Dynamic Time Quantum based Round Robin CPU Scheduling Algorithm

Yosef Berhanu
Department of Computer Science
University of Gondar
Ethiopia

Abebe Alemu
Department of Computer Science
University of Gondar
Ethiopia

Manish Kumar Mishra
Department of Computer Science
University of Gondar
Ethiopia

ABSTRACT
In multiprogramming environment, CPU scheduling is the process to determine the most efficient way to service the requests of the processes waiting in the ready queue to be executed. The CPU scheduling algorithms focus on maximizing CPU utilization by minimizing waiting time, turnaround time and number of context switches for a set of processes. In time shared systems the preferred choice is Round Robin (RR) CPU scheduling in which performance of the system depends on the choice of the time quantum. This paper presents a dynamic time quantum based Round Robin CPU scheduling algorithm to enhance the CPU performance using the features of an Improved Round Robin (IRR) and an improved Round Robin CPU scheduling algorithm with varying time quantum (IRRVQ). The experimental and simulation results show that the proposed algorithm is proven better than the RR, IRR and IRRVQ in terms of decreasing the average waiting time, average turnaround time and number of context switches.

General Terms
CPU Scheduling, Operating System

Keywords
CPU Scheduling, Round Robin, Dynamic Time Quantum, Waiting Time, Turnaround Time

1. INTRODUCTION
In multiprogramming environment multiple processes are being kept in memory for maximum utilization of CPU [1]. Switching CPU among waiting processes in the ready queue and running some process all the time can maximize the CPU utilization [2]. The CPU scheduling algorithms focus on maximizing CPU utilization by minimizing waiting time, turnaround time and number of context switches for a set of processes. This study focuses on improving the efficiency of Round Robin CPU scheduling algorithm by minimizing waiting time, turnaround time and number of context switches using dynamic quantum size.

1.1 Performance Parameters
A process can compete for the CPU as soon as it is loaded from input queue to the ready queue. CPU should be busy all the time to optimize the systems performance. CPU is allocated to a waiting process from the ready queue whenever CPU is free [2]. The total CPU time needed by a process to complete execution is known as burst time of the process. The time a process loaded from input queue to the ready queue is known as arrival time of the process. When a process loaded into the ready queue, it may have to wait for some time before CPU is allocated to it. The total time a process is waiting for the CPU in the ready queue is known as waiting time of the process. The sum of burst time and waiting time is known as turnaround time of the process. When CPU becomes idle, it switches to another process waiting in the ready queue. This process is known as context switch. In the proposed dynamic time quantum based Round Robin (DTQRR) CPU scheduling algorithm, three parameters namely waiting time, turnaround time and number of context switches have been considered for performance evaluation. Our focus is to minimizing waiting time, turnaround time and number of context switches using dynamic quantum size by scheduling the processes from the ready queue effectively.

1.2 CPU Scheduling Algorithms
CPU scheduling algorithms allocate CPU to the processes waiting in the ready queue to be executed. Different CPU scheduling algorithms have been proposed by the research scholars. The process that loaded first from input queue to the ready queue gets the CPU first in First-Come-First-Served (FCFS) CPU scheduling algorithm. In this non-preemptive approach, CPU is occupied by one process until it finishes execution or terminates [2]. Since the average waiting time is quite long in FCFS, it is not suitable for the time sharing systems. Shortest Job First (SJF) CPU scheduling assigns the CPU to the process having minimum burst time. Compared to FCFS, SJF gives better performance in terms of reducing average waiting time and turnaround time. SJF is also not suitable for the time sharing systems since it does not give fair CPU time to the waiting processes in the ready queue. A process with highest priority gets the CPU time first in the Priority scheduling algorithm. Priority scheduling algorithm is also not considered appropriate for time sharing systems because of the same reason. Round Robin (RR) CPU scheduling is considered suitable for time sharing systems. A fixed time slice of the CPU is assigned to each process waiting for execution in the ready queue. This algorithm gives equal preference to all waiting processes. The performance of RR scheduling depends on time quantum. Time quantum is the CPU time that is assigned to each process. If time quantum is very large, RR behaves like FCFS scheduling. If it is too short, number of context switches increased. Time quantum affects the processes waiting time, turnaround time, response time and number of context switches. In this paper we have proposed a new algorithm that combined the approaches proposed in IRR [3] and IRRVQ [4] to reduce the waiting time, turnaround time and number of context switches.

2. RELATED WORK
Several CPU scheduling algorithms have been proposed by research scholars for improving the system performance. We have reviewed some CPU scheduling algorithms that are relevant to our study. A fixed time quantum value is allocated

to the processes waiting in the ready queue, only in first cycle and then SJF is used to select next process for execution in "An Improved Round Robin Scheduling Algorithm for CPU Scheduling" [1]. The CPU is allocated to the first process from the ready queue for a time interval of up to one time quantum in "An Improved Round Robin (IRR) CPU Scheduling Algorithm" [3]. After completion of process's time quantum, the remaining CPU burst time of the currently running process is compared with the time quantum. If the burst time of the currently running process is less than one time quantum, the CPU is again allocated to the same process so that it can finish execution and removed from the queue. It reduces the waiting time of the process in the ready queue and hence improves the performance. "An improved Round Robin CPU scheduling algorithm with varying time quantum (IRRVQ)" combines the features of SJF and RR scheduling algorithms with varying time quantum [4]. In this approach initially the processes in the ready queue are arranged in the ascending order of their remaining burst time. CPU is allocated to the processes using RR scheduling with time quantum value equal to the burst time of first process in the ready queue. After each cycle, processes in the ready queue are arranged in the ascending order of their remaining burst time and CPU is allocated to the processes using RR scheduling with time quantum value equal to the remaining burst time of first process in the ready queue. In "Self-Adjustment Time Quantum in Round Robin Algorithm" [5], time quantum is continuously adjusted according to the remaining burst time of the processes. The job mix order for the algorithm in [5] is used in "Dynamic Quantum with Readjusted Round Robin Scheduling Algorithm" [6]. A robust time quantum value has been proposed in [7] after arranging the process in the ascending order and calculating the average of minimum and maximum burst time of the processes waiting in the ready queue. In "Burst Round Robin (BRR)", a new weighting technique is introduced for CPU Schedulers [8]. The processes having smaller burst time are given more CPU time, so that these processes can be removed from the ready queue in a short time span. Debashree Nayak et. al. [9] did the similar work as [5] [6]. After every cycle of execution, the optimal time quantum is assigned to each process waiting in the ready queue. Optimal time quantum is the average of highest CPU burst time and the median. In the "New Method of Adaptive CPU Scheduling using Fonseca and Fleming's Genetic Algorithm" [10], the running processes are scheduled based on three parameters of CPU. These parameters are burst time, I/O service time, and priority of processes. A "New fare-share scheduling with weighted time slice" [11] assigns a weight to each process waiting in the ready queue. The process having the minimum burst time is assigned the highest weight. Weighted time slice method is used to calculate the time quantum dynamically. "A new dynamic Round-robin and SRTN algorithm using variable original time slice and dynamic intelligent time slice for soft real time system" [12] calculates the original time slice suited to the burst time of each processes and then dynamic ITS (Intelligent Time Slice) is found out in conjunction with the SRTN algorithm [2]. The scheduling algorithm "A New Proposed Two Processor Based CPU Scheduling Algorithm with Varying Time quantum for Real Time Systems" [13] uses two processors, one processor is used to execute CPU-intensive processes only and the other processor is used to execute I/O-intensive process. This gives better result in a two processor environment than [6]. "A New Round Robin based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average" [14] is using dynamic time quantum equal to the average burst time of the processes waiting in the ready queue after every cycle. In the algorithm, "Fair Priority Round Robin with Dynamic Time Quantum" [15], the processes are scheduled for execution from ready queue by giving importance to both the user priority and shortest burst time priority. In this approach, deciding factor for individual time quantum for each process is based on both user priority and the burst time priority. "An Additional Improvement in Round Robin (AAIRR) CPU Scheduling Algorithm" [16] propose an improvement in the conventional RR and IRR [3] CPU scheduling. This approach is similar to the IRR but it chooses the process for execution from the ready queue whose remaining burst time is shortest. "A New Improved Round Robin (NIRR) CPU Scheduling Algorithm" [17] is again an improvement of IRR [3] which uses two queues, ARRIVE queue and REQUEST queue. It is giving significance performance improvement compared to IRR. "An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum" [18] uses the dynamic time quantum assigned to the processes waiting in the ready queue. The processes are first arranged in the ascending order of their burst time and the dynamic time quantum is set equal to the mean of the burst times of the processes after every cycle of execution. They compare their results with some existing algorithms and found that the proposed algorithm provides better performance metrics.

3. PROPOSED DTQRR CPU SCHEDULING ALGORITHM

The proposed dynamic time quantum based Round Robin (DTQRR) CPU scheduling algorithm takes the advantage of the approaches proposed in IRR [3] and IRRVQ [4]. It combines these two approaches to reduce the waiting time, turnaround time and number of context switches

Initially the processes in the ready queue are arranged in the ascending order of their remaining burst time. Time quantum is assigned equal to the burst time of first process in the ready queue. CPU is allocated to the first process from the ready queue for a time interval of 1 time quantum. After completion of currently running process's time quantum, the remaining burst time is checked. If it is less than 1 time quantum, CPU is allocated again to the same process for remaining burst time. In this case this process finishes execution and removed from the ready queue. The scheduler then proceeds to the next process in the ready queue. Otherwise, if the remaining burst time of the currently running process is more than 1 time quantum, the running process is put at the tail of the ready queue. The CPU scheduler then selects the next process in the ready queue. After each cycle, processes in the ready queue are arranged in the ascending order of their remaining burst time and CPU is allocated to the processes using RR scheduling with time quantum equal to the burst time of first process in the ready queue.

Following is the proposed DTQRR CPU scheduling algorithm

1. Make a ready queue RQUEUE of the processes submitted for execution.
2. DO steps 3 to 9 WHILE queue RQUEUE becomes empty.
3. Arrange the processes in REQUEUE in the ascending order of their remaining burst time.
4. Set the time quantum value equal to the burst time of first process from RQUEUE.
5. Allocate CPU to first process from RQUEUE for a time interval of up to 1 time quantum. Remove the

currently running process from RQUEUE, since it has finished execution and the remaining burst time is zero.

6. Select the next process from RQUEUE, and allocate CPU for a time interval of up to 1 time quantum.
7. IF the currently running process has finished execution and the remaining CPU burst time of the currently running process is zero, remove it from RQUEUE.
8. IF the remaining CPU burst time of the currently running process is less than 1 time quantum THEN allocate CPU again to the currently running process for remaining CPU burst time. Remove the currently running process from RQUEUE, since it has finished execution and the remaining burst time is zero. ELSE put it at the tail of RQUEUE.
9. REPEAT steps 6 to 8 for each process in RQUEUE.

3.1 Illustration

For illustration purpose, a ready queue with four processes P1, P2, P3 and P4 has been considered. The processes are arriving at time 0 with burst time 19, 9, 25 and 12 respectively. According to our proposed algorithm, the processes P1, P2, P3 and P4 are arranged in the ascending order of their burst time in the ready queue which gives the sequence P2, P4, P1 and P3. The time quantum value is set equal to the burst time of first process in the ready queue i.e. TQ = 9. CPU is allocated to the first process P2 from the ready queue for a time quantum of 9 milliseconds (ms). Process P2 has finished execution and remaining burst time is 0. Hence it is removed from the ready queue. Now CPU is allocated to the next process P4 from the ready queue for a time quantum of 9 ms. After executing P4 for 9 ms, the remaining CPU burst time of P4 is 3 ms. Since the remaining CPU burst time of P4 is less than the TQ, CPU will be allocated again to P4 for a time interval of 3 ms. P4 has finished execution, it will be removed from the ready queue. CPU is allocated to the next process P1 from the ready queue for a time quantum of 9 ms. Since the remaining CPU burst time of P1 is 10 which is more than the TQ, CPU will be allocated to the next process P3 from the ready queue for a time quantum of 9 ms. Remaining CPU burst time of P3 is 16 which is more than the TQ, it will not get CPU time again in the first cycle. After first cycle, the remaining burst time for P2, P4, P1 and P3 are 0, 0, 10 and 16 respectively. The remaining processes P1 and P3 from the ready queue will be arranged in the ascending order of their remaining burst time which gives the sequence P1 and P3. The time quantum value will be updated and it will be set equal to the burst time of first process in the ready queue i.e. TQ = 10. CPU is allocated to the first process P1 from the ready queue for a time quantum of 10 ms. Process P1 has finished execution and remaining burst time is 0. Hence it is removed from the ready queue. Now CPU is allocated to the next process P3 from the ready queue for a time quantum of 10 ms. After executing P3 for 10 ms, the remaining CPU burst time of P3 is 6 ms. Since the remaining CPU burst time of P3 is less than the TQ, CPU will be allocated again to P3 for a time interval of 6 ms. Process P3 has finished execution and remaining burst time is 0. Hence it is removed from the ready queue.

The waiting times are 30, 0, 40 and 9 for the processes P, P2, P3 and P4 respectively. The average waiting time is 19.75 ms. The average turnaround time is 36 ms and the number of context switch is 5.

With the same set of process with same arrival and CPU burst times, the average waiting time is 25 ms, average turnaround time is 41.25 ms and number of context switch is 8 in IRRVQ algorithm.

With the same set of process with same arrival and CPU burst times, the average waiting time is 26.25 ms, average turnaround time is 41.5 ms and number of context switch is 8 in IRR algorithm for time quantum 5 ms. For time quantum 10 ms, IRR gives average waiting time 24.25 ms, average turnaround time 40.5 ms and number of context switch 4.

With the same set of process with same arrival and CPU burst times, the average waiting time is 33.75 ms, average turnaround time is 50 ms and number of context switch is 12 in RR algorithm for time quantum 5 ms. For time quantum 10 ms, RR gives average waiting time 31.75 ms, average turnaround time 48 ms and number of context switch 7.

Above illustration clearly shows that the proposed DTQRR CPU scheduling algorithm is giving better performance than the IRRVQ, IRR and RR CPU scheduling algorithms in terms of reducing average waiting time, average turnaround time and number of context switches.

4. EXPERIMENTAL ANALYSIS

4.1 Assumptions

For performance evaluation, we have assumed that all the processes joining the ready queue are having equal priority in uniprocessor environment. The number of processes and their respective burst times are known before submitting the processes for the execution. The CPU overhead has not been taken into account for arranging the processes in ascending order in the ready queue. All processes submitted for execution are CPU bound i.e. no process is I/O bound. The burst time and time quantum are measured in milliseconds (ms).

4.2 Experiments Performed

Two different cases have been taken for performance evaluation of our proposed DTQRR algorithm. In Case I, CPU burst time is in random orders and processes arrival time is assumed zero. In Case II, CPU burst time is in random orders and processes arrival time is assumed non zero. The CPU burst time in ascending or descending orders have not been considered since it gives the same result as the CPU burst time in random orders.

4.2.1 Case I – Zero Arrival Time

In this case arrival time has been considered zero and CPU burst time has been taken in random orders. A ready queue with five processes P1, P2, P3, P4, and P5 has been considered as shown in table 1.

Table 1. Processes with their arrival and burst time (Case I)

Process	Arrival Time	Burst Time
P1	0	12
P2	0	28
P3	0	8
P4	0	25
P5	0	15

The comparison result of RR, IRR, IRRVQ and proposed DTQRR is shown in Table 2. Figure 1 and Figure 2 show the Gantt chart representation of RR with time quantum 5 and 10 respectively. Figure 3 and Figure 4 show the Gantt chart representation of IRR with time quantum 5 and 10 respectively. Figure 5 shows the Gantt chart representation of IRRVQ with dynamic time quantum. Figure 6 shows the Gantt chart representation of our proposed DTQRR with dynamic time quantum.

Table 2. Comparison of RR, IRR, IRRVQ and DTQRR (Case I)

Algorithm	Time Quantum (TQ in ms)	Average Waiting Time (ms)	Average Turnaround Time (ms)	Number of Context Switches
RR	5, 10	47.6, 47.2	65.2, 64.8	18, 10
IRR	5, 10	40.2, 34.0	57.8, 51.6	15, 6
IRRVQ	8, 4, 3, 10, 3	37.2	54.8	13
DTQRR	8, 17	26.2	43.8	6

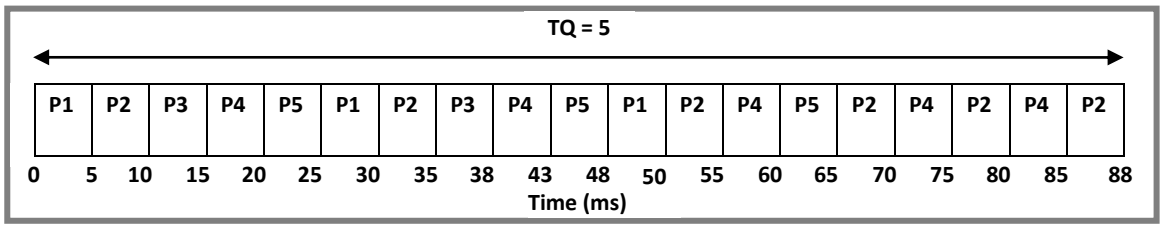

Fig 1: Gantt chart representation of RR with TQ = 5 (Case I)

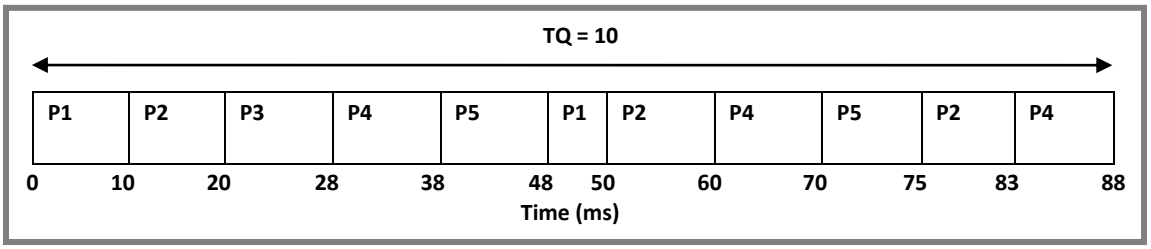

Fig 2: Gantt chart representation of RR with TQ = 10 (Case I)

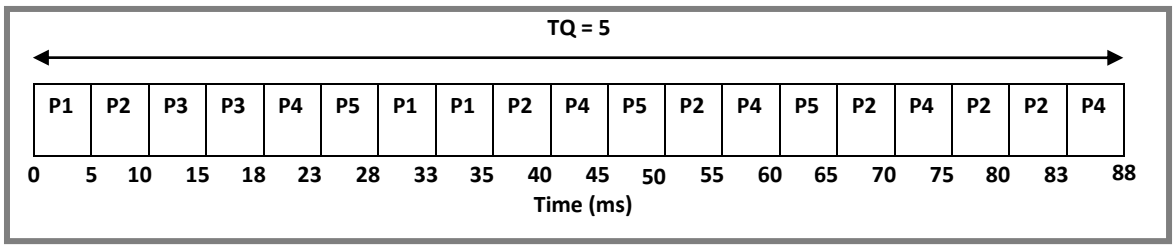

Fig 3: Gantt chart representation of IRR with TQ = 5 (Case I)

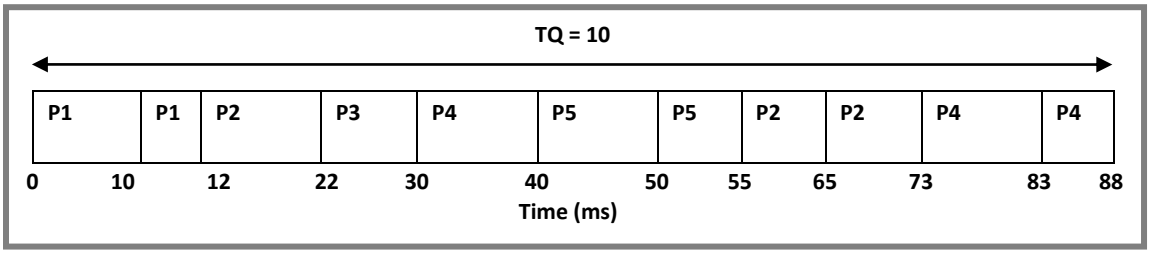

Fig 4: Gantt chart representation of IRR with TQ = 10 (Case I)

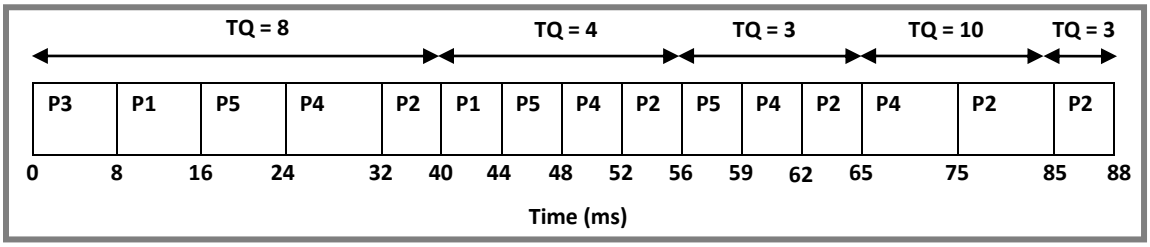

Fig 5: Gantt chart representation of IRRVQ with dynamic TQ (Case I)

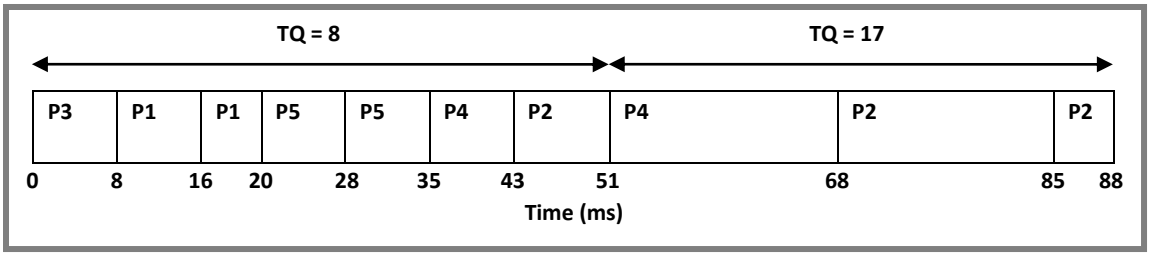

Fig 6: Gantt chart representation of DTQRR with dynamic TQ (Case I)

4.2.2 Case I – Non - Zero Arrival Time

In this case arrival time has been considered non-zero and CPU burst time has been taken in random orders. A ready queue with five processes P1, P2, P3, P4, and P5 has been considered as shown in table 3.

Table 3. Processes with their arrival and burst time (Case II)

Process	Arrival Time	Burst Time
P1	0	9
P2	3	21
P3	6	7
P4	11	29
P5	15	16

The comparison result of RR, IRR, IRRVQ and proposed DTQRR is shown in Table 4. Figure 7 and Figure 8 show the Gantt chart representation of RR with time quantum 5 and 10 respectively. Figure 9 and Figure 10 show the Gantt chart representation of IRR with time quantum 5 and 10 respectively. Figure 11 shows the Gantt chart representation of IRRVQ with dynamic time quantum. Figure 12 shows the Gantt chart representation of our proposed DTQRR with dynamic time quantum.

Table 4. Comparison of RR, IRR, IRRVQ and DTQRR (Case II)

Algorithm	Time Quantum (TQ in ms)	Average Waiting Time (ms)	Average Turnaround Time (ms)	Number of Context Switches
RR	5, 10	35.0, 29.0	51.4, 45.4	17, 9
IRR	5, 10	25.0, 23.0	41.4, 39.4	12, 6
IRRVQ	9, 7, 5, 8	23.6	40.0	8
DTQRR	9, 12	22.2	38.6	6

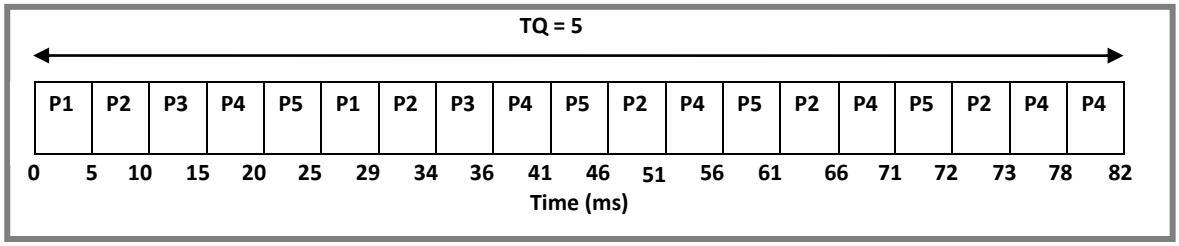

Fig 7: Gantt chart representation of RR with TQ = 5 (Case II)

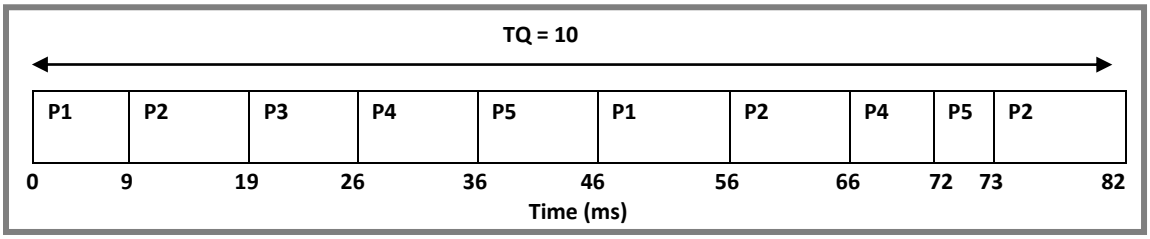

Fig 8: Gantt chart representation of RR with TQ = 10 (Case II)

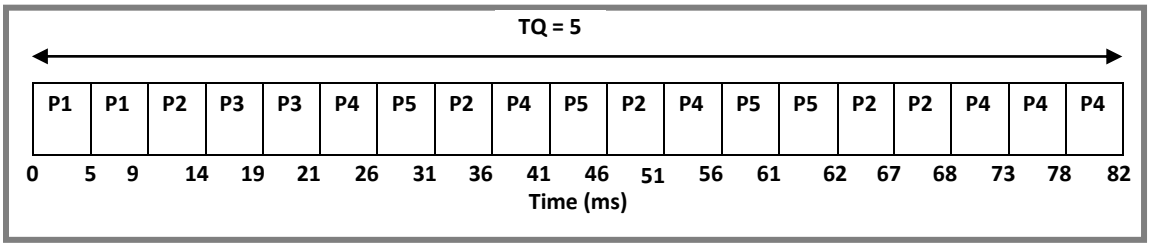

Fig 9: Gantt chart representation of IRR with TQ = 5 (Case II)

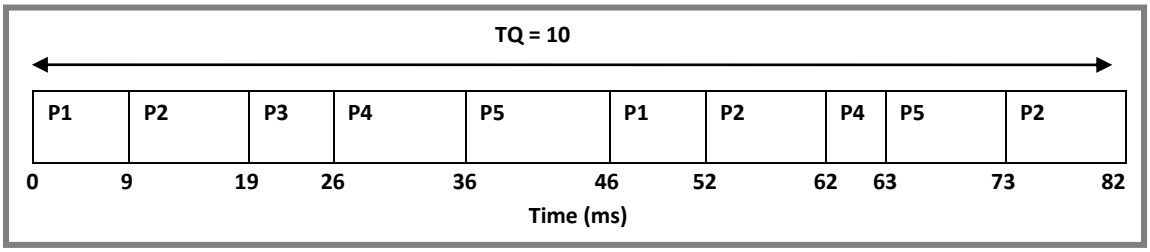

Fig 10: Gantt chart representation of IRR with TQ = 10 (Case II)

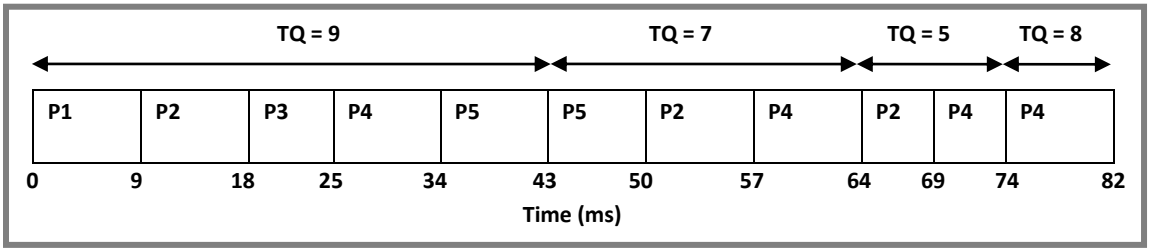

Fig 11: Gantt chart representation of IRRVQ with dynamic TQ (Case II)

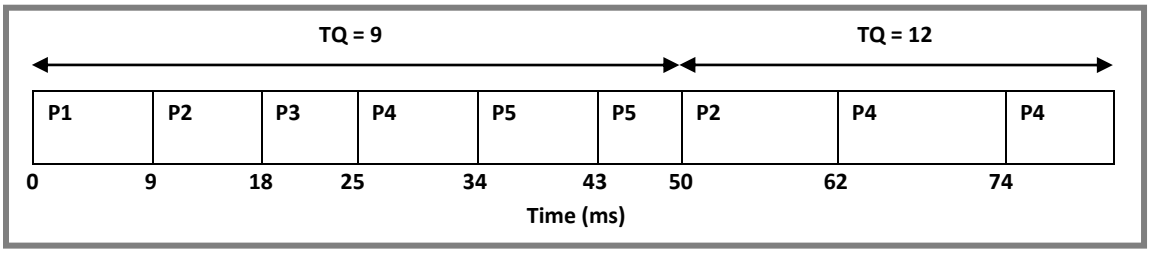

Fig 12: Gantt chart representation of DTQRR with dynamic TQ (Case II)

4.2.3 Analysis of Results

In this section, we provide the comparative analysis of results of our experiments with the help of bar chart. The graphical representation of results of case I (zero arrival time) and case II (non- zero arrival time) are shown in Figure 13 and Figure 14 respectively. The comparative results clearly prove that the proposed DTQRR algorithm is giving better result (reduction in average waiting time, average turnaround time and number of context switches) than RR, IRR and IRRVQ CPU scheduling algorithms.

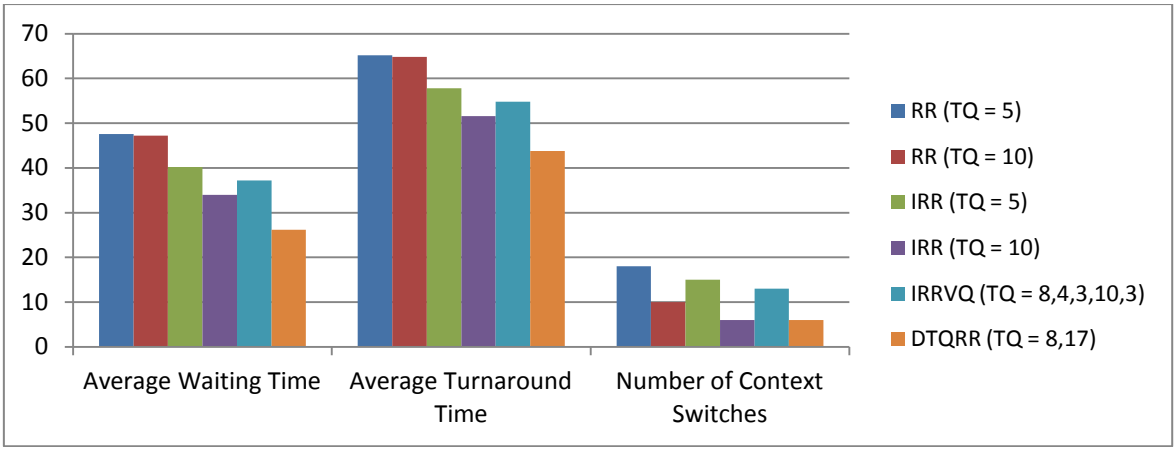

Fig 13: Bar chart representation of average waiting time, average turnaround time and number of context switches (Case I)

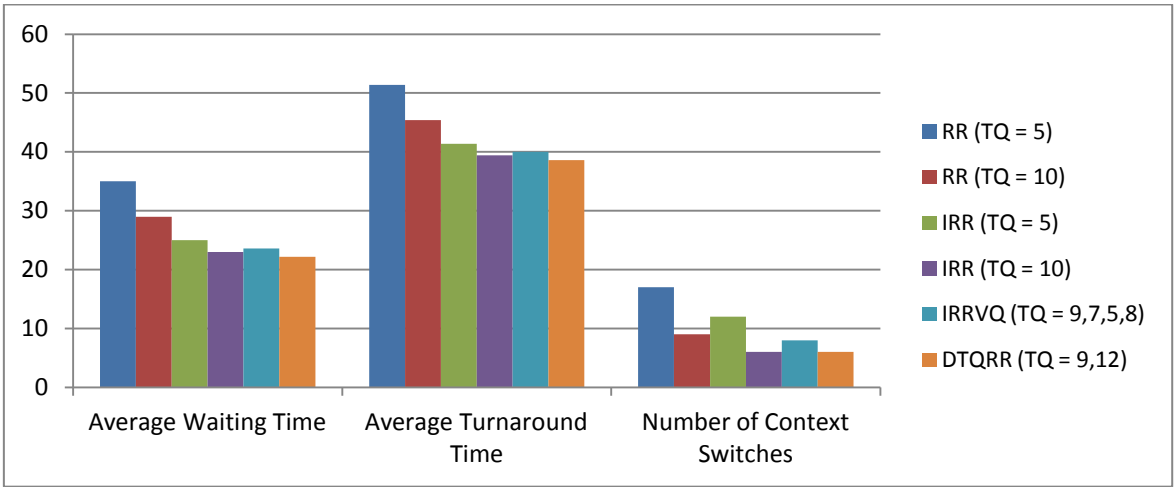

Fig 14: Bar chart representation of average waiting time, average turnaround time and number of context switches (Case II)

5. CONCLUSION

In this paper, a new approach of CPU scheduling algorithm was proposed. The proposed DTQRR CPU scheduling algorithm together with RR, IRR, and IRRVQ CPU scheduling algorithms were implemented and their results were compared based on three scheduling criterion waiting time, turnaround time and context switch. The experimental results show that DTQRR performs better than the RR, IRR and IRRVQ in terms of reducing average waiting time, average turnaround time and number of context switches. Simulation results also prove the correctness of the theoretical results. The proposed algorithm can be integrated to improve the performance of the systems in which RR is a preferable choice.

6. REFERENCES

[1] R. K. Yadav, A. K. Mishra, N. Prakash, and H. Sharma, "An Improved Round Robin Schedduling Algorithm for CPU Scheduling", International Journal on Computer Science and Engineering, Vol. 2, No. 4, 2010, 1064-1066.

[2] Silberschatz, A., Galvin, P. B., and Gagne, G. 2005. Operating System Concepts. John Wiley and Sons Inc.

[3] M. K. Mishra, and A. K. Khan, "An Improved Round Robin CPU Scheduling Algorithm", Journal of Global Research in Computer Science, Vol. 3, No. 6, 2012, 64-69.

[4] M. K. Mishra, and F. Rashid, "An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum", International Journal of Computer Science, Engineering and Applications (IJCSEA), Vol.4, No.4, 2014, 1-8.

[5] R. J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences, Vol. 6, No. 10, 2009, 1831-1837.

[6] H. S. Behera, R. Mohanty and D. Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis", International Journal of Computer Applications, Vol. 5, No. 5, 2010, 10-15.

[7] M. Lavanya, and S. Saravanan, "Robust Quantum Based Low-power Switching Technique to improve System Performance", International Journal of Engineering and Technology, Vol. 5, No. 4, 2013, 3634-3638.

[8] T. Helmy, and A. Dekdouk, "Burst Round Robin as a Proportional-share Scheduling Algorithm", IEEEGCC, 2007, Available: http://eprints.kfupm.edu.sa/1462/.

[9] D. Nayak, S. K. Malla, and D. Debadarshini, "Improved Round Robin Scheduling using Dynamic Time Quantum", International Journal of Computer Applications, Vol. 38, No. 5, 2012, 34-38.

[10] M. Neshat, M. Sargolzaei, A. Najaran, and A. Adeli, "The New method of Adaptive CPU Scheduling using Fonseca and Fleming's Genetic Algorithm", Journal of Theoretical and Applied Information Technology, Vol. 37, No. 1, 2012, 1-16.

[11] H. S. Behera, R. Mohanty, J. Panda, D. Thakur, and S. 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, 2011, 54-60.

[12] H. S. Behera, S. Patel, and B. Panda, "A new dynamic Round-robin and SRTN algorithm using variable original time slice and dynamic intelligent time slice for soft real time system". International Journal of Computer Applications, Vol. 2, No.1, 2011, 54-60.

[13] H. S. Behera, J. Panda, D. Thakur, and S. Sahoo, "A New Proposed Two Processor Based CPU Scheduling Algorithm with Varying Time quantum for Real Time Systems", Journal of Global Research in Computer Science, Vol. 2, No. 4, 2011, 81-87.

[14] A. Noon, A. Kalakech, and S. Kadry, "A New Round Robin based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average", International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, 2011, 224-229.

[15] M. K. Srivastav, S. Pandey, I. Gahoi, and N. K. Namdev, "Fair Priority Round Robin with Dynamic Time Quantum", International Journal of Modern Engineering Research, Vol. 2, Issue 3, 2012, 876-881.

[16] A. Abdulrahim, S. Aliyu, A. M. Mustapha, and S. E. Abdullahi, "An Additional Improvement in Round Robin (AAIRR) CPU Scheduling Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, Issue 2, 2014, 601-610.

[17] A. Abdulrahim, S. E. Abdullahi, and J. B. Sahalu, "An New Improved Round Robin (NIRR) CPU Scheduling Algorithm", International Journal of Computer Applications, Vol. 90, No. 4, 2014, 27-33.

[18] A. R. Dash, S. K. Sahu, and S. K. Samantra, "An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum", International Journal of Computer Science, Engineering and Information Technology, Vol. 5, No. 1, 2015, 7-25.