International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 1 217 - 221

Modified RR Algorithm with Dynamic Time Quantum for Externally Prioritized Tasks

Lipika Datta
Computer Science and Engineering Department,
College of Engineering and Management Kolaghat
KTPP Township, West Bengal, India
lipika.datta@cemk.ac.in

***Abstract*—** The objective of this paper is to modify Round Robin scheduling for soft real time systems. It introduces a variation of round robin algorithm which can schedule tasks considering their priorities in a round robin fashion. Simple Round Robin scheduling algorithm and Priority scheduling algorithm, both have their own drawbacks. The proposed algorithm removes the drawbacks of Round Robin scheduling and Priority scheduling. The proposed scheduling algorithm calculates different time slices for individual processes considering their priorities. Experimental analysis reveals that the proposed algorithm produces better average turnaround time, average waiting time and less number of context switches than some existing algorithms useful for interactive systems.

Keywords- *Operating System; Scheduling; Round Robin Algorithm; Context switch; Turnaround time; Average Waiting time*

*****

I. INTRODUCTION

A real-time operating system is intended to serve a real-time application process which is expected to respond within some small upper bound on response time and any late result leads to system's performance degradation. Priority scheduling may cause starvation and lower priority tasks may not satisfy their deadlines. Simple round robin scheduling is also not efficient for soft real time systems because of higher context switching rate, high waiting time. The proposed algorithm has the advantage that processes with less CPU burst will have a better waiting time and turnaround time than compared to simple round robin architecture and covers the drawbacks of round robin architecture. It schedules processes in round robin fashion while the time slice of processes' execution in CPU depends on their priorities. The drawback of priority scheduling i.e. starvation can also be overcome as it is ensured that after certain amount of time the process will execute in CPU.

Ideally we want to maximize the CPU utilization and throughput and minimize Average waiting time, average turnaround time and response time. To guarantee that all users get good service, we try to minimize the maximum turnaround time. But this is not always possible. Instead, we choose a scheduling algorithm based on its ability to satisfy a policy.

- Minimize average response time - provide output to the user as quickly as possible and process their input as soon as it is received.
- Minimize average turnaround time – real time application processes satisfy their deadlines. So turnaround time must be minimized.
- Maximize throughput by minimizing overhead (OS overhead, context switching) and by efficient use of system resources (CPU, I/O devices)
- Minimize waiting time by giving each process the same amount of time on the processor. This might actually increase average response time.

Soft real time systems have no hard deadlines for tasks but missing of deadlines in soft real time systems will degrade the system performance. Minimum turnaround time implies higher quality of service. Number of context switching is also a dominant performance measuring metric as it is a pure overhead of a process.

II. RELATED WORK

RR algorithm performs optimally in timeshared systems, but it is not suitable for soft real time systems. Because it gives more number of context switches, larger waiting time and larger response time. Some researchers have already introduced some variations of RR scheduling algorithm. But these algorithms have some limitations. In [1] authors have proposed an algorithm in which according to the given priority the CPU is allocated to the processes only once in RR fashion for a given time quantum. Then processes are arranged in increasing order of their remaining CPU burst time in the ready queue. New priority is assigned to each process following the rule that lesser the remaining burst time higher the priority. Then processes are allocated CPU according to non-preemptive priority scheduling algorithm. If this algorithm is used after first response from the system user may have to wait long for next response. Fairness criteria is not held. In [2] different time slices are calculated for different processes based on three aspects: user defined priority, average CPU burst, context switch avoidance time. An assumption is made on average CPU burst. In [3] also different time slices are calculated for different processes based on priority, shortest CPU burst time and context switch avoidance time. In different rounds the time quantum for a process goes on changing depending on the parameters i.e. the authors have introduced dynamic time quantum. The authors only have taken into consideration the processes with highest priority. Rest of the processes' priorities is ignored. In [4] the authors have introduced a concept called intelligent time slicing which depends on priority and context switch. The time slice is static. This algorithm is modified to get different time slice values in different rounds for different processes in [6]. It calculates the initial time slice for each process as the previous algorithm [6] and in each round the time slices are modified. In [6] the authors have made the priority and time slice for a process dynamic by calculating the weighted mean values of time quantums and priorities of the processes and considering the burst time of the processes. The author have considered only the response time in [7] but ignored turnaround time and waiting time, which are also important in soft real time applications.

A. *My contribution*

In my work, I have proposed an improved algorithm which calculates different time slices for individual processes considering their priorities. Experimental result shows that my algorithm is better than [2]in terms of average turnaround time, average waiting time and number of context switches. The result reveals that the proposed algorithm is better than [3] in terms of number of context switches.

B. *Organization of paper*

Section III presents the illustration of my proposed algorithm. In section IV, Experimental results and its comparison with existing algorithms is presented. Section V contains the conclusion.

III. PROPOSED ALGORITHM

The proposed algorithm eliminates the drawbacks of implementing a simple round robin architecture in real time system by introducing a concept of assigning different time slices to individual processes depending on their are priorities. The priority of a process is assigned by user externally. In the proposed architecture when a new process arrives in the system it is queued at a small processor. This small dedicated processor is used to calculate the time slices of each process, arranges the processes in ascending order of their burst times and then creates the ready queue for the main processor. This small dedicated processor is used to reduce the burden of the main processor. The processes then execute in the main processor according to round robin scheduling algorithm with their individual time slices. Whenever a new process arrives in the system ready queue, its time slice is calculated and enqueued to the main processor's ready queue. Whenever a process completes its execution it is removed from both the system ready queue and the main processor ready queue. The process continues until the main processor ready queue becomes empty. I am assuming that lesser number implies higher priority.

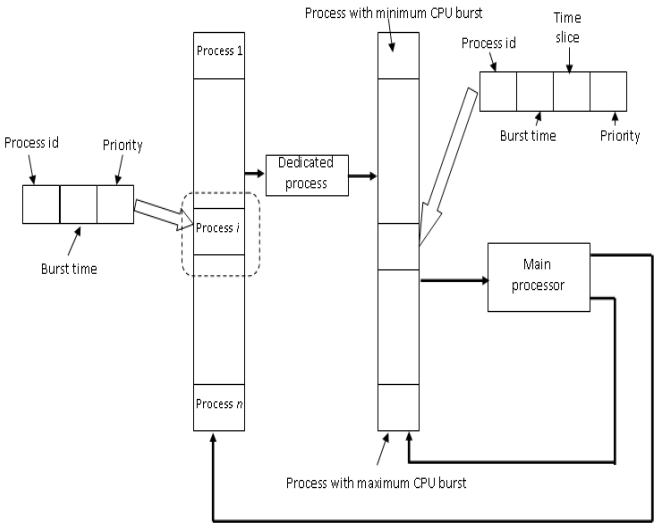

Figure1 .Proposed architecture

Begin

1. Input: Process id, Burst time, Process priority.
   Output: Average turnaround time, Average waiting time, Number of Context Switches
2. Initialize i=1
3. Sort the processes in the system ready queue (SRQ) according to their burst time.
4. while (SRQ!=NULL)
   {
   Find average burst time (T) of the processes.
   Let SB as the burst time of the process at the front of the queue (i.e. smallest burst time).
   Let n be the maximum priority number present in the SRQ.
   Let x be the priority of i^{th} process.
   Calculate Time Slice (TS) for process i as SB + (T – SB)*(n – x)/n.
   Increment i by 1.
   }
5. Enqueue processes in the same sequence as SRQ with calculated time slices in Processor ready queue (PRQ).
6. Schedule the processes in the main processor according to RR scheduling algorithm with the calculated time slices.
7. If new process arrives repeat step 4 and enqueue the process at rear of PRQ.
8. While a process completes execution remove it from SRQ and PRQ.
9. Calculate Average turnaround time, Average waiting time, number of context switch.

End.

IV. EXPERIMENTAL RESULTS

A. *Assumptions:*

Experiments are performed in single processor environment and on independent processes. All the parameters like number of processes, and burst time, priority of all the processes are known before submitting the processes to the processor. All processes are CPU bound and none I/O bound. Context switching overhead and time taken by the dedicated processor for calculating the time slices are zero while calculating Average Turnaround time and average waiting time.

B. *Data Set:*

To compare the performance of the algorithm with the algorithms described in [2] (SARTT) and [3] (PBDRR) 2 different data sets are taken. For all the cases arrival time is considered as 0. Again comparison is done between algorithms introduced in [4] (MRR) and [5] (TSPBRR) with a different set of data. Another set of data is applied to Shortest RR, Intelligent time slice and the proposed algorithm.

1) Same data set applied to SARTT, PBDRR and Proposed Algorithm:

a) Case a: Processes with increasing Burst Time:

TABLE 1. Inputs for case 1a

Process id	Priority	Burst time
P1	2	5
P2	3	12
P3	1	16
P4	4	21
P5	5	23

 ISSN: 2321-8169
217 - 221

TABLE 2. Calculation of time slice for proposed algorithm for case 1a

Process ID	Priority	Burst time	Time slice
P1	2	5	12
P2	3	12	10
P3	1	16	14
P4	4	21	8
P5	5	23	5

TABLE 3. Comparison between algorithms for case 1a

Algorithm	Average Turnaround Time	Average Waiting Time	No. of Context Switch
SARTT	51.2	35.8	19
PBDRR	46.4	31	17
Proposed Algorithm	47.2	31.8	10

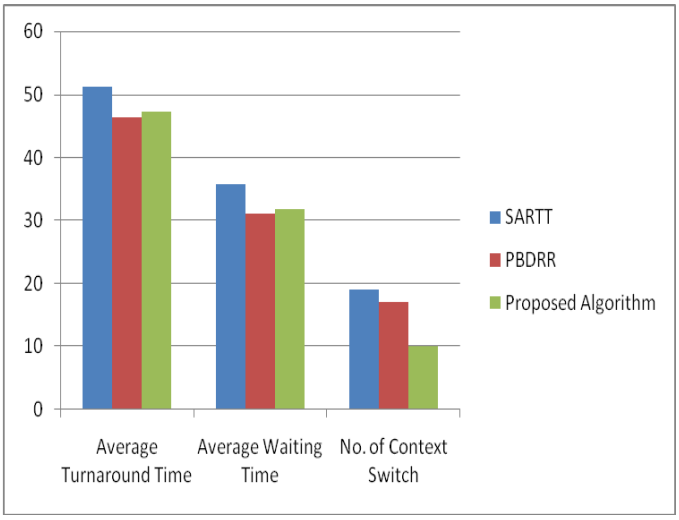

Figure2 . Analysis of performance among algorithms (case 1a)

b) Case b: Processes with decreasing burst time:

TABLE 4. Inputs for case 1b

Process id	Priority	Burst time
P1	2	31
P2	1	23
P3	4	16
P4	5	9
P5	3	1

TABLE 5. Calculation of time slice for proposed algorithm for case 1b

Process id	Priority	Burst tme	Time Slice
P1	2	31	10
P2	1	23	13
P3	4	16	4
P4	5	9	1
P5	3	1	7

TABLE 6. Comparison between algorithms for case 1b

Algorithm	Average Turnaround Time	Average Waiting Time	No. of Context Switch
SARTT	54	38	18
PBDRR	50.4	34.4	12
Proposed Algorithm	54.8	38.8	15

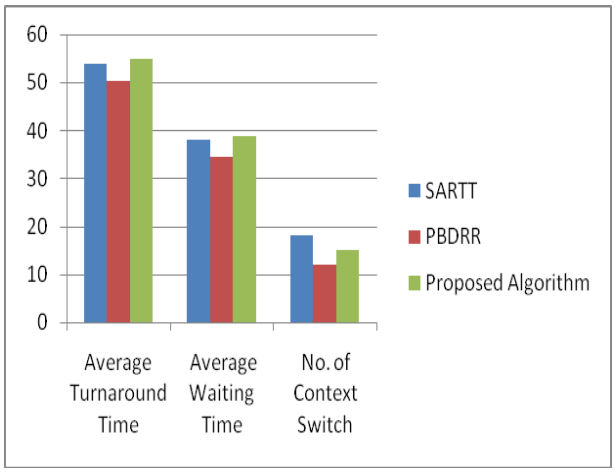

Figure3 .Analysis of performance among algorithms (case 1b)

c) Case c:Processes with Random Burst Time:

TABLE 7. Inputs for case 1c

Process id	Priority	Burst time
P1	3	11
P2	1	53
P3	2	8
P4	4	41
P5	5	20

TABLE 8. Calculation of time quantum for proposed algorithm for case 1c

Process id	Priority	Burst time	Time Slice
P1	3	11	16
P2	1	53	24
P3	2	8	20
P4	4	41	12
P5	5	20	8

TABLE 9. Comparison between algorithms for case 1c

Algorithm	Average Turnaround Time	Average Waiting Time	No. of Context Switch
SARTT	80.8	54.2	29
PBDRR	76	49.4	18
Proposed Algorithm	79.8	53.2	11

ISSN: 2321-8169
217 - 221

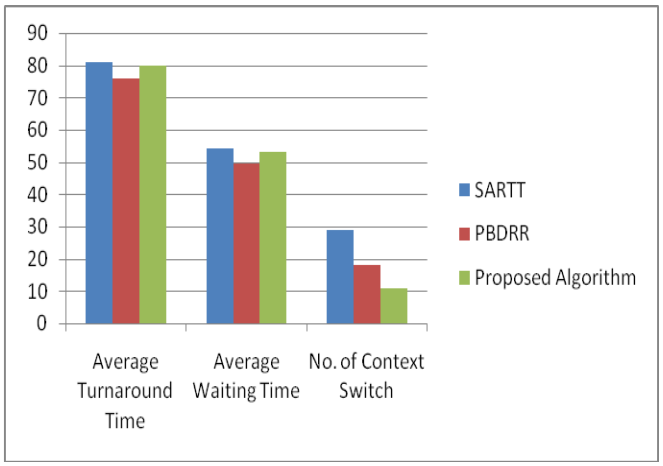

Figure4 .Analysis of performance among algorithms (case 1c)

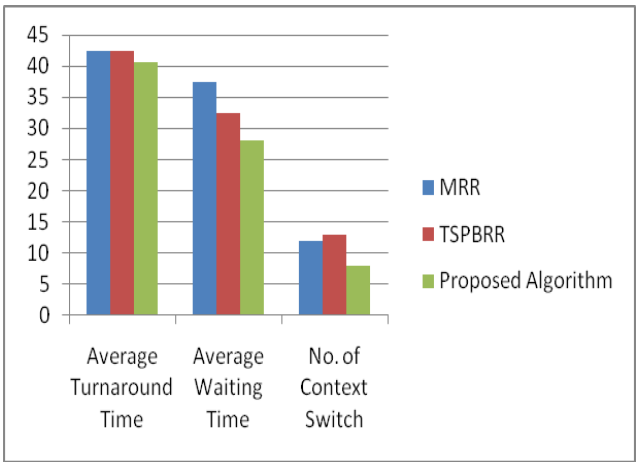

Figure5 .Analysis of performance among algorithms (data set 2)

2) Data set applied to MRR, TSPBRR and Proposed Algorithm:

TABLE 10. Input data set

Process id	Priority	Burst time
P1	2	25
P2	3	5
P3	1	15
P4	4	10
P5	5	8

TABLE 11. Calculation of time quantum for proposed algorithm for data set 2

Process id	Priority	Burst time	Time Slice
P1	2	25	10
P2	3	5	9
P3	1	15	12
P4	4	10	7
P5	5	8	5

TABLE 12. Comparison between algorithms for data set 2

Algorithm	Average Turnaround Time	Average Waiting Time	No. of Context Switch
MRR	42.5	37.4	12
TSPBRR	42.4	32.4	13
Proposed Algorithm	40.6	28	8

3) Same data set applied to Shortest RR and Intelligent Time Slice and Proposed Algorithm:

TABLE 13. Input data Set

Process id	Priority	Burst time
P1	2	25
P2	3	5
P3	1	15
P4	2	8
P5	1	10

TABLE 14. Calculation of time quantum for proposed algorithm for data set 3

Process id	Priority	Burst time	Time Slice
P1	2	25	8
P2	3	5	5
P3	1	15	11
P4	2	8	8
P5	1	10	11

TABLE 15. Comparison between algorithms for data set 3

Algorithm	Average Turnaround Time	Average Waiting Time	No. of Context Switch
Shortest RR	22	36	14
Intelligent TS	25	37	9
Proposed Algorithm	17.4	30	6

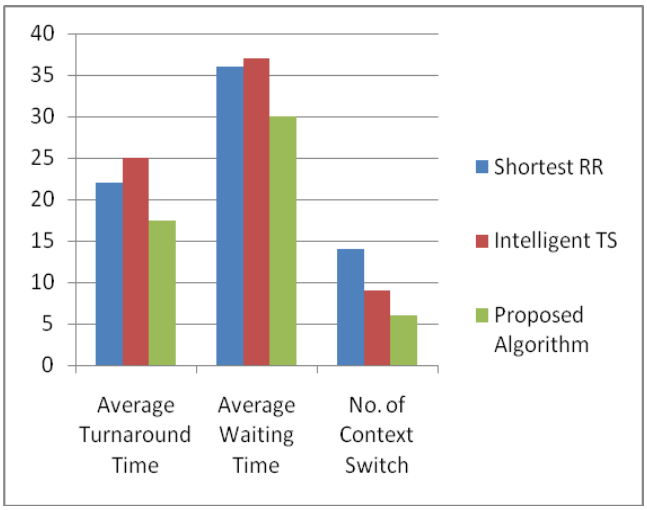

Figure6 .Analysis of performance among algorithms (data set 3)

Fig 2, Fig 3, Fig 4 graphically represents the performances of the SARTT, PBDRR and proposed algorithms in terms of Average Turnaround Time, Average Waiting Time and number of context switching. Fig 5 graphically represents the performances of the MRR, TSPBRR and proposed algorithms and Fig 6 graphically represents the performances of the Shortest RR, Intelligent time slice and the proposed algorithm in terms of the same parameters.

V. CONCLUSION

From the above comparisons it can observed that the proposed algorithm is better than some existing algorithms for real time task scheduling in terms of Average Turnaround time, Average Waiting time and number of context switches in most of the cases. If the CPU burst times of the processes vary very widely the algorithm doesn't produce good result. In the other cases the quality of service can improved and overhead can be reduced. Thus memory space which is an important constraint for embedded system applications can be saved. Deadlines of tasks can be considered in future as a new parameter while calculating the time quantum in each round.

REFERENCES

[1] Ishwari Singh Rajput, Deepa Gupta "A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems", International Journal of Innovations in Engineering and Technology (IJIET) Vol. 1 Issue 3 Oct 2012

[2] C.Yaashuwanth, Dr.R.Ramesh "A New Scheduling Algorithms for Real Time Tasks", International Journal of Computer Science and Information Security, Vol. 6, No.2, 2009

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

[4] Yaashuwanth .C, Dr.R. Ramesh, " Design of Real Time scheduler simulator and Development of Modified Round Robin architecture" IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.3, March 2010

[5] Subasish Mohapatra, Subhadarshini Mohanty, K.Smruti Rekha," Analysis of Different Variants in Round Robin Algorithms for Load Balancing in Cloud Computing", International Journal of Computer Applications (0975 – 8887) Volume 69– No.22, May 2013

[6] H.S.Behera, Sabyasachi Sahu, Sourav Kumar Bhoi, "Weighted Mean Priority Based Scheduling for Interactive Systems" Journal of Global Research in Computer Science, Volume 2, No. 5, May 2011

[7] Lipika Datta, "Modified RR Algorithm with Dynamic Time Quantum for Externally Prioritized Tasks", International Journal of Engineering Research and Technology, Vol 3 – Issue 3 (March 2014)