A new Hybridized Multilevel Feedback Queue Scheduling with Intelligent Time Slice and its Performance Analysis

H.S.Behera , Reena Kumari Naik, Suchilagna Parida
Department of Computer Science and Engineering
Veer Surendra Sai University of Technology (VSSUT), Burla, Sambalpur, Odisha,768018, India.

Abstract

In this paper we have proposed a new hybridized multilevel feedback queue with intelligent time slice (ITS). The processes that are entering into the system are assigned to the first ready queue according to their priority which is decided by using HRRN algorithm and then gradually shifted to the next lower level queues upon expiration of time slice. In multilevel feedback queue, the ITS increases gradually while entering to the lower level queues. HRRN scheduling policy reduces the indefinite postponement of the long processes thus reducing starvation. Here control flow diagram has been used to describe the flow of control of execution of processes in respective queues. The proposed approach shows a better and reduced turnaround time, average waiting time and throughput than the other papers.

Keywords *--- CPU burst time, intelligent time slice(ITS), shifting to lower queues, MLFQ Scheduling algorithm, Round Robin scheduling, HRRN scheduling, turnaround time, waiting time, and throughput.*

1. Introduction

Scheduling in multitasking and multiprocessing environment is the way the processes are assigned to execute on the available CPUs. The main goal of scheduling is to minimize the various parameters such as CPU utilization, throughput, turnaround time, waiting time, context switches etc. There are various type of scheduling are used to schedule various processes. Multilevel feedback queue scheduling is a scheduling policy which allows processes entering to the system to move among several queues. Here the processes do not come with any priority but during scheduling they goes down to the lower level queues according to its CPU burst time and calculated time slice(ITS). Here the HRRN scheduling algorithm is merged with multilevel feedback queue to improve the performance. HRRN scheduling policy is similar to the SJN (shortest job next) which decide the priority of the processes based on the execution time and the waiting time. Priority of processes increases as long as they wait in the queue which prevents the indefinite postponement (process starvation). The scheduling is used in the real time applications like routing of data packets in computer networking, controlling traffic in airways, roadways and railways etc. This motivates us to implement multilevel feedback queue scheduling algorithm with sorted remaining burst time with dynamic time quantum concept.

A. Scheduling algorithms

When there are number of processes in the ready queue, the algorithm which decides the order of execution of those processes is called *scheduling algorithm*. The various well known CPU scheduling algorithms are First Come First Serve (FCFS), Shortest Job First (SJF), Highest Response Ratio Next (HRRN) and Priority. All the above four algorithms are non-pre-emptive in nature and are not suitable for time sharing systems. Shortest Remaining Time Next (SRTN) and Round Robin (RR) are pre-emptive in nature. RR is most suitable for time sharing systems.

B. Related Work

There are various type of approaches proposed in different papers in order to increase the overall performance of the MLFQ. Paper[3], a parametric multilevel feedback queue scheduling algorithm that has been proposed to solve the problems regarding the scheduling and also increase the overall performance . Here the priority has also played a most important role. A very small time quantum has been assigned to the very high priority queue and decreased the time

quantum by 1 and doubles the time slice as the level of queue increases. In paper [4], it is proposed that the process does not come with any priority rather it is decided during scheduling. The time quantum assigned were gradually increasing as the priority increases. An approach is given for the long processes whose burst time is so much that they starve during scheduling to get the CPU time. They also proposed a well organized control flow diagram. In another paper [6],Recurrent neural network has been proposed to optimize the quantum of each queue. The RNN can give the most effective model for recognizing the trend information of the time series data .The input of the RNN are the quantum of queues and average response time. Average response time enters as the input to neural network so, that the network obtains a relation between the change of quantum of specified queue with the average response time and with the quantum of other queues and by a change in the quantum of specified queue. In Paper[8], a different type of analysis has been described which is the combination of both best case analysis as well as worst case analysis .The performance is analyzed in terms of time complexity. It was also an effective method for better performance of the multilevel feedback queue scheduling. In another paper [10], the multilevel feedback queue scheduling is implemented in linux kernel. In another paper, multilevel feedback queue with dynamic time quantum has been proposed which shows a better performance. Here the time quantum is calculated based on the mean and median of the processes. Likewise various algorithms were proposed to achieve better performance and reliability of using the scheduling.

C. Our Contribution

Here best suited OTS is calculated based on the execution time and waiting time. Dynamic ITS calculated for each queue with the point that the value of ITS increases as the processes go downward. HRRN scheduling policy is used to prevent the process starvation. Thus it is observed that there is an improvement in the overall performance.

II.A. PROPOSED ALGORITHM

In the proposed approach, the original time slice (OTS) is calculated which is based on the waiting time and the run time or burst time. The ITS calculated for each queue is based on the calculated OTS.

A. Proposed Algorithm

In this algorithm, the first process is assigned to the Q_1 since at the beginning all processes have the same priority and then the Response Ratio or priority of other processes is calculated which is based on the waiting time and the execution time . As long as the process waits in the ready queue the priority increases. Then according to the priority the processes are assigned to the ready queue for execution. Then the OTS is calculated for each process based on the burst time. ITS is calculated based on the OTS (original time slice), PC (priority component), SC(shortness component) and CSC (context switch component) for each process and average of the ITS of each processes is assigned as the time slice for the Q_1. Upon expiration of the ITS the processes move toward the lower level queue till completion. The ITS increases as processes move towards lower level queue. The response of

$$\text{Response Ratio} = \frac{\text{waiting time + expected run time}}{\text{expected run time}}$$

The Response Ratio is calculated for each queue before entering to the queue. According to the response time the processes are scheduled. The process having higher response ratio will be assigned first to the ready queue for execution.
To calculate ITS we introduce some parameters and those are described below:
OTS (original time slice): It is calculated for each process based on the execution time. It depends upon the range, number of processes and priority.

$$\text{Range} = \frac{\text{Max burst time+ Min burst time}}{\text{Max burst time- Min burst time}}$$

OTS= range+ (no. of processes) + (priority of current Process).

PN (priority number): It is decided based on the burst time of each process (process having smallest burst time is given highest priority).

PC (priority component): It can be calculated using PN.

$$PC=1/PN \begin{cases} 1, \text{ if } PN=1 \\ 0, \text{ if } PN>1 \end{cases}$$

SC (shortness component): It is calculated based on burst time of current process and previous process.

$$SC= \begin{cases} 1, \text{ if burst time (current-previous) process}<0 \\ 0, \text{ otherwise} \end{cases}$$

CSC (context switch component): It is calculated based on burst time, PC, SC and OTS.

$$SC=\begin{cases} 1, \text{ if (burst time-(PC+SC+OTS))} < 0 \\ 0, \text{ otherwise} \end{cases}$$

ITS (intelligent time slice): It is calculated based on all the above parameters. It should not be greater than the maximum burst time.

PSEUDO CODE:

1. Let n: number of processes

   m: number of levels

   l: level

   b[i]: burst time of ith process

   rb[i]: remaining burst time of ith process

   OTS: original time slice,

   ITS: intelligent time slice

   PC: priority component, SC: shortness component, CSC: context switch component

   Initialize: l=1,avg tat=0

2. Insert the processes p_1 to Q_1

   then

   Priority of remaining Pi where i=2 to 5 are calculated using HRRN.

   Assign all P_i to Q_1. According to their priority

3. Calculate the OTS

Range = $\frac{\text{Max burst time+ Min burst time}}{\text{Max burst time- Min burst time}}$

OTS= range + (no. of processes) +(priority of current process)

4. Calculate ITS

   for(m=1 to 5)

   {

   While (ready queue!=NULL)

   {

ITS= OTS+PC+SC+CSC// for each processes

ITS1=sum of all ITS / total no. of processes

Q_1<-ITS_1

$Q_{2<--}2*ITS_1$

   }

   }

5. If(b_t>=ITS)

   {

   $Rb_t=b_t$-ITS;

   $Q_{m+1<}$--Rb_t;

   }

   Else

   $Q_{m<}$--P[i];

6. if(m>=5)

{

Sort the all Rb_t of processes in ascending order;

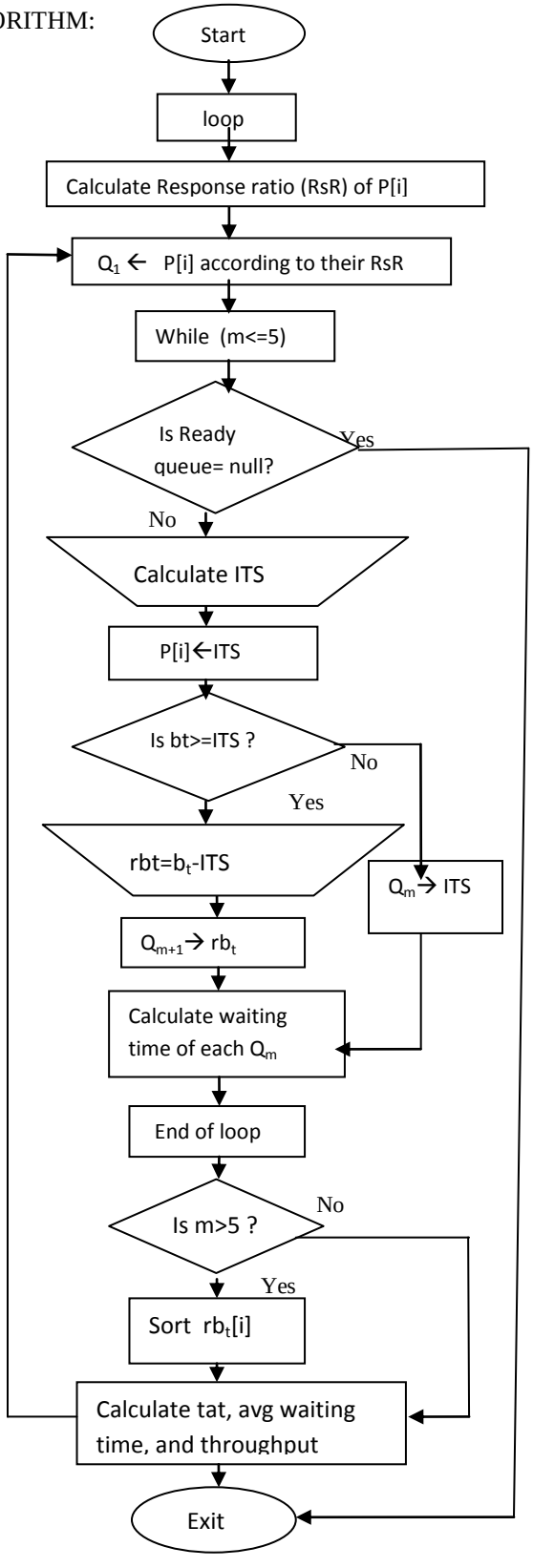
Then resend them to the respective queues for complete execution.

}

7. Calculate avg tat, waiting time and throughput.

8. stop and exit.

Fig2. FLOW CHART OF THE PROPOSED ALGORITHM:



II. B. CONTROL FLOW DIAGRAM OF THE PROPOSED ALGORITHM

A control flow diagram (CFD) describes the sequence of flow of control of process or program. It can consist of a subdivision to show sequential steps, with different conditional statements, repetitions and case conditions.

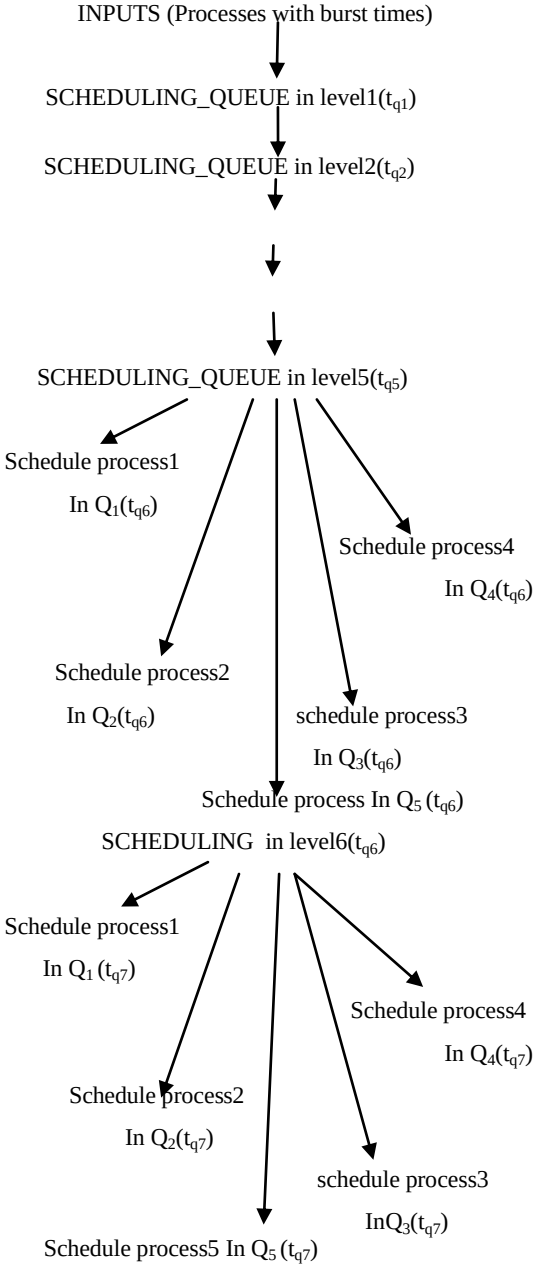


Figure 2.Control Flow Diagram showing the scheduling process.

This algorithm will work for n processes but here for experimental purpose we have taken 5 queues (Q_1, Q_2, Q_3, Q_4 and Q_5) .If any process does not

complete its execution within these queues and reaches to the lowest queue i.e. Q_5 then the remaining processes are to be rescheduled in their respective queues. The processes are arranged according to their remaining CPU burst time and send them to their respective queues. The processes are in increasing or decreasing order and the process having minimum burst time is sent to Q_1 and next process to Q_2 and in the same way processes are assigned up to Q_5 so that Q_5 must get the process with highest remaining burst time. The same procedure repeats till all the processes have finished their execution.

The process of execution repeats till all the processes are finished within their given burst times. Here the order of time quanta is

$$t_{q1} < t_{q2} < t_{q3} < t_{q4} < t_{q5} < t_{q6} < t_{q7}$$

III. EXPERIMENTAL ANALYSIS

A. Assumptions

The environment is assumed to be time sharing, multiprocessor and multitasking. OTS and ITS are assumed to be not more than the maximum burst time. All the attributes like CPU burst time, number of processes, OTS and ITS of all the processes are known before submitting the processes to the processor. All processes are CPU bound. No processes are I/O bound.

B. Data set

We have performed three experiments for evaluating various performances. We have taken three different cases for evaluation in increasing, decreasing and random order of burst time.

C. Performance Metrics

There should be some significance output for better performance and those are as follows:
1) Turnaround time (TAT): The average turnaround time should be less for better performance.
2) Waiting time (WT): waiting time is the time of the process that waits for the CPUs to execute .The average waiting time should be less for better performance.
3) Throughput: Throughput of the algorithm should be more for better performance.

D. Experiments Performed

Here to evaluate the performance of the proposed algorithm we have taken 5 processes in increasing, decreasing and random order for each cases . The algorithm works effectively for any type of processes. In each case, we have compared the experimental results of our proposed algorithm with the previous proposed MLFQ algorithms. Here we have taken a dynamic ITS as time slice for each queue. And hence the turnaround time, average waiting time and throughput are calculated.
The OTS is calculated using the following formula:

$$\text{Range} = \frac{\text{Max burst time+ Min burst time}}{\text{Max burst time- Min burst time}}$$

OTS= range +no. of processes +priority of current process

To calculate IT'S THE formula used is

ITS=OTS+(total no. of processes)+(priority of current process).

For example: table 1:

Process	Burst Time	PN	PC	SC	CSC	OTS	ITS
P1	290	1	1	0	0	10	11
P2	300	2	0	0	0	11	11
P3	324	3	0	0	0	12	12
P4	400	4	0	0	0	13	13
P5	520	5	0	0	0	14	14

The average ITS = (11+11+12+13+14)/5=61/5=

12.2(rounding up) =12

The various ITS for each queue is calculated as follows;

ITS1=12

$ITS_2=2* ITS_1=24$

$ITS_3=2*ITS_2 =48$

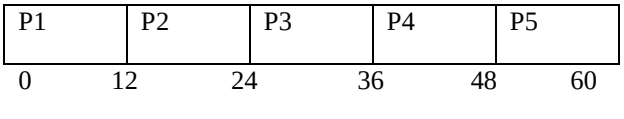
$ITS_4=2*ITS_3 =96$

$ITS_5=2*ITS_4 =192$

III.A.GANTT CHART

Gantt chart for case 1:

$ITS_1=12$; arrival time =0



ITS_2=24, arrival time=60

P1	P2	P3	P4	P5	
60	84	108	132	156	180

ITS_3=48, arrival time=180

P1	P2	P3	P4	P5	
180	228	276	324	372	420

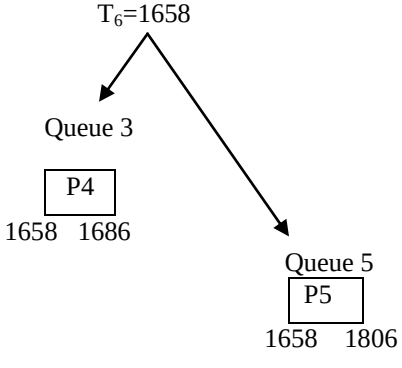
ITS_4=96, arrival time=420

P1	P2	P3	P4	P5	
420	516	612	708	804	900

ITS_5=192, arrival time=900

P1	P2	P3	P4	P5	
900	1010	1130	1274	1466	1658

When the processes reach to the lowest queue but have not completed then the remaining CPU burst time of each process are calculated and sort all the values in ascending order .Rescheduled the processes by sending them into their required respective queues. The process having least burst time is send to Q_1 and the next process to Q_2. In the same manner all processes have been sent to different queues according to their values. Q_5 must have the process with highest remaining burst time. The same procedure is repeated till all the processes get executed. Here ,since two processes are remaining those are P4and P5 so those are scheduled as follows,



For this case the turnaround time is 1806.

average waiting time =

$$\frac{1}{\text{no. of processes}} \sum_{i=1}^{n} \text{waiting time of queue at each level}$$

$$\text{Throughput} = \frac{\text{no. of processes completed}}{\text{total time taken}}$$

Here we have taken few test cases of different CPU burst times which gives a comparison based on the idea of proposed approach and other MLFQs. We have taken 5 different processes for each test case which are to be scheduled.

Table 2(a): FIRST TEST CASE INPUT:

(Five processes with CPU burst times in increasing order)

Process	Burst time	PN	PC	SC	CSC	OTS	ITS
P1	290	1	1	0	0	10	11
P2	300	2	0	0	0	11	11
P3	324	3	0	0	0	12	12
P4	400	4	0	0	0	13	13
P5	520	5	0	0	0	14	14

Table 2(b): FIRST TEST CASE OUTPUT:

algorithms	Turnaround time	Average waiting time	throughput
Power MLFQ	1834	648	$2.73*10^{-3}$
EMLFQ	1834	648	$2.73*10^{-3}$
Proposed approach	1806	473	$2.76*10^{-3}$

Table 3(a): SECOND TEST CASE INPUT:

(Five processes with CPU burst times in decreasing order)

Process	Burst time	PN	PC	SC	CSC	OTS	ITS
P1	522	5	0	0	0	9	11
P2	390	4	0	1	0	10	12
P3	326	3	0	1	0	11	12
P4	280	2	0	1	0	12	12
P5	276	1	1	1	0	13	13

Table 3(b): SECOND TEST CASE OUTPUT:

algorithms	Turnaround time	Average waiting time	Throughput
Power MLFQ	1794	641	$2.78*10^{-3}$
EMLFQ	1794	641	$2.79*10^{-3}$
Proposed approach	1788	541	$2.80*10^{-3}$

Table 4(a): THIRD TEST CASE INPUT:

(Five processes with CPU burst times in random order)

Process	Burst time	PN	PC	SC	CSC	OTS	ITS
P1	328	4	0	0	0	9	11
P2	282	5	1	1	0	10	13
P3	580	1	0	0	0	11	11
P4	360	3	0	1	0	12	12
P5	420	2	0	0	0	13	11

Table 4(b): THIRD TEST CASE OUTPUT:

algorithms	Turnaround time	Average waiting time	throughput
Power MLFQ	1970	661	$2.538*10^{-3}$
EMLFQ	2970	661	$2.54*10^{-3}$
Proposed approach	1934	556	$2.90*10^{-3}$

In all the above test cases the calculated turnaround time, average waiting time & throughput are better than previous MLFQ papers.

III.B. PERFORMANCE ANALYSIS

Here we have analyzed the performance of our proposed algorithm with other MLFQ algorithms in terms of graph. The graph shows the turnaround time, average waiting time and throughput along the Y-axis and the number of test cases along the X-axis.

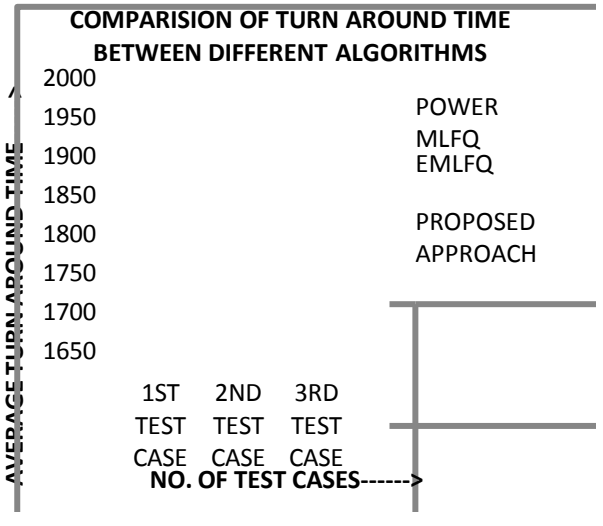


Figure 3.turnaround time of various proposed approaches for 1st test case, 2nd test case and 3rd test case are shown.

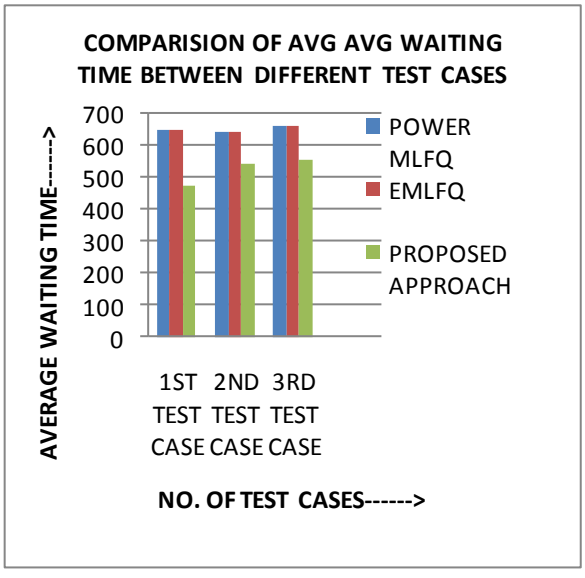


Figure 4.Average waiting time of the queues at each level of various proposed approaches for 1st test case, 2nd test case & 3rd test case are shown

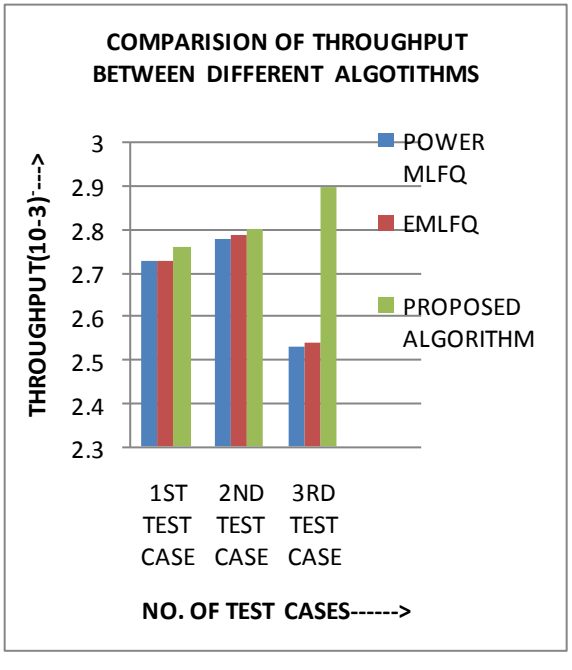


Figure 5.throughput of various proposed approaches for 1st test case, 2nd test case and 3rd test case are shown.

IV.CONCLUSION AND FUTURE WORK

Here the multilevel feedback queue scheduling and HRRN scheduling are merged to improve the performance as well as to prevent the indefinite postponement of the processes. The starvation is reduced by this proposed approach.

The number of queues and the time quantum of each queue also affect the performance a lot. Here a new ITS introduced in order to enhance overall performances. As a result reduction in the turnaround time and the waiting time in Multilevel Feedback Queue Scheduling was achieved. The throughput for each queue has also been increased. As the turnaround time and waiting time decreases, the execution becomes faster. Throughput increases and hence the CPU utilization also increases.

Here we found less turnaround time, average waiting time and high throughput than the previous proposed algorithm so we can conclude that this proposed approach is optimal.

This algorithm can be used on multitasking, multiprocessing, time sharing and distributed systems in an effective way that the research is still being continued in these fields.

V. REFERENCES

[1]Shafigh Parsazad, E. Saboori(2010)" A new scheduling algorithm for server farms load balancing", International Journal Conference on Industrial and information System.

[2]. H.S. Behera, Reena Kumari Naik, Suchilagna Parida,"Improved Multilevel Feedback Queue Scheduling using Dynamic Time Quantum and Its Perfarmance Analysis",IJCSIT,Vol.3(2),2012.

[3].Baney, Jim and Livny, Miron (2000)," Managin NetworkResources in Condor", 9th IEEE Proceedings of the International Symposium on High Performance Distributed Computing, Washington, DC, USA

[4].Parvar, Mohammad R.E, Parvar, M.E. and Safari, Saeed (2008),"A Starvation Free IMLFQ Scheduling Algorithm Based on Nueral Network", International Journal of Computational Intelligence Research ISS 0973-1873 Vol.4, No.1 pp.27- 36 .

[5] Wolski, Rich, Nurmi, Daniel and Brevik, Jhon (2007)," An Analysis of Availability Distributions in Condor", IEEE, University of California, Santa Barbara .

[6] .Litzkow, Micheal J., Linvey, Miron and Mutka, Matt W. (1988)," Condor –A Hunter of Idle Workstations" IEEE, Department of Computer Sciences, University of Wisconsin, Madison .

[7]. Abawajy, Jemal H. (2002),"Job Scheduling Policy for High Throughput Computing Environments", Ninth IEEE International Conferences on Parallel and Distributed Systems, Ottawa, Ontario, Canada .

[8] Liu, Chang, Zhao, Shawn and Liu, Fang (2009), "An Insight into the architecture of Condor-a Distributed Scheduler", IEEE, Beijing, China .

VI. AUTHOR'S BIOGRAPHY

1. Dr. H. S. Behera is currently working as a Faculty in Dept. of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Sambalpur, 768018, Odisha, India. His areas of interest include Distributed Systems, Data Mining, and Soft Computing.
2. Reena Kumari Naik is a final year B.Tech student in Dept. of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Sambalpur, 768018, Odisha, India.
3. Suchilagna Parida is a final year B.Tech student in Dept. of Computer Science and Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Sambalpur, 768018, Odisha, India.