© INTERNATIONAL JOURNAL FOR RESEARCH PUBLICATION & SEMINAR
ISSN: 2278-6848 | Volume: 07 Issue: 03 | April-June 2016
Paper is available at www.jrps.in | Email : info@jrps.in

An Advanced Mean Round Robin (AMRR), CPU Scheduling Algorithm

[1]PRINCY THAREJA, Research Scholar, Universal Institute of Technology,Garhi , (Hansi)
[2]SUNITA, Asst. Professor, Universal Institute of Technology,Garhi , Hansi)

ABSTRACT: The Round Robin (RR) CPU scheduling algorithm is an impartial scheduling algorithm that gives same time quantum to all processes. The selection of the time quantum is very critical as it affects the algorithm's performance. This paper suggests a new algorithm that improved on the Round Robin (RR) CPU scheduling algorithm. The proposed algorithm was implemented and benchmarked against other algorithms available in the literature. The proposed algorithm compared with the other algorithms, produces minimal average waiting time (AWT), average turnaround time (ATAT), and number of context switches (NCS). It also improves average response time. Built on these results, the proposed algorithm should be preferred over other scheduling algorithms for systems that adopt RR CPU scheduling.

© IJRPS International Journal for Research Publication & Seminar

KEYWORDS: CPU scheduling, RR and SJF Schedule Algorithms, Turnaround Time, Waiting Time, Response Time, Burst Time, Context Switching, Gantt Chart, Scheduling Criteria, Completion Time.

I. INTRODUCTION

Multiprogramming is one of the most significant aspects of operating systems. Multiprogramming became possible when virtual memory concept was introduced to the computing world. The concept of multiprogramming depends on the capability of a computer to store instructions for long-time use. The objective is to decrease CPU idle time by allowing new jobs to take over the CPU whenever the presently running job needed to wait. Process scheduling is an important part of a Multiprogramming operating system. A CPU scheduling algorithms is used for better utilization of CPU. Main goal is to increase system performance in accordance with the chosen set of conditions. Here process's state changes from ready state to running state. CPU scheduler chooses from among the processes that are ready to execute and assigns the CPU to one of them. CPU scheduling is the core of multiprogramming systems. Maximum CPU utilization can be obtained by using multiprogramming. CPU scheduling can be more complex when multiple CPUs are available.

CPU scheduling is the way by which a resource is assigned to a process or task. For scheduling task, many scheduling algorithms are used such as FCFS, SJF, RR, and Priority CPU scheduling algorithm. The processes are scheduled as per to the given arrival time, burst time and priority. The execution of processes needs number of resources such as Memory, CPU time etc. [1]. A scheduling decision implies to the theory of selecting the next process to execute. In each scheduling decision, a context switch may occur, meaning that the current process will stop executing and put back to the ready queue (or some other place) and another process will be dispatched. We define the scheduling overhead cost which includes context switching time. Since processes are switching the so CPU performance will be decreased. Scheduling algorithms are extensively used in communication networks and in operating systems to assign resources to competing processes. This research examines several popular CPU scheduling algorithms by means of analysis and comparing their results under different workloads.

© INTERNATIONAL JOURNAL FOR RESEARCH PUBLICATION & SEMINAR
ISSN: 2278-6848 | Volume: 07 Issue: 03 | April-June 2016
Paper is available at www.jrps.in | Email : info@jrps.in

II. CPU SCHEDULING ALGORITHM

The fundamental CPU scheduling algorithms are First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR) and Priority Scheduling (PS). The FCFS scheduling is the simplest CPU scheduling algorithm, which assigns CPU to the processes on the basis of their arrival time to the ready queue. The Arrived process is inserted into the rear of the ready queue and the process after execution is removed from the front of the ready queue. A longer CPU-bound process will dominate the CPU time and may force shorter CPU-bound processes to wait for long periods. In SJF, the scheduler sort processes according to minimum burst time in the ready queue, so that the process having least burst time is scheduled first. If two processes have same burst times, then FCFS is followed. Processes which require large time may wait for prolonged periods, because the CPU will execute short processes first. It has been proven as the fastest scheduling algorithm, but it suffers from one major problem:-

How do we know how long the subsequent CPU burst is going to be? [2], the priority scheduling (PS) associates each process with a priority value. The CPU is given to the process with the highest priority. If there are many processes with same priority, then FCFS will decide. Less priority processes may starve, because the CPU will execute higher priority processes at first. Round Robin (RR) is particularly designed for time-sharing systems; each process have a small unit of CPU time (time quantum). This algorithm allows a process in the ready queue to run until its time quantum finishes, and then executes the next process in the ready queue.

III. SCHEDULING CRITERIA

Different CPU scheduling algorithms have its own properties as mentioned above. The choice of a specific algorithm may favour one class of processes over another. For choice of an algorithm for a particular situation, the characteristics of various algorithms must be taken in consider [4]. Many criteria have been proposed for comparing CPU scheduling algorithms. These characteristics are used to compare and to make a significant difference in which algorithm is judged to be the best one. The criteria are the following:

Context Switch Time: Time taken in the process of storing and restoring context of a pre-empted process, so that execution can be resumed from exactly same point at a later time.

Throughput: Number of processes completed per unit time.

CPU Utilization: This is a measure of how much time CPU is busy.

Turnaround Time: Total time taken by the CPU to execute a process from time of entering.

Waiting Time: It is the total time a process has been in waiting state.

Response Time: Time at which a process get the CPU at first time from the time of entering.

So, a good scheduling algorithm should have the following features [2]:

Maximum throughput.

© INTERNATIONAL JOURNAL FOR RESEARCH PUBLICATION & SEMINAR
ISSN: 2278-6848 | Volume: 07 Issue: 03 | April-June 2016
Paper is available at www.jrps.in | Email : info@jrps.in

Maximum CPU utilization. Minimum context switches. Minimum turnaround time.

Minimum waiting time. Minimum response time.

The performance of Round Robin CPU scheduling is depends on time quantum selection, because if time quantum is large then RR will works the same as the FCFS. If the time quantum is very small then context switches will be very high. Different values of time quantum will lead to different performances and will affect the algorithm's efficiency by affecting the turnaround time, waiting time, response time and number of context switches.

In this paper, a new algorithm is proposed that calculates the time quantum dynamically, by taking the average of the available burst time of processes. This algorithm together with FCFS, SJF, RR are implemented and their results are compared based on average turnaround time, average waiting time, average response time and number of context switches. Results of the analysis shows that the proposed algorithm is very promising as it outclass other algorithms in term of average turnaround time, average waiting time, average response time and number of context switches.

IV. LITERATURE REVIEW

Various improvements to RR CPU scheduling algorithm have been proposed by many authors. These modifications can be classified as follows:

Statically Allocated Time Quantum:

Ajit et al [3] suggested an algorithm that assigns the CPU to every process in RR manner for an initial time quantum equals to t unit time. After executing first cycle, it doubles the initial time quantum (2t units) and assigns the CPU to the processes in SJF manner. It interchanges the doubling and halving of the time quantum if processes are still in the ready queue after executing any cycle.

Ishwari and Deepa [4] suggested an algorithm that assigns the CPU to every process in RR manner for one time quantum only. The CPU is then assigned to the remaining processes in the ready queue in SJF manner.

Manish and Abdul Kadir [5] suggested an algorithm that assigns the CPU to processes in RR manner. After execution of each process for one time quantum, it checks whether the remaining burst time of the currently executing process is less than the time quantum. If so, it assigns the CPU to the process for the remaining burst time, else it moves the process to the rear of the ready queue and execute next process.

Dynamically Determined Time Quantum:

Behera et al [6] suggested an algorithm that sorts the processes in the ready queue in ascending order of burst time and time quantum is calculated. To get an optimal time quantum, it takes the median of the burst time of the processes in the ready queue. The time quantum is recalculated using remaining burst time of the processes in each cycle.

© INTERNATIONAL JOURNAL FOR RESEARCH PUBLICATION & SEMINAR
ISSN: 2278-6848 | Volume: 07 Issue: 03 | April-June 2016
Paper is available at www.jrps.in | Email : info@jrps.in

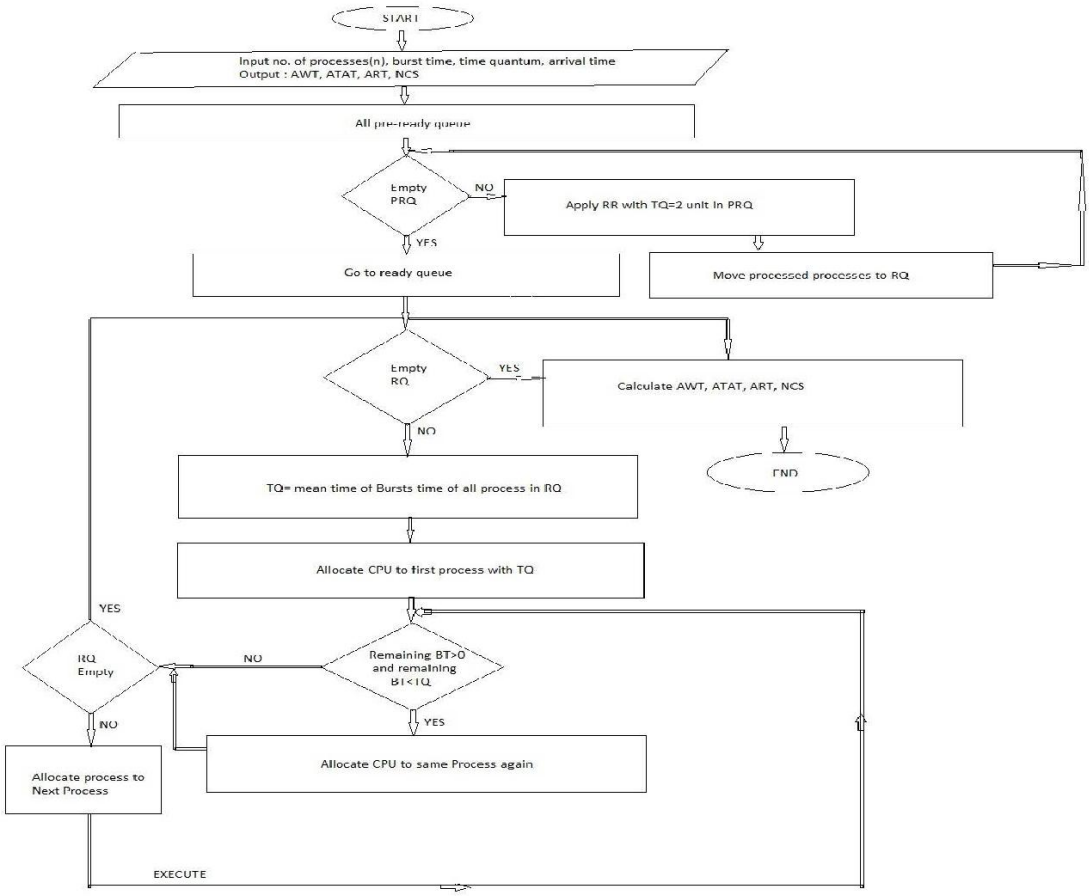
Lalit et al [7] suggested an algorithm that sort the processes in ascending order of burst time and put time quantum for RR equals to the average of the burst times of the processes. This algorithm assumes that all processes are arrived at the time t=0.
Soraj and Roy [8] suggested an algorithm that sort the processes in ascending order of burst time, and then it calculate the smart time slice (STS), which is mostly dependant on the number of processes. STS is equal to the burst time of the mid process.
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.
Ali Jbaeer Dawood [9] suggested, The time quantum T study to increase the efficiency of RR and reduce the number of Context Switching, Average Waiting Time, and Average Turn Around Time that an overhead on the system. Thus, the new method was proposed to find out the value of the time quantum T, known as Ascending Quantum and Minimum Maximum Round Robin (AQMMRR). The processes were arrange with shortest remaining burst time and calculate the T from multiply the summation of minimum and maximum BT by 80%. The simulation result shows that AQMMRR performs better than RR.

This paper proposed a modification in RR CPU scheduling algorithm by calculating the time quantum dynamically. Based on results of a simulation, use of this suggested algorithm in time sharing and real time systems will result as increase in performance of the systems by decreasing average turnaround time, average waiting time, average response time and number of context switches.

V. PROPOSED ALGORITHM

The proposed algorithm work in the following steps:
Step 1: Start
Step 2: Create two queues PREREADYQUEUE (PRQ) and READYQUEUE (RQ) Step 3: new processes will put into PREREADYQUEUE
Step 4: while (PRQ is not empty)
Perform Round Robin with time quantum equals to two unit time. Move all processed process to RQ from PRQ.
Step 5: In READYQUEUE, apply following steps.
Step 6: TQ = mean of burst time of processes present in RQ. Step 7: Allocate CPU to first process present in READYQUEUE.
Step 8: If the remaining CPU burst time of the currently running process is less than time quantum then allocate the CPU again to the currently running process for remaining CPU burst time. After completion of execution, remove the process from the ready queue and allocate CPU to next process.
Step 9: If the remaining CPU burst time of the currently running process is longer than time quantum then allocate CPU to next process in READYQUEUE.
Step 10: If a new process arrives in the system during this time, put it into PREREADYQUEUE and it will wait till current process get executed completely in READYQUEUE. Now go to step 4.
Step 11: WHILE READYQUEUE is not empty.
Step 12: Calculate AWT, ATAT, ART and NCS.
Step 13: END

© INTERNATIONAL JOURNAL FOR RESEARCH PUBLICATION & SEMINAR
ISSN: 2278-6848 | Volume: 07 Issue: 03 | April-June 2016
Paper is available at www.jrps.in | Email : info@jrps.in



Flowchart 1: Proposed Algorithm

VI. ILLUSTRATIVE EXAMPLE

We consider the queue with seven process P0, P1, P2, P3, P4, P5, P6 arriving at time 0, 2, 3, 4, 1, 6, 3 with burst time 14, 58, 18, 30, 28, 46, 7 respectively.

Process Name	Arrival Time	Burst Time
P0	0	14
P1	2	58
P2	4	18
P3	5	30
P4	1	28
P5	6	46
P6	3	7

Table 1: Process Detail

© INTERNATIONAL JOURNAL FOR RESEARCH PUBLICATION & SEMINAR
ISSN: 2278-6848 | Volume: 07 Issue: 03 | April-June 2016
Paper is available at www.jrps.in | Email : info@jrps.in

The gantt chart for FCFS, SJF, RR and AMRR is shown below:

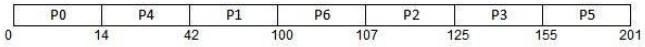


Fig 1: Gantt Chart representation of FCFS

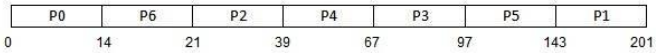


Fig 2: Gantt Chart representation of SJF

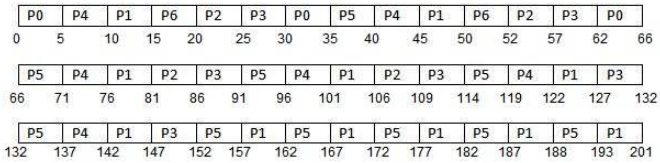


Fig 3: Gantt Chart representation of RR

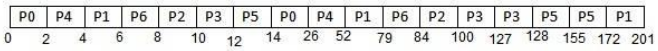


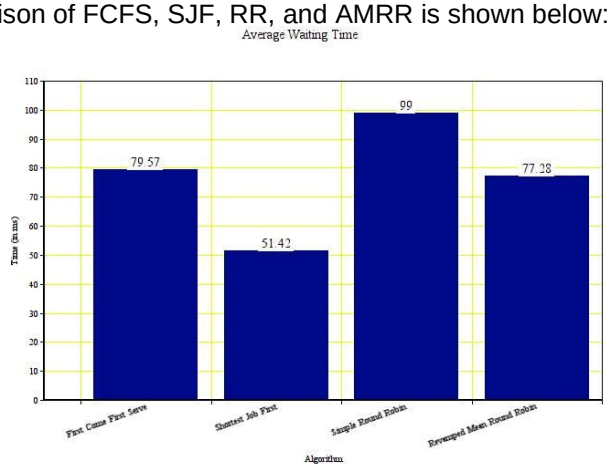
Fig 4: Gantt Chart representation of AMRR

The comparison of average waiting time, average turnaround time, response time and context switches for FCFS, SJF, RR, AMRR is shown below:

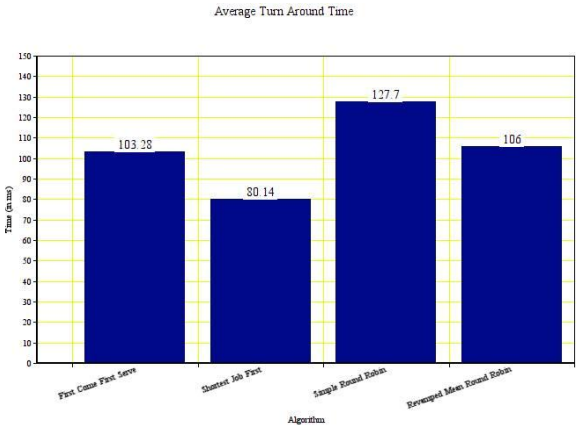
ALGORITHM	AVG WAITING TIME	AVG TURN AROUND TIME	RESPONSE TIME	NUMBER OF CONTEXT SWITCH
FCFS	74.57	103.28	77.57	6
SJF	51.42	80.14	54.42	6
RR	99	127.7	15.71	42
RMRR	77.28	106	6	14

Table 2: Comparison of AWT, ATAT, RT and NCS for FCFS, SJF, RR, AMRR

The graph for the comparison of FCFS, SJF, RR, and AMRR is shown below:

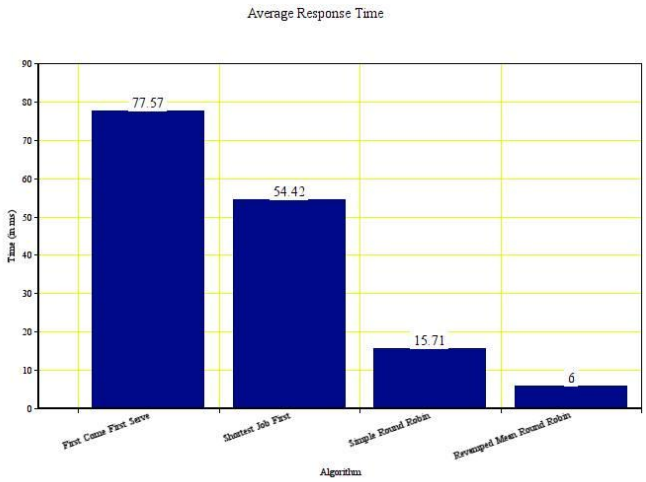


Graph 1: Comparison of Average Waiting Time

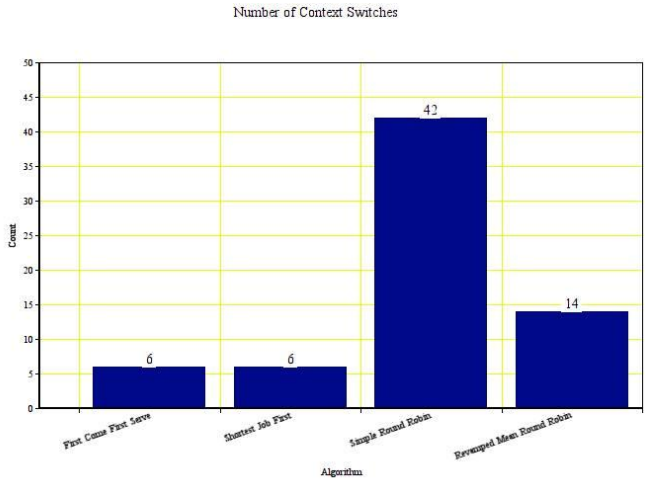


© INTERNATIONAL JOURNAL FOR RESEARCH PUBLICATION & SEMINAR
ISSN: 2278-6848 | Volume: 07 Issue: 03 | April-June 2016
Paper is available at www.jrps.in | Email : info@jrps.in

Graph 2: Comparison of Average Turn Around Time



Graph 3: Comparison of Response Time



Graph 4: Comparison of Number of Context Switches

VII. CONCLUSION

In this paper we proposed new algorithm known as Advanced Mean Round Robin (AMRR) CPU Scheduling algorithm .This algorithm is based on improvement and revised on the Simple Round Robin (RR) CPU Scheduling algorithm. Time Quantum is always an important factor for the Round Robin algorithm and always a Question arise – What is the optimal Time Quantum to be used in Round Robin Algorithm? For the effectiveness and better performance of the algorithm, result of this work provides an answer to this question by using dynamic time quantum. This proposed algorithm (AMRR) together with the RR algorithm were implemented in python and result were compared based on the based on the following scheduling criteria namely, AWT, ATAT, NCS.

Based on the results obtained, it is observed that the proposed algorithm (AMRR) is preferred for the system because it produces minimal AWT, ATAT, NCS compared to RR algorithm.

REFERENCES

1. Seltzer, M P. Chen and J outerhout, 1990.Disk scheduling revisited in USENIX. Winter technical conference. Shamim H M 1998. Operating system, DCSA-2302.
2. E.O. Oyetunji, A. E. Oluleye," Performance Assessment of Some CPU Scheduling Algorithms", Research Journal of Information Technology,1(1): pp 22-26, 2009
3. Ajit, S, Priyanka, G and Sahil, B (2010): An Optimized Round Robin Scheduling Algorithm for CPU Scheduling, International Journal on Computer Science and Engineering (IJCSE), Vol. 02, No. 07, 2383-2385, pp 2382-2385.
4. Ishwari, S. R and Deepa, G (2012): 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.
5. Manish K. M. and Abdul Kadir K. (2012): An Improved Round Robin CPU Scheduling Algorithm, Journal of Global Research in Computer Science, ISSN: 2229-371X, Volume 3, No. 6, pp 64-69.
6. Lalit, K, Rajendra, S and Praveen, S (2011): Optimized Scheduling Algorithm, International Journal of Computer Applications, pp 106-109.
7. 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).
8. Rakesh Kumar Yadav, Abhishek K Mishra, Navin Prakash, Himanshu Sharma," An Improved Round Robin Scheduling Algorithm for CPU Scheduling", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 1064-1066, 2010
9. Ishwari Singh Rajput," A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems", (IJIET)International Journal of Innovations in Engineering and Technology Vol. 1 Issue 3 Oct 2012
10. H.S. Behera, Rakesh Mohanty, Jajnaseni Panda, Dipanwita Thakur and Subasini Sahoo, "Experimental Analysis of a New Fare-Share Scheduling Algorithm with Waited Time Slice for Real Time Systems", Journal of Global Research in Computer Science, Vol. 2, No. 2, February 2011, pp. 54-60.
11. http://nptel.ac.in/courses/operating system.
12. http://en.wikipedia.org/wiki/Scheduling.

BIOGRAPHY

Princy Thareja is a student in the Computer Science Department, Universal Institute of Technology, Hansi. She is pursuing her M.Tech in Computer Science from Universal Institute of Technology. Her research interests are Operating System and Design and Analysis of Algorithm.