International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.3, May-June 2012 pp-876-881 ISSN: 2249-6645

Fair Priority Round Robin with Dynamic Time Quantum: FPRRDQ

Asst. Proff. M. K. Srivastav[1]
Sanjay Pandey[2], Indresh Gahoi[3], Neelesh Kumar Namdev[4]
M. M. M. Engineering College Gorakhpur

Abstract—
Round Robin, considered as the most widely adopted CPU scheduling algorithm, undergoes severe problems directly related to quantum size. If time quantum chosen is too large, the response time of the processes is considered too high. On the other hand, if this quantum is too small, it increases the overhead of the CPU. Round Robin (RR) scheduling algorithm is not suitable for real time operating system because of high context switch rate, larger waiting time, and larger response time. In this paper, we have proposed an improved algorithm which is a variant of RR. Our proposed Fair Priority Round Robin with Dynamic Time Quantum(FPRRDQ) algorithm calculates optimum individual time slice for each task in each round according to the priority and the burst time of that task. Our Experimental results show that FPRRDQ algorithm performs better than Priority Based Simple Round Robin Algorithm (PBSRR) and Shortest execution First Dynamic Round Robin (SEFDRR) by decreasing the number of context switches, average waiting time, and average turnaround time .

***Keywords*—Operating System; Real Time System; Scheduling; Round Robin, Time slice; Priority**

1.INTRODUCTION

Modern Operating Systems are moving towards multitasking environments which mainly depends on the CPU scheduling algorithm since the CPU is the most effective or essential part of the computer. Round Robin is considered the most widely used scheduling algorithm in CPU scheduling [8, 9], also used for flow passing scheduling through a network device [1].

CPU Scheduling is an essential operating system task, which is the process of allocating the CPU to a specific process for a time slice. Scheduling requires careful attention to ensure fairness and avoid process starvation in the CPU. This allocation is carried out by software known as scheduler and dispatcher [8, 9].

There are many different scheduling algorithms which varies in efficiency according to the holding environments, which means what we consider a good scheduling algorithm in some cases which is not so in others, and vice versa. The Criteria for a good scheduling algorithm depends, among others, on the following measures [8]:

- Fairness: all processes get fair share of the CPU according to their priority and burst time,
- Efficiency: keep CPU busy 100% of time,
- Response time: minimize response time,
- Turnaround: minimize the time batch users must wait for output-
- Throughput: maximize number of jobs per hour.

Moreover, we should distinguish between the two schemes of scheduling: preemptive and non preemptive algorithms. Preemptive algorithms are those where the burst time of a process being in execution is preempted when a higher priority process arrives. Non preemptive algorithms are used where the process runs to complete its burst time even a higher priority process arrives during its execution time.

1.1 WELL KNOWN CPU SCHEDULING ALGORITHMS

First-Come-First-Served (FCFS)[8, 9] is the simplest scheduling algorithm, it simply queues processes in the order that they arrive in the ready queue. Processes are dispatched according to their arrival time on the ready queue. Being a non preemptive discipline, once a process has a CPU, it runs to completion. The FCFS scheduling is fair in the formal sense or human sense of fairness but it is unfair in the sense that long jobs make short jobs wait and unimportant jobs make important jobs wait [8, 9].

Shortest Job First (SJF) [8, 9] is the strategy of arranging processes with the least estimated processing time remaining to be next in the queue. It works under the two schemes (preemptive and non-preemptive). It's provably optimal since it minimizes the average turnaround time and the average waiting time. The main problem with this discipline is the necessity of the previous knowledge about the time required for a process to complete. Also, it undergoes a starvation issue especially in a busy system with many small processes being run[8,9].

Round Robin (RR) [8, 9]which is the main concern of this research is one of the oldest, simplest and fairest and most widely used scheduling algorithms, designed especially for

time-sharing systems. It's designed to give a better responsive but the worst turnaround and waiting time due to the fixed time quantum concept. The scheduler assigns a fixed time unit (quantum) per process usually 10-100 milliseconds, and cycles through them. RR is similar to FCFS except that preemption is added to switch between processes [2, 3, and 8].

1.2RELATED WORK

Matarneh [2] founded that an optimal time quantum could be calculated by the median of burst times for the set of processes in ready queue, unless if this median is less than 25ms. In such case, the quantum value must be modified to 25ms to avoid the overhead of context switch time [2]. Other works [7], have also used the median approach, and have obtained good results.

Helmy et al. [3] propose a new weighting technique for Round-Robin CPU scheduling algorithm, as an attempt to combine the low scheduling overhead of round robin algorithms and favor short jobs. Higher process weights means relatively higher time quantum; shorter jobs will be given more time, so that they will be removed earlier from the ready queue [3]. Other works have used mathematical approaches, giving new procedures using mathematical theorems [4].

Mohanty and others also developed other algorithms in order to improve the scheduling algorithms performance [5], [6] and [7]. One of them is constructed as a combination of priority algorithm and RR [5] while the other algorithm is much similar to a combination between SJF and RR [6].

1.2 OUR CONTRIBUTION

In our work, we have scheduled the processes giving importance to both the user priority and shortest burst time priority rather than using single parameter. A new calculated factor based on both the user priority and the burst time priority, decides the individual time quantum for each process . We have compared the performance of our proposed Fair Priority Round Robin with Dynamic Time Quantum(FPRRDQ) algorithm with the Priority Based Static Round Robin(PBSRR) algorithm and Shortest Execution First Dynamic Round Robin(SEFDRR). Experimental results show that our proposed algorithm performs better than PBSRR and SEFDR .

1.4.ORGANIZATION OF THE PAPER

In Section II, the pseudo code and illustration of our proposed FPRRDQ algorithm is presented. Section III shows the results of experimental analysis of FPRRDQ and its comparison with PBSRR and SEFDR. Conclusion and directions for future work is given in Section IV.

2. OUR PROPOSED ALGORITHM

2.1. UNIQUENESS OF OUR APPROACH

Generally with every process two factors are associated. These factors are user priority and burst time. Above factors play an important role to decide in which sequence the processes will be executed. Sorting according to the importance of these factors, user priority comes first, and then the burst time. In FCFS, SJF and Priority algorithms, only one among these two factors are taken into consideration. If we mix up all these factors to calculate the individual time quantum of each process then average waiting time, average turnaround time and number of context switches will be decreased. But FCFS, SJF and Priority scheduling algorithms are non-preemptive in nature and they can't be used in time sharing systems. So to increase the responsiveness of the system, RR algorithm should be used. Generally in RR algorithm, processes are taken from the ready queue in FCFS manner for execution. But in our algorithm, is calculated for each process.

Since , in the previously existed algorithsm like PBSRR and EFDR ,they don't pay more attention regarding the user priority and burst priority (weight of the process given according to the burst time i.e. shorter burst process having more weight) of the process that the process with higher user priority and Weight (burst) should get more time quantum value for the execution of that process. That's why , we can say that the time quatum given to a process is inversely proportional to the user priority(Pr_i) and directly proportional to the weight of the process (i.e. given according to the burst time of the process W_i).

So for the time quantum calculation for processs i is give as:-

$$\mathbf{TQ=(\sum^{N}_{i=1}Bt_i/N)*W_i/pr_i} \quad \text{.....(1)}$$

2.2. PSEUDO CODE FOR FPRRDQ ALGORITHM

Here,

N = No. of processes

W_i = Weight of Pi based on burst time of the process.(Shorter burst processes are assigned more weight).

Input: No of processes(P1, P2, ……., Pn),

Burst time of processes (Bt1, Bt2,….,Btn),

Priority of processes (Pr1, Pr2,……,Prn).

Output: Tav = Average turnaround time,

Wav = Average waiting time,

Ncs = Number of context switches.

Method:

1. According to the ascending order of the burst time value, the processes are sorted in the ready queue.

2. While(ready queue != null)

{

(a) calculate TQ as follows.

TQ = average (remaining burst time of all the processes) *W_i/pr_i

(b) Assign TQ to process P_i

If (i<n) then go to step 2(a)

}

End of while

3. Average waiting time, average turnaround time and context switch are calculated

End

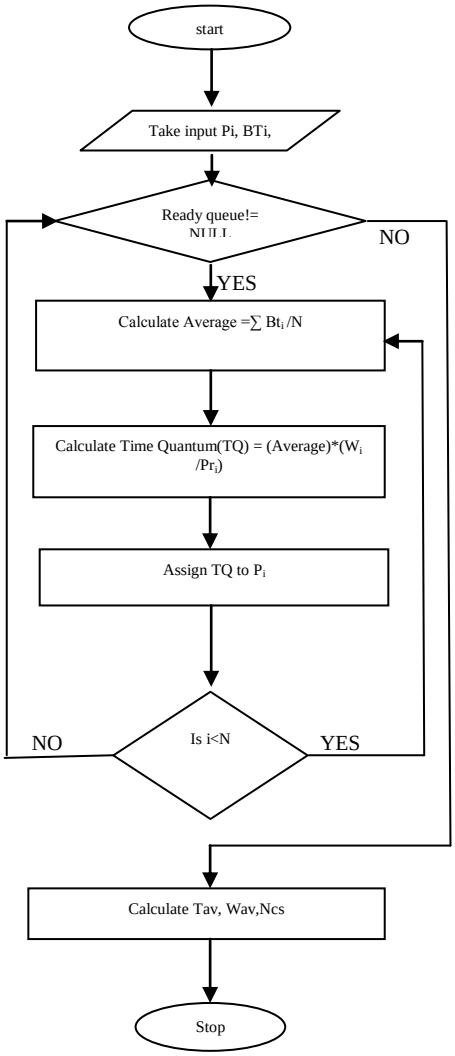

FIG 1: FLOWCHART FOR FPRRDQ ALGORITHM

3. EXPERIMENTAL RESULTS

3.1. ASSUMPTIONS

In a uni-processor environment, all the experiments are performed and all the processes are independent. Time slice is assumed to be not more than the maximum burst time. The attributes like burst time, number of processes and the user-priorities of all the processes are known before submitting the processes to the processor. All processes are CPU bound. No processes are I/O bound.

3.2 EXPERIMENTAL FRAME WORK

Taking various inputs and output parameters we have performed many experiments. The input parameters consist of the number of processes, burst time and user-priorities. The output parameters consist of average waiting

time, average turnaround time and number of context switches.

3.3. PERFORMANCE METRICS

We have used three performance metrics for our experimental analysis. Turn Around Time (TAT): For the better performance of the algorithm, average turnaround time should be less. Waiting Time (WT): For the better performance of the algorithm, average waiting time should be less. Number of Context Switches (CS): For the better performance of the algorithm, the number of context switches should be less.

3.4. EXPERIMENTS PERFORMED

To evaluate the performance of our proposed algorithm, we have taken a set of five processes in four different cases. The algorithm works effectively even if it used with a very large number of processes. In each case, we have compared the experimental results of our proposed algorithm with the priority based RR scheduling algorithm(PBSRR) and Shortest Execution First Dynamic Round Robin(SEFDRR) with dynamic time quantum Q.

CASE 1: We Assume five processes with increasing burst time (P1 = 5, P2 = 12, P3 = 16, P4 = 21, p5= 23) and priority (p1=2, p2=3, p3=1, p4=4, p5=5) as shown in Table below.

Table shows the output using PBSRR , SEFDRR algorithm and our new proposed FPRRDQ algorithm respectively.

CASE 2: We Assume five processes with decreasing burst time (P1 = 63, P2 = 54, P3 = 30, P4 = 12, p5= 5) and priority (p1=3, p2=2, p3=4, p4=1, p5=5) as shown in Table below. The Table shows the output using PBSRR, SEFDRR and our proposed FPRRDQ algorithm respectively.

Process	Burst Time	Priority
1	63	3
2	54	2
3	30	4
4	12	1
5	5	5

Process	Burst Time	Priority
1	5	2
2	12	3
3	16	1
4	21	4
5	23	5

ALGORITHM	TAV	WAV	NCS
PBSRR	109.8	77	14
SEFDRR	106.4	73.6	10
FPRRDQ	81.8	49	6

CASE 3: We Assume five processes with random burst time (P1 = 30, P2 = 8, P3 = 24, P4 = 19, p5= 46) and priority (p1=5, p2=3, p3=2, p4=1, p5=4) as shown in Table-below. The Table shows the output using PBSRR, SEFDRR algorithm and our proposed FPRRDQ algorithm respectively.

ALGORITHM	TAV	WAV	**NCS**
PBSRR	47.2	31.8	17
SEFDRR	42.2	26.8	11
FPRRDQ	38.2	22.8	8

Process	Burst Time	Priority
1	30	5
2	8	3
3	24	2
4	19	1
5	46	4

ALGORITHM	TAV	WAV	NCS
PBSRR	74.4	48	15
EFDRR	73.4	47	10
FPRRDQ	61.4	36	8

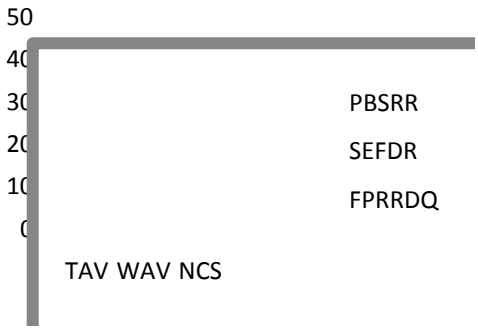

FIG. 2 : COMPARISON AMONG PBSRR , SEFDR AND FPRRDQ(CASE 1)

CASE 4: : We Assume five processes with same burst time (P1 = 10, P2 = 23, P3 = 15, P4 = 34, p5= 15) and distinct priority (p1=2, p2=4, p3=1, p4=3, p5=5) as shown in Table-below. The Table shows the output using PBSRR, SEFDRR algorithm and our proposed FPRRDQ algorithm respectively .

Process	Burst Time	Priority
1	10	2
2	23	4
3	15	1
4	34	3
5	15	5

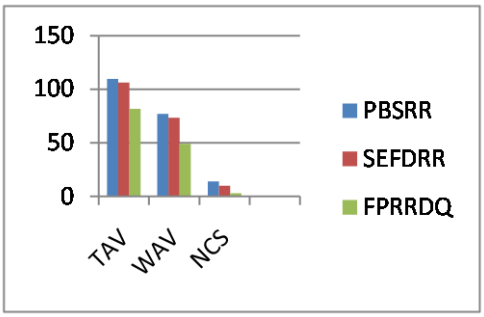

FIG. 3 : COMPARISON AMONG PBSRR , SEFDR AND FPRRDQ (CASE 2)

ALGORITHM	TAV	WAV	NCS
PBSRR	61.6	41.8	13
SEFDRR	56.8	36.8	10
FPRRDQ	52.4	33	9

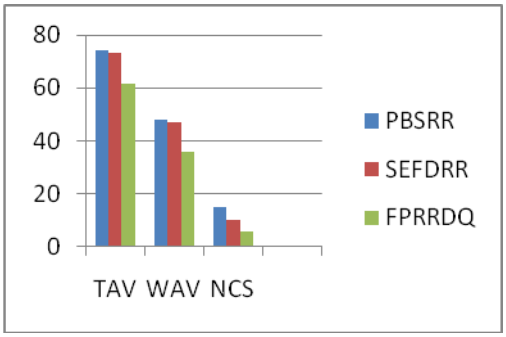

FIG. 4 : COMPARISON AMONG PBSRR , SEFDR and FPRRDQ(CASE 3)

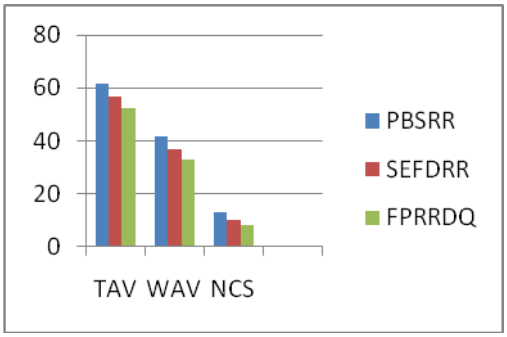

FIG. 5 : COMPARISON AMONG PBSRR , SEFDR AND FPRRDQ (CASE 4)

Where

TAV: Average Turn around time

WAV: Average Wating time

NCS: No. Of context switch

4. CONCLUSION

From the experimental results, we found that FPRRDQ performs better than the PBSRR and SEFDR in terms of decreasing the number of context switches, average waiting time and average turnaround time. The algorithm also gives a fair value of time quantum to each process according to the priority and burst time of that process.

5. REFERENCES

[1] Weiming Tong, Jing Zhao, "Quantum Varying Deficit Round Robin Scheduling Over Priority Queues", International Conference on Computational Intelligence and Security. pp. 252- 256, China, 2007.

[2] Rami J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes", American Journal of Applied Sciences, Vol 6, No. 10, 2009.

[3] Tarek Helmy,Abdelkader Dekdouk, "Burst Round Robin as a Proportional-Share Scheduling Algorithm", In Proceedings of The fourth IEEE-GCC Conference on Towards Techno-Industrial Innovations, pp. 424-428, Bahrain, 2007.

[4] Samih M. Mostafa, S. Z. Rida, Safwat H. Hamad, "Finding Time Quantum Of Round Robin Cpu Scheduling Algorithm In General Computing Systems Using Integer Programming", International Journal of Research and Reviews in Applied Sciences (IJRRAS), Vol 5, Issue 1, 2010.

[5] Rakesh Mohanty, H. S. Beheram Khusbu Patwarim Monisha Dash, M. Lakshmi Prasanna , "Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No.2, February 2011.

[6] Rakesh Mohanty, H. S. Behera, Khusbu Patwari, Monisha Dash, "Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm", In Proceedings of International Symposium on Computer Engineering & Technology (ISCET), Vol 17, 2010.

[7] Rakesh Mohanty, H. S. Behera, Debashree Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis", International Journal of Computer Applications (0975 – 8887), Volume 5– No.5, August 2010.

[8] Silberschatz ,Galvin and Gagne, Operating systems concepts, 8th edition, Wiley, 2009

[9] Lingyun Yang, Jennifer M. Schopf and Ian Foster, "Conservative Scheduling: Using predictive variance to improve scheduling decisions in Dynamic Environments", SuperComputing 2003, November 15-21, Phoenix, AZ, USA..

[10] A.S. Tanenbaun, Modern Operating Systems.3rd Edn, Prentice Hall, ISBN:13: 9780136006633, 2008.

[11] C. Yaashuwanth and R. Ramesh, : A New Scheduling Algorithm for Real Time System, International Journal of Computer and Electrical Engineering (IJCEE), Vol. 2, No. 6, pp 1104-1106, December, 2010.