real Time Scheduling, , Broker Policy, Earliest Deadline First.**

## I. INTRODUCTION

Nowadays, Cloud computing is the most recent technology It provide users with services through networks in which a lot of application should be handled appropriately in order to provide a convenient service specially real time applications such as video conference and HDTV, so providers must understand the characteristics of these applications and the network factors that ultimately affect the performance to help exploit techniques and mechanism that guarantee an acceptable level of service behavior hence develop better cloud systems

(Barrie, 2011) refers to Cloud computing as an applications and services Iaas, Paas or Saas runs on a distributed network using virtualized resources accessed by common internet protocols and networking standards. It is distinguished by the notion that resources are virtual and limitless in a way that resources are pooled and shared, storage can be provisioned as needed from a centralized infrastructure, the details of the physical systems on which software runs are abstracted from the user , applications run on physical systems that aren't specified, data is stored in locations that are unknown, administration of systems is outsourced to others, and access by users is ubiquitous.

Since cloud computing builds the illusion of limitless resources that dynamically scaled up or down because of hardware virtualization the high performance computing need an appropriate methods to be considered for job scheduling. Scheduling in cloud computing is not fully developed and needs further elaboration as in [1] [6] and it will be discussed in section I.A. Any designed scheduling algorithm must seek a way to maximize the performance of the system by avoiding unnecessary delays. Hence an earliest deadline first algorithm (section I.B) has been implemented in the cloud broker side to schedule tasks to hosts in a cloud data center using CloudSim Tool.

### *A. Scheduling in Cloud Computing*

The clouds large pool of virtualized resources are managed by software called the hypervisor, this software assign and transfer tasks by mapping them to available resources on the basis of tasks' characteristics and requirements. and resources should be utilized efficiently without affecting the service parameters of the cloud. This scheduling process can be generalized into three stages [2]:

A. Resource discovering and filtering –the datacenter broker discovers the resources present in the network system and collects status information related to them.
B. Resource selection – the target resource is selected based on certain parameters of the task requiring a specific processing usually advised by the broker which is at this decision stage.
C. Task submission -Task is submitted to the target resource.

The simplified scheduling steps mentioned above are shown in Figure.1

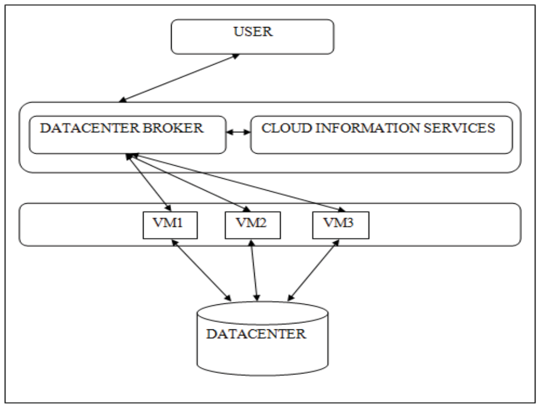


Fig 1. Scheduling steps [2]

### *B. Earliest Deadline First Algorithm*

The Earliest Deadline First (EDF) is implemented in this study as a cloud broker scheduling policy; this algorithm was introduced by Liu and Layland [3] basically it sort jobs by deadline so that jobs with the earliest absolute deadline should get executed first hence finish earlier and avoid the possibility of missing deadline.

The optimality of the EDF algorithm has been proven among all scheduling algorithms on a uni-processor, in the sense that if a real-time task set can not be scheduled by EDF, then this task set can not be usually scheduled by any algorithm. Usually a real time tasks arrive at time *ri* with two main characteristics a relative deadline *Di* and worst case execution time *Ci*, for periodic tasks the deadline is equal to it period *Ti,* thus each job receives an absolute deadline:

$$di = ri + Di \qquad (1)$$

the relative deadline and the worst execution time are hard to predict so in this work the value 1.25 from the execution time time of each cloudlet (task) is taken as a deadline.

## II. PROBLEM STATEMENT

This paper seeks to reflect a design of a cloud computing scheduler (broker) that deals with users requests for many applications that are subject to time constraints and require certain platform of virtual resources, thus adopting a real time scheduling algorithm to assign these application tasks to an available virtual machines that's virtualizes these resources to be executed without exceeding the time limit, and determining the effects of the number of the available resources in the cloud infrastructure and the flexibility of the time constraints on the performance of these applications.

## III. METHODOLOGY

This research work attain to examine tasks performance that are time sensitive in a cloud computing environment through ,Investigate the behavior of the cloud broker and implement EDF scheduling policy, to assign a certain number of cloudlets (tasks) to resources in cloud computing data center simulated by cloudsim , and evaluate the performance of cloudlets by testing the number of hosts and deadline value variation when submitting the same number of cloudlets using the EDF broker and analyze the cloudlets (tasks) performance results for each number of hosts that is chosen, And comparing the cloudlet performance results under the EDF broker with first come first served Scheduling policy to evaluate the performance. So the study was divided into four parts. First part, Simulation Scenario a data center represents the cloud computing environment was designed using CloudSim with number of hosts that was increased several times through testing, cloudlets was processed in this data center as the applications requests. The submission process is done via the data center broker which was modified by this research to implement the earliest deadline first algorithm.

Secondly is the implementation of the algorithm on the broker side simulated in cloudsim as the *data center broker* class which had been modified using java to sort the submitted cloudlets by their deadline and send these cloudlets to be processed by a cloud host.

The third part is the comparison between the cloudlets performance under the original *data center broker* which implies First come first served policy and the modified one named *EDFbroker* to measure the performance in the same cloud data center.

The fourth part is testing the design for several cases, each of which had a different number of hosts and analyze the simulation results to find an appropriate number of hosts needed for any cloud infrastructure in order for all cloudlets to meet the deadline while the wall time and the waiting time remain acceptable.

### *A. Simulation Scenario*

The simulation Scenario was a cloud computing environment with attributes of x86 system architecture, LINUX as an operating system, Xen as a virtual machine monitor, and number of hosts that was increased several times through testing, cloudlets represents an application component responsible of delivering the data in the cloud service model. It has all the characteristics the length and the input and output file sizes parameter it should be greater than or equal to 1.and it also contains various ids for data transfer and application hosting policy [6], the cloudlets assigned to hosts through the EDFbroker or the data center broker to be executed, after finishing the execution cloudsim outputs the result for each cloudlet performance under each broker, these results are analyze and compared to observe how the brokers performance varies.

The simulation submits 1000 independent cloudlets with Random length, Each cloudlet executed on one processing element, the Execution time for each cloudlet to be the length (MI) divided by processing element speed (MIPS)

And the Deadline to be proportional to the cloudlet execution time

The cloudlets are submitted by the broker to a different number of hosts and virtual machines: in three cases as scenarios were taken 100 hosts, 150 and finally 200 hosts and evaluate the performance of the algorithm under each case, the evaluation considered:

A. Number of missed deadline cloudlets in each case
B. The average waiting time in each case
C. The average execution time in each case
D. The average delay after deadline in each case
E. Furthermore the deadline value was increased several times for each case and the performance of the algorithm was observed in term of the number of missed deadline cloudlet.

The evaluation process was applied for both the *data center broker* and the *EDFbroker*

### *B. Determine Deadlines*

The deadline is the most important assumption in this research, for each cloudlet usually the estimation of the deadline is done based on a complicated analysis of the task history which include the quality of service, the application requirement in order to avoid this complexity one processing element is assigned to each VM and one VM to each host so each cloudlet will be executed on one processor the whole time of its execution. Since the length of the cloudlets and the processors speed are known, the execution time can be calculated for each cloudlet, additionally, a proportional value of the execution time is added to the execution time because of the complexity of estimating the deadline , therefore the deadline can be calculated by summing the arrival time at a cloud resource and the execution time and the assumed proportional value this value has been changed several times through testing. The values 0.15, 0.25 and 0.5 were used.

In cloudsim a new cloudlets characteristic introduced as cloudlet deadline in the cloudlets class, *setDeadline()* is a method designed to gets the length of each cloudlet and processing element speed of each processing element and returns the deadline.

### *C. Implementation of the Earliest Deadline First Broker*

The proposed implementation method of implementing this algorithm is designing a broker class that creates a new cloudlets list from the cloudlets list submitted from the users which in this work the broker itself, this new cloudlets list will be ordered according to the earliest deadline first algorithm in which the cloudlet with the earliest deadline will be placed on the front of the list, and then the cloudlets will be assigned to hosts from this new cloudlets list using the coded sendNow() method that is responsible of assigning the cloudlets to hosts in the broker implementation by first com first served which provided by cloudsim.

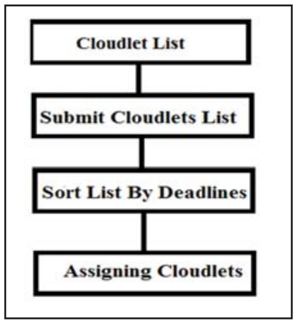


Fig 2. Implementation Method

The hosts executes these cloudlets in a four processing elements, 512 MB of RAM , 1000 BW virtual machines, considering that each host have one virtual machine, 2048 MB of RAM then each cloudlet will be executed by the virtual machine independently in one processing element with a speed of 1000 million instructions per second (MIPS)

**Algorithm:**

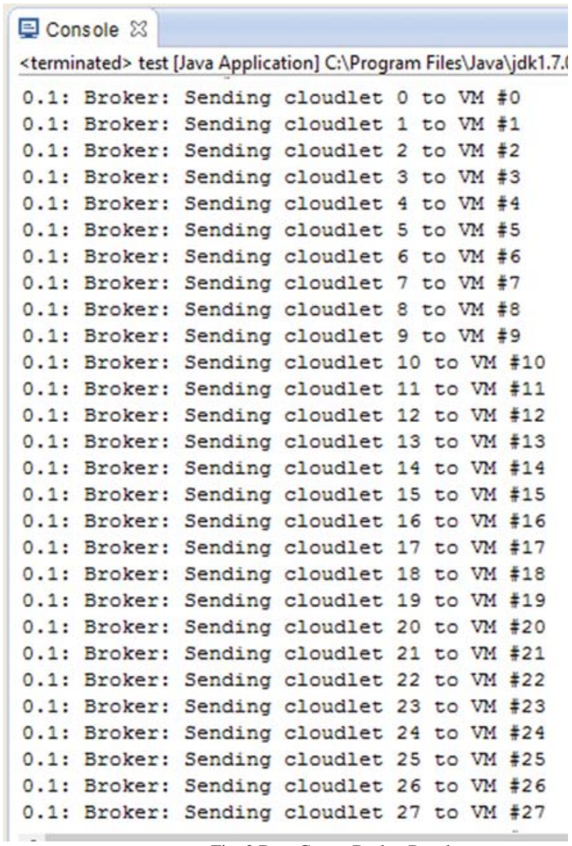
```
Data:
Cloudletlet List: the cloudlets
Deadline: the cloudlet deadline
VmList: the vm list
I : Itration variable
Procedure:
        Sort cloudletList By Deadline
        For  V < VmList  do
                For x < cloudletList Length do
                    Run CloudletList [I] on Vm
                    Remove CloudletList[I]
                End for
        End for
```

## IV. Results

CloudSim outputs a flow of the execution for each cloudlet at every moment of time First Cloudsim starts the datacenter and brokers and then creates the datacenter with hosts and virtual machines Second, the Broker assigns the cloudlets to the hosts with virtual machines, the EDFbroker sends the cloudlet with the earliest deadline to the first virtual machine as appear in fig. 4, then the cloudlet get processed by the processing elements allocated in that virtual machine, Unlike the EDFbroker the Datacenter broker assigns the cloudlets to VMs by first come first served policy as shown in fig.3

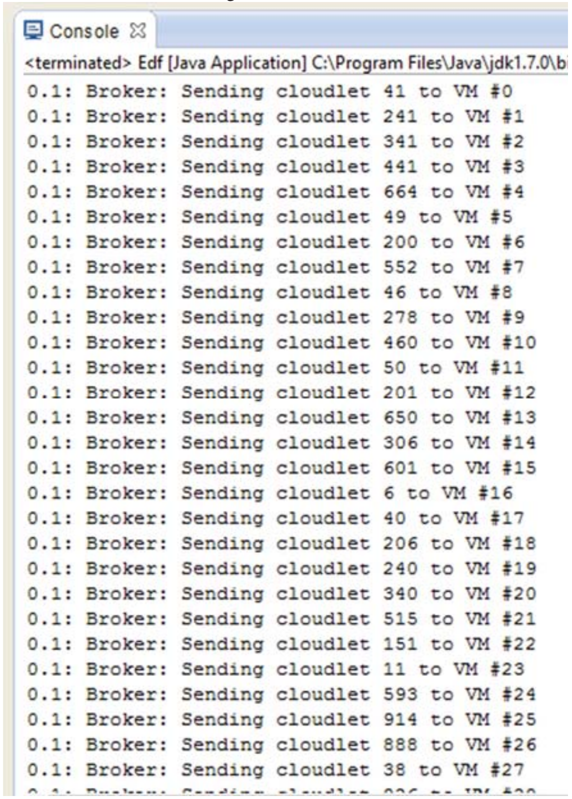
Third, CloudSim presents the result of processing the cloudlets which are the execution time, the finish time, the waiting time and the deadline along with the deadline status in terms of 0 indicates the cloudlet have missed the deadline and 1 indicates the opposite, This results was analyzed to demonstrate the performance of the cloudlets on different numbers of hosts, the analyzed data are gathered through the following testing steps:

*1)* Run the simulation for submitting cloudlets by the *Edfbroker* to 100 hosts.
*2)* Note the cloudlet performance data (the number of missed deadline cloudlets ,and the waiting time, the wall time and the delay after the deadline per cloudlet in seconds)
*3)* Run the simulation for submitting cloudlets by the *Edfbroker* to 150 hosts.
*4)* Repeat step (2).
*5)* Run the simulation for submitting cloudlets by the *Edfbroker* to 200 hosts.
*6)* Repeat step (2).
*7)* Continue until find a proper number of hosts to execute 1000 cloudlets.
*8)* Repeat the steps for the *Datacenter Broker.*
*9)* Repeat the steps for 1.15, 1.25, 1.50 deadline values.



```
Console
<terminated> test [Java Application] C:\Program Files\Java\jdk1.7.
0.1: Broker: Sending cloudlet 0 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #1
0.1: Broker: Sending cloudlet 2 to VM #2
0.1: Broker: Sending cloudlet 3 to VM #3
0.1: Broker: Sending cloudlet 4 to VM #4
0.1: Broker: Sending cloudlet 5 to VM #5
0.1: Broker: Sending cloudlet 6 to VM #6
0.1: Broker: Sending cloudlet 7 to VM #7
0.1: Broker: Sending cloudlet 8 to VM #8
0.1: Broker: Sending cloudlet 9 to VM #9
0.1: Broker: Sending cloudlet 10 to VM #10
0.1: Broker: Sending cloudlet 11 to VM #11
0.1: Broker: Sending cloudlet 12 to VM #12
0.1: Broker: Sending cloudlet 13 to VM #13
0.1: Broker: Sending cloudlet 14 to VM #14
0.1: Broker: Sending cloudlet 15 to VM #15
0.1: Broker: Sending cloudlet 16 to VM #16
0.1: Broker: Sending cloudlet 17 to VM #17
0.1: Broker: Sending cloudlet 18 to VM #18
0.1: Broker: Sending cloudlet 19 to VM #19
0.1: Broker: Sending cloudlet 20 to VM #20
0.1: Broker: Sending cloudlet 21 to VM #21
0.1: Broker: Sending cloudlet 22 to VM #22
0.1: Broker: Sending cloudlet 23 to VM #23
0.1: Broker: Sending cloudlet 24 to VM #24
0.1: Broker: Sending cloudlet 25 to VM #25
0.1: Broker: Sending cloudlet 26 to VM #26
0.1: Broker: Sending cloudlet 27 to VM #27
```

Fig .3 Data Center Broker Results



```
Console
<terminated> Edf [Java Application] C:\Program Files\Java\jdk1.7.0\b
0.1: Broker: Sending cloudlet 41 to VM #0
0.1: Broker: Sending cloudlet 241 to VM #1
0.1: Broker: Sending cloudlet 341 to VM #2
0.1: Broker: Sending cloudlet 441 to VM #3
0.1: Broker: Sending cloudlet 664 to VM #4
0.1: Broker: Sending cloudlet 49 to VM #5
0.1: Broker: Sending cloudlet 200 to VM #6
0.1: Broker: Sending cloudlet 552 to VM #7
0.1: Broker: Sending cloudlet 46 to VM #8
0.1: Broker: Sending cloudlet 278 to VM #9
0.1: Broker: Sending cloudlet 460 to VM #10
0.1: Broker: Sending cloudlet 50 to VM #11
0.1: Broker: Sending cloudlet 201 to VM #12
0.1: Broker: Sending cloudlet 650 to VM #13
0.1: Broker: Sending cloudlet 306 to VM #14
0.1: Broker: Sending cloudlet 601 to VM #15
0.1: Broker: Sending cloudlet 6 to VM #16
0.1: Broker: Sending cloudlet 40 to VM #17
0.1: Broker: Sending cloudlet 206 to VM #18
0.1: Broker: Sending cloudlet 240 to VM #19
0.1: Broker: Sending cloudlet 340 to VM #20
0.1: Broker: Sending cloudlet 515 to VM #21
0.1: Broker: Sending cloudlet 151 to VM #22
0.1: Broker: Sending cloudlet 11 to VM #23
0.1: Broker: Sending cloudlet 593 to VM #24
0.1: Broker: Sending cloudlet 914 to VM #25
0.1: Broker: Sending cloudlet 888 to VM #26
0.1: Broker: Sending cloudlet 38 to VM #27
```

Fig 4. Earliest deadline firstbroker Results

The results are displayed as a comparison between the two brokers in plotted graphs which indicate the number of missed deadline cloudlets, waiting time, wall time and the delay after the deadline of a particular number of hosts and it's expected to achieve the following hypotheses:

A. Sending the earliest cloudlets first to be executed in a cloud host.
B. This sending process should reduce the number of missed deadline cloudlets.
C. Also a reduction in the average waiting time of each cloudlet compared to the *DCbroker* is expected and it should affect the average wall time.

### *A. Missed deadline*

The data that observed from CloudSim analyzed to note the number of missed deadline cloudlets by each broker the *EDFbroker* and the *Dcbroker* in every scenario case using 1.25 deadline value as shown in the Table I bellow

TABLE I. Number of Misssed Deadline Cloudlets

| ***First case : 100 hosts*** | | |
|---|---|---|
| Deadline Status | ***Datacenterbroker*** | ***Edf Broker*** |
| Missed Deadline | 30.7% | 19.4% |
| Met Deadline | 69.3% | 80.6% |
| ***Second case: 150 hosts*** | | |
| Deadline Status | ***Datacenterbroker*** | ***Edf Broker*** |
| Missed Deadline | 52.7% | 44% |
| Met Deadline | 47.3% | 56% |
| ***Third case: 200 hosts*** | | |
| Deadline Status | ***Datacenter broker*** | ***Edf Broker*** |
| Missed Deadline | *12.2%* | 0% |
| Met Deadline | 87.8% | 100% |

Figure 5 shows that the number of missed deadline cloudlets decreases as the number of hosts increases as expected since the cloudlets had more available hosts to be processed, also the EDF broker shows less number of missed deadline each time

.

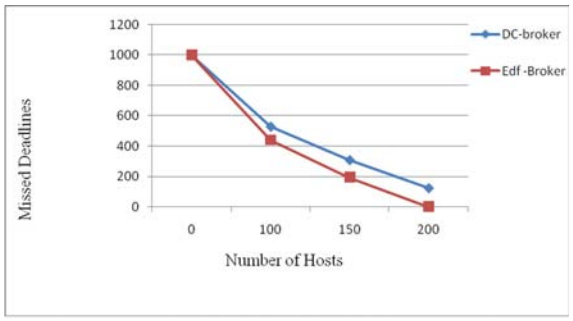


Fig 5 Missed Deadlines Cloudlets

### *B. Average Waiting Time*

The average waiting time decreased by *EDF broker* more than the DC broker meaning that the cloudlet does not waited as much as it waited by the *DC broker*

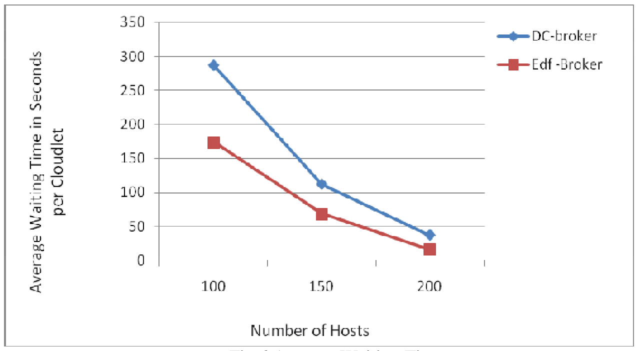


Fig 6 Average Waiting Time

### C. Average Wall Time

Since the execution time for each cloudlet in the processing element is constant and the waiting time decreased more by the EDFbroker it is expected that the wall time to be decreased also as shown in the Fig 7

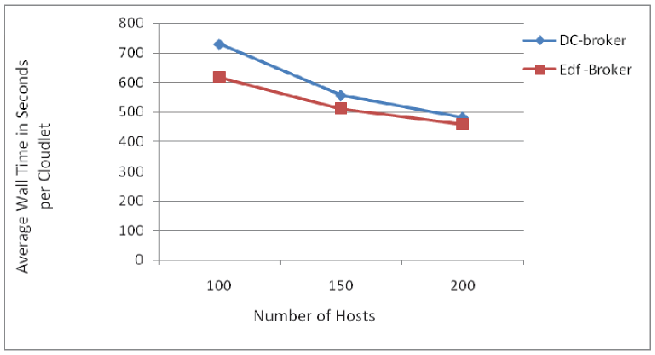


Fig 7Average Wall Time

### D. Delay after Deadline

The average delay after deadline is decreased as shown in Fig.8 implies that cloudlet did not wait as much as it did by the *DCBroker* also because the number of missed deadline cloudlets is decreased.

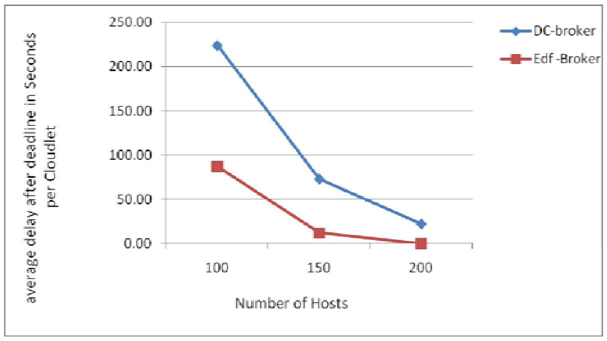


Fig 8 Average Delay after deadline

Additionally when increasing or decreasing the deadline and implement it on the same design every time for 100 and 150 and 200 hosts the number of missed deadline cloudlets affected by this change as shown in the Figures bellow.

100 hosts

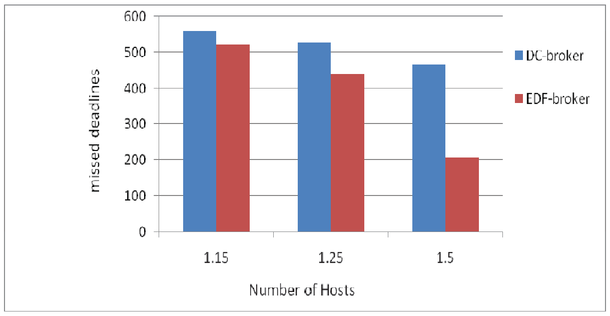


Fig 9 Missed Deadlines for 100 Hosts

150 hosts

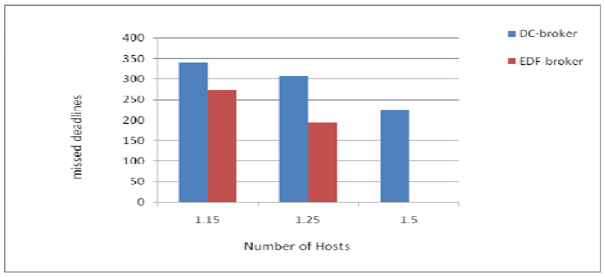


Fig 10 Missed Deadlines for 150 Hosts

200hosts

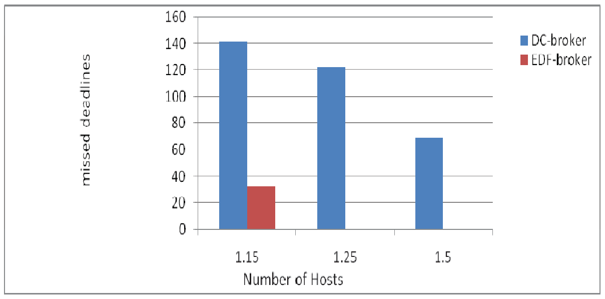


Fig 11 Missed Deadlines for 200 Hosts

### E. Result Discussion

The Earliest Deadline First algorithm results showed a better performance than First Come First Serve algorithm in which the number of missed deadline cloudlet is always lesser in all cases.

The number of missed deadline cloudlet is affected majorly by the number of hosts and the deadline value (greater deadline value and hosts number less missed deadline cloudlets) in which as shown in the graphs the lesser number of hosts means there is no resources available for cloudlets to be executed, so it should Waite for a host to be available and most of cloudlets will miss the deadline.

The Average waiting time for each cloudlet is decreased by the EDFbroker since the cloudlets with earliest deadline has been sent first and executed first by the particular host and the cloudlet with bigger deadline has to wait, but it's still smaller than the waiting time of the cloudlets submitted by the *DCbroker*, also for less number of hosts the cloudlet have to wait longer and vice versa.

The Average Wall time for each cloudlet is the summation of the waiting time and the execution time by each processor element which is fixed in both brokers for each cloudlet because the processor element speed is constant, the degradation in the waiting time leads to a decrease in the total execution time, this degradation growth, while the number of hosts decreases because of the shortage in the available resources.

The Average delay after deadline for each cloudlet is decreased by the *EDFbroker* more, since the number of missed deadline cloudlets is lesser in all cases, also this delay is affected by the number of hosts in which more hosts available shorter delay.

The results of the deadline value 1.15 in (fig 9, fig10, and fig11 ) showed that it's short so the cloudlet does not have much time to finish before the deadline but it could affect the performance greatly with a bigger number of hosts in both brokers , although the *EDFbroker* has a better results, also the deadline value 1.25 results showed lesser number of missed deadline cloudlets than the 1.15 value specially in 200 number of hosts where there is no missed deadline cloudlets, with 1.5 deadline value showed a better results than the two past values in which in 150 and 200 number of hosts cases no cloudlets missed the deadline.

These results concluded that the average waiting time have dropped gradually as the number of hosts increased for each of the cloudlets hence meeting the deadline, and reduced the average wall time for each cloudlets hence improved the hosts utilization , and even reduced the delay after deadline for the cloudlets that had missed the deadline, also when using a greater deadline value the cloudlets takes more time to finish before it considered to be a missed deadline cloudlet hence decreasing the number of missed deadline cloudlets, the waiting time, the wall time and the delay after the deadline under both brokers, but the *EDF broker* allowed the cloudlets to finish earlier. also the greater values of deadlines allows for a use of less resources (number of hosts) and still keep the performance acceptable.

## V. Conclusion

In conclusion, the results of this study provide a wide range of view to the cloud providers to construct the cloud infrastructure in a way that to fulfill the users applications time requirements, Earliest Deadline First algorithm has been implemented to assign tasks (cloudlets) to resources in the broker side by deadline to make sure that each task (cloudlet) meet it's deadline as possible and to reduce the waiting time for the missed deadline cloudlets in order to find a proper infrastructure design for the applications requirements, the results showed the earliest deadline first algorithms reduced the number of missed deadline cloudlets in general also we observed that as the deadline value increases the number of missed deadline cloudlets decreased .

Additionally comparing the results of *EDFbroker* and the *Dcbroker* results showed that the *Edfbroker* reduced the missed deadline cloudlets in all cases more than the *DCbroker*

In general the cloudlets assigned by the *EDfbroke*r showed a better performance Than the *DCbroke*r in term of number of missed deadline cloudlets , waiting time, wall time and the delay after the deadline.

To make the implementation more globally and adaptable apriority based policy could be implemented in case of equal deadlines. Also Predict the deadline of each cloudlet in multiple processor environment by consider more than one processing units for each cloudlet in the implementation scenario and design the broker to Generate cloudlets randomly by interval time.

## References

[1] Nidhi Purohit, Richa Sinha, Hiteishi Diwanji "Resource Optimality Improvisation in Cloud Environment by Efficiency Enhancement through Two Level Scheduling" International Journal of Computer Sci ence And Technology IJCST Vol. 3, Iss ue 2, April - June 2012.

[2] Monika Choudhary, Sateesh Kumar Peddoju" A Dynamic Optimization Algorithm for Task Scheduling in Cloud Environment" International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622I.

[3] C. L. Liu, James W. Layland" Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment" Journal of the Association for Computing Machinery, Vol, 20, No. 1, January 1973, pp. 46-61.

[4] Arezou Mohammadi , Selim G. Akl" Scheduling Algorithms for Real-Time Systems" Technical Report No. 2005-499,2005

[5] Soumya Ray and Ajanta De Sarkar "execution analysis of load balancing algorithms in cloud computing environment" International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.2, No.5, October 2012

[6] Ashish Tiwari, A. Nagaraju, Mehul Mahrishi "An Optimized Scheduling Algorithm for Cloud Broker using Adaptive Cost Model" 978-1-4673-4529-3/12c 2012 IEEE, 2013 3rd International Advance computing Conference (IACC)

[7] Barrie Sosinsky "Cloud Computing Bible" by Wiley Publishing, Inc., Indianapolis, Indiana 2011.