iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 12 **Issue:** VII **Month of publication:** July 2024

DOI: https://doi.org/10.22214/ijraset.2024.63675

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 12 Issue VII July 2024- Available at www.ijraset.com

Various Variants of Round RobinScheduling Algorithm

Roshi Bhati[1], Rahul Thambi[2], Nikhil Nerurkar[3]
Btech IT MPSTME, NMIMS) MUMBAI, INDIA

Abstract: ***In order to use the computer processor efficiently, different scheduling algorithms are used to manage the executionof multiple processes. One popular algorithm is called Round Robin (RR), where each process is given a short amount of time to run on the processor before switching to the next process inthe queue. The length of this time period, called a time quantum, is important in determining how efficient the scheduling is. If the time quantum is too long, it can increase the time it takes for a process to respond. If it is too short, it can increase the amountof time the processor spends switching between processes instead of actually executing them. In this paper, we explore different variants of the RR algorithm and compare their performancein terms of waiting time, turnaround time, and the number of times the processor switches between processes.***

I. INTRODUCTION

An operating system manages computer programs, in- cluding scheduling them to use the CPU efficiently. Dif- ferent scheduling algorithms are used, such as ”First Come First Serve”, ”Shortest Job First”, ”Fixed Priority Preemptive Scheduling”, and ”Round Robin Scheduling”. The main goal of these algorithms is to maximize processor utilization and throughput, while minimizing waiting time, turnaround time, and response time. However, context switching can lead to wasted time and memory. CPU scheduling involves various algorithms to determine the order in which processes access the CPU. Key algorithms include First Come First Serve (FCFS), where processes are scheduled in the order they arrive, and Shortest Job First (SJF), which gives the CPUto the process with the shortest burst time first. SJF can be non-preemptive or preemptive, with the latter allowinga process to be preempted if a shorter job arrives. Fixed Priority Preemptive Scheduling assigns the CPU based onprocess priority, scheduling the highest priority process first. Round Robin Scheduling assigns each process a fixed time quantum in a cyclic order, preempting the process when its time quantum expires and placing it at the end of the ready queue. The main aim of scheduling algorithms is to maximize throughput and processor utilization. Performance parameters include context switches, which store and restore the stateof preempted processes; processor utilization, which measureshow busy the processor is; throughput, defined as the number of processes completed per unit of time; waiting time, whichis the total time a process waits in the ready queue; turnaround time, the interval from submission to completion of a process; and response time, the time from submission of a request tothe first response.

II. RELATED WORK

In recent years, numerous CPU scheduling algorithms have been developed. For instance, Rami J. Matarneh [6] proposeda method where the time quantum is selected based on the median of all processes in the ready queue. If the medianis greater than 25, the time quantum is set to the median; otherwise, it is set to 25. This approach ensures that 50% ofthe remaining processes will be completed in each round.

Negi [7] introduced an approach that extends the time quantum for processes that require only a slightly more amountof time than the allocated fixed time quantum to complete.

Hiranwal et al. [8] suggested arranging processes in non- decreasing order of their burst time and using a smart time quantum for servicing the processes. If the number of pro- cesses is odd, the time quantum equals the burst time of the middle process; if even, the average burst time is used.

Dawood [9] proposed a method where the time quantumis determined by summing the maximum and minimum burst times and then multiplying the result by 0.80, based on the rationale that 80

III. DIFFERENT ROUND ROBIN TECHNIQUES:

1) Normal Round Robin scheduling algorithm assigns a fixed time slice to each process, regardless of their specific characteristics or resource requirements.

©IJRASET: All Rights are Reserved

This fixed time slice may not be optimal for all processes, as some may require more time to complete their tasks, while others may need less. Therefore, the overall systemperformance may not be optimal with a fixed time slice.

2) Adaptive Round Robin scheduling algorithm adjusts the time slice dynamically based on the characteristics and resource requirements of each process. The algorithmmonitors the behavior of the processes and adjuststhe time slice accordingly. If a process is using fewer resources, it is assigned a smaller time slice, while if a process is using more resources, it is assigned a larger time slice. This helps to ensure that each process is given enough time to complete its tasks, while also maximizing the overall system performance. That is we initially sort the processes in order of there BurstTimeand then take the median value of the burst time as the Time quantum (q).

3) Modified Additive Round Robin uses a dynamic time slice that changes according to the priority of the process and the time it has spent waiting in the ready queue.in MARR, the time slice is calculated by adding a dynamic factor to the base time slice, based on the priority and waiting time of the process. TQ = [(Burst Time of all processes) / Number of processes] + C

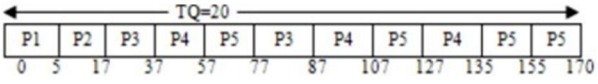
4) SARR is an adaptive algorithm that adjusts the time quantum for each process based on its behavior. SARR starts by allocating a small time quantum, and if a process finishes its work in that time, it gets a shortertime quantum in the next round. If a process requires more time, it gets a longer time quantum. This way, the algorithm can adapt to the behavior of different processes and optimize the use of CPU time. Time Quantum = Base Quantum * Round Robin Factor * Aging Factor Base Quantum = (Total Burst Time of all Processes)/(Number of Processes * RR fact) Aging Factor = (Time in Ready Queue + Base Quantum) / Base Quantum

5) AQMMRR stands for Adaptive Queue Management Multi-Round Robin scheduling algorithm. It is a CPU scheduling algorithm that aims to improve the perfor- mance of Round Robin by dynamically adjusting the time quantum for each process based on the process behavior and system load. This algorithm utilizes multiple queues to manage the processes based on their priority levels and their past execution behavior. The time quantum for each process is calculated using a dynamic formula that takes into account the priority level, the waiting time, and the execution history of the process. AQMMRR also includes a mechanism for preemptive execution of higher priority processes and ensures fairness by periodically redistributing the processes across the queues.[1,2,3] Now, as we have looked at the various forms of RoundRobin (RR), let's implement them to a problemand see the difference in their performances. Also, for abetter visual representation we will plot a bar graph forthe same.

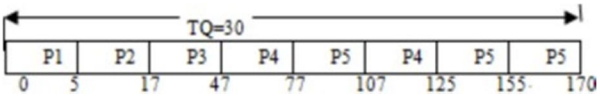
IV. IMPLEMENTATION OF THE ALGORITHMS TO A PROBLEM

PROCESS	BURST TIME
P1	8
P2	19
P3	30
P4	54
P5	83

1) *BASIC RR GANTT CHART*

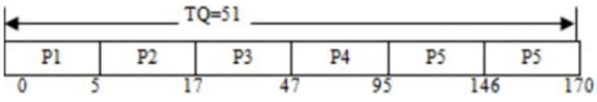


2) *ADAPTIVE RR GANTT CHART*

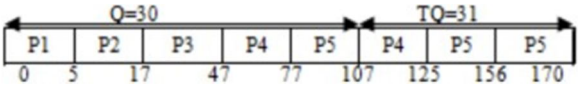


©IJRASET: All Rights are Reserved

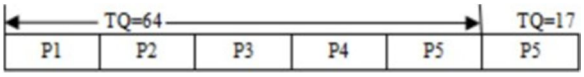
3) MARR GANTT CHART



4) SARR GANTT CHART



5) AQMMRR GANTT CHART



V. COMPARING THE RESULTS

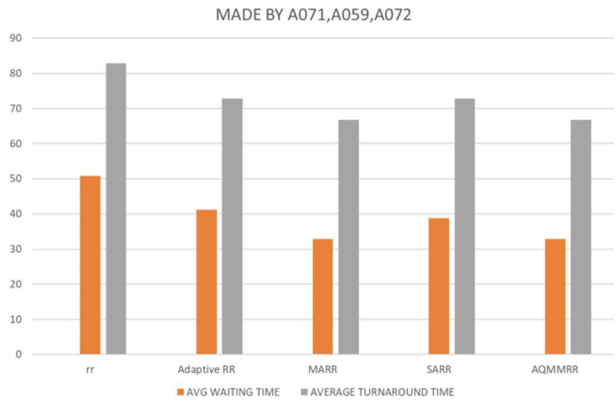
Algorithm	Time Quantum	Average Waiting Time	Average Turnaround Time
RR	20	50.8	82.8
Adaptive RR	30	41.2	72.8
MARR	51	32.8	66.8
SARR	30,31	38.8	72.8
AQMMRR	64,17	32.8	66.8

VI.

We can Conclude that MARR and AQMMRR are giving thelowest Avg.Waiting and Avg.Turnaround Time i.e. 32.8 and 66.8, respectively.

Let's combine AQMMRR and MARR which will have abetter average wait time and average turn around time.

PLOTTING :



VII. HOW THEY CAN BE COMBINED

Modified Additive Round Robin (MARR) and Adaptive Queue Management Multi-Round Robin (AQMMRR) are two modified versions of the round robin algorithm that aim to improve performance by reducing average response time and average waiting time, respectively. These two algorithmscan be combined to achieve more accurate results by using the following approach:

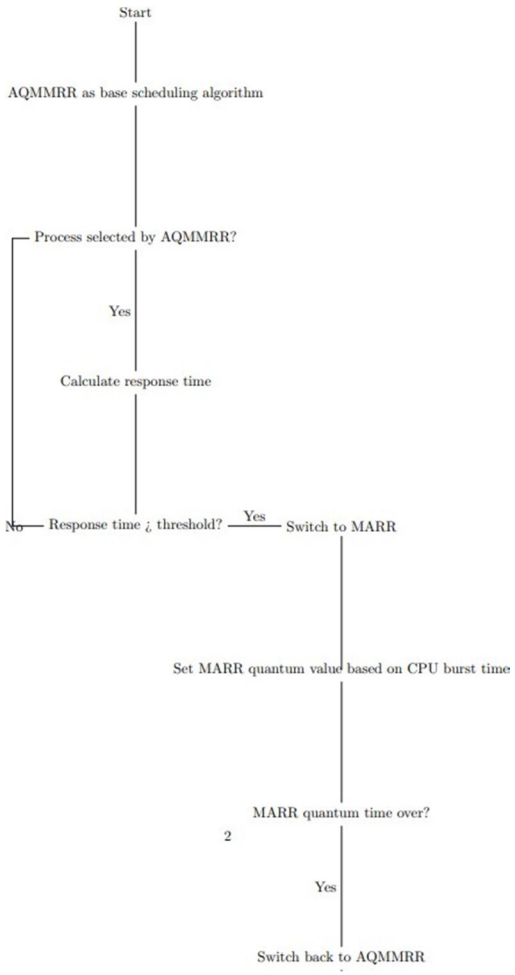
1) Start with AQMMRR as the base scheduling algorithm, as it can provide better average waiting times compared to MARR.

2) Whenever a process is selected by AQMMRR, calculate its response time. If the response time is greater than a certain threshold, switch to MARR for that process.

3) In MARR, dynamically adjust the quantum value based on the CPU burst time required by the process. Shorter processes should be given a smaller quantum value, while longer processes should be given a larger quantum value.

4) Once the MARR quantum time is over, switch back to AQMMRR for the process and continue using the approximate quantum-multiples for the remaining timeslices.

VIII. HERE IS A FLOWCHART CREATED BY US FOR ABETTER UNDERSTANDING



This approach can help to achieve more accurate results by reducing both the average waiting time and the average response time for processes. It can also provide more fairnessin the allocation of CPU time among processes of different lengths. However, it may require additional computationalresources to calculate the response time and switch between scheduling algorithms, which should be taken into consideration when implementing this approach.[3,4,5]

Implementation of the above algorithm in a python code:from collections import deque class Process:

In this implementation, the Process class represents a processto be scheduled, with attributes like its burst time and response time. The MARR AQMMRR Scheduler class has aqueue of processes, a time quantum value for AQMMRRmode, a threshold value for switching to MARR mode, aremaining time slice value, and a boolean flag to keep track of the current mode (AQMMRR or MARR)[5]. To use the scheduler, you can create an instance of MARR AQMMRR Scheduler, add some processes to it using the add process method, and then call the run method to execute the processes:

Solving an Example: Consider a system with five processesP1, P2, P3, P4, and P5, with the following burst times

- P1: 3
- P2: 5
- P3: 2
- P4: 7

- P5: 4

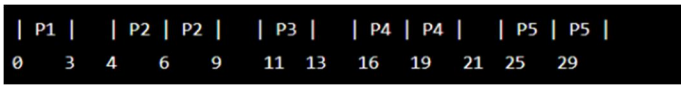
Assume that the time quantum for AQMMRR is 2 units andthe threshold for switching to MARR is 5 units.

A. Using the above algorithm, we can schedule theprocesses as follows:

- First, we start with AQMMRR and assign a timequantum of 2 units to each process.
- he first process to execute is P1, which completes itsexecution in 3 units of time.
- Next, P2 is selected by AQMMRR and is given a timequantum of 2 units.
- After executing for 4 units of time, the response time for P2 reaches the threshold of 5 units, and we switchto MARR for this process.
- n MARR, we adjust the time quantum based on the burst time of the process. Since P2 has a burst time of 5units, we give it a time quantum of 5/2 = 2.5 units.Afterexecuting for 2.5 units of time in MARR, P2's remaining burst time is 2.5 units, and we switch back toAQMMRR for this process.
- The next process to execute is P3, which completes itsexecution in 2 units of time.
- Next, P4 is selected by AQMMRR and is given a timequantum of 2 units.
- After executing for 4 units of time, the response time for P4 reaches the threshold of 5 units, and we switchto MARR for this process.
- n MARR, we adjust the time quantum based on the burst time of the process. Since P4 has a burst time of 7units, we give it a time quantum of 7/2 = 3.5 units.After executing for 3.5 units of time in MARR, P4's remaining burst time is 3.5 units, and we switch back toAQMMRR for this process.
- The last process to execute is P5, which completes itsexecution in 4 units of time.
- The total execution time for all processes using the above algorithm is 18.5 units of time. The response time for each process is:
- P1: 0 units (arrived at time 0 and completed at time 3)
- P2: 4 units (arrived at time 0 and completed at time 9)
- P3: 5 units (arrived at time 0 and completed at time 5)
- P4: 9 units (arrived at time 0 and completed at time 16
- P5: 10 units (arrived at time 0 and completed at time14)

Therefore, this algorithm provides a reasonable trade-off between average response time and average waiting time forthe processes.

Gantt Chart:



IX. CONCLUSION

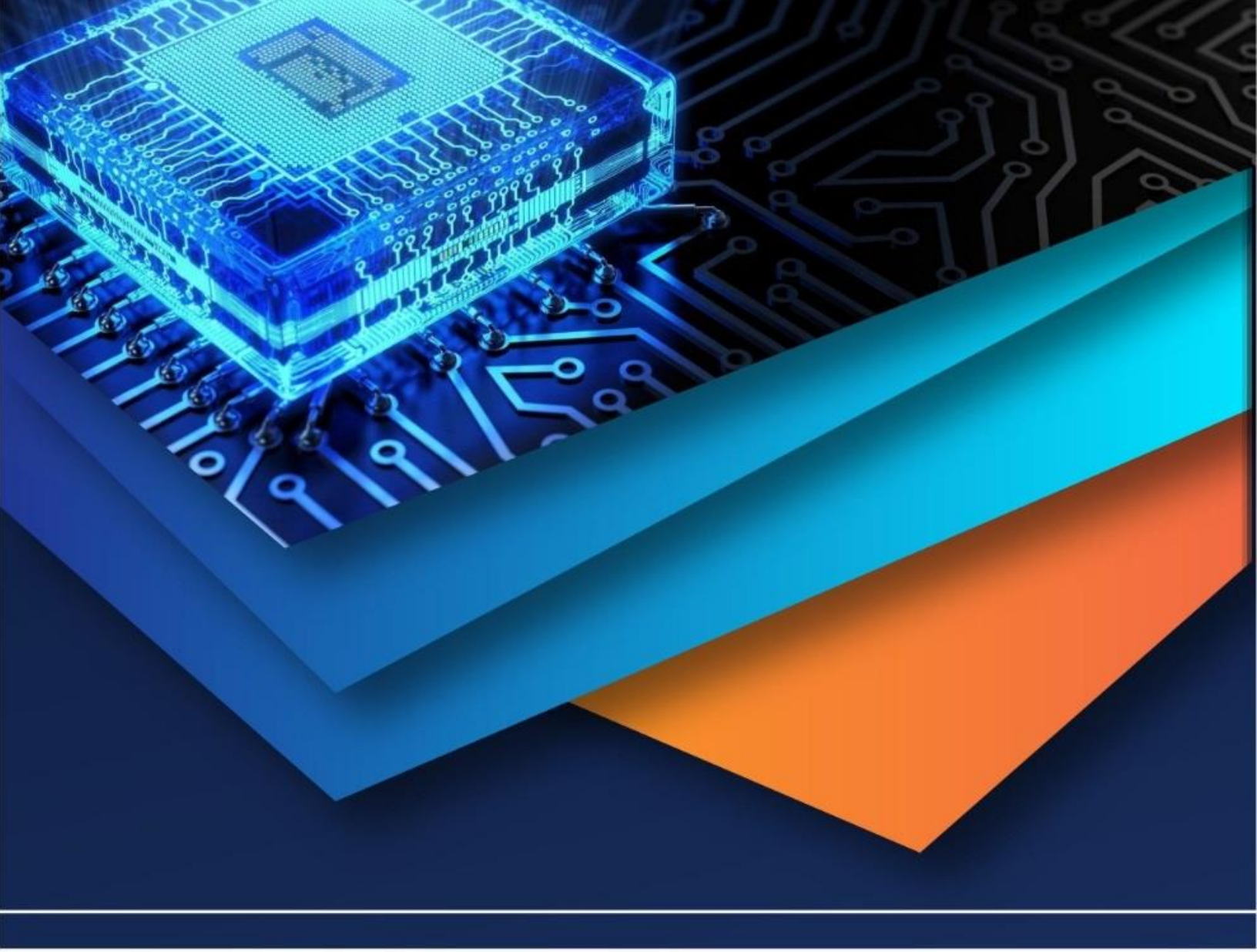
this paper explores different forms of the Round Robin (RR) algorithm, which is used to manage the execution of multiple processes and maximize processor utilization while minimizing waiting time, turnaround time, and response time. Different variants of the RR algorithm are compared based on their performance in terms of waiting time, turnaround time, and the number of times the processor switches between processes. The paper discusses Normal RR, Adaptive RR, Modified Additive RR, SARR, and AQMMRR scheduling algorithms. An implementation of these algorithms to a problem is provided, and a comparison is made based on the average waiting time and turnaround time. MARR and AQMMRR are found to give the lowest average waiting and turnaround time. Finally, the paper suggests combining MARR and AQMMRR for better performance. Hence, we have successfully found a new advanced algorithm which can be implemented for a far better average waiting and an average turn around time.

REFERENCES

[1] Silberschatz, A., Galvin, P. B., Gagne, G. (2018). Operating System Concepts (10th ed.). John Wiley Sons, Inc. Tanenbaum, A. S., Bos, H. (2014). Modern OperatingSystems (4th ed.). Prentice Hall Press.

[2] Singh, S., Mahajan, S. (2019). Comparative analysis ofround robin scheduling algorithms. International Journal of Computer Sciences and Engineering, 7(8), 649-654.

[3] Zahran, A. H. (2018). Dynamic time slice in round robinscheduling algorithm. Journal of Computer Science and Technology, 18(2), 234-240.
[4] Singh, A., Gupta, A., Saxena, N. (2017). AQMMRR: A novel adaptive queue management multi-round robin scheduling algorithm for real-time systems. Journal of Systems and Software, 125, 91-105.
[5] Sandeep Negi, Priyanka Kalra A Comparative Performance Analysis of Various Variants of Round Robin Scheduling Algorithm International Journal of Information &Computation Technology ISSN 0974-2239 Volume 4,
[6] Number 7 (2014), pp. 765-772
[7] Matarneh Rami J., "Self adjustment time quantum in round robin algorithm depending on burst time of the now running process", Department of Management InformationSystems, American Journal of Applied Sciences 6 (10):1831-1837, 2009, ISSN 1546-9239.
[8] Negi Sandeep, "An Improved Round Robin Approach using dynamic time quantum for improving average waitingtime", International Journal of Computer Applications: Vol. 69 No 14, 2013.
[9] Hiranwal Saroj, Dr. Roy K. C., " Adaptive round robin scheduling using shortest burst approach based on smart timeslice", Volume 2, No. 2, July-Dec 2011, pp:319-32.
[10] Dawood Ali Jbaeer, " Improving efficiency of roundrobin scheduling using ascending quantum and
[11] minimum-maximum burst time", Journal of University ofambar for pure science : Vol. 6: No 2, 2012,



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)