Volume 2, Issue 8, August 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performance Evaluation of a New Proposed Average Mid Max Round Robin (AMMRR) Scheduling Algorithm with Round Robin Scheduling Algorithm

Pallab Banerjee[1], Probal Banerjee[2], Shweta Sonali Dhal[3]
[1]*Computer Science and Engineering*
[2]*Electronics and Communication Engineering*
[3]*Electrical and Electronics Engineering*
Cambridge Institute of Technology, Ranchi University.

Abstract- Round Robin scheduling algorithm is a preemptive CPU scheduling algorithm which switches between the processes when static time quantum expires. In Round Robin scheduling algorithm the time quantum is fixed and process are scheduled in such a way that no process get CPU time more than one time quantum in one go. Round Robin scheduling algorithm is designed especially for time sharing operating system but it has its disadvantages that are its Longer Average Waiting Time, Higher Context Switches and Higher Turnaround Time. In this scheduling algorithm the main idea is to adjust the time quantum dynamically so that (AMMRR) perform better performance than Round Robin scheduling algorithm.

Keywords-Operating System, Round Robin, Average Mid Max Round Robin, Turnaround time, Waiting time, Context Switch.

I. INTRODUCTION

An operating system is a program that manages the computer hardware. It provides a platform in which user can interact with hardware and execute program in an efficient manner. Modern operating systems have become more complex, they have evolved from a single task to a multitasking environment in which processes run in a concurrent manner. In multitasking and multiprocessing environment the way the processes are assigned to run on the available CPUs is called scheduling. Allocating CPU to a process requires careful awareness to assure justice and avoid process starvation for CPU. The main goal of the scheduling is to maximize the different performance metrics viz. CPU utilization, throughput and to minimize response time, waiting time and turnaround time and the number of context switches [4].

II. PRELIMINARIES

A program in execution is called a process. The time for which a process holds the CPU is known as burst time. The time at which a process arrives for execution is its arrival time. Turnaround time is the amount of time to execute a particular process, while waiting time is the amount of time a process has been waiting in the ready queue. Time expired from the submission of a request by the process till its first response is defined as the response time. Scheduler selects a process from queues in a manner, for its execution such that the load balance is effective. In non-preemption, CPU is assigned to a process, it holds the CPU till its execution is completed. But in preemption, running process is forced to release the CPU by the newly arrived process. In time sharing system, the CPU executes multiple processes by switching among them very fast. The number of times CPU switches from one process to another is called as the number of context switches. When process enters into the main memory and are ready and waiting to execute are kept in the data structure called Ready Queue. When a process assign to the CPU, it execute or while waiting for some event to occur. The processes which are waiting for I/O request are kept in Device Queue. The Long term scheduler or job scheduler select process from job pool and load them into main memory for execution. Short term scheduler or CPU scheduler select from among the processes that are ready to execute and allocates the CPU to one of them. Medium term scheduler is used in time sharing system. The main advantage of Medium term scheduler is sometimes it remove processes from main memory and thus reduce degree of multiprogramming. Later the process can be reintroduced into memory, and its execution can be continued where it left off. This scheme is called as swapping. So, the process is swapped out, and is later swapped in, by the medium term scheduler [4].

III. SCHEDULING ALGORITHM

CPU scheduling algorithm decides which of the processes in the Ready Queue (RQ) are to be allocated to the CPU. There are many different CPU scheduling algorithms used like First Come First Serve (FCFS), Shortest Job First (SJF),Round Robin (RR),Priority scheduling algorithm and Short Remaining Time Next (STRN) algorithm. The processes are scheduled according to the given burst time, arrival time, time quantum and priority.

IV. RR SCHEDULING ALGORITHM

Round Robin (RR) is the oldest, simplest and most widely used proportional share scheduling algorithm. It is similar to FCFS scheduling, but preemption is added to switch between processes. In Round Robin algorithm a small unit of time slice are required which is called Time Quantum (TQ). The CPU scheduler goes around Ready Queue and allocates the CPU to each processes by the help of Dispatcher for a time interval of up to 1 Time Quantum(TQ)[4]. If new process arrives then it is added to the tail of Circular Queue. The CPU scheduler picks the first process from the Ready Queue sets a timer to interrupt after one Time Quantum and dispatches the process [10]. After TQ is expired, the CPU preempts the process and the process is added to the tail of the Circular Queue. If the process finishes before the end of the TQ, the process itself preempts the CPU willingly [2]. In this paper, we tried to solve the Time Quantum problem by adjusting the Time Quantum Dynamically with respect to the existed set of processes in Ready Queue.

V. PERFORMANCE METRICS

The proposed algorithm is designed to meet all scheduling criteria such as maximum CPU utilization, maximum throughput, minimum turnaround time, minimum waiting time and context switches. Here we are considering three performance criteria in each case of our experiment.

Turnaround Time (TAT)=Finish Time–Arrival Time.

Average Turnaround Time should be less

Waiting Time (WT)= Start Time- Arrival Time.

Average Waiting Time should be less.

Context Switch

The number of context Switch should be less

VI. RELATED WORK

In the last few years different approaches are used to increase the performance of Round Robin scheduling like Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice[1], Multi-Dynamic time Quantum Round Robin (MDTQRR)[5].Min-Max Round Robin (MMRR)[2], Self-Adjustment Time Quantum in Round Robin (SARR)[10], Dynamic Quantum with Re-adjusted Round Robin (DQRRR)[11],Average Max Round Robin Algorithm (AMRR)[8].

VII. PROPOSED APPROACH

Let's assume that the burst time of the processes is taken as sorted increasing order so that it will give better turnaround time and waiting time. Generally in Round Robin algorithm the performance depends upon the size of fixed or static Time Quantum (TQ). If TQ is too large then Round Robin algorithm approximate to First Come First Served (FCFS). If the Time Quantum is too small then there will be many context switching between the processes. So, our approach solved this problem by taking a dynamic TQ. Where TQ is the Average of the summation of Mid and Max process.

Mid=(Min + Max)/2

TQn=(Mid+ Max)/2

VIII. PROPOSED ALGORITHM

In our proposed algorithm, processes are already present in the Ready Queue (RQ). By default, Arrival Time (AT) is assigned to zero. The number of processes 'n' and CPU Burst Time (BT) are accepted as input and Average Turnaround Time (ATT), Average Waiting Time (AWT) and number of Context Switch (CS) are produced as output. Let TQn be the new time quantum. The pseudo code for the algorithm is presented in Figure 1 and the flowchart of the algorithm is presented in Figure 2.

1. All the processes in the Ready Queue are sorted in ascending order.

Where

// N= Number of Processes in the ready queue

// BT= Burst Time of the Processes

2. While(RQ != NULL)

// RQ= Ready Queue

//Mid= Mid of all the processes in the Ready Queue

//Min= First process in the Ready Queue having minimum Burst Time

// Max=Last process in the Ready Queue having maximum Burst Time

Mid=(Min + Max)/2

//(Use Use round off function in Mid)

TQn=(Mid+ Max)/2

//TQn=New Time Quantum

//(Use round off function in TQ)

(Remaining Burst time of the process)

//If one process is there then after calculation TQn is equal to BT itself

3. // Assign TQ to (1 to n) processes

for i=0 to N loop

{

Pi->TQn

}

End of for

//Assign TQn to all the available processes

4. Calculate the remaining Burst time of the processes.

5. If (new process arrived and BT!=0 or new process is arrived and BT==0)

then go to step1

else if (new process is not arrived and BT!=0)

then go to step 2

else

go to step 6

end of if

end of while

6. Calculate ATT,AWT,CS

//ATT=Average Turnaround time

//AWT=Average waiting time

//CS=Number of context switch

7. End

Figure 1. Pseudo code for Average Mid-Max Round Robin (AMMRR) algorithm

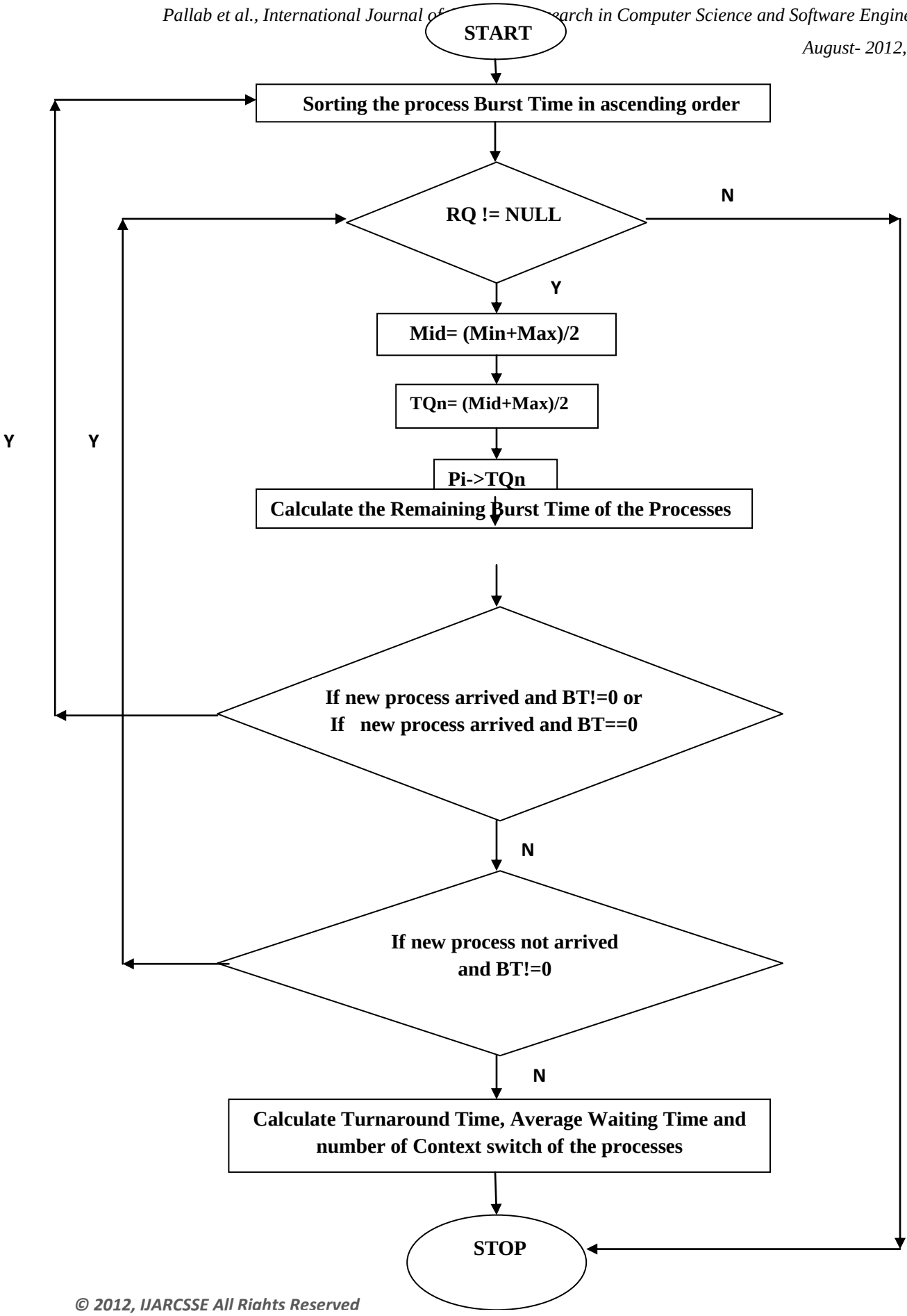

Figure 2. Flowchart of Average Mid Max Round Robin (AMMRR)

IX. ILLUSTRATION

Given the burst time of the process in unsorted sequence:(P1=57,P2=69,P3=45,P4=20,P5=85) taking arrival time=0. Initially the burst time of all the processes were sorted in ascending order which resulted in sequence : (P4=20,P3=45,P1=57,P2=69,P5=85). Then the burst time of Mid process is find which is P1=57 by using the formula Mid=(Min+Max)/2. After that TQn is calculated. Where TQn is the Average of the summation of Mid processes and Max process i.e TQn=(Mid+Max)/2. So TQn=(57+85)/2=71. After first iteration the remaining CPU burst time sequence is P4=0,P3=0,P1=0,P2=0 and P5=14. In this case, processes P4,P3,P1 and P2 are deleted from the Ready Queue. Now there is only one process left P5 with burst time. So TQn=14. After second iteration P5=0. Now there is no process in the RQ, it completes its execution and TAT, AWT and CS are calculated. In this case, TAT=134.8, AWT=79.6, CS=4.

X. EXPERIMENTAL ANALYSIS

In every case we will compare the result of the proposed Average Mid Max Round Robin (AMMRR) method with Round Robin (RR) scheduling algorithm. Here we have taken 20 as the static time quantum (TQ) for RR algorithm

CASE 1:-Let's consider five processes with Burst time (P1=15,P2=35,P3=55,P4=85,P5=95) and Arrival Time =0 as shown in the Table 1. Table 2 shows the output using RR algorithm and AMMRR algorithm. Figure 3 and Figure 4 shows Gantt chart of both RR and AMMRR algorithm respectively.

Table 1. Processes with Burst Time

Processes	Arrival Time	Burst Time
P1	0	15
P2	0	35
P3	0	55
P4	0	85
P5	0	95

Table 2: Comparison between RR algorithm and our new proposed AMMRR algorithm (CASE 1).

Algorithm	Time Quantum	Turnaround Time	Average Waiting Time	Context Switch
RR	20	173	116	15
AMMRR	75,20	144	87	6

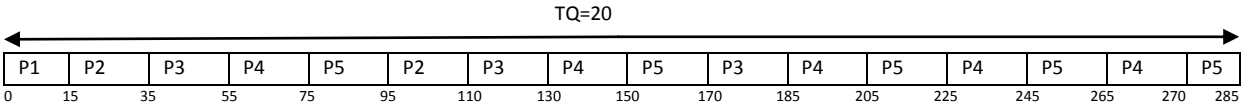

Fig.3: Gantt chart of RR from Table 1 of CASE 1.

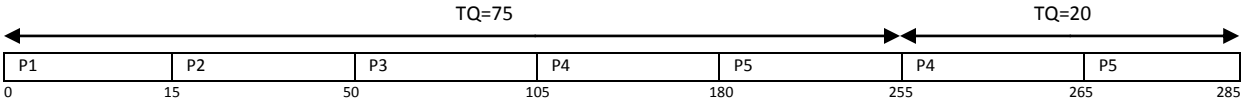

Fig.4: Gantt chart of AMMRR from Table 1 of CASE 1.

CASE 2:-Let's consider five processes with Burst time (P1=31, P2=32, P3=33,P4=34,P5=35) and Arrival Time =0 as shown in the Table 3. Table 4 shows the output using RR algorithm and AMMRR algorithm. Figure 5 and Figure 6 shows Gantt chart of both RR and AMMRR algorithm respectively

Table 3. Processes with Burst Time

Processes	Arrival Time	Burst Time
P1	0	51
P2	0	52
P3	0	53
P4	0	54
P5	0	55

Table 4: Comparison between RR algorithm and our new proposed AMMRR algorithm (CASE 2).

Algorithm	Time Quantum	Turnaround Time	Average Waiting Time	Context Switch
RR	20	237	180	14
AMMRR	54,1	104	157	4

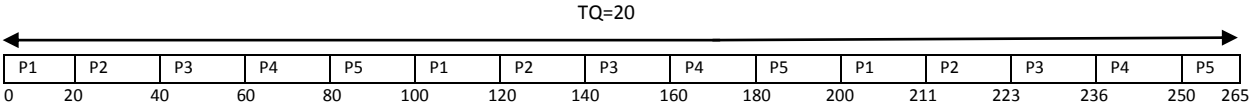

Fig.5: Gantt chart of RR from Table 3 of CASE 2.

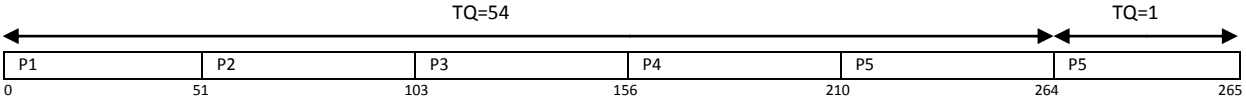

Fig.6: Gantt chart of AMMRR from Table 3 of CASE 2.

CASE 3:-Let's consider five processes with Burst time (P1=10,P2=20,P3=40,P4=80,P5=160) and Arrival Time =0 as shown in the Table 5. Table 6 shows the output using RR algorithm and AMMRR algorithm. Figure 7 and Figure 8 shows Gantt chart of both RR and AMMRR algorithm respectively

Table 5. Processes with Burst Time

Processes	Arrival Time	Burst Time
P1	0	10
P2	0	20
P3	0	40
P4	0	80
P5	0	160

Table 6: Comparison between RR algorithm and our new proposed AMMRR algorithm (CASE 3).

Algorithm	Time Quantum	Turnaround Time	Average Waiting Time	Context Switch
RR	20	134	72	11
AMMRR	100,60	114	52	4

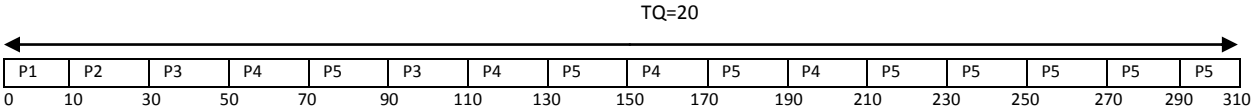

Fig.7: Gantt chart of RR from Table 5 of CASE 3.

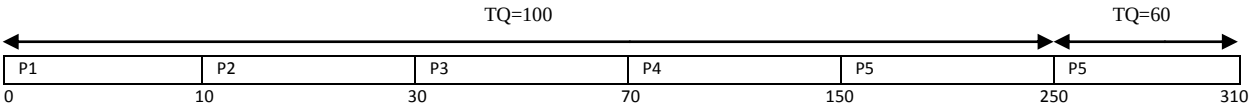

Fig.8: Gantt chart of AMMRR from Table 5 of CASE 3.

CASE 4:-Let's consider five processes with Burst time (P1=20,P2=30,P3=35,P4=50,P5=70) and Arrival Time as (P1=0,P2=5,P3=15,P4=25,P5=30) shown in the Table 7. Table 8 shows the output using RR algorithm and AMMRR algorithm. Figure 9 and Figure 10 shows Gantt chart of both RR and AMMRR algorithm respectively.

Table 7. Processes with Burst Time

Processes	Arrival Time	Burst Time
P1	0	20
P2	5	30

P3	15	35
P4	25	50
P5	30	70

Table 8: Comparison between RR algorithm and our new proposed AMMRR algorithm (CASE 4).

Algorithm	Time Quantum	Turnaround Time	Average Waiting Time	Context Switch
RR	20	112	71	10
AMMRR	53,17	84	43	4

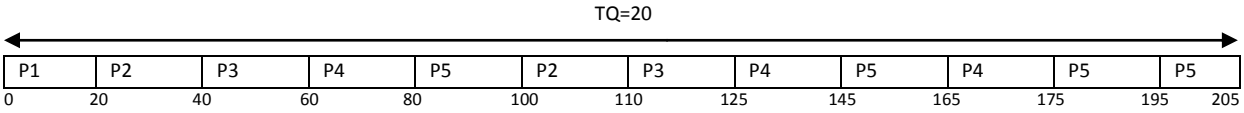

Fig.9: Gantt chart of RR from Table 7 of CASE 4.

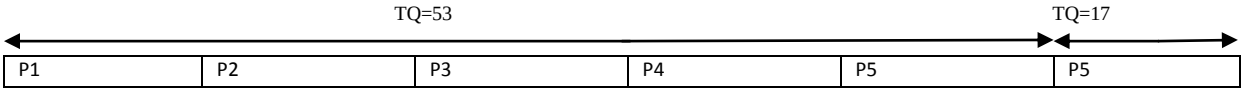

Fig.10: Gantt chart of AMMRR from Table 7 of CASE 4.

CASE 5:-Let's consider five processes with Burst time (P1=10,P2=40,P3=55,P4=80,P5=90) and Arrival Time as (P1=0,P2=5,P3=25,P4=30,P5=40) shown in the Table 9. Table 10 shows the output using RR algorithm and AMMRR algorithm. Figure 11 and Figure 12 shows Gantt chart of both RR and AMMRR algorithm respectively.

Table 9. Processes with Burst Time

Processes	Arrival Time	Burst Time
P1	0	10
P2	5	40
P3	25	55
P4	30	80
P5	40	90

Table 10: Comparison between RR algorithm and our new proposed AMMRR algorithm (CASE 5).

Algorithm	Time Quantum	Turnaround Time	Average Waiting Time	Context Switch
RR	20	145	88	13
AMMRR	73,17	119.6	48.6	6

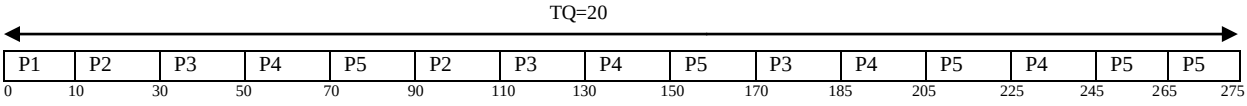

Fig.11: Gantt chart of RR from Table 9 of CASE 4.

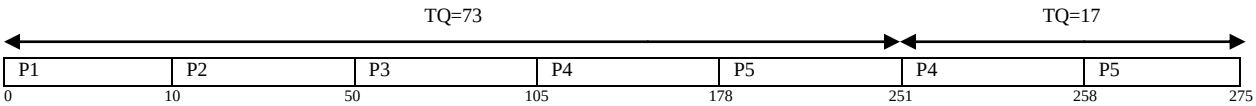

Fig.12: Gantt chart of AMMRR from Table 9 of CASE

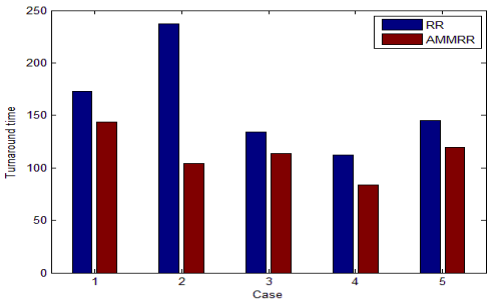

Fig.13: Comparison of average turnaround time of RR and AMMRR taking arrival time into consideration.

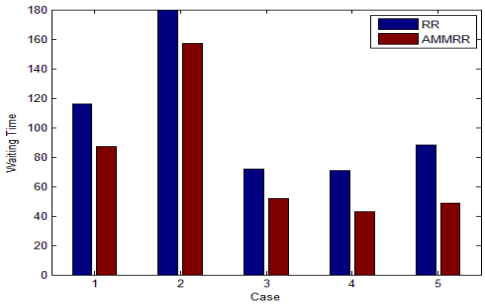

Fig.14. Comparison of average waiting time of RR and AMMRR taking arrival time into consideration.

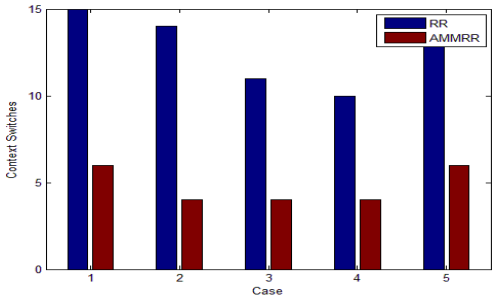

Fig.15. Comparison of Context Switches of RR and AMMRR taking arrival time into consideration.

XI. CONCLUSION

From the above study we conclude that the proposed Average Mid Max Round Robin Scheduling algorithm shows better performance as compared to Round Robin Scheduling algorithm by decreasing the Total Turnaround Time, Average Waiting Time and Number of context switching .This is achieved by increasing the Time Quantum Dynamically.

REFERENCE

[1] Sarojhiranwal and D.r. K.C.Roy"Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice".volume 2,issue 3.

[2] Sanjay Kumar Panda and Saurav Kumar Bhoi, "An Effective Round Robin Algorithm using Min-Max Dispersion Measure" ISSN : 0975-3397 ,Vol. 4 No. 01, January 2012.

[3] "Tanebaun, A.S., 2008" Modern Operating Systems. 3rd Edn., Prentice Hall, ISBN: 13:9780136006633, pp: 1104.

[4] "Silberschatz, A., P.B. Galvin and G. Gagne, 2008" Operating Systems Concepts. 7th Edn., John Wiley and Sons, USA., ISBN: 13: 978-0471694663, pp: 944.

[5] H. S. Behera, Rakesh Mohanty, Sabyasachi Sahu and Sourav Kumar Bhoi." Comparative performance analysis of multi-dynamic time quantum round robin (mdtqrr) algorithm with arrival time", ISSN : 0976-5166, Vol. 2, No. 2, Apr-May 2011.

[6] "Tarek Helmy, Abdelkader Dekdouk" Burst Round Robin: As a Proportional-Share Scheduling Algorithm, IEEE Proceedings of the fourth IEEE-GCC Conference on towards Techno-Industrial Innovations, pp. 424-428, 11-14 November,2007

[7] "Yaashuwanth .C & R. Ramesh" Inteligent time slice for round robin in real time operating system, IJRRAS 2 (2), February 2010.

[8] Pallab banerjee, probal banerjee, shweta sonali dhal,"Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm using static Time Quantum",IJITEE,ISSN: 2278-3075, Volume-1, Issue-3, August 2012.

[9] J. Nieh, C. Vaill and H. Zhong, "Virtual-Time Round-Robin: An O(1) Proportional Share Scheduler", Proceedings of the USENIX

[10] R. J. Matarneh, "Seif-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Proceses", American Journal of Applied Sciences 6 (10), pp. 1831-1837, 2009.

[11] H. S. Behera, R. Mohanty, and D. Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis," vol. 5, no. 5, pp. 10-15, August 2010.

[12] A. Bhunia, "Enhancing the Performance of Feedback Scheduling", IJCA, vol. 18, no. 4, pp. 11-16, March 2011.

[13] "Prof. Rakesh Mohanty, Prof. H. S. Behera, Khusbu Patwari, Manas Ranjan Das, Monisha Dash,Sudhashree" Design and Performance Evaluation of a New Proposed Shortest Remaining Burst RoundRobin (SRBRR) Scheduling Algorithm, Am. J. Applied Sci., 6 (10): 1831-1837, 2009.

