STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **3**, June 2015, pp 189–196.
Published online in International Academic Press (www.IAPress.org)

New Design and Implementation of MLFQ Scheduling Algorithm for Operating Systems Using Dynamic Quantum

Seifedine Kadry [1,*], Armen Bagdasaryan [1]

[1] *Department of Mathematics and Statistics, American University of the Middle East, Kuwait*

Received: 13 January 2014; Accepted: 10 February 2015
Editor: Reza Shadnam

Abstract The new design of multilevel feedback queue (MLFQ) will depend on the usage of a new technique in computing the quantum in order to produce Auto Detect Quantum (ADQ) which is relied on the burst of each process that has been enrolled to the system. By summating the burst time of each process that has arrived and dividing it by the number of available processes, we can obtain the dynamic quantum in each level of scheduling. The processes are scheduled and shifted down from queue to other one according to their remaining bursts time that should be updated periodically. Every queue has a unique auto detected quantum which is gradually increased or decreased from top level to bottom level queues according to the case of arriving processes. Depending on the results of graphical simulating algorithm on case studies, we can discover that a dynamic quantum is very suitable to accommodate low priority processes that still for a long duration to complete their requests, i.e. avoid the starvation of CPU bounded processes. Although, it is still compatible with high priority processes (I/O-Bounded) to provide a fair interactivity with them. In comparison to traditional MLFQ, the performance of the new scheduling technique is better and practical according to the applied results. Additionally, we developed suitable software to simulate the new design and test it on different cases to verify and validate it.

Keywords Multilevel Feedback Queue, Auto Detect Quantum, Dynamic Quantum, Scheduling Algorithm, Operating Systems, CPU Processes

AMS 2010 subject classifications 68M20

DOI: 10.19139/soic.v3i2.58

1. Introduction

In this paper, we propose a new design of multilevel feedback queue technique and propose suitable implementation for it. The new design of multilevel feedback queue will depend on the usage of a new technique in computing the quantum in order to produce Auto Detect Quantum which is relied on the burst of each process that has been enrolled to the system. By adding the burst of each process that has arrived for processing and dividing it by the number of available processes, we can obtain the quantum dynamically in each level of scheduling, i.e. depending on previous and posterior knowledge of process length. In this design, processes are scheduled and shifted down from queue to queue according to their remaining bursts time that should be updated periodically. Every queue has a unique auto detected time quantum that gradually increases or decreases from top level to bottom level queues according to the case of arriving processes. In addition, the processes do not come with any priority, so the arrival time will decide which process is enrolled first to the system and is given the dynamic quantum periodically, i.e.

*Correspondence to: Seifedine Kadry (Email: skadry@gmail.com). Department of Mathematics and Statistics, American University of the Middle East. Block 3, Building 1, Egaila, Kuwait.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

the (Round-Robin) procedure will distribute the dynamic quantum among them fairly. If the processes have the same arrival times, they will be ordered ascending or descending according to the lengths of the processes.

The main purposes of applying a dynamic quantum instead of a static quantum are to accommodate low priority jobs that still for a long duration to complete their request and to maximize the resource utilization, i.e. avoid the starvation of CPU-bounded processes.

The new MLFQ will be more efficient in dealing with low priority process (CPU-Bounded) and compatible with high priority process (I/O-Bounded), i.e. it solves the problems of good response time but large turnaround time. So the proposed MLFQ scheduling algorithm works better compared to the traditional MLFQ.

The rest of the paper is organized as follows. Section 2 presents how the traditional MLFQ works. In Section 3, a graphical representation algorithm of the new MLFQ is presented. The simulation of the MLFQ scheduling algorithm using dynamic quantum as well as static quantum is presented in Section 4. Finally, Section 5 concludes the paper.

2. MLFQ: How it works?

In MLFQ, jobs are scheduled according to their remaining CPU burst and they are shifted down from queue to queue as they have some remaining CPU burst, as we see in Fig. 1 [4]. Each queue has a static time quantum that progressively increases from higher level queue to lower level queue. When a process is enrolled to the system, it is placed at the highest priority. If a process consumes an entire time quantum while running, its priority is reduced and moved down one queue. The process will be done if it consumes all burst time during an entire time quantum as we see in Fig. 1. So, the CPU-bounded processes are descended from upper queues to lower queues progressively for getting completed. As a result, lower priority queue is crowded with CPU-bounded processes, i.e. these processes will suffer from a starvation for a long time. The basic MLFQ model has several advantages in scheduling and the algorithm is simple and easy to verify and understand [1].

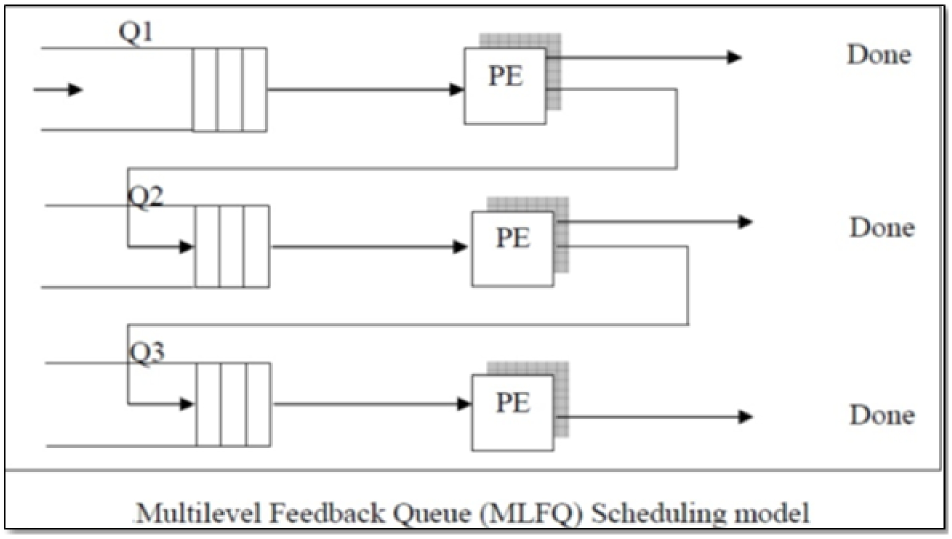


Figure 1. MLFQ-Model

3. Graphical representation algorithm

In the proposed algorithm, four major functions are implemented in each layer of the new multilevel feedback queue. Those functions insert new processes one by one in each queue checking the arrival time of the new processes, compute dynamic quantum for each new level and calculate the waiting time for each process that has been enrolled in queue.

This new design describes the mechanism of how processes are scheduled from high level queue to the low level queues. The proposed mechanism will be clear after producing the results and Gantt chart.

The pseudo code of new MLFQ Scheduling Algorithm is:

```
1. Start of program
• For the first queue (high priority process), we have the following:
2. Set initial quantum to one /Q=1/
3. Start insertion of processes
4. Check every 1 ms for new processes
5. If the system has a new one process at this time then /different arrival time/
6. Update the first dynamic quantum /new quantum = burst of first process/
7. Else /same arrival time/
8. Update the first dynamic quantum /new quantum = summation the bursts of arriving processes at same time/
9. Set out scheduling in Round-Robin with auto dynamic quantum: New burst = burst (pi) - Q /Calculate new
   burst/
10. Compute (Response time, Waiting time, Turnaround time) for each process in level 1
11. If new burst = 0 then the process will be done
12. Else, process will move to the second level
• For the second queue, we have the following:
13. Sum = 0 /initialize value of burst summation buffer in each new level/
14. For i=1 to n do /n = # of processes/
15. Sum = sum + burst (pi)
16. Update the second dynamic quantum /new quantum = dividing the summation of remaining bursts by the
    number of available processes/
17. Start second scheduling in round robin with auto dynamic 2nd quantum
18. Calculate wait, turn (cumulatively) for available processes
19. New burst = burst (pi) - Q /Calculate new burst/
20. If new burst = 0 then the process will be done
21. Else process will move to the third level
22. Update number of available processes
• For the third queue (low priority processes), we have the following:
23. Update the third dynamic quantum /new quantum = burst time of current process/
24. Start final scheduling in FCFS
25. Calculate Final wait, Final turn (cumulatively) for all processes
26. Display the graphical representation /Gantt Chart/
27. END processing
```

4. Simulations and Result Analysis of MLFQ

In this section we present the results of four simulation cases, two for static quantum (traditional) and two for dynamic quantum (new design), for five processes (P1, P2, P3, P4, P5). By comparing the results of two main cases (static, dynamic), we get the following: the response time of each process increases slightly and the wait time decreases extremely when applying the new design; in turn, this will minimize the total turnaround time of starved processes and provide a fair interactivity of the system. Now, we will run the graphical simulating program by providing an input data as follows:

4.1. *Simulation one: Same arrivals and constant quantum*

In first case, MLFQ scheduler has three queues having static time quantum, suppose 8, 16. This time quantum is taken to show the increasing order of time and how MLFQ will behave according to it. Note: arrival times for all processes are equal (see Fig. 2).

Process	Burst time	Arrival time	Response time	Wait time	Turn-around time
P1	15	0	0	32	47
P2	29	0	8	87	116
P3	33	0	16	92	125
P4	45	0	24	101	146
P5	52	0	32	122	174

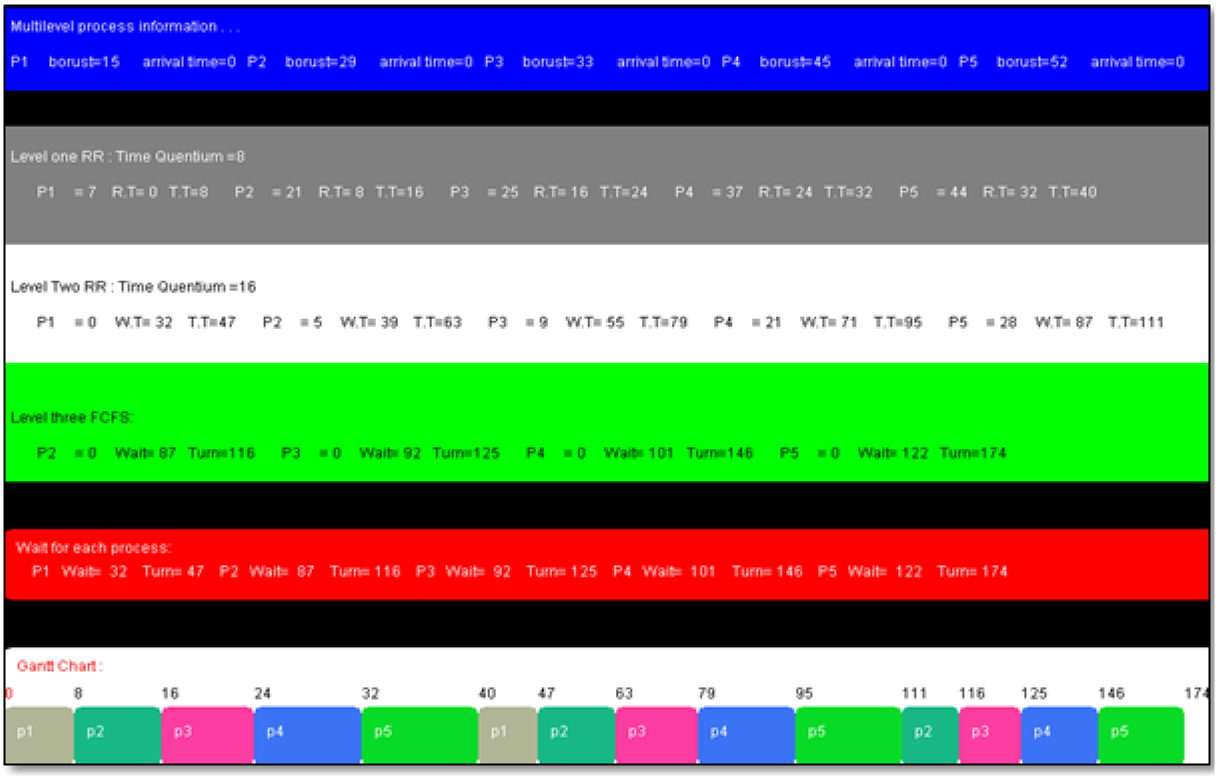
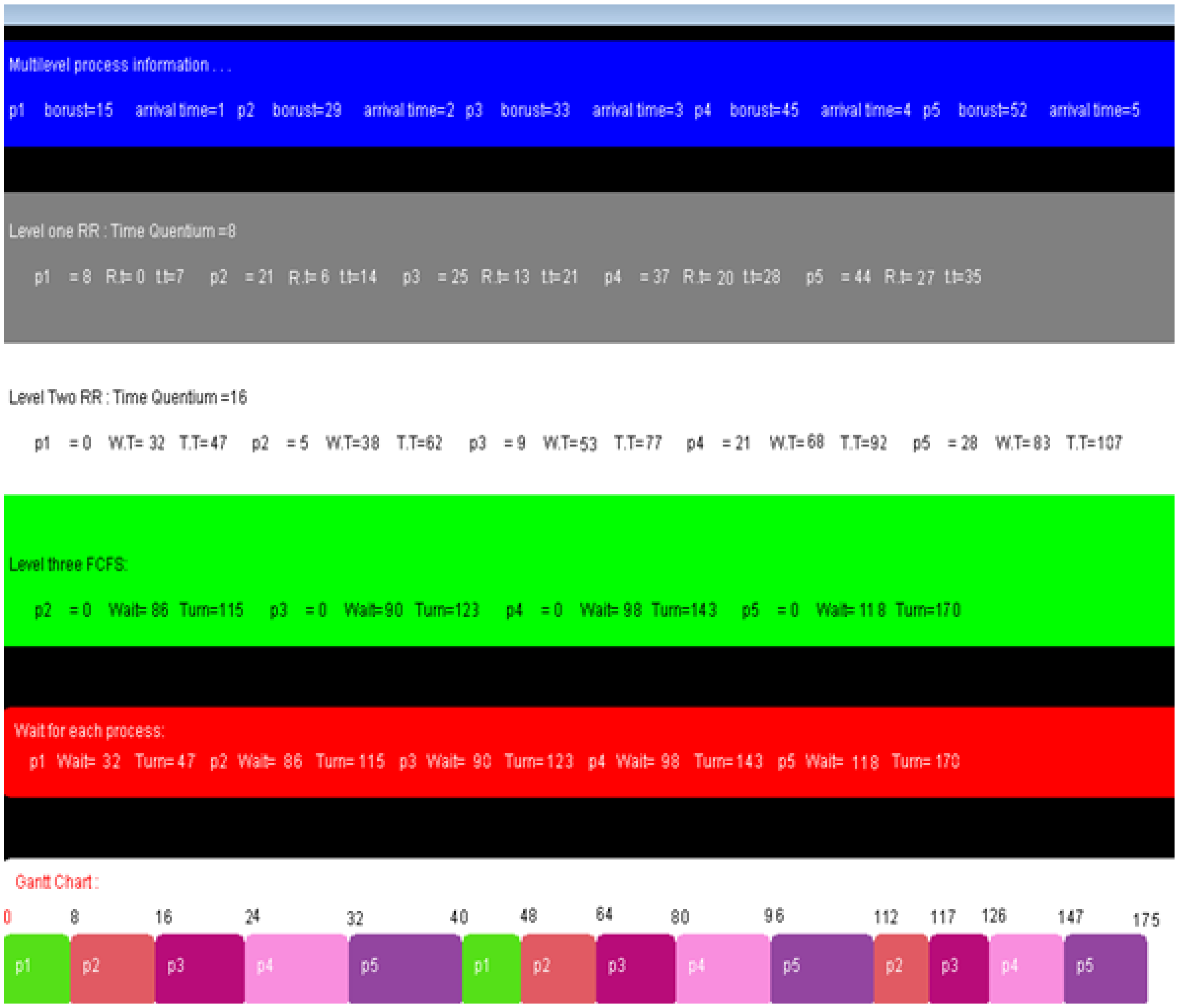

```
Multilevel process information . . .
P1   borust=15   arrival time=0  P2   borust=29   arrival time=0  P3   borust=33   arrival time=0  P4   borust=45   arrival time=0  P5   borust=52   arrival time=0

Level one RR : Time Quentium =8
    P1   = 7   R.T= 0  T.T=8    P2   = 21   R.T= 8  T.T=16    P3   = 25   R.T= 16  T.T=24    P4   = 37   R.T= 24  T.T=32    P5   = 44   R.T= 32  T.T=40

Level Two RR : Time Quentium =16
    P1   = 0   W.T= 32   T.T=47    P2   = 5   W.T= 39   T.T=63    P3   = 9   W.T= 55   T.T=79    P4   = 21   W.T= 71   T.T=95    P5   = 28   W.T= 87   T.T=111

Level three FCFS:
    P2   = 0   Wait= 87  Turn=116    P3   = 0   Wait= 92  Turn=125    P4   = 0   Wait= 101  Turn=146    P5   = 0   Wait= 122  Turn=174

Wait for each process:
  P1  Wait=  32   Turn= 47   P2  Wait=  87   Turn= 116   P3  Wait=  92   Turn= 125   P4  Wait=  101   Turn= 146   P5  Wait=  122   Turn= 174

Gantt Chart :
```

Figure 2. Graphical representation of MLFQ with static quantum and same arrival time

4.2. *Simulation two: Different arrivals and constant quantum*

In second case, MLFQ scheduler has three queues having static time quantum, suppose 8, 16. This time quantum is taken to show the increasing order of time and how MLFQ will behave according to it. Note: Arrival times for all processes are different (see Fig. 3).

Process	Burst time	Arrival time	Response time	Wait time	Turn-around time
P1	15	1	0	32	47
P2	29	2	6	86	115
P3	33	3	13	90	123
P4	45	4	20	98	143
P5	52	5	27	118	170



```
Multilevel process information . . .
p1  borust=15  arrival time=1  p2  borust=29  arrival time=2  p3  borust=33  arrival time=3  p4  borust=45  arrival time=4  p5  borust=52  arrival time=5

Level one RR : Time Quentium =8
    p1  =8  R.t=0  t.t=7    p2  =21  R.t=6  t.t=14    p3  =25  R.t=13  t.t=21    p4  =37  R.t=20  t.t=28    p5  =44  R.t=27  t.t=35

Level Two RR : Time Quentium =16
    p1  =0  W.T=32  T.T=47    p2  =5  W.T=38  T.T=62    p3  =9  W.T=53  T.T=77    p4  =21  W.T=68  T.T=92    p5  =28  W.T=83  T.T=107

Level three FCFS:
    p2  =0  Wait=86  Turn=115    p3  =0  Wait=90  Turn=123    p4  =0  Wait=98  Turn=143    p5  =0  Wait=118  Turn=170

Wait for each process:
  p1  Wait= 32  Turn=47  p2  Wait= 86  Turn= 115  p3  Wait= 90  Turn=123  p4  Wait= 98  Turn= 143  p5  Wait= 118  Turn= 170

Gantt Chart :
```

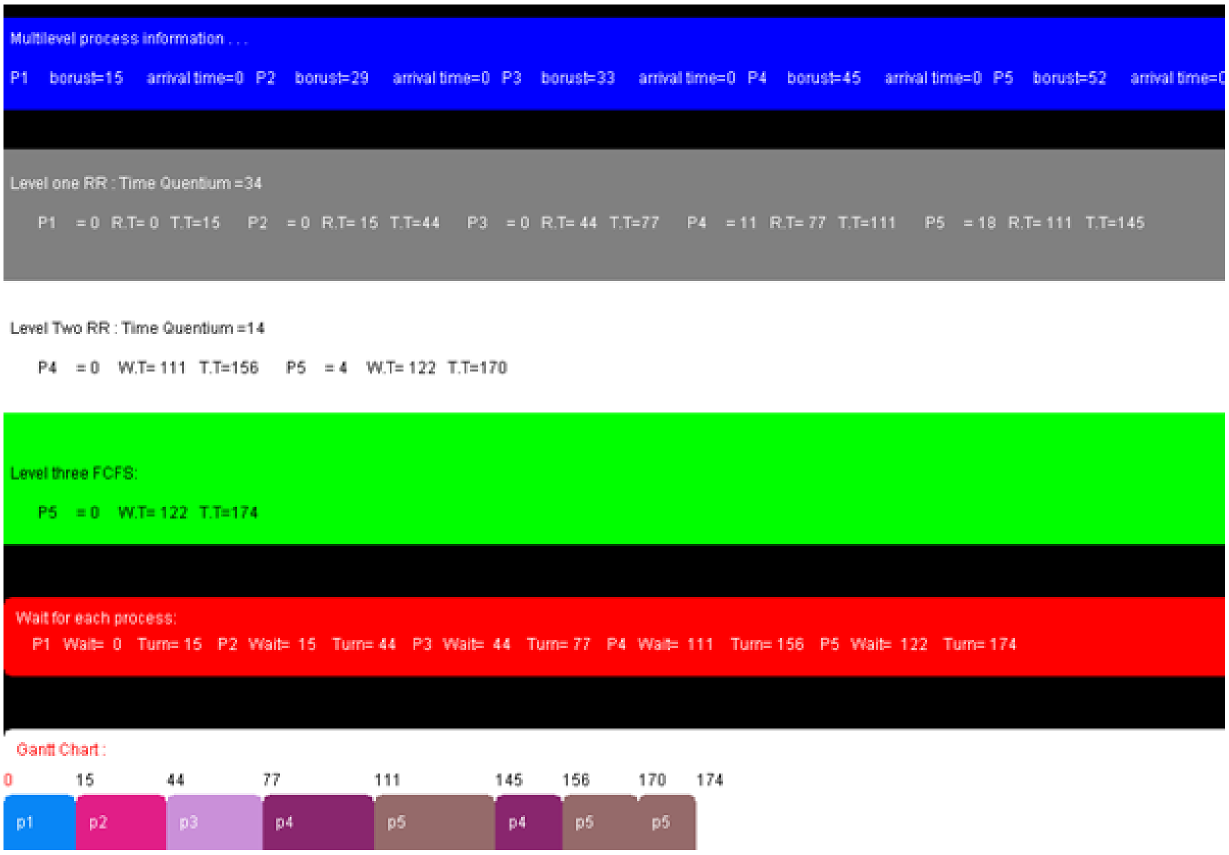
Figure 3. Graphical representation of MLFQ with static quantum and different arrival time

From Fig. 2 and Fig. 3 one can see that traditional MLFQ will provide good response time but bad turnaround time, the number of starved processes is increased to about 80% of them.

4.3. *Simulation three: Same arrivals and dynamic quantum*

In third case, MLFQ scheduler has three queues having dynamic time quantum. This time quantum is taken to show the increasing or decreasing order of time and how MLFQ will behave according to it. Note: Arrival times for all processes are equal (see Fig. 4).

Process	Burst time	Arrival time	Response time	Wait time	Turn-around time
P1	15	0	0	0	15
P2	29	0	15	15	44
P3	33	0	44	44	77
P4	45	0	77	111	156
P5	52	0	111	122	174



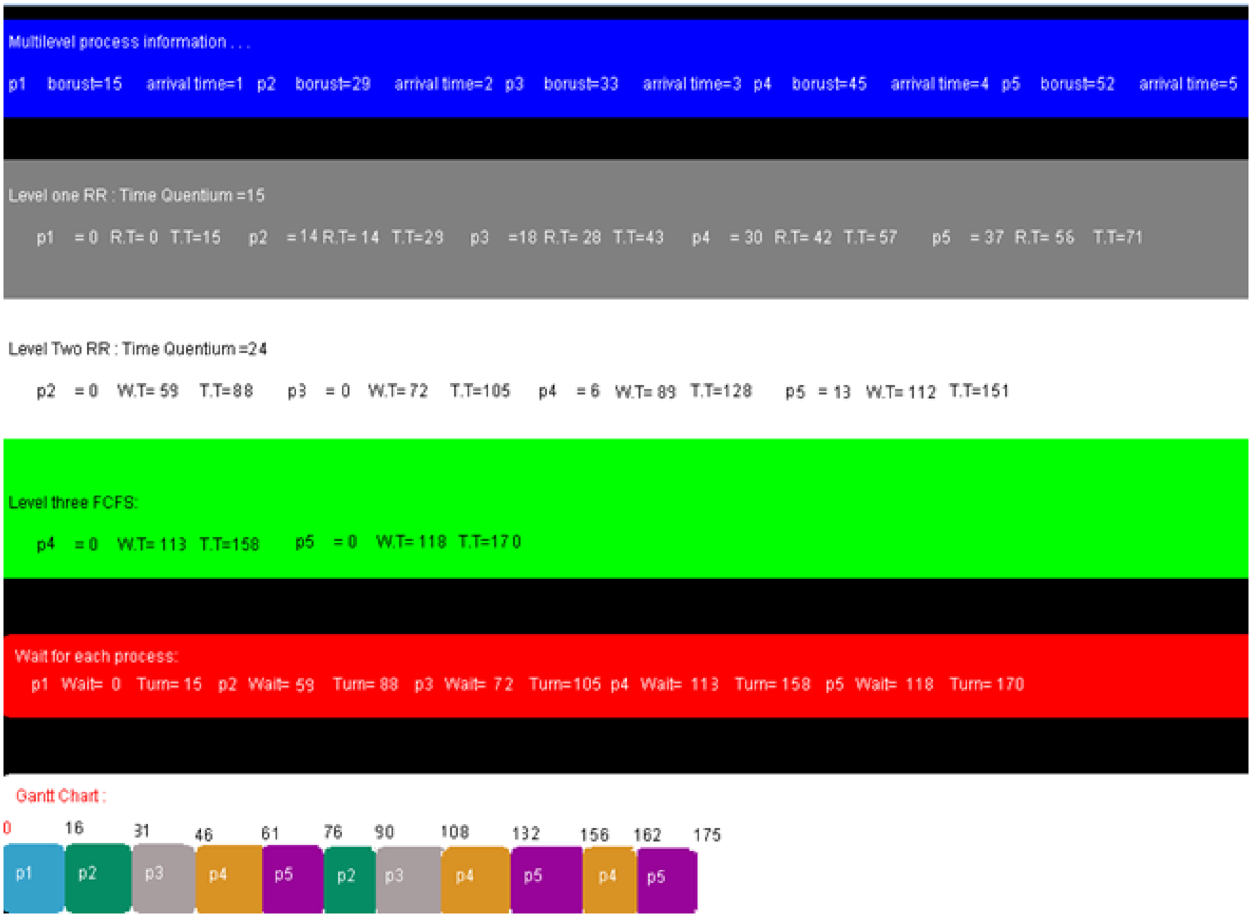
```
Multilevel process information . . .
P1    borust=15    arrival time=0  P2    borust=29    arrival time=0  P3    borust=33    arrival time=0  P4    borust=45    arrival time=0  P5    borust=52    arrival time=0

Level one RR : Time Quentium =34
    P1    = 0  R.T= 0  T.T=15    P2    = 0  R.T= 15  T.T=44    P3    = 0  R.T= 44  T.T=77    P4    = 11  R.T= 77  T.T=111    P5    = 18  R.T= 111  T.T=145

Level Two RR : Time Quentium =14
    P4    = 0   W.T= 111  T.T=156    P5    = 4   W.T= 122  T.T=170

Level three FCFS:
    P5    = 0   W.T= 122  T.T=174

Wait for each process:
   P1  Wait=  0   Turn= 15   P2  Wait=  15   Turn= 44   P3  Wait=  44   Turn= 77   P4  Wait=  111   Turn= 156   P5  Wait=  122   Turn= 174

Gantt Chart :
```

Figure 4. Graphical representation of new MLFQ with dynamic quantum and different arrival time

4.4. *Simulation four: Different arrivals and dynamic quantum*

In fourth case, MLFQ scheduler has three queues having dynamic time quantum. This time quantum is taken to show the increasing or decreasing order of time and how MLFQ will behave according to it. Note: Arrival times for all processes are different (see Fig. 5).

Process	Burst time	Arrival time	Response time	Wait time	Turn-around time
P1	15	1	0	0	15
P2	29	2	14	59	88
P3	33	3	28	72	105
P4	45	4	42	113	158
P5	52	5	56	118	170



```
Multilevel process information . . .
p1   borust=15    arrival time=1  p2   borust=29    arrival time=2  p3   borust=33    arrival time=3  p4   borust=45    arrival time=4  p5   borust=52    arrival time=5

Level one RR : Time Quentium =15
    p1   = 0  R.T= 0  T.T=15     p2   = 14 R.T= 14  T.T=29     p3   =18 R.T= 28  T.T=43     p4   = 30  R.T= 42  T.T= 57     p5   = 37  R.T= 56   T.T=71

Level Two RR : Time Quentium =24
    p2   = 0   W.T= 59    T.T=88     p3   = 0   W.T= 72    T.T=105     p4   = 6   W.T= 89  T.T=128     p5  = 13   W.T= 112  T.T=151

Level three FCFS:
    p4   = 0   W.T= 113  T.T=158     p5   = 0   W.T= 118  T.T=170

Wait for each process:
  p1  Wait= 0   Turn= 15  p2  Wait= 59   Turn= 88  p3  Wait= 72  Turn=105  p4  Wait= 113  Turn= 158  p5  Wait= 118  Turn= 170

Gantt Chart :
0     16     31     46     61     76     90     108    132    156    162    175
| p1 | p2 | p3 | p4 | p5 | p2 | p3 | p4 | p5 | p4 | p5 |
```

Figure 5. Graphical representation of new MLFQ with dynamic quantum and different arrival time

In Fig. 4 and Fig. 5 it is clearly shown that the new design of MLFQ will provide fair response time but very good turnaround time, the number of starved processes is decreased to about 40% of them.

5. Conclusion

The paper describes a new design of Multilevel Feedback Queue Scheduling (MLFQ) that provides a comfortable solution for starvation problem in traditional MLFQ by adding ADQ (Auto Detect Quantum) or dynamic quantum instead of static quantum for each new level queue. This means that processes with lower priority will be completed quickly, without migrating to the lower level queue. As we observed in the previous four case studies, new MLFQ is able to increase the system usage by providing a suitable dynamic quantum able to cover burst time for most processes at the same level. This means that the new MLFQ does not allow delaying the execution of any long length process in the queue in order to get high throughput, have good response time, and increase the machine utilization.

Since starvation in the lower level queue is an important issue in MLFQ, the main goal of the new design is to prevent higher priority processes to keep resources occupied, thus generating starved low priority waiting process.

Acknowledgement

The authors wish to thank the anonymous referees for useful comments and remarks that improved the presentation of the main results of this paper.

REFERENCES

1. Li Lo, Liang-Teh Lee, and Huang-Yuan Chang, *A Modified Interactive Oriented Scheduler for GUI-based Embedded Systems*. Tatung University, Taiwan, 2008.
2. Ayan Bhunia, *Enhancing the Performance of Feedback Scheduling*, Int. J. Comput. Appl., vol, 18, no. 4, 2011.
3. A. Silberschatz, P.B. Galvin, and G. Gagne, *Operating system concepts*. 7th ed., USA, 2004.
4. Dharamendra Chouhan, and S.M. Dilip Kumar, and Jerry Anatony Ajay, *A MLFQ Scheduling technique using M/M/c queues for grid computing*, Int. J. Computer Sci. Issues, vol. 10(2), no. 1, 2013.
5. Dalibor Klusacek, and Hana Rudova, *Alea 2 - Job Scheduling Simulator*. SIMUTools-2010, March 15-19, Torremolinos, Malaga, Spain.
6. Riky Subrata, Albert Y. Zomaya, and Bjorn Landfeldt, *Game-Theoretic Approach for Load Balancing in Computational Grids*, IEEE Trans. Parallel and Distributed Systems, vol. 19, no. 1, 2008.
7. Parvar Mohammad, M.E. Parvar, and Safari Saeed, *A Starvation Free IMLFQ Scheduling Algorithm Based on Neural Network*, Int. J. Computat. Intell. Res., vol. 4, no.1, 27–36, 2008.
8. L.A. Torrey, J. Coleman, and B.P. Miller, *A Comparison of the Interactivity in the Linux 2.6 Scheduler and an MLFQ Scheduler*, Software Practice and Experience, vol. 37, no. 4, 347–364, 2008.