Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems

Article *in* International Journal of Advanced Computer Science and Applications · March 2011

DOI: 10.14569/IJACSA.2011.020209 · Source: DOAJ

Rakesh Mohanty
Veer Surendra Sai University of Technology
23 PUBLICATIONS **144** CITATIONS
SEE PROFILE

Dr. H. S. Behera
Veer Surendra Sai University of Technology(VSSUT), Govt. of Odisha, India
125 PUBLICATIONS **885** CITATIONS
SEE PROFILE

Monisha Dash
San Jose State University
2 PUBLICATIONS **48** CITATIONS
SEE PROFILE

Priority Based Dynamic Round Robin with Intelligent Time Slice and Highest Response Ratio Next Algorithm for Soft Real Time System

Zena Hussain Khalil [1], *Ameer Basim Abdulameer Alaasam* [2, 3]

[1] *MSc, Department of statistic and informatics, College of Computer and mathematics sciences, University of Al-Qadisiyah, Iraq*

[2] *General Directorate for Education in Al-Qadisiyah, Ministry of Education, Iraq*

[3] *MSc Computer Sciences student, Georgian Aviation University, Tbilisi, Georgia*

Abstract:-*In this paper, we have improved the work of Round Robin (RR) algorithm to make it suitable for Soft Real-Time system and that because Round Robin (RR) algorithm suitable for timeshared systems but it is not suitable for Soft Real-Time system due to a large waiting time , large number of context switch and large response time, for this we have improved it by proposing a new algorithm to improve the concept of Dynamic quantum time in [3], by take the Highest Response Ratio Next (HRRN) for each process in each round to select the next process from ready queue , because one of the most important challenges in scheduling is how choose the next process for execute, where we found that the proposed algorithm gives better results than in [3] in terms of reducing the number of context switches, average waiting time and average turnaround time.*

Keywords:-*Real time system; Operating System; Scheduling; Round Robin Algorithm; Highest response ratio next; Context switch; Waiting time; Turnaround time .*

I. INTRODUCTION

Real-time systems (RTS) designed to accomplish tasks within a specific time and this includes all tasks by type of system and several other factors, such as the importance and execution time given to task and other factors. There are two main types of the real-time systems, the first is Hard Real-Time System and the second is Firm or Soft Real-Time System [2]. One of the most important reasons for this classification is a handle method for each Real-time systems types with the execution time required, the Hard Real-Time System is very strict in dealing with the time is determine the fixed deadlines for the tasks and in case of exceed deadline will the system failure arise forcibly, while the Firm or Soft Real-Time System, it is also deals with the deadlines and exceeded it is undesirable but in normally may not arise the case of failure, but it is considered as degraded performance in this case, because of this it is considered as soft in dealing with cases that exceeded the deadline. Some of the applications of real time systems it is as follow: money withdrawal from ATM, weather forecast, seismic detection, video conferencing, audio conferencing, flight reservation and etc. For this, the Round Robin algorithm (RR) in its simple case cannot be applied in real-time systems because it gives a large waiting time and a long response time, but can take advantage of it because Round robin algorithm help to save the processor busy all the time, so it is need to developed to become compatible with (RTS) .The (RTS) must be multi-tasking and pre-emptable ,the scheduler should be able to pre-empt any task in the system and give the resource to the task that needs it most. An (RTS) should also handle multiple levels of interrupts to handle multiple priority levels [1]. Prof. Rakesh Mohanty and et. al [3] focused their work on the (time slice) imposed by the Round robin algorithm (RR) for all process, They created a mechanism to calculate a new time slice for individual Process which changed during implementation. We've improved this algorithm by propose a mechanism for process selection from the ready queue by taking the concept of Highest Response Ratio Next (HRRN) to finding a better way to determine what the next process will choose to execute, and we get better results than in [3].

A. Round Robin algorithm (RR)

A clock interrupt is generated at periodic intervals. When the interrupt occurs, the currently running process is placed in the ready queue, and selected next ready job to execute. This technique is also known as time slicing**,** because each process is given a slice of time before being pre-empted [5]. The most important challenges facing the (RR) is the amount of the (Quantum - time slice that is given to process before being pre-empted) where if it was largest of the biggest process, (RR) faces the risk of switching to (FCFS) and if it very small, then a smaller process will need two rounds at least to complete its execution. As well as the throughput rate may degrade because of the large waiting time and long turnaround time and large number of context switch. To make (RR) suitable to work with (RTS), it is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time. In most cases, we optimize the average measure [8] in addition to minimization of context switch.

B. Highest Response Ratio Next (HRRN)

Scheduling algorithm that is a non-preemptive discipline in which the priority of each job is dependent on its estimated service time or burst time and also the amount of time it has spent waiting [6]. (HRRN) based the following calculation:

R = (W + S) / S

Where:
R: response ratio.
W: waiting time.
S: service time.

C. Related work

Many researchers have considered RR algorithm as headline in their work; some of their published works are the following, Rakesh Mohanty and et.al have proposed a scheduling algorithm for soft real time systems where the processes are scheduled using RR with inelegant time slice ITS as time quantum, and then calculate dynamic time quantum depend on ITS for all the processes [3]. H.S. Behera and et.al proposed algorithm to find the (Original Time Slice 'OTS') depending on the range, priority and total number of processes, where the range is defined by finding the total of the largest process burst time plus smaller process burst time, and then calculating the (ITS) depending on the (OTS) but the select of next process depend on shortest remaining burst time [4]. H.S. Behera and et.al proposed Round Robin with Highest Response Ratio Next algorithm with dynamic time quantum which computed by taking the mean of the remaining burst time [6]. Ishwari Singh Rajput and Deepa Gupta focused on the priority for each process with the adoption of original quantum the work adopted of change the priority for each process in each round, where give the process that have shortest remaining CPU burst time a highest priority and every process is arranged in each round passed this concept [7].

II. Our proposed algorithm

Because the (RR) depend on share the processor time between the ready process after each interrupt generated, then there are two important challenges, first is the amount of time quantum given for the process and second is any one of the ready process will be choose to next execute. The PBDRR [3] proposed to solve the first problem a dynamic time quantum for each process in each round depending on the priority and burst time quantum. In this paper we propose to solve the second problem by using HRRN, where the response ratio offering greater efficiency than the rest of priority algorithms which may cause a state of neglect for some process such as (shortest remaining time and shortest burst time), while HRRN is suitable for all process either it is small or large burst time because it based on waiting time together with service time making it more effective and fairness.

A. Methodology

Our algorithm improved the work in [3] that is not discuss how to chose the next process, while we improved this work by combines the importance of burst time and waiting time for each process to find any process will be next, where we have created a algorithm that synchronizes the concept of HRRN with Round Robin that have dynamic time quantum for each process. This is done by calculating the response ratio for each process with calculation of dynamic time quantum for each process in the ready queue and then processor will begin execute ready process for current round by choice the next process in order from highest to lowest response ratio in each round, this done to be executing all process in ready queue according to each time quantum.

B. Algorithm

In our algorithm at first step the Intelligent Time Slice (ITS) is calculated to give a different time quantum to each process based on the difference between CPU burst time, context switch avoidance time and priority.

Let the original time slice (OTS) is the time slice to be given to any process in normal case. Priority component (PC) is assigned 0 or 1 based on the priority assigned by the user, where process with highest priority is assigned 1 and rest is assigned 0.Shortness Component (SC) is calculated by difference between the burst time of current process and burst time of its previous process, if the difference is less than 0, then SC is assigned 1, else is assigned 0. For calculation of Context Switch Component (CSC) subtract the sum of PC, SC and OTS from the burst time, If CSC less than OTS, its value will be considered, else equal to 0 .Finally to get intelligent time slice(ITS) for individual process, adding all the values like SC, PC, OTC and CSC. Each round 'Round' will be begin with recorded waiting time 'WT', remaining burst time 'RM' and calculate response ratio 'RS' for each process in ready queue with calculate dynamic time quantum 'TQ' for each process in current round to select the ready process that have highest response ratio to execute one by one till finish all process in ready queue for this round ,and then the waiting time and remaining burst time will be updated for each process and Round increments by 1 and go to next round while ready queue not equal to NULL. Fig.1 in section C of II shows the pseudo code of our proposed algorithms.

C. Pseudo Code for Algorithm

```
1. Get Intelligent Time Slice (ITS) for all the
   processes in the ready queue.
2. Round = 1
3. While (ready queue != NULL)
      {
       3. A. For i=1 to n do
          {
```

```
        RS_i =(WT_i + RMi) / RM_i, if Round > 1
        RS_i =1, otherwise
        if (Round =1)
        {
        TQi = ½ ITS, if SC= 0
        TQi = ITS, otherwise
        }
        Else
        {
        TQi = TQ i + ½ TQi, if SC=0
        TQi= 2 * TQ i, otherwise
        }
        If (RM i -TQ i ) <=2
          TQ i = remaining burst time
      } End of For
    3. B. Start execute all processes in order from
          highest RS to last lowest RS
    3. C. The WT and RM for each processes
          will be updated
    3. D. Round=Round+1
  } End of while
4. Calculate average turnaround time, Average
   waiting time and number of context switches
```

Figure 1: Pseudo Code of Our Proposed Algorithms

III. Experiment And Simulation Result

A. Simulation

We have built a simulator for our propose algorithm to perform our experiments algorithms, A simulators are build in C++, where No processes are I/O bound, all processes are CPU bound. The input parameters of simulator consist of burst time, original time quantum, priority and the number of processes. The output parameters consist of three performance metrics which are average waiting time ' Avg WT ', average turnaround time ' Avg TAT ' and number of context switches 'CS', which can be define as: Average Waiting time is a total time for all processes has been waiting in the ready queue divided by number of process .Average Turnaround is a total time between submission of a processes and its completion divided by number of process. Context switches is the number of times that a CPU switches from one process to another. It should be noted that, for the better performance of the algorithm, average turnaround time, average waiting time and number of Context Switches should be less in most case.

B. Case Study

To evaluate the performance of our proposed algorithm and PBDRR algorithm, we have taken a set of five processes in three different cases. Here for simplicity, we have taken 5 processes. 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 PBDRR algorithm presented in [3].

CASE 1: We assume five processes arriving at time = 0, with random burst time (P1 = 11, P2 = 53, P3 = 8, P4 = 41, p5= 20) and priority (p1=3, p2=1, p3=2, p4=4, p5=5). The TABLE-1 show the ITS, Table-2 and Table-3 show the output using algorithm PBDRR and our proposed algorithm respectively Table-4 shows the comparison between the two algorithms.

Table 1: (ITS – Case 1)

Process id	Burst time	Priority	OTS	PC	SC	CSC	ITS
P1	11	3	4	0	0	0	4
P2	53	1	4	1	0	0	5
P3	8	2	4	0	1	3	8
P4	41	4	4	0	0	0	4
P5	20	5	4	0	1	0	5

Table 2: (PBDRR – Case 1)

Round	Pi - TQi				
1	P1-2	P2-3	P3-8	P4-2	P5-5
2	P1-3	P2-5	P4-3	P5-10	
3	P1-6	P2-8	P4-5	P5-5	
4	P2-12	P4-8			
5	P2-18	P4-12			
6	P2-7	P4-11			

Table 3: (Our Proposed Algorithm- Case 1)

Round	Pi - TQi				
1	P1-2	P2-3	P3-8	P4-2	P5-5
2	P5-10	P1-3	P2-5	P4-3	
3	P1-6	P5-5	P4-5	P2-8	
4	P4-8	P2-12			
5	P4-12	P2-18			
6	P2-7	P4-11			

Table 4: (Comparison between PBDRR and Our Proposed Algorithm - Case 1)

Algorithm	Avg TAT	Avg WT	CS
PBDRR	76	49.4	19
Our proposed algorithm	73.4	46.7	19

For CASE-1 Figure-2 and Figure-3 show Gantt chart for algorithms PBDRR and our proposed algorithm respectively Figure-4 shows Comparison of Performance of Algorithms - PBDRR with our proposed algorithm.

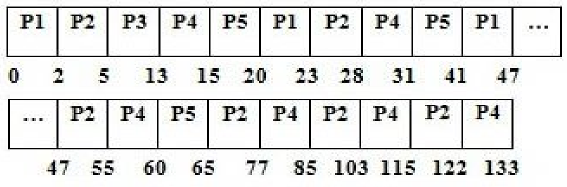

Figure 2: Gantt chart for BPDRR (CASE 1)

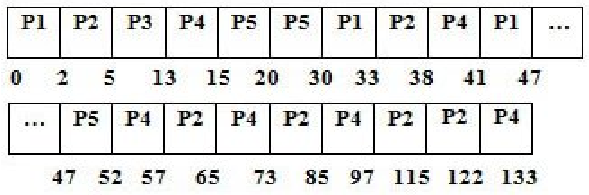

Figure 3: Gantt chart for Our Proposed Algorithm (CASE 1)

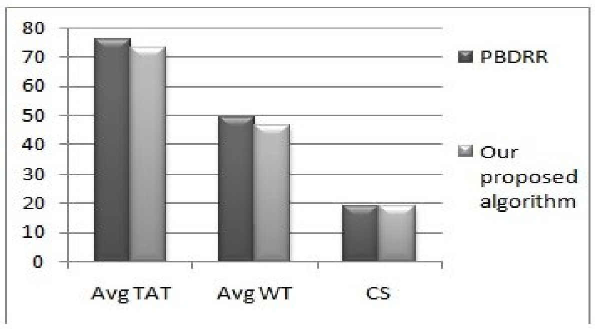

Figure 4: Comparison of Performance of Algorithms - PBDRR with Our Proposed Algorithm (Case 1)

CASE 2: We Assume five processes arriving at time = 0, with decreasing burst time (P1 = 31, P2 = 23, P3 = 16, P4 = 9, p5= 1) and priority (p1=2, p2=1, p3=4, p4=5, p5=3) The TABLE-5 show the ITS, Table-6 and Table-7 show the output using algorithm PBDRR and our proposed algorithm respectively. Table-8 shows the comparison between the two algorithms. Figure-5 and Figure-6 show Gantt chart for algorithms PBDRR and our proposed algorithm respectively. Figure-7 shows Comparison of Performance of Algorithms - PBDRR with our proposed algorithm.

Table 5: (ITS – Case 2)

Process id	Burst time	Priority	OTS	PC	SC	CSC	ITS
P1	31	2	4	0	0	0	4
P2	23	1	4	1	1	0	6
P3	16	4	4	0	1	0	5
P4	9	5	4	0	1	0	5
P5	1	3	4	0	1	0	1

Table 6: (PBDRR – Case 2)

Round	Pi - TQi				
1	P1-2	P2-6	P3-5	P4-5	P5-1
2	P1-3	P2-12	P3-11	P4-4	
3	P1-5	P2-5			
4	P1-21				

Table 7: (Our Proposed Algorithm- Case 2)

Round	Pi - TQi				
1	P1-2	P2-6	P3-5	P4-5	P5-1
2	P4-4	P1-3	P2-12	P3-11	
3	P2-5	P1-26			

Table 8: (Comparison Between PBDRR and Our Proposed Algorithm- Case 2)

Algorithm	Avg TAT	Avg WT	CS
PBDRR	50.4	34.4	12
Our proposed algorithm	45	29	11

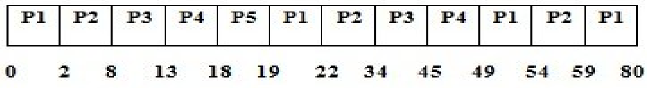

Figure 5: Gantt Chart for BPDRR (CASE 2)

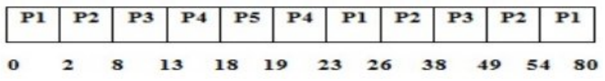

Figure 6: Gantt Chart for Our Proposed Algorithm (CASE 2)

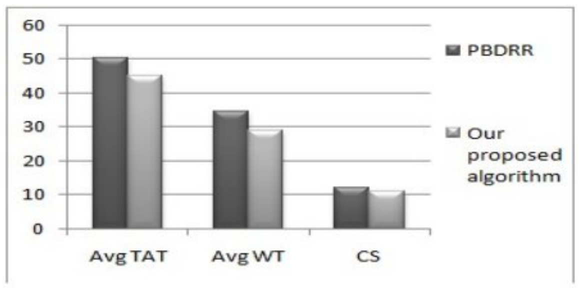

Figure 7: Comparison of Performance of Algorithms - PBDRR with Our Proposed Algorithm (Case-2)

CASE 3: We assume five processes arriving at time = 0, 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). The Table-9 show the ITS, Table-10 and Table-11 show the output using algorithm PBDRR and our proposed algorithm respectively.Table-12 shows the comparison between the two algorithms. Figure-8 and Figure-9 show Gantt chart for algorithms PBDRR and our proposed algorithm respectively. Figure-10 shows Comparison of Performance of Algorithms - PBDRR with our proposed algorithm.

Table 9: (ITS – Case 3)

Process id	Burst time	Priority	OTS	PC	SC	CSC	ITS
P1	5	2	4	0	0	1	5
P2	12	3	4	0	0	0	4
P3	16	1	4	1	0	0	5
P4	21	4	4	0	0	0	4
P5	23	5	4	0	0	0	4

Table 10: (PBDRR – Case 3)

Round	Pi - TQi				
1	P1-5	P2-2	P3-3	P4-2	P5-2
2	P2-3	P3-5	P4-3	P5-3	
3	P2-7	P3-8	P4-5	P5-5	
4	P4-8	P5-8			
5	P4-3	P5-5			

Table 11: (Our Proposed Algorithm- Case 3)

Round	Pi - TQi				
1	P1-5	P2-2	P3-3	P4-2	P5-2
2	P2-3	P3-5	P4-3	P5-3	
3	P2-7	P3-8	P4-5	P5-5	
4	P4-8	P5-8			
5	P4-3	P5-5			

Table 12: (Comparison between PBDRR and Our Proposed Algorithm - Case 3)

Algorithm	Avg TAT	Avg WT	CS
PBDRR	46.4	31	17
Our proposed algorithm	46.4	31	17

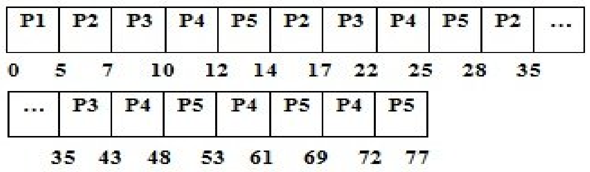

Figure 8: Gantt Chart for BPDRR (CASE 3)

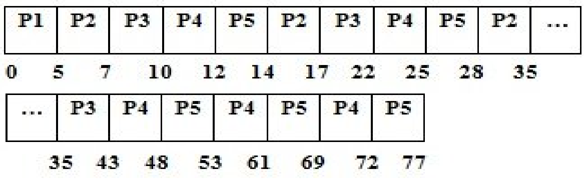

Figure 9: Gantt Chart for Our Proposed Algorithm (CASE 3)

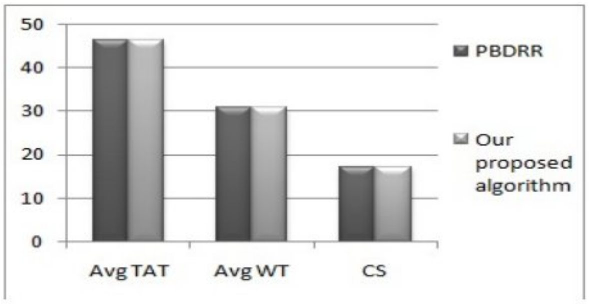

Figure 10: Comparison of Performance of Algorithms - PBDRR with Our Proposed Algorithm (Case-3)

IV. Conclusion

From the previous work and comparisons, we observed that our proposed algorithm is performing better than the algorithm PBDRR proposed in paper [3] in most case in terms of average waiting time, average turnaround time and number of context switches thereby saving the memory spaces and in addition to that, we enabled the algorithm to choose any of the ready process will be the next based on the waiting time and service time, so in the future work could develop this algorithm by adding deadline parameter in addition to the priority in the proposed algorithm to work on Hard Real Time Systems that have hard deadline, which causes system fail events.

References

[1] S. Baskiyar and N. Meghanathan, "*A Survey of Contemporary Real-time Operating System,"* Informatica ,29, pp 233–240, 2005.

[2] M.Kaladevi and S.Sathiyabama," *A Comparative Study of Scheduling Algorithms for Real Time Task,"* International Journal of Advances in Science and Technology, Vol. 1, No. 4, 2010.

[3] Rakesh Mohanty, H. S. Behera, Khusbu Patwari , Monisha Dash and 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 (IJACSA)Vol. 2, No.2, Feb 2011.

[4] H.S. Behera, Simpi Patel and Bijayalakshmi Panda, "*A New Dynamic Round Robin and SRTN Algorithm with Variable Original Time Slice and Intelligent Time Slice for Soft Real Time Systems,*" International Journal of Computer Applications (0975 – 8887) Volume 16– No.1, February 2011.

[5] William Stallings, "*Operating System Internals and Design Principles, 7th Edition,*" Prentice Hall, Mar. 10, 2011.

[6] H.S. Behera ,Brajendra Kumar Swain, Anmol Kumar Parida and Gangadhar Sahu, "*A New Proposed Round Robin with Highest Response Ratio Next (RRHRRN) Scheduling Algorithm for Soft Real Time Systems*," International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-1, Issue-3, Feb 2012.

[7] Ishwari Singh Rajput and Deepa Gupta, "*A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems*," International Journal of Innovations in Engineering and Technology (IJIET) ISSN: 2319 – 1058 Vol. 1 Issue 3 Oct 2012.

[8] Abraham Silberschatz , Peter Baer Galvin and Greg Gagne, "*Operating system concepts, 9th Edition*," John Wiley and Sons, Dec 17 .2012.