Volume 5, Issue 12, December 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performance Analysis of SORR CPU Scheduling Algorithm with Other Dynamic Round Robin Algorithms

Rahul Joshi*
M.Tech Scholar
Department of Computer Science & Engineering
Tula's Institute of Engineering, Uttarakhand, India

Shashi Bhushan Tyagi
Asst. Professor
Department of Computer Science & Engineering
Tula's Institute of Engineering, Uttarakhand, India

Abstract: ***CPU scheduling is the most essential and important aspect of Operating System. There are various CPU scheduling algorithms are used to perform multitasking. The most widely used scheduling algorithm is Round Robin (RR) CPU scheduling algorithm. In round robin algorithm the time quantum is static and the performance of these algorithms is totally depending upon the selection of time quantum. This static time quantum decrease the performance of CPU scheduling like Average waiting time, Average turnaround time and CPU utilization. To overcome these problems of Round Robin CPU scheduling, instead of static time quantum dynamic time quantum can be used to get optimal performance. There are various research have been done in past for the selection of dynamic time quantum in Round Robin algorithm to improve its performance. The objective of this paper is to performed the performance analysis or comparison of our proposed new Smart optimized Round Robin CPU scheduling algorithm with other proposed Dynamic Round Robin algorithms in terms of minimizing average waiting time, average turnaround time and number of context switches.***

Keywords: Context switching, CPU scheduling, Gantt chart, Response time, Round Robin CPU scheduling algorithm, Turnaround time, Waiting time.

I. INTRODUCTION

Now a day's the entire operating systems support multitasking environment in which they can run multiple processes concurrently at a same time. In a single-processor system, processor run only one process at a time in CPU and other processes have to wait in the ready queue until the CPU becomes free. The operating system must decide through the scheduler the order of execution of the processes in ready state. The objective of multiprogramming is to have some process running at all times to maximize CPU utilization. Scheduling is a fundamental operating-system function. Almost all computer resources are scheduled before using. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating-system design. CPU scheduling determines which processes run when there are multiple run-able processes. CPU scheduling is important because it can have a big effect on resource utilization and the overall performance of the system. In general we want to optimize the behavior of the system.
The goals of scheduling may be categorized as user based scheduling goals and system based scheduling goals [1]. User based goals are the criteria that benefit the user. Some User-based scheduling goals are:

- Turnaround Time: The time elapsed between the submission of a job and its termination is called the turnaround time.

$$tr = wt + x$$

Where *tr* is turnaround time of a process
wt is waiting time of the process in the ready queue.
x is the execution time of the process.

The scheduling algorithm should be designed such that turnaround time is minimized.

- Waiting Time: The time spent by the process in the ready queue is the waiting time. The scheduling algorithm should be designed such that waiting time is less.
- Response Time: It is the time interval between the time of submission of a process and the first response given by the process to the user. The scheduling algorithm should be designed such that the response time is within an acceptable range.
- Predictability: The algorithm should take care that a process does not take too long in processing as compared to the predictable behavior of the process.
- Deadlines: The scheduling algorithm should be designed such that real-time processes will execute within their deadlines.

Some system-based scheduling goals are:

- Throughput: Throughput is the number of processes completed in a unit time. The scheduling algorithm should be designed in such a way that throughput in a system is maximized.

- CPU Utilization: It is the percentage of time that the CPU is busy in executing a process. The fundamental goal of scheduling is to keep the processor busy all the time.
- Fairness: All processes in the system should be treated in the same way unless there is some preference or priority for a specific process. In that case also processes with lower priority should not be ignored to avoid starvation.
- Context Switch: Context switching is the procedure of storing the state of an active process and restoring the state of another process for the CPU when it has to start executing the later process. Context switch is total overhead to the system and leads to wastage of CPU time. The scheduling algorithm should be designed such that the context switch is minimized.

So we can conclude that a good scheduling algorithm for real time and time sharing system must possess following characteristics:

- Minimum context switches.
- Maximum CPU utilization.
- Maximum throughput.
- Minimum turnaround time.
- Minimum waiting time.
- Minimum response time.

II. RELATED WORK DONE

In the last few years different approaches are used to enhance the performance of Round Robin scheduling. Research has been conducted to achieve good fairness in a dynamic environment while having a low scheduling overhead [2]. In [3] the authors have arranged the processes in ascending order of burst time. Time slice is chosen as the CPU burst of the mid process in case of odd number of processes, otherwise time slice is equal to the average CPU burst of all running processes. In [4] the authors have proposed an algorithm (ORR) executed in three phases. In first phase the processes execute according to RR scheduling with an initial time quantum. Then in the next round the time quantum doubles and the remaining processes are scheduled according to RR scheduling. In next phase they select the shortest process to execute. In [5] the authors have selected the median process and the time slice is calculated as the time slice of the median process plus the number of processes. In [6] authors have introduced weighted dynamic RR scheduling for scheduling cells in ATM switches to reduce waiting time by considering delay in each queue. In [7] the authors have assigned weights to processes. They have assigned more weight to the process with small CPU burst and they have also considered the waiting time of the processes for I/O and accordingly modified their weights. For heterogeneous processes i.e. CPU bursts of some processes are very smaller or larger than other processes, time quantum for RR scheduling is calculated using Arithmetic mean and Harmonic mean respectively in [8]. Time slice for different rounds of RR algorithm is dynamically calculated depending on the remaining CPU bursts of currently running processes in [9]. In [10] in DQRRR algorithm time Quantum is dynamically calculated for each round as the remaining CPU burst of the median process. In first round the processes are sorted according to the ascending order of their CPU burst time and in the following rounds they are arranged in the lowest followed by highest CPU burst processes among the currently running processes. Most of the aforesaid algorithms do not consider the waiting time of a process while calculating time slice for the next round of the RR scheduling. In [11] author proposed Efficient Round Robin Scheduling Algorithm with Dynamic Time Slice which eliminates the drawbacks of implementing simple round robin architecture in real time system by introducing a concept of assigning different time quantum to different rounds of RR scheduling algorithm. At the beginning of each round of the RR algorithm the following matrices are calculated for each process:

RRB = Remaining CPU burst/Original CPU burst

$$WR = \frac{\text{Waiting time of the process till now}}{\text{Total waiting time of all currently Running processes till now}}$$

Precedence Factor (PF) = WR/RRB

The proposed algorithm eliminates the drawbacks of implementing simple round robin architecture in real time system by introducing a concept of assigning different time quantum to different rounds of RR scheduling algorithm. At the beginning of each round of the RR algorithm the following matrices are calculated for each process.

III. PROPOSED ALGORITHM

The proposed algorithm (SORR)[12] focuses on the improvement on the selection of time quantum for the scheduling of processes. Instead of giving static time quantum in the CPU scheduling algorithms, our algorithm calculates the Smart time quantum itself according to the burst time of all processes. The proposed algorithm eliminates the discrepancies of implementing simple round robin architecture. In the first stage SORR CPU scheduling algorithms all the processes are arranged in the increasing order of CPU burst time. It means it automatically assign the priority to the processes. Process having low burst time has high priority to the process have high burst time. Then in the second stage the algorithm calculates the mean of the CPU burst time of all the processes. After calculating the mean, it will set the time quantum dynamically i.e. average of mean and highest burst time .Then in the last stage algorithm pick the first process from the ready queue and allocate the CPU to the process for a time interval of up to 1 Smart time quantum. If the remaining burst

time of the current running process is less than 1 Smart time quantum then algorithm again allocate the CPU to the Current process till it execution. After execution it will remove the terminated process from the ready queue and again go to the stage 3[12].

The steps of the proposed algorithm are as follow.

Step 1: START

Step 2: Arrange the processes in the increasing order of CPU burst time in the ready queue.

Step 3: Calculate the Mean of the CPU burst time of all the Processes

$$\text{Mean (M)} = \frac{P1+P2+P3\ldots\ldots\ldots P_n}{N}$$

Step 4: Set the Smart time quantum (STQ) according to the following method

$$\text{STQ} = \frac{\text{Mean+ Highest burst time}}{2}$$

Step 5: Allocate the CPU to the First process in the Ready Queue for the time period of 1Smart Time Quantum.

Step 6: If the remaining CPU burst time of the current process is less than or equal to 1 time quantum

- Reallocate the CPU to current process again for the remaining burst time. After the complete execution of the current process, remove it from the ready queue.
- Otherwise remove the current process from the ready queue and put it on the tail of the ready queue for further execution.

Step 7: Pick the next process from the ready queue and allocate the CPU to it up to the time period of 1 Smart time quantum and then again to the step 6.

Step 8: Process the queue until it will be empty.

Step 9: Calculate the No. of context switch, Average waiting time and Average Turnaround time of all process.

Step 10: END

The flowchart for proposed algorithm is shown below in figure 1:

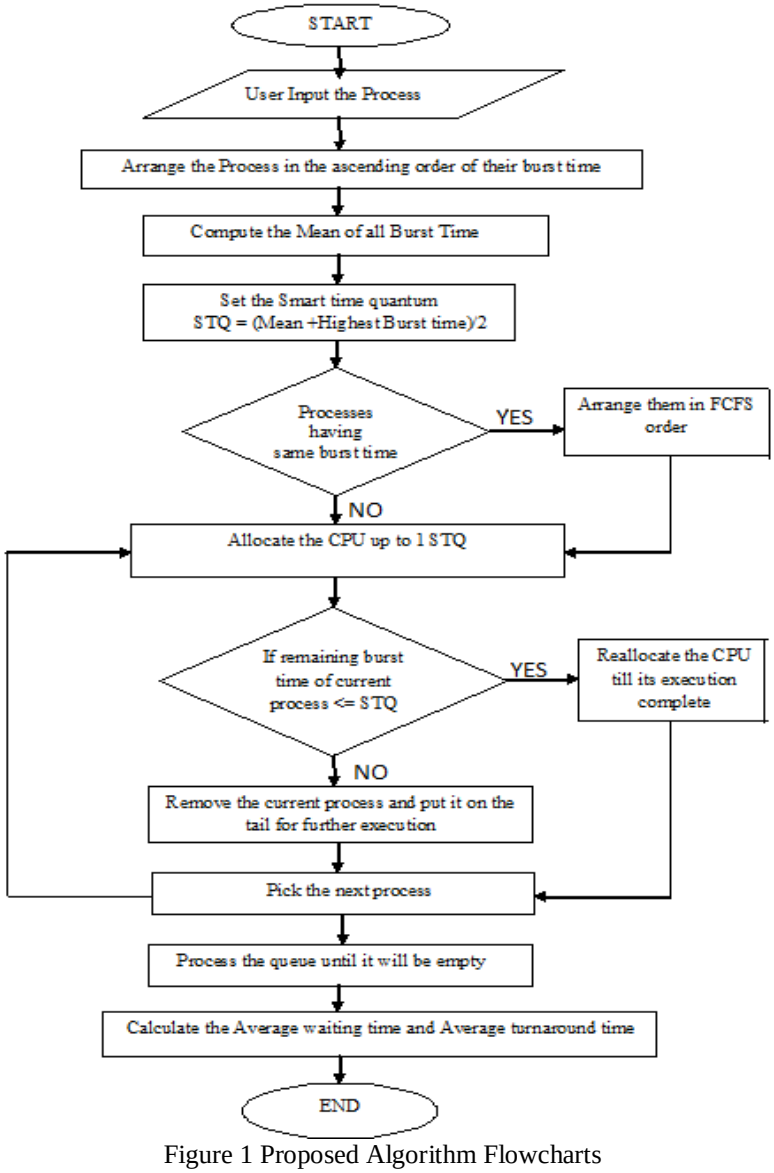

Figure 1 Proposed Algorithm Flowcharts

IV. EXPERIMENTAL ANALYSIS

Experiments are performed in single processor environment and on independent processes. All the parameters like number of processes, and burst time 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 for calculating the time slices are ignored while calculating average turnaround time and average waiting time [11].

The experimental analysis that will be adopted in this paper uses all the assumptions and experiments performed in [11], [10] and [4] and then compared the results in the original paper along with the results by the proposed algorithm. In this paper we compare the performance analysis of our proposed algorithm Smart Optimized Round Robin CPU scheduling (SORR) [12] with Efficient Round Robin Scheduling Algorithm with Dynamic Time Slice (ERRDTS) [11], Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm (DQRRR) [10] and Optimized Round Robin Scheduling Algorithm for CPU Scheduling [4].

Comparison between DQRR [10], ERRDTS [11] and SORR [12] CPU scheduling algorithms:

1. **Process with increasing Burst time**: The ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0 with increasing burst time respectively was considered.

Table 1: Inputs for the Case 1

Process ID	Burst Time
P1	30
P2	42
P3	50
P4	85
P5	97

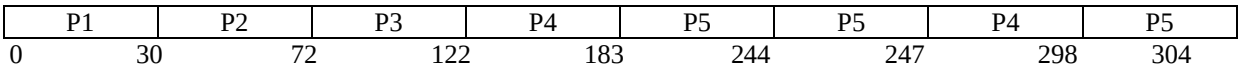

Figure 2: Gantt chart representation of ERRDTS

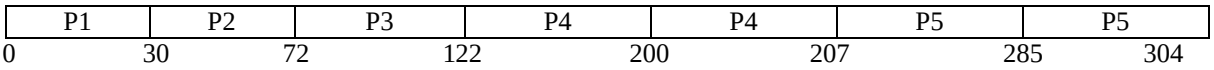

Figure 3: Gantt chart representation of SORR

Table 2: Comparative Result of DQRR, ERRDTS and SORR

Algorithms	Average Waiting Time (ms)	Average Turnaround Time(ms)	Context Switch
DQRR	134.4	195.2	7
ERRDTS	104.4	165.2	6
SORR	101.8	162.6	4

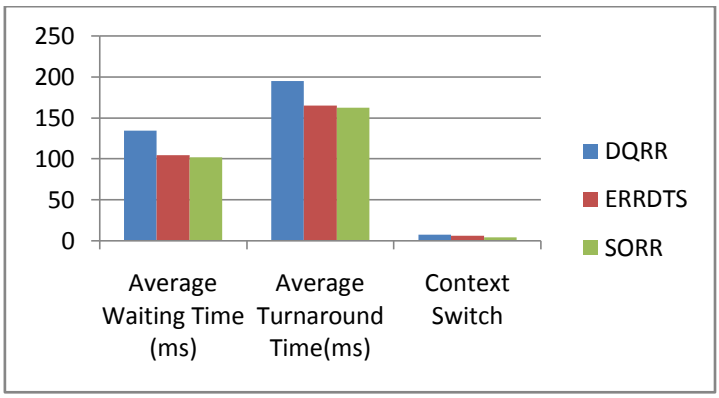

Figure 4: Bar chart of Case 1 (Increasing order)

2. **Process with decreasing Burst time**: The ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0 with decreasing burst time respectively was considered.

Table 4: Inputs for the Case 2

Process ID	Burst Time
P1	105
P2	90

P3	60
P4	45
P5	35

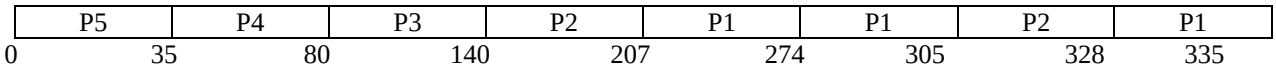

Figure 5: Gantt chart representation of ERRDTS

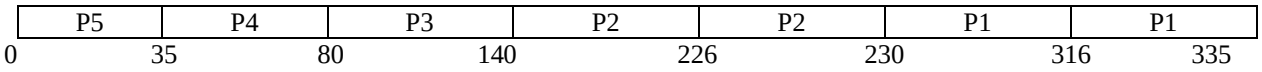

Figure 6: Gantt chart representation of SORR

Table 3: Comparative Result of DQRR, ERRDTS and SORR

Algorithms	Average Waiting Time (ms)	Average Turnaround Time(ms)	Context Switch
DQRR	152.4	219.4	7
ERRDTS	116.6	183.6	7
SORR	114.1	181.1	4

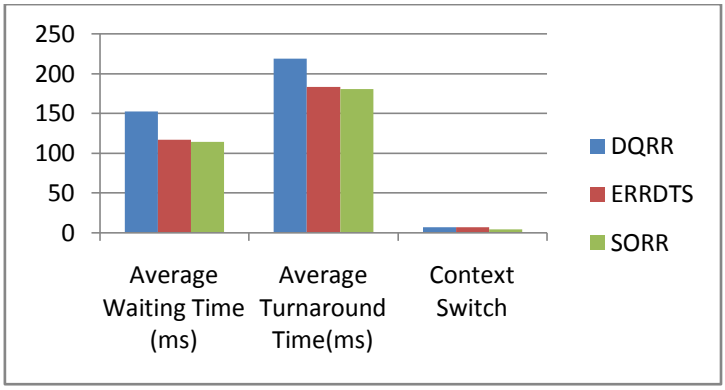

Figure 7: Bar chart of Case 2 (Decreasing order)

3. **Process with Random Burst time**: The ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0 with random burst time respectively was considered.

Table 4: Inputs for the Case 3

Process ID	**Burst Time**
P1	92
P2	70
P3	35
P4	40
P5	80

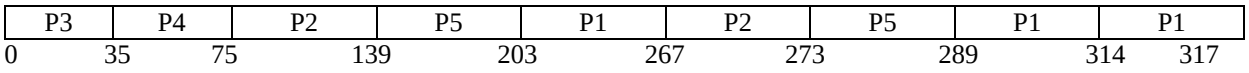

Figure 8: Gantt chart representation of ERRDTS

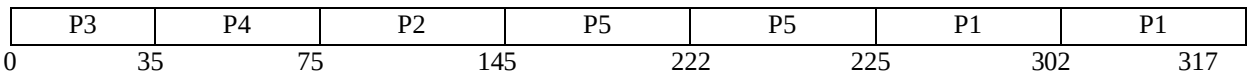

Figure 9: Gantt chart representation of SORR

Table 4: Comparative Result of DQRR, ERRDTS and SORR

Algorithms	Average Waiting Time (ms)	Average Turnaround Time(ms)	Context Switch
DQRR	150.2	215.6	7
ERRDTS	134.4	197.8	8
SORR	111.4	174.8	4

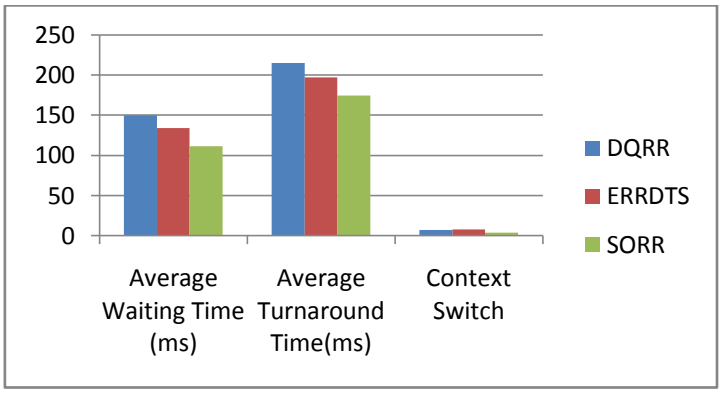

Figure 10: Bar chart of Case (Random order)

V. CONCLUSION AND FUTURE SCOPE

In this paper, we have done performance analysis of our Smart and Optimized Round Robin CPU scheduling algorithm [12] with other already proposed Dynamic Round Robin CPU scheduling algorithms as DQRR [10] and ERRDTS [11]. Results have shown that our algorithm SORR gives better results in terms of average waiting time, average turnaround time and number of context switches in all cases of process categories than the DQRR and ERRDTS. The general problem in any form of Round Robin CPU scheduling algorithm, So, a future scope is to determine more accurate dynamic time quantum or specific time quantum for each process so as to improve the performance and we can add other parameters also as Deadline concept. In future we can also improve our algorithm for multiple processors also to attain better results for multiprogramming.

REFERENCES

[1] Principles of Operating System, Naresh Chauhan, Oxford University Press, 2014.

[2] Kevin Jeffay, F. Donelson Smith, Arun Moorthy, James Anderson, "Proportional Share Scheduling of Operating System Services for Real-Time Applications", In Proceedings of the 19th IEEE Real-Time Systems Symposium, Madrid, Spain, December 1998.

[3] Saroj Hiranwal, Dr. K.C. Roy, "Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice", International Journal of Computer Science and Communication Vol. 2, No. 2, July-December 2011, pp. 319-323.

[4] Ajit Singh, Priyanka Goyal, Sahil Batra, "An Optimized Round Robin Scheduling Algorithm for CPU Scheduling", International Journal on Computer Science and Engineering Vol. 02, No. 07, 2010, 2383-2385.

[5] Chhayanath Padhy, Dillip Ranjan Nayak, "Revamped Round Robin Scheduling Algorithm", Journal of Engineering Computers & Applied Sciences (JECAS) ISSN No: 2319-5606 Volume 3, No.4, April 2014.

[6] Taeck-Geun Kwon, Sook-Hyang Lee, and June-Kyung Rho, "Scheduling Algorithm for Real-Time Burst Traffic using Dynamic Weighted Round Robin", R&D Lab., LG Information & Communications, LTD., 1998 IEEE.

[7] Tarek Helmy**,** Abdelkader Dekdouk, "Burst Round Robin as a Proportional - Share Scheduling Algorithm", IEEEGCC, http://eprints.kfupm.edu.sa/1462.

[8] Ashkan Emami Ale Agha, Somayyeh Jafarali Jassbi," A New Method to Improve Round Robin Scheduling Algorithm with Quantum Time Based on Harmonic-Arithmetic Mean (HARM)", I.J. Information Technology and Computer Science, 2013, 07, 56-62 Published Online June 2013 in MECS (http://www.mecs-press.org/).

[9] 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 6 (10): 1831-1837, 2009 ISSN 1546-9239 © 2009 Science Publications.

[10] H.S.Behera, R. Mohanty, 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.

[11] Lipika Dutta, "Efficient Round Robin Scheduling Algorithm with Dynamic Time Slice",http://www.mecs-press.net/ijeme June 2015.

[12] Rahul Joshi, Shashi Bhushan Tyagi, "Smart Optimized Round Robin (SORR) CPU Scheduling Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering Volume 5, Issue 7, July 2015 ISSN: 2277 128X.

