"A Paper on Modified Round Robin Algorithm"

Neha Mittal[1], Khushbu Garg[2], Ashish Ameria[3]

[1,2]*Arya College of Engineering & I.T, Jaipur, Rajasthan*

[3]*JECRC UDML College of Engineering, Jaipur, Rajasthan*

Abstract: **CPU scheduling algorithms are the main factor for performance of multitasking operating system. Where CPU is the main computer resource and round robin algorithm is mainly used as a CPU scheduling algorithm. This is the drawbacks like more perspective switches, higher average waiting time, and higher turnaround time of Simple Round Robin Scheduling algorithm which quantum repeatedly selecting according to the no of processes by arranging a processes in rising order.**

Based on the experiments and calculation that in this thesis the new modified algorithm solve the fix time quantum problem which considered a challenge for round robin algorithm. The use of this scheduling algorithm increased the performance and constancy of the operating system, which means that the system is who will adapt itself to the requirements Robin Algorithm , This reduces the drawbacks like more is used for the time sharing system In this thesis a new proposed variant of this algorithm presented, elaborate in detail, tested and verified. The new proposed algorithm called Modified Round Robin based on a new method called modified time quantum; the concept of this approach is to make the time context switches higher average waiting time, and higher turnaround time of Simple Round Robin Scheduling algorithm which is used for the time sharing system. This is used for high throughput of the system. When throughput is high then the system performance will increase. This research is useful in the future with the arrival time of the jobs.

Keywords: Central Processing Unit, scheduling, round-robin, time quantum, multitasking, throughput

I. INTRODUCTION

CPU scheduling is a essential operating is central to operating system design. When there is more than one process in the ready system task; therefore its scheduling queue or job pool waiting its turn to be assigned to the CPU, the operating system must decide through the scheduler the order of execution. Allocating CPU to a process requires careful consciousness to assure justice and avoid process starvation for CPU. Scheduling decision try to reduce the following: turnaround time, response time and average waiting time for processes and the number of context switches. Scheduling algorithms are the method by which a resource is allocated to a process or task.

CPU scheduling is the method by which a resource is allocated to a process or task and executes in different ways. For scheduling many scheduling algorithms are used like FCFS, SJF, RR, and Priority scheduling algorithm. The processes are scheduled according to the given burst time, arrival time and priority. The execution of processes used number of resources such as Memory, CPU etc. [6] .A scheduling decision refers to the theory of selecting the next process for execution. During each scheduling decision, a context switch occurs, meaning that the current process will stop its execution and put back to the ready queue and another process will be dispatched. We define the scheduling overhead cost when more context switches and all process are switching the finally CPU performance will be decreased. Scheduling algorithms are widely used in communications networks and in operating systems to allocate resources to competing tasks. This research investigates several well known CPU scheduling algorithms by means of analysis and compares their concert under different workloads.

Basic Concepts: Scheduling Criteria

Different CPU scheduling algorithms have different properties and may favor one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the different properties of the various algorithms.

CPU Utilization: We want to keep the CPU as busy as possible. CPU utilization may range from 0 to 100 percent. In a real system, it should range from 40 percent (for a lightly loaded system) to 90 percent (for a heavily used system).
Throughput: If the CPU is busy executing process, then work is being done. One calculate of work is the number of processes that are completed per time unit, called throughput. For long processes, this rate may be one process per hour; for short transactions, throughput might be 10 processes per second.

Turnaround time: From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.

Waiting Time: The CPU scheduling algorithm does not affect the amount of time during which a procedure executes or does I/O; it affects only the amount of time that a process spends waiting in the prepared queue. Waiting time is the sum of the periods spent waiting in the ready queue.

Response time: In an interactive system, turnaround time may not be the best criterion. Often, a process can produce some output fairly easy, and can continue computing new results while earlier results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced. This measure, called response time, is the time that it takes to start responding, but not the time that it takes to output that response. The turnaround time is generally incomplete by the speed of output device.

II. MOTIVATION

The aim of process scheduling is to assign the processor or processors to the put of processes in a way that meets system and user objectives such as response time, throughput or processor efficiency. Various scheduling algorithms have been future; many are well understood. This algorithm focuses to enable a quantitative comparison of the performance of the Round Robin algorithm under a range of workloads.

Process execution consists of CPU execution (CPU Burst) and I/O waits (I/O Burst)

Purpose and Scope of the Study

Keeping above mentioned inspiration in mind, we are trying to develop a new algorithm to find resourceful time slice so that our average waiting time and average turnaround time will reduce. We do so by arranging all the processes ascending order of their burst time and scheduled them according to shortest job first.

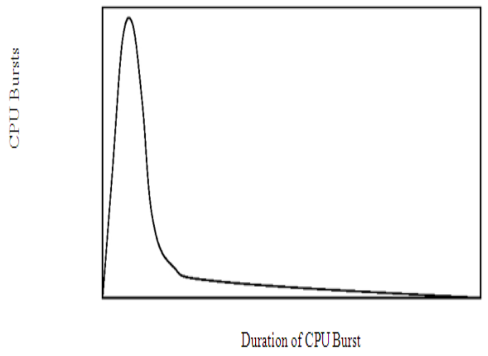

FIG 1.1: Histogram of CPU Burst times

CPU Scheduler select a process from the ready queue when the current process releases the CPU, Since the scheduler executes very regularly, it must be Very fast to avoid wasting CPU time . Very efficient (i.e., the problem of selecting a CPU algorithm for a particular system. Note that all the PCBs (Process Control Blocks) in the ready queue represent the processes in CPU burst state.

Round Robin Algorithm

The round-robin (RR) scheduling algorithm is designed especially for time-sharing systems. It is similar to FCFS scheduling, but pre-emption is further to switch between processes.

A small unit of time, called a time quantum or time slice, is distinct. A time quantum is generally from 10 to 100 milliseconds. The ready queue is treated as a circular queue. To implement RR scheduling, we keep the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process. The process may have a CPU burst of less than 1 time quantum. In this case, the process itself will discharge the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. Otherwise, if the CPU burst of the currently running process is longer than 1 time quantum. the timer will go off and will cause an interrupt to the OS.

A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the prepared queue.

Flow Chart: Round Robin Algorithm

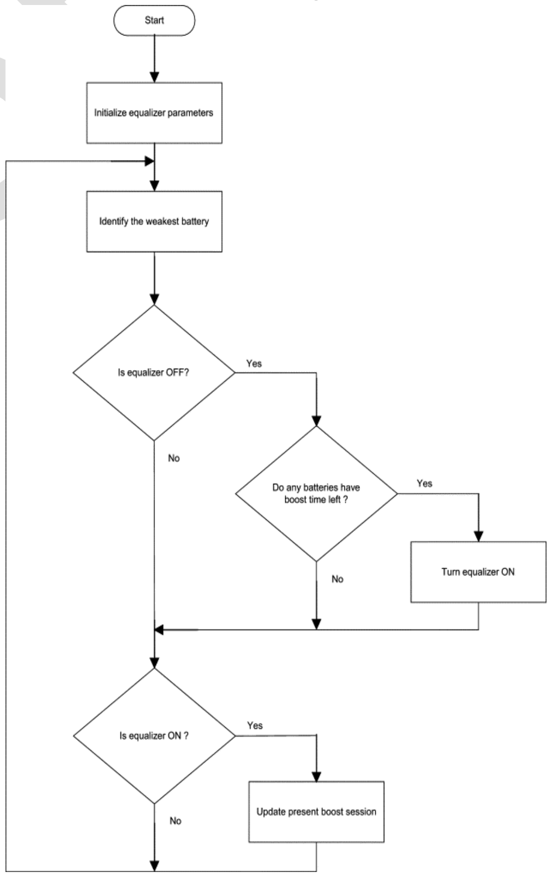
Start
Initialize equalizer parameters
Identify the weakest battery
Is equalizer OFF?
Yes
No
Do any batteries have boost time left ?
Yes
No
Turn equalizer ON
Is equalizer ON ?
Yes
No
Update present boost session

Fig 2.1 Flow chart or Round Robin Algorithm

Example RR Algorithm:

We Assume five processes arriving at time = 0, with increasing burst time (P1 = 14, P2 =45, P3 = 36, P4 = 25, P5= 77) as shown in Table 2.1. Figure-2.2 shows Gantt chart for simple round robin algorithm

Table 2.1 Example of Round Robin Algorithm

Processes	Arrival Time	Burst Time
P1	0	14
P2	0	45
P3	0	36
P4	0	25
P5	0	77

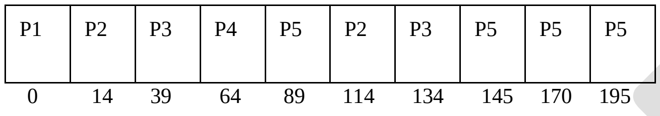

Figure: 2.2 Gantt chart for simple round robin Algorithm

AWT = 70.2

TAT = 109.6

CS = 7

III. INTRODUCTION: MODIFIED ROUND ROBIN ALGORITHM

In our proposed algorithm, the number of processes is residing in the ready queue, we assume their arrival time is assigned to zero and burst times are allocated to the CPU. The burst time and the number of processes (n) are accepted as input. Now first of all we arrange all processes in increasing order according to their given burst time and choose modified time slice according some conditions. The modified time slice will be depends on the inputting number of processes burst time. If number of processes are vary then customized time slice will be vary.

In this algorithm some number of processes and their burst time are given. And we have to find out their context switching time their turnaround time and their waiting time

In modified round robin algorithm shortest job first and round robin algorithms are mixed up. So that this algorithm can take the advantages of both algorithm. And execute more efficiently

A) STEPS FOR MODIFIED ROUND ROBIN ALGORITHM

Step 1: first check the status of ready queue
Step 2: change the status of all the processes to ready state
Step 3: Arrange all the processes in increasing order. or sort processes according to their burst time
Step 4: check whether ready queue is null or not
Step 5: If not than calculate modified time quantum

(i) Find whether the no of processes are even or odd

If no of processes are odd

Than modified time slice would be burst time of mid process

Else

Modified time slice would be the average time of burst time of all processes

Step 6: assign modified time quantum to every process
Step 7 : if burst time of process is smaller and equal to time quantum than process complete its execution s
Otherwise Repeat cycle and give the time quantum to each process
Step 8: repeat step 6 until all processes does not complete their execution

B) PSEUDO CODE FOR MODIFIED RR

```
1. Check whether ready queue status
2 All the processes are assigned into the ready queue, if ready queue is empty
3. All the processes are set in increasing order according to their burst time. Lower burst time process gets higher priority and larger burst time process get lower priority.
4. While (ready queue! = NULL)
5. Calculate modified time quantum:
If (No. of process%2= = 0)
        MTQ = average CPU burst time of all processes
        Else
MTQ = mid process burst time
6. Assign modified time quantum to the jth process:
        Px<- MTQ
        x=x+1
7. If (x< Number of process) then go to step 6.
8. If a new process is arrived update the ready queue, go to step 2.
9. End of While
10. Calculate average waiting time, turnaround time, context switches.
11. End
```

STEPS FOR MODIFIED ROUND ROBIN ALGORITHM

Step 1: first check the status of ready queue

Step 2: change the status of all the processes to ready state

Step 3: Arrange all the processes in increasing order. or sort processes according to their burst time

Step 4: check whether ready queue is null or not

Step 5: If not than calculate modified time quantum

(i) Find whether the no of processes are even or odd

If no of processes are odd

C) *Flow Chart of MRR Algorithm*

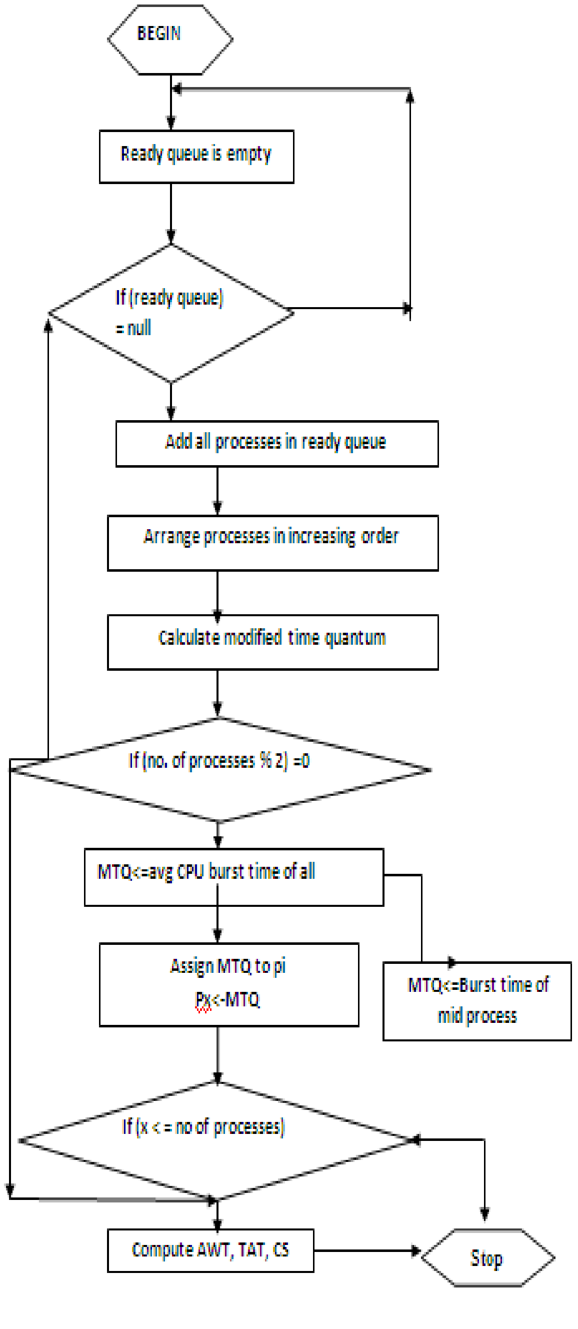
BEGIN
Ready queue is empty
If (ready queue) = null
Add all processes in ready queue
Arrange processes in increasing order
Calculate modified time quantum
If (no. of processes % 2) =0
MTQ<=avg CPU burst time of all
Assign MTQ to pi
Px<-MTQ
MTQ<=Burst time of mid process
If (x < = no of processes)
Compute AWT, TAT, CS
Stop

Fig 3.1 Flow chart of modified round robin algorithm

D) *Mathematical implementation of MRR Algorithm*

If number of processes are odd:

a) According RR mechanism for odd no of processes

We Assume five processes arriving at time = 0, with increasing burst time (P1 = 14, P2 =45, P3 = 36, P4 = 25, P5= 77) as shown in Table 3.1. Figure 3.2 shows Gantt chart for simple round robin algorithm algorithms

Example of SRR for odd no of processes

Table 3.1 Example of round robin algorithm for odd no of processes

Processes	**Arrival Time**	**Burst Time**
P1	0	14
P2	0	45
P3	0	36
P4	0	25
P5	0	77

Gantt chart for simple round robin algorithm

P1	P2	P3	P4	P5	P2	P3	P5	P5

0 14 39 64 89 114 134 145 170 195 197

Figure: 3.2 Gantt chart for simple round robin algorithm

AWT = 70.2

TAT = 109.6

CS = 7

b) *According MRR mechanism for odd no of processes:*

First of all we arrange the processes in ready queue according their given burst time in increasing order that is P1=14, P4=25, P3=36, P2=45 and P5=77 (as shown in table no 3.1) and after that we choosing the time quantum according my algorithm, the time quantum is 36. All the processes have arrival time 0. Figure-3.3 shows Gantt chart for Modified round robin algorithm algorithms

Gantt chart for MRR

P1	P4	P3	P2	P5	P2	P5	P5

Figure: 3.3: Gantt chart for MRR for even no of processes

AWT = 56.8

TAT = 96.2

CS = 6

c) Comparison of SRR and MRR for odd no. of processes

Table 3.2: Comparison of SRR and MRR for odd no of processes:

Algorithm	STS	CS	AWT	TAT	Throughput
Simple RR	25	7	70.2	109.6	Low
Modified RR	36	6	56.8	96.2	High

2) If number of processes are even:

We Assume four processes arriving at time = 0, with random burst time (P1 = 20, P2 =32, P3 = 9, P4 = 19) with time quantum =16 as shown in Table 3.3. The Figure:3.4 & 3.5 the output using RR algorithm and Modified RR algorithm.

According RR mechanism

Table 3.3 example of Simple round robin algorithm for even no. of processes

Process	Arrival Time	Burst Time (ms)
P1	0	20
P2	0	32
P3	0	9
P4	0	19

Figure: 3.4 Gantt chart for SRR algorithm for even no. of processes

P1	P2	P3	P4	P1	P2	P4

0 16 32 41 57 61 77 80

AWT = 44.78

TAT = 64.7

CS = 6

According Modified RR mechanism:

First of all I arrange the processes in ready queue according their given burst time in increasing order that is P3=9, P4=19, P1=20 and P2=32 and after that we choosing the time quantum according, Modified RR algorithm, the time quantum is the average processes burst time if the given processes are even, that is 20.

P3	P4	P1	P2	P2

0 9 28 48 68 80

Figure 3.5 Gantt chart for Modified RR for even no of processes

AWT = 21.2

TAT = 41.2

CS = 3

TABLE: 3.4 COMPARISON OF SRR AND MRR FOR EVEN NO. OF PROCESSES

Algorithm	STS	CS	AWT	TAT	Throughput
Simple RR	16	6	44.78	64.2	Low
Modified	20	3	21.2	41.2	High

IV) Performance Matrix

A) For odd no of processes:

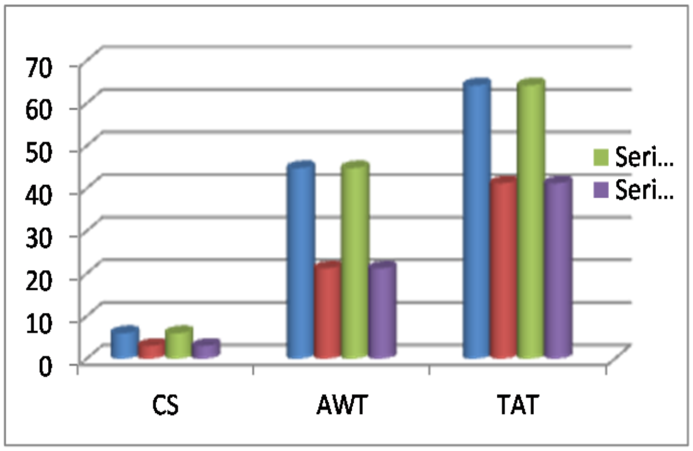

Fig 4.1 performance analysis of SRR and MRR algorithm for odd no of processes

Performance Matrix

B) For EVEN no of processes

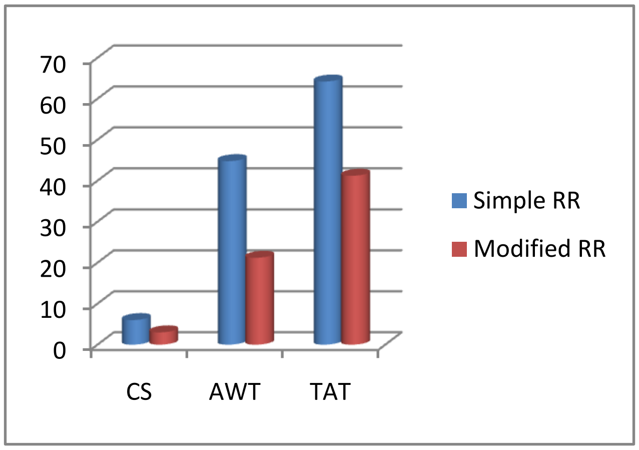

Fig 4.2 performance analysis of SRR and MRR algorithm for odd no of processes

IV. CONCLUSIONS AND FUTURE WORK

The problem of scheduling is which computer process run at what time on the central processing unit (CPU) or the processor is explored. Some CPU scheduling algorithms has been briefed

Since we expect the system to work for the user process, naturally it will be better, if it is very efficient. To make better use of the resources we go for scheduling so that work takes place in a controlled fashion. In simple words, the objective of Scheduling is to optimize the system performance. A system designer must consider a variety of factors when developing a scheduling discipline, such as what type of systems and what are user's needs. Depending on the system, the user and designer might expect the scheduler to Better Rationing of Resources for better control and co-ordination

• Fair Treatment to all the processes Provide time slice for every process.

• Efficient utilization of the CPU

• Increase the level of CPU utilization, thereby providing a very high degree of Multiprogramming

• Achieve a Balance in the usage of resources.

Provide good response, A system can accomplish these goals in several ways. The scheduler can prevent indefinite postponement of processes through aging. The scheduler can upgrade throughput by favoring processes whose requests can be satisfied quickly, or whose completion frees other processes to run.

It is concluded from the above experiments that is proposed algorithm performs better than the previous developed algorithm in terms of performance metrics such as average waiting time, average turnaround time and total number of context switches and the time and space complexity is reduced.

Future work can be enhanced to implement the proposed algorithm for adaptive and hard real time system

REFERENCES

[1]. Department of Operations Research, University of Aarhus, Ny Munkegade, Building 1530, 8000 Aarhus C, Denmark Tepper School of Business, Carnegie Mellon University, Pittsburgh PA 15213, USA "Round robin scheduling - a survey" Rasmus V. Rasmussen and Michael A. Trick
[2]. I.J. Information Technology and Computer Science, 2012, 10, 67-73 Published Online September 2012 in MECS (http://www.mecs-press.org/)
[3]. DOI: 10.5815/ijitcs.2012.10.08" Determining the Optimum Time Quantum Value In Round Robin Process Scheduling Method"
[4]. http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node125.html.
[5]. Silberchatz, Galvin and Gagne ,2003 .operating systems concepts,(6thedn, John Wiley and Sons)
[6]. Seltzer, M P. Chen and J outerhout, 1990.Disk scheduling revisited in USENIX. Winter technical conference. Shamim H M 1998. Operating system, DCSA-2302.
[7]. School of sciences and Technology. Bangladesh open university Gazipur 1705
[8]. D. M. Dhamdhere Operating Systems A Concept Based Approach , Second edition Tata McGraw-Hill, 2006
[9]. .Operating Systems Sibsankar Haldar 2009 , Pearson Education, India
[10]. Department of computer science and engineering, Barla, sambalpur, orrisa, India "Design and Performance Evaluation of a New Proposed Shorest Remaining Burst Round Robin(SRBRR) Scheduling Algorithm " Prof. Rakesh Mohanty, Prof. H.s Behera.
[11]. Yaashuwanth .C & R. Ramesh" Inteligent time slice for round robin in real time operating system, IJRRAS 2 (2), February 2010.