A New Proposed Round Robin with Highest Response Ratio Next (RRHRRN) Scheduling Algorithm for Soft Real Time Systems

H.S. Behera, Brajendra Kumar Swain, Anmol Kumar Parida, Gangadhar Sahu

Abstract*: *The efficiency and performance of multitasking operating systems mainly depend upon the use of CPU scheduling algorithms. Round Robin (RR) performs optimally in timeshared system but it is not suitable for real time system because it gives more number of context switches, larger waiting and turnaround time. In this paper, we have proposed a new Round Robin with Highest Response Ratio Next (RRHRRN) scheduling algorithm, which uses Highest Response Ratio (HRR) criteria for selecting processes from Ready Queue. Our experimental result shows that our proposed algorithm performs better than algorithm in DQRRR [1] in terms of reducing the number of context switches, average waiting time and average turnaround time.

Index Terms*: *Context Switch, Highest Response Ratio Next Algorithm, Real Time Operating System, Response Ratio, Round Robin Algorithm, Scheduling, Turnaround Time, Waiting Time.

I. 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. Scheduling refers to the way processes are assigned to run on the available CPUs, since there are typically many more processes running than available CPUs. 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 [7].

There are many different scheduling algorithms which vary in efficiency according to the environment. The Criteria for a good scheduling algorithm depends, on the following measures [8]:

- Fairness: all processes get fair share of the CPU,
- 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.

First-Come-First-Served (FCFS) 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) 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) 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. RR is similar to FCFS except that preemption is added to processes [8, 9].

Highest Response Ratio Next (HRRN) scheduling is a non-preemptive discipline similar to Shortest Job Next (SJN) 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. Jobs gain higher priority the longer they wait.

$$Response\ Ratio = \frac{Waiting\ Time + Service\ Time}{Service\ Time}$$

In this paper, we proposed a new algorithm (RRHRRN) in which the Highest Response Ratio Next (HRRN) in conjunction with Round Robin

Revised Manuscript Received on February 05, 2012.

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, Odisha, India. (e-mail: hsbehera_india@yahoo.com).

Brajendra Kumar Swain is a Final Year Undergraduate B.Tech student in Dept. of Computer Science & Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India. Mobile No. 09090553319, (e-mail: brajendra.swain@gmail.com).

Anmol Kumar Parida is a Final Year Undergraduate B.Tech student in Dept. of Computer Science & Engineering in Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India. Mobile No. 07205784574, (e-mail: anmolparida@gmail.com).

Gangadhar Sahu is a Final Year Undergraduate B.Tech student in Dept. of Computer Science &Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India. Mobile No. 09437275515, (e-mail: ganga.sahu29@gmail.com).

(RR). It is similar to RR as processes are added to the tail of the Ready Queue according to arrival time. And after taking the CPU for a time quantum the process is added again to the tail of the Process Queue, but selection of the processes from the ready queue is based on HRRN criteria i.e. CPU is assigned to the process having highest response ratio. The proposed algorithm takes dynamic time quantum into account. Section-3 describes the proposed method in details. Section-4 discusses the experimental analysis and comparison of our proposed algorithm with DQRRR [1] algorithm, before concluding this paper in the last section.

II. RELATED WORK

Abielmona[4] provided an analytical overview of scheduling algorithms. The Proportional Share Scheduling Algorithm proposed by Helmy and Dekdouk [5] combines low overhead of round robin algorithm and favor shortest jobs. Many works has been done related to this. DQRRR is based upon dynamic time quantum where the quantum time was taken by taking the median of the burst time. PBDRR [11] is combination of priority algorithm and RR while the SBRRR [6] is much similar to a combination between SJF and RR. Other works have used mathematical approaches, giving new procedures using mathematical theorems [3].

III. PROPOSED APPROACH

Proposed Round Robin with Highest Response Ratio Next (RRHRRN) algorithm finds the dynamic time quantum by taking the mean of burst time of the processes and fills the Ready Queue according to arrival time. We calculate the Response ratio of each process and assign the CPU to the process with Highest Response Ratio. After the process is being assigned to the CPU again the Response Ratio is calculated with the updated waiting time of the processes. This loop is continued until all the processes are being executed by the CPU. The dynamic time quantum is computed by taking the mean of the remaining burst time. The processes with shorter burst time and higher waiting time are executed first resulting in better turnaround time and better waiting time.

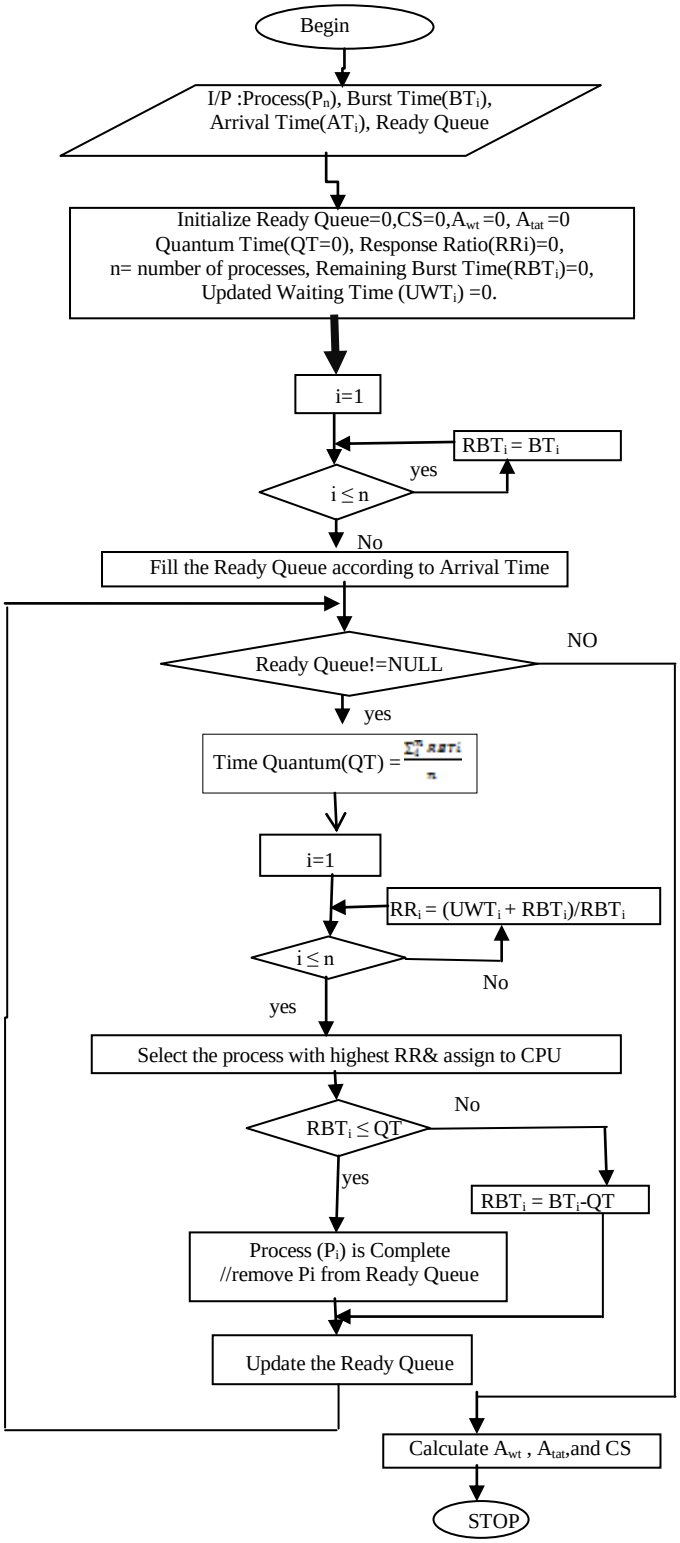
A. Algorithm (Pseudo Code)

1) I/P : Process(P_n), Burst Time(BT_i), Arrival Time(AT_i), Ready Queue ;
   O/P: Context Switch (CS), Average Waiting Time(A_{wt}), Average Turnaround Time(A_{tat}).
2) Initialize Ready Queue=0,CS=0,A_{wt} =0, A_{tat}=0, Quantum Time (QT=0), n=number of processes;
3) /* initialization */
   For (i=1 to n)
   {
   Remaining Burst Time (RBT_i) =0;
   Response Ratio (RRi)=0;
   Updated Waiting Time (UWT_i) =0;
   }
4) For process (i = 1 to n)
   $RBT_i = BT_i$;
5) Fill the Ready Queue according to Arrival Time
6) While(Ready Queue!=NULL)
   /*find the time quantum using Mean of remaining burst time */
   $QT = \frac{\sum_i^n RBTi}{n}$
7) For process (i = 1 to n)
   /* calculate Response Ratio (RR_i) */
   $RR_i = \frac{(UWTi + RBTi)}{RBTi}$
8) for process (i =1 to n)
   Select the process with highest RR_i and Assign to the CPU;
9) if ($RBT_i \leq QT$)
   Process (P_i) is Complete;
   /*remove from Ready Queue */
   else
   $RBT_i = BT_i - QT$;
   /*update the remaining burst time */
   Update the Ready Queue
10) for process(i=1 to n)
   Update UWT_i
   end while;
11) Calculate A_{wt} , A_{tat}, and CS;
12) STOP & EXIT;

B. Steps

step-1 : Processes with Arrival time and Burst Time are considered.
step-2 : Ready Queue filled according to arrival times.
step-3 : Time Quantum is calculated by taking the mean of Remaining Burst Times.
step-4 : Response Ratio is calculated for each process.
step-5 : Process with highest response ratio is executed first.
step-6 : Check which processes are finished and which require more processor time.
step-7 : Remaining Burst Time is updated.
step-8 : Repeat steps 3-7 until all the processes are not completed.

C. Flow Chart for Proposed Algorithm



IV. EXPERIMENTAL ANALYSIS

A. Assumptions

The environment where all the experiments are performed is a single processor environment and all the processes are independent. Time slice is assumed to be not more than the maximum burst time. All the attributes like burst time, number of processes and the time slice of all the processes are known before submitting the Processes to the processor. All processes are CPU bound. No processes are I/O bound. Also, a large number of processes is assumed in the ready queue for better efficiency. Since, the cases are assumed to be close to ideal, the Context Switching Time is equal to zero i.e. there is no Context Switch Overhead incurred in transferring from one job to another.

B. Experimental Frame Work

Our experiment consists of several input and output parameters. The input parameters consist of burst time, arrival time, time quantum and the number of processes. The output parameters consist of average waiting time, average turnaround time, number of context switches, Overhead, Throughput and CPU Utilization.

C. Data set

We have performed six experiments for evaluating performance of our proposed algorithm. For the first three we have considered a data set as the process with arrival time in decreasing, increasing and random order respectively. Same order of data set was taken for processes with zero arrival time.

D. Performance Parameters

The significance of our performance metrics for Experimental analysis is as follows:

1) Turnaround time (TAT): For the better performance of the algorithm, average turnaround time should be less.

2) Waiting time (WT): For the better performance of the algorithm, average waiting time should be less.

3) Number of Context Switches (CS): For the better performance of the algorithm, algorithm, the number of context switches should be less.

E. Result Obtained

This algorithm can work effectively with large no of data. In each case we have compared our proposed algorithm's result with DQRRR [1].

We take 5 processes for consideration

Case-1: **Processes with Zero Arrival Time**. **In Increasing Order**

Table 1: Data in Increasing Order (Case-1)

Process No	Arrival Time	Burst Time
1	0	30
2	0	42

3	0	50
4	0	85
5	0	97

Table 2: Gantt chart for DQRRR

p1	p5	p2	p4	p3	p5	p4	p5

0 30 80 122 172 222 263 298 304

Table 3: Gantt chart for RRHRRN

p1	p2	p3	p4	p5

0 30 72 122 207 304

Table 4: comparison between DQRRR &RRHRRN

PARAMETERS	ALGORITHM	
	DQRRR	RRHRRN
q_t	50,41,6	61,69,78,91,97
Cs	7	4
a_{wt}	134.4	88.6
a_{tat}	195.2	147
Overhead	high	low

In Decreasing Order

Table 5: Data in Decreasing Order (Case-1)

Process No	Arrival Time	Burst Time
1	0	80
2	0	72
3	0	65
4	0	50
5	0	43

Table 6: Gantt chart for DQRRR

p5	p1	p4	p2	p3	p1	p2	p1

0 35 95 140 200 260 297 327 335

Table 7: Gantt chart for RRHRRN

p1	p5	p3	p1	p1	p2

0 67 102 147 20 245 335

Table 8: comparison between DQRRR &RRHRRN

PARAMETERS	ALGORITHM	
	DQRRR	RRHRRN
q_t	80,57,11,4	67,54,59,63,64,90
cs	7	5
a_{wt}	147.8	140.2
a_{tat}	209.8	207.2
overhead	high	avg.

In Random Order

Table 9: Data in Random Order (Case-1)

Process No	Arrival Time	Burst Time
1	0	92
2	0	70
3	0	35
4	0	40
5	0	80

Table 10: Gantt chart for DQRRR

p3	p1	p4	p5	p2	p1

0 35 115 155 235 305 317

Table 11: Gantt chart for RRHRRN

p1	p3	p4	p1	p2	p5

0 64 99 139 167 237 317

Table 12: comparison between DQRRR & RRHRRN

PARAMETERS	ALGORITHM	
	DQRRR	RRHRRN
q_t	80,11	64,51,55,60,75,80
Cs	5	5
a_{wt}	150.2	128.4
a_{tat}	215.6	171.8
Overhead	avg.	avg.

Case-2: Processes without Zero Arrival Time In Increasing Order

Table 13: Data in Increasing Order (Case-2)

Process No	Arrival Time	Burst Time
1	0	28
2	2	35
3	6	50
4	6	82
5	8	110

Table 14: Gantt chart for DQRRR

p1	p2	p5	p3	p4	p5	p4	p5

0 28 63 129 179 245 275 291 305

Table 15: Gantt chart for RRHRRN

p1	p2	p3	p4	p5

0 28 63 113 195 305

Table 16: comparison between DQRRR and RRHRRN

PARAMETERS	ALGORITHMS	
	DQRRR	RRHRRN
q_t	28,66,30,14	61,70,81,96,110
Cs	7	4
a_{wt}	112.2	75.4
a_{tat}	173.2	97.0
overhead	High	low

In Decreasing Order

Table 17: data in decreasing order

Process No	Arrival Time	Burst Time
1	0	80
2	2	72
3	3	65
4	4	50
5	5	43

Table 18: Gantt chart for DQRRR

p1	p5	p2	p4	p3	p2	p3	p2

0 80 123 180 230 287 298 306 310

Table 19: Gantt chart for RRHRRN

p1	p5	p1	p4	p3	p2

0 62 105 123 173 138 310

Table 20: comparison between DQRRR and RRHRRN

PARAMETERS	ALGORITHMS	
	DQRRR	RRHRRN
q_t	80.57,11,4	62,50,52,63,69,72
Cs	7	5
a_{wt}	147.8	104.16
A_{TAT}	209.8	187.00
OVERHEAD	HIGH	AVG.

Random Order

Table 21: Data in Random Order

Process No	Arrival Time	Burst Time
1	0	26
2	1	82
3	2	70
4	5	31
5	7	40

Table 22: Gantt chart for DQRRR

p1	p4	p2	p5	p3	p2	p3	p2

0 26 57 112 152 207 228 243 249

Table 23: Gantt chart for RRHRRN

p1	p4	p5	p3	p2

0 26 57 97 167 249

Table 24: comparison between DQRRR and RRHRRN

PARAMETERS	ALGORITHMS	
	DQRRR	RRHRRN
q_t	26,55,21,6	50,56,64,76,82
cs	7	4
a_{wt}	95.6	66.8
a_{tat}	145.4	116.2
overhead	high	low

F. Comparison Graph

Case 1(With Zero Arrival)

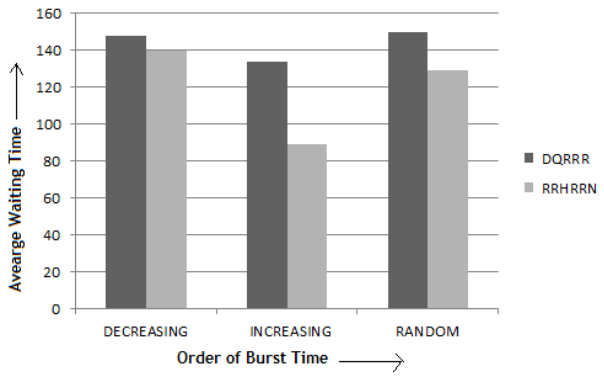


Fig. 1: Comparison Graph For Avg. Waiting Time (case-1)

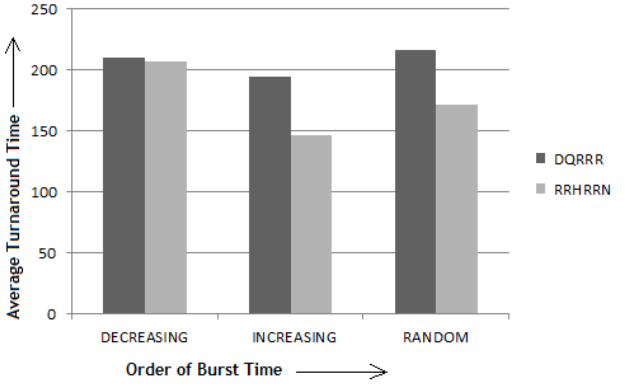


Fig. 2: Comparison Graph Of Average Turnaround Time (case-1)

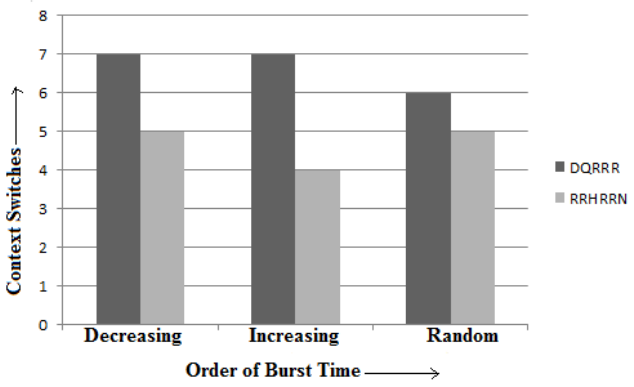


Fig. 3: Comparison Graph For Context Switch (case-1)

Case-2: Without Zero Arrival Time

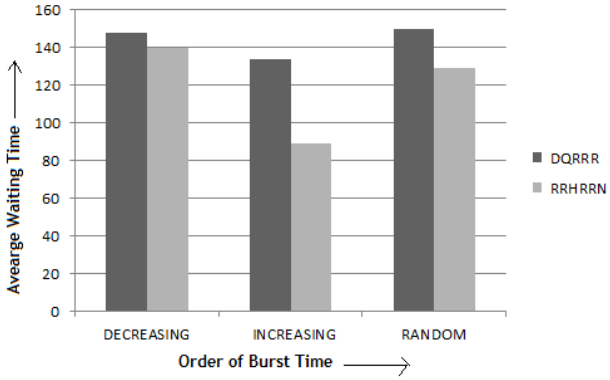


Fig. 4: Comparison Graph For Average Waiting Time (case-2)

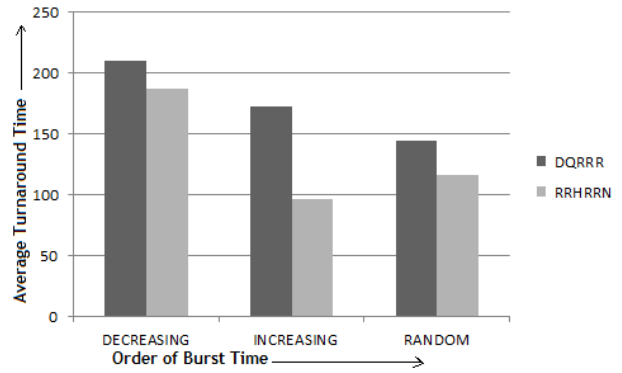


Fig. 5: Comparison Graph for Average Turnaround Time (case-1)

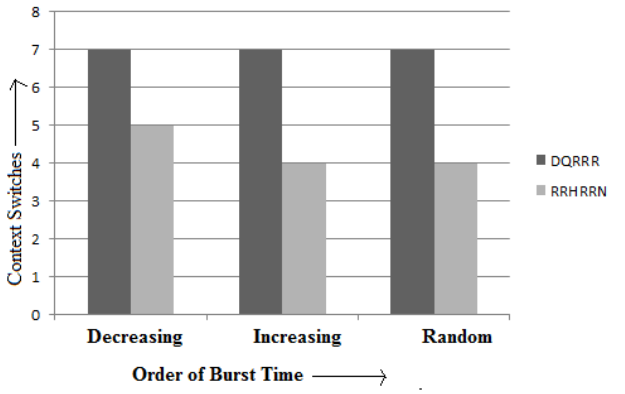


Fig. 6: Comparison Graph for Context switch

V. CONCLUSION

From the above comparisons, we observed that our new proposed algorithm RRHRRN is performing better than the algorithm DQRRR proposed in paper [1] in terms of average waiting time, average turnaround time, and no of context switches there by reducing the overhead and saving the memory spaces. Our proposed algorithm can be further investigated to be useful in providing more and more task-oriented results in future along with developing adaptive algorithms to fit the varying situations in today's multifaceted complex working of operating system.

REFERENCES

1. H.S.Behera, R.Mohanty, Debashree Nayak " A New Proposed Dynamic Quantum With Re-Adjusted Round Robin Scheduling Algorithm & its Performance ",International Journal of Computer Applications(0975-8887),Vol 05-No.5, August 2010.
2. Yaashuwanth.C & R.Ramesh "Intelligent Time Slice For Round Robin In Real Time Operating Systems" Ijrras 2 (2) - February 2010.
3. 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 AppliedSciences (IJRRAS), Vol 5, Issue 1, 2010.
4. Rami Abielmona, Scheduling Algorithmic Research,Department of Electrical and Computer Engineering Ottawa-Carleton Institute, 2000.
5. TarekHelmy, Abdelkader, Dekdouk, "Burst Round Robin: As a Proportional-Share Scheduling Algorithm", IEEE Proceedings of the fourth IEEE-GCC Conference on towardsTechno-Industrial Innovations, pp. 424-428, 11- 14 November,2007.
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. 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.
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", Super Computing 2003, November 15-21, Phoenix, AZ, USA.
10. Abbas Noon, Ali Kalakech, Seifedine Kadry, "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, May 2011,ISSN (Online): 1694-0814,
11. 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" , (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No.2, February 2011

AUTHORS PROFILE

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, Odisha, India. His areas of interest include Distributed Systems, Data Mining and Soft Computing.

Brajendra Kumar Swain is a Final Year B.Tech student in Dept. of Computer Science & Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India.

Anmol Kumar Parida is a Final Year B.Tech student in Dept. of Computer Science & Engineering in Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India.

Gangadhar Sahu is a Final Year B.Tech student in Dept. of Computer Science &Engineering, Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India.