IJournals: International Journal of Software & Hardware Research in Engineering
ISSN-2347-4890
Volume 4 Issue 10 October, 2016

An Optimum Dynamic Priority based Multilevel Round Robin CPU Scheduling Algorithm

Durgesh Samariya

M.Tech (Software Technology), VIT University, Vellore, Tamilnadu, India

Samariya.durgesh@gmail.com

Abstract: *The main objective of this paper is to improve the Round Robin scheduling algorithm using the priority of the processes. CPU Scheduling gets to be essential in fulfilling the working framework (OS) outline objectives. The expectation ought to be permitted whatever number as could be allowed running courses of action at record-breaking with a specific end goal to make best utilization of CPU. CPU planning has solid impact on asset usage and additionally general execution of the framework. Round Robin calculation performs ideally in timeshared frameworks, 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. In this paper, a new CPU scheduling algorithm called An Optimum Dynamic Priority based Multilevel CPU Round Robin Scheduling Algorithm is proposed, which figures shrewd time cut and changes after every round of execution as per need of methodology. The proposed calculation was assessed on some CPU Scheduling goals and it was watched that this calculation gave great execution when contrasted with the other existing CPU Scheduling calculations.*

Keywords: Operating System, CPU scheduling, Round Robin Scheduling, Burst Time, Response Time, Waiting Time, Context Switch, Time Quantum

1. INTRODUCTION

In a single-processor systems are one where we only execute one process at a time other process must wait until already executing process completes and then they can be scheduled. Whereas multiprogramming ensures that CPU is not ideal at anytime and we have at least one process running all the time [1]. Scheduling is very important operation of operating system as each and every resource need to be scheduled and processer is very important resource as computers performance depends on CPU. CPU scheduling is a process where it decides show jobs will be executed on processer keeping in mind that CPU is used very efficiently [2]. Scheduling not only just focuses on process but also have to keep track of resources they use. This king of resource sharing between processes is also form of scheduling. And as CPU is very important resource of System it's scheduling is centre of focus [3]. It is also important to give efficient use of resource that are sharable between users or processes [4]. With all this in mind main objective of multiprogramming systems is optimal resource sharing between different users that are using same recourses [3]. A piece of the explanation behind utilizing multiprogramming is that the working framework itself is actualized as one or more methodologies, so there must be a route for the working framework and application procedures to impart the CPU. An alternate primary reason is the requirement for courses of action to perform I/O Operations in the typical course of calculation. Since I/O operations normally oblige requests of greatness of an opportunity time to finish than do CPU guidelines, multiprogramming frameworks designate the CPU to an alternate methodology at whatever point a procedure summons an I/O operation [5].

Goals for Scheduling

- Utilization/Efficiency: CPU doing important work all the time
- Throughput: No of processes executed for particular time minutes, hours etc.
- Turnaround time: It is the time between job submission and job completion.
- Waiting time: The amount of time between submissions of job till completion of job where job was not executing.
- Response Time: amount of time from process in queue till its first execution on CPU.
- Fairness: Each process gets equal CPU time and no process is neglected.

SchedulingAlgorithms:There mainly five scheduling algorithm is available:

First Come First Served Scheduling Algorithm (FCFS): FCFS is the least difficult Scheduling calculation. In this calculation prepared line keep up as FIFO. PCB of a methodology submitted to the framework is connected to the tail of the line. The calculation dispatches forms from the leader of the prepared line for execution by the CPU. At the point when a methodology has finished its undertaking it ends and is erased from the framework. The following procedure is then dispatched from the leader of the prepared line.

Shortest Job First Scheduling Algorithm (SJF): For this calculation the prepared line is kept up in place of CPU Burst length, with the most limited blast length at the leader of the line. A PCB of a procedure submitted to the framework is connected to the line as per its CPU Burst length. The calculation dispatches forms from the leader of the prepared line for execution by the CPU. At the point when a procedure has finished its undertaking it ends and is erased from the framework. The following methodology is then dispatched from the leader of the prepared line.

Priority Based Scheduling: In this algorithm, priority is associated with each process and on the basis of that priority CPU is allocated to the processes. Higher priority processes are executed first and lower priority processes are executed at the end. If multiple processes having the same priorities are ready to execute, control of CPU is assigned to these processes on the basis of FCFS [3]. Priority Scheduling can be pre-emptive and non- pre-emptive in nature.

Round Robin Scheduling Algorithm (RR): For this calculation the prepared line is kept up as a FIFO line. A PCB of a methodology submitted to the framework is connected to the tail of the line. The calculation dispatches forms from the leader of the prepared line for execution by the CPU. Methods being executed are appropriated on expiry of a period quantum, which is a framework characterized variable. A seized process' PCB is connected to the tail of the prepared line. At the point when a procedure has finished its undertaking, i.e. prior to the expiry of the time quantum, it ends and is erased from the framework. The following methodology is then dispatched from the leader of the prepared line.

OS may feature up to 3 distinct types of schedulers :A long haul scheduler (otherwise called an affirmation scheduler or abnormal state scheduler), a mid-term or medium-term scheduler and a fleeting scheduler (otherwise called a dispatcher or CPU scheduler). In Round Robin (RR) each methodology has meet need and is given a period quantum or time cut after which the procedure is acquired.Although RR gives improved response time and uses shared resources efficiently. Its impediment are bigger waiting time, undesirable overhead and bigger turnaround time for processes with variable CPU bursts for the static time quantum This motivates us to implement RR algorithm with sorted remaining burst time with dynamic time quantum concept. Another concept employed in this algorithm is the use of more than one cycle instead of a single Round Robin.

According to Seltzer, M P. Chen and J Outerhout 1990[7], the last thirty years have seen an enormous amount of research in the area of disk scheduling algorithm. The core objective has been to develop scheduling algorithms suited for certain goals sometimes with provable properties. According to F.C.D, Sabrina, Nguyen, D. Platt, S.Jha, and F. Safaei[2] says Scheduling is a fundamental function of operating system. Before use of computer resource all resource is scheduled. The CPU is only one of the primary resources. Thus its scheduling is central to Operating system design. CPU scheduling determines which process run when there are multiple run able processes CPU scheduling is important because it can have a big effect on resources utilization and overall performance of the system.

SCHEDULING CRITERIA : calculation may support one class of procedures over another. In picking which calculation to use in a specific circumstance, we must consider the properties of the different calculations. Numerous criteria have been suggested for comparing CPU scheduling algorithms. Which characteristics are recommended for looking at CPU booking calculations. Which qualities are utilized for correlation can have a significant effect in which calculation is judged to be best. The criteria incorporate the accompanying:**Utilization/Efficiency**: keep the CPU busy 100% of the time with useful work :

Throughput: maximize the number of jobs processed per hour.
Turnaround time: from the time of submission to the time of completion, minimize the time batch users must wait for output
Waiting time: Sum of times spent in ready queue - Minimize this
Response Time: time from submission till the first response is produced, minimize response time for interactive users
Fairness: make sure each process gets a fair share of the CPU

2. ORGANIZATION OF THE PAPER

The paper is divided into four sections. **Section I** gives a brief introduction on the various aspects of the scheduling algorithms, the approach to the current paper and the motivational factors leading to this improvement. **Section II** presents the proposed algorithm and illustration of our proposed new algorithm (OMDRR). In **Section III**, an experimental analysis and Result of our algorithm (OMDRR) and its comparison with the static RR algorithm. Conclusion is presented in **Section IV** followed up by the references used.

3. RELETED WORK

Several authors have proposed various modifications to round robin CPU scheduling algorithm. These modifications can be classified as follows:

Statically allocated time quantum

Ajit [13] proposed an algorithm that allocates the CPU to every process in RR fashion for an initial time quantum (say k units). After completing first cycle, it doubles the initial time quantum (2k units) and allocates the CPU to the processes in SJF format. It alternates the doubling and halving of the time quantum if processes remain in the ready queue after completing any execution cycle.

Ishwari and Deepa [15] proposed an algorithm that allocates the CPU to every process in RR fashion for only one time quantum. The CPU is then allocated to the remaining processes in the ready queue after completion of the execution in SJF fashion.

Manish and AbdulKadir [17] proposed an algorithm that allocates the CPU to processes in RR fashion. After executing each process for one time quantum, it checks if the remaining currently running process's burst time less than the time quantum. If so, it allocates the CPU to the process for the remaining burst time, else it moves the process to the tail of the ready queue.

Dynamically determined time quantum

Behera and all [14] developed an algorithm that make ready queue from the processes in ascending order of burst time. Then, the time quantum is calculated. For finding an optimal time quantum, it takes the median of the processes in the ready queue. The time quantum is recalculated taking the remaining burst time into account after each execution cycle.

Lalit et al [16] developed an algorithm that arranges the processes in ascending order of burst time, and then calculate the time quantum for RR by taking the average of the burst times. This algorithm assumes that all processes arrive at the time t=0.

Soraj and Roy [19] presented a new algorithm that arranges the processes in ascending order of burst time, and then it chooses the smart time slice (STS), which is mainly dependant on the number of processes. It is equal to the burst time of the mid process when number of processes is odd and average of the processes burst times when the number of processes is even. This algorithm assumes that all processes arrive at the time t=0.

Abdullahi and Junaidu [21] made an improvement to the Longest Job First (LJF) CPU scheduling algorithm. It works by sorting the processes in descending order of their burst times and then it determines a threshold known as Combined Weighted Average (CWA) which is the average of the processes. This threshold is used to categorize the processes into long and short processes. A Long process is a process with burst time greater than Cwa while a short process is one with burst time less than or equal to Cwa. New burst times are created from this categorization by merging two consecutive shorter processes until no shorter process has one to merge with or no shorter process exist in the categorization. After the merging, new queue is created by sorting the categorized and merged processes in descending order of burst times. The CPU is then allocated to the processes based on Longest Job First.

This paper presents a modification in RR CPU scheduling algorithm by modifying [17] and also determining the time quantum dynamically. Based on results of a simulation, application of this proposed algorithm in time sharing and real time systems will increase the performance of the systems by reducing average waiting time, average turnaround time and number of context switches.

4. PROPOSED ALGORITHM

The proposed CPU scheduling algorithm is a modification of the algorithm presented in [17]. It assumes another queue called the Priority that holds the priority of Process. Higher priority show by min number 1. Higher priority process gets first chance in CPU allocation.

The Optimum Dynamic Priority based Multilevel Round Robin CPU Scheduling Algorithm (ODPMRR), in this algorithm we just do some minor change in Round Robin and we analyze its more efficient. This algorithm picks the process with the higher priority from the process queue and allocates the CPU to it for a 1time quantum. If currently running process's remaining burst time is less than 1 time quantum, then again CPU allocated to for remain BT to current running process. For this process, it will finish execution and removed from ready queue. OR in other hand if the burst time of the running process is more than 1 TQ, the

process will be transfer at the end of the RQ. Now select next process from the ready queue with next priority value.

Following is the proposed Optimum Dynamic Priority based Multilevel Round Robin CPU Scheduling Algorithm

1. Start
2. Create a queue, ARRIVE, PRIORITY, where process will be placed when they arrive the system before they moved to ready queue.
3. Create a ready queue, REQUEST.
4. Do
5. Set time_quantum.
6. If (process_index >=1)
{
}
}
7. Remove the currently finish process from running process and put it at the end of the ready queue.
8. END

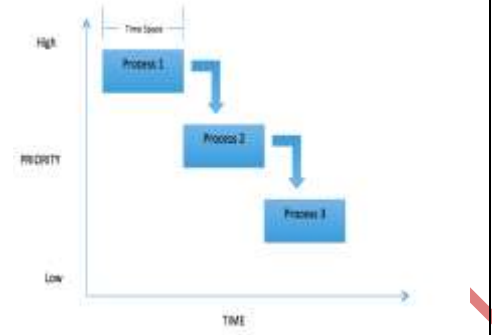

Figure: Process Architecture for ODMPRR

4.1 Work Flow:

Considering a Ready Queue (RQ) with five processes at arriving at time 0 and process is P1, P2, P3, P4 and P5 burst time (BT) 15, 7, 28, 20, and 3 respectively; priority of this process is 3, 1, 5, 2, and 4 respectively. We assumed Time quantum (TQ) = 10 milliseconds (ms). Our proposed Optimum Dynamic Priority based Multilevel Round Robin CPU Scheduling Algorithm picks the process with high priority (P2 is higher priority) from the RQ and allocate the CPU to P2 for a 1 time quantum time interval is 10 ms so give CPU for 10 ms. After executing P2 remaining burst time is 7 ms. So it's a less than 1time quantum. Again P2 process takes CPU for 7 ms. P2 will finish execution and removed from the ready queue. Now take Next Process from the RQ is P4 (Priority 2) with CPU
burst time 20. CPU allocated to process P4 for a 1TQ of 10ms. After executing process P4 for 1TQ the remaining BT is 10ms for P4 Process. So the

Give CPU upto time_quantum to process with priority 1
Move the first process (pr[Priority]) to REQUEST queue
If(Brust_Time < time_quantum)
{
Again allocate the CPU to same process for the next Burst time.
After completion of process execution, this process has been removed from ready queue and goes to step 4.
}
Else
{
There is no process in ready queue.

remaining BT of process P4 is more than the 1TQ. So it's put in end of the queue. Next process in the RQ is P1 (Priority 3) burst time is 15. CPU allocated to Process P1 for a 1TQ. After the execution process P1 remaining BT is 5ms. Since the remaining BT of P1 is less than the 1TQ, so Now CPU will be allocated to again for 5ms. P1 it will be deleted from the RQ when it has finished execution. Next Process in the RQ is P5 (Priority 4) with BT 3. CPU will be allocated to process P5 for a time interval. P5 will be removed from the RQ when it will finish execution. Next process in the RQ is P3 with Priority of 5 and BT is 28. CPU will allocate to P3 for a time interval of 1TQ value. After executing 10ms of time, the remaining CPU BT of P3 is not less than the 1 TQ.After First Cycle the process reaming in ready queue are P3 with BT 18 and P4 with BT 10. Now P4 is executing before P3, because P4 have higher priority than P3. The first process will be selected from ready queue and execute up to a time interval of 10ms. P4 has finish execution and it will be deleted from ready queue. Next process in ready queue is P3 with 18-burst time. CPU will allocated to P3 for a time interval of 10 ms. after executing P3 for 10ms, the remaining CPU burst of P3 is 8. Since the reaming CPU burst time of P3 is less than time Quantum. CPU will be allocated again to P3 for a time interval of 8ms. P3 has finished execution; it will be removed from the ready queue.

4.2 Experiment:

For performance evaluation of proposed ODPMRR algorithm, we take two different cases. In the first case arrival time has been considered zero and Priority has been taken increasing, decreasing and random orders. In second case arrival time has been considered non-zero and Priority has been taken in increasing, decreasing and random orders.

CASE 1 – When Arrival Time is considered as ZERO: In this case we take arrival time of all processes has been considered as zero and priority

of all processes has been in increasing, decreasing and random orders. TQ is 10 milliseconds.

CPU Priority in increasing order: We consider the RQ with five processes P1, P2, P3, P4, and P5 arriving at time 0 according to our case and BT is 5, 12, 20, 26, and 34 respectively, and priority is 1, 2, 3, 4 and 5 respectively. This comparison result of RR, IRR, and ODPMRR are shown in table 2, figure 5, 6, and 7 show the Gantt chart representation of RR, IRR, and ODPMRR respectively.

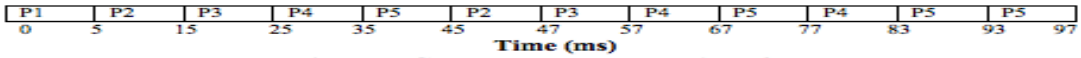

Figure 5. Gantt chart representation of RR

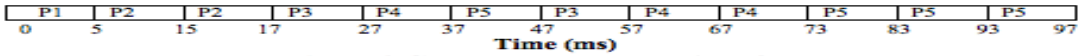

Figure 6. Gantt chart representation of IRR

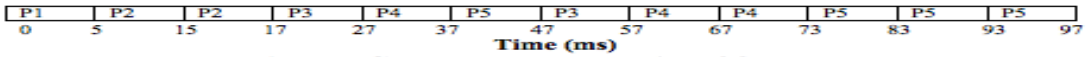

Figure 7. Gantt chart representation of ODPMRR

Algorithm	Average Waiting Time(ms)
RR	38.4
IRR	30.4
ODPMRR	30.4

Table 2. Comparison of RR, IRR and ODPMRR.

Priority in Decreasing Order: We consider the RQ with five processes P1, P2, P3, P4, and P5 arriving at time 0 with BT 5, 12, 20, 26, and 34 respectively, and priority is 5, 4, 3, 2 and 1 respectively. This comparisonresult of RR, IRR, and proposed ODPMRR are shown in table 3, figure 8, 9, and 10 show the Gantt chart representation of RR, IRR, and ODPMRR respectively.

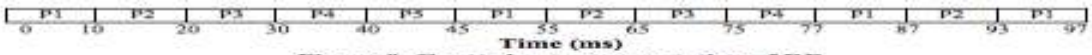

Figure 8. Gantt chart representation of RR

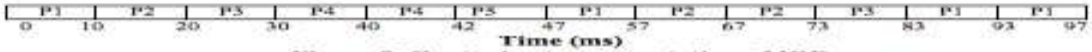

Figure 9. Gantt chart representation of IRR

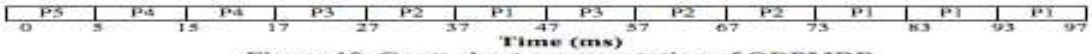

Figure 10. Gantt chart representation of ODPMRR

Algorithm	Average Waiting Time(ms)
RR	58
IRR	49
ODPMRR	30.4

Table 3. Comparison of RR, IRR and ODPMRR.

Priority in Random Order: We consider the RQ with five processes P1, P2, P3, P4, and P5 arriving at time 0 with BT 20, 34, 5, 12, and 26 respectively. And priority is 3, 5, 1, 2 and 4 respectively. This comparison result of RR, IRR, and proposed ODPMRR are shown in table 4, figure 11, 12, and 13 show the Gantt chart representation of RR, IRR, and ODPMRR respectively.

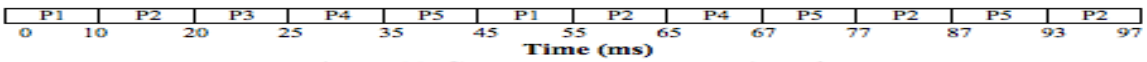

Figure 11. Gantt chart representation of RR

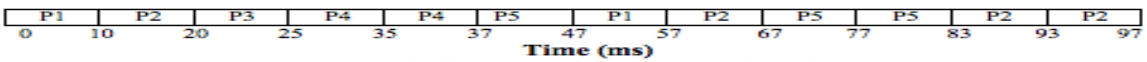

Figure 12. Gantt chart representation of IRR

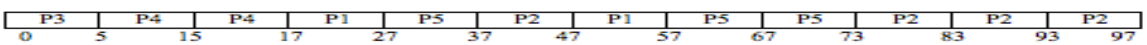

Figure 13. Gantt chart representation of ODPMRR

Algorithm	Average Waiting Time(ms)
RR	47
IRR	40.4
ODPMRR	30.4

Table 4. Comparison of RR, IRR and ODPMRR.

CASE 2 – When Arrival Time is greater than ZERO: In this case arrival time of all processes has been considered as non-zero and Priority has been taken in increasing, decreasing and random orders. Time quantum is 10 milliseconds.

Priority in Increasing Order: We consider the RQ with five processes P1, P2, P3, P4 and P5 arriving time 0, 4, 10, 15, and 17 with BT 7, 18, 27, 30 and 36 respectively and Priority of processes is 1, 2, 3, 4 and 5 respectively. The comparison result of RR, IRR and ODPMRR are shown in Table 5, figure 14, 15, and 16 show the Gantt chart representation of RR, IRR and ODPMRR respectively.

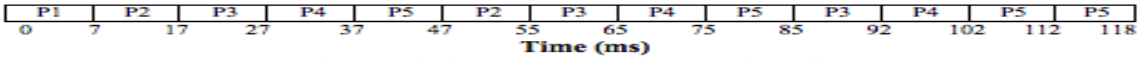

Figure 14. Gantt chart representation of RR

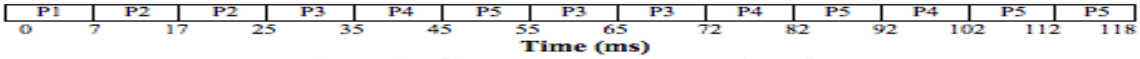

Figure 15. Gantt chart representation of IRR

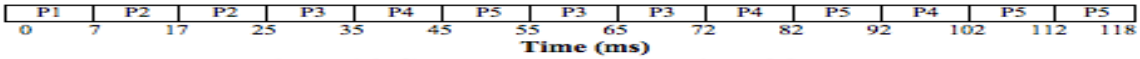

Figure 16. Gantt chart representation of ODPMRR

Algorithm	Average Waiting Time(ms)
RR	42
IRR	32
ODPMRR	32

Table 5. Comparison of RR, IRR and ODPMRR.

Priority in Decreasing Order: We consider the RQ with five processes P1, P2, P3, P4 and P5 arriving time 0, 4, 10, 15, and 17 with BT 36, 30, 27, 18 and 7 respectively and Priority of processes is 5, 4, 3, 2 and 1 respectively. The comparison result of RR, IRR and ODPMRR are shown in Table 6, figure 17, 18, and 19 show the Gantt chart representation of RR, IRR and ODPMRR respectively.

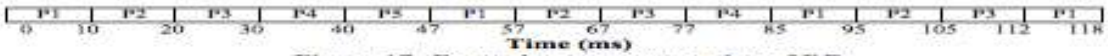

Figure 17. Gantt chart representation of RR

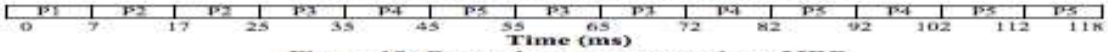

Figure 18. Gantt chart representation of IRR

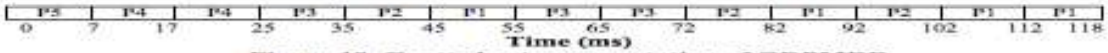

Figure 19. Gantt chart representation of ODPMRR

Algorithm	Average Waiting Time (ms)
RR	60.6
IRR	51.4
ODPMRR	32

Table 6. Comparison of RR, IRR and ODPMRR.

Priority in Random Order: We consider the RQ with five processes P1, P2, P3, P4 and P5 arriving time 0, 4, 10, 15, and 17 with BT 27, 7, 30, 36 and 18 respectively and Priority of processes is 1, 3, 2, 5 and 4 respectively. The comparison result of RR, IRR and ODPMRR are shown in Table 7, figure 20, 21, and 22 show the Gantt chart representation of RR, IRR and ODPMRR respectively.

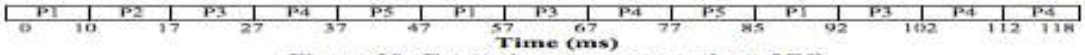

Figure 20. Gantt chart representation of RR

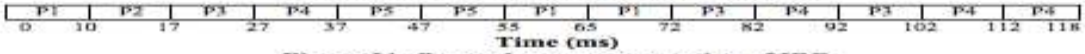

Figure 21. Gantt chart representation of IRR

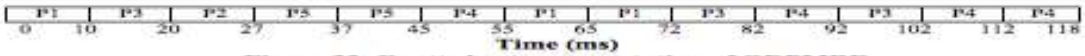

Figure 22. Gantt chart representation of ODPMRR

Algorithm	Average Waiting Time(ms)
RR	52
IRR	40
ODPMRR	32

Table 7. Comparison of RR, IRR and ODPMRR.

5. CONCLUSION

We all know that CPU is the heart of the computer. CPU scheduling checks penning process and examine them carefully and determine the best way efficient way to service the request. This Paper mainly defines the existing algorithm and one more new algorithm called Optimum Dynamic Priority based Multilevel Round Robin CPU Scheduling Algorithm (ODPMRR). This new algorithm compared with all existing algorithms. From example it is clear that Optimum Dynamic Priority based Multilevel Round Robin Scheduling Algorithm gives best and optimized average waiting time compared with other existing algorithm and when time quantum is increased Optimum Dynamic Priority based Multilevel Round Robin Scheduling Algorithm give best average waiting time.

REFERENCES

1. Silberschatz A. P.B. Galvin and G. Gagne, "Operating System Concepts, 8th edition, Wiley India (2012).
2. Sabrian, F., C.D. Nguyen, S. Jha, D. Platt and F. Safaei, "Processing resource scheduling in programmable networks Computer communication" (2005).
3. Syed Nasir Shah, Ahmad Mahmood, Alan Oxley, "Hybrid Scheduling and Dual Queue Scheduling", IEEE 978-1-4244-4520-2/09 Conference.
4. Galvin, Silberschatz, Gange John, "Operating System concepts 7th Edition", Wiley & Sons, 2005
5. http://www1bpt.bridgeport.edu/sed/projects/cs503/ Spring_2001/kode/os/scheduling.htm
6. Milan Milenkovic, "Operting System Concepts and Design", Second Edition McGraw Hill International, 1992
7. Leland L. Beck, "System Software", Addison Wesley, 3rd Edition, 1997
8. Seltzer, M P. Chen and J outerhout, "Disk scheduling revisited", USENIX Winter technical conference in 1990.
9. Operating Systems Sibsankar Haldar ,Pearson Education, India 2009.
10. D.M. Dhamdhere, "Operating Systems A Concept Based Approach", Second edition, Tata McGraw-Hill, 2006.
11. Behera S. H., Mohanty R., Sahu S. & Bhoi K. S., "Design and Performance Evaluation of Multi Cyclic Round Robin (MCRR) Algorithm Using Dynamic Time Quantum", Journal of Global Research in Computer Science, Volume 2, Issue No. 2, ISSN-2229-371X, February 2011
12. Behera S. H., Patel S. & Panda B., "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.
13. Ajit S., Priyanka G., and Sahil B., "An Optimized Round Robin Scheduling Algorithm for CPU Scheduling", (2010), International Journal on Computer Science and Engineering (IJCSE), Vol. 02, No. 07, pp 2382-2385.
14. Behera H.S., Mohanty R. and Debashree N., "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, pp 10-15, 2010.
15. Ishwari S. R. and Deepa G "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, pp 1-11, 2012.
16. Lalit K., Rajendra S. and Praveen S., "Optimized Scheduling Algorithm, International Journal of Computer Applications, pp 106-109, 2011.
17. Manish K. M. and Abdul Kadir K. , "An Improved Round Robin CPU Scheduling Algorithm", Journal of Global Research in Computer Science, ISSN: 2229- 371X, Volume 3, No. 6, pp 64-69, 2012.
18. Soraj H., and Roy K.C, "Adaptive Round Robin scheduling using shortest burst approach, based on smart time slice", International Journal of Data Engineering (IJDE), Volume 2, Issue 3.
19. Suri P.K., and Sumit M. ,"Design of Stochastic Simulator for Analyzing the Impact of Scalability on CPU Scheduling Algorithms", International Journal of Computer Applications (0975 – 8887) Volume 49, No.17, pp 4-9, 2012.
20. Abdullahi I., and Junaidu S. B., "Empirical Framework to Migrate Problems in Longer Job First Scheduling Algorithm (LJF+CBT)", International Journal of Computer Applications (0975 – 8887) Volume 75– No.14, pp 9-14, 2013.
21. Abdulrazaq Abdulrahim, Saleh E Adbullahi, Januaidu B. Sahalu, "A New Improved Round Robin (NIRR) CPU Scheduling Algorithm", International Journal of Computer Application ((0975 - 8887) Volume 90-No 4, pp 27-33,March 2014.

