

RESEARCH PAPER

Available Online at www.jgrcs.info

AN IMPROVED ROUND ROBIN CPU SCHEDULING ALGORITHM

Manish Kumar Mishra^{*1}, Abdul Kadir Khan²

^{*1}Department of Information Technology, Haramaya University, Dire Dawa, Ethiopia
mishrasoft1@gmail.com¹

²Department of Computer Science, Haramaya University, Dire Dawa, Ethiopia
qadirforu@gmail.com²

Abstract: One of the most important components of the computer resource is the CPU. CPU scheduling involves a careful examination of pending processes to determine the most efficient way to service the requests. CPU scheduling is the basis of multiprogrammed operating systems. Most CPU scheduling algorithms concentrate on maximizing CPU utilization and throughput and minimizing turnaround time, waiting time, response time and number of context switching for a set of requests. Some of the popular CPU scheduling algorithms are First-Come-First-Served (FCFS), Shortest Job First (SJF), Priority Scheduling and Round Robin (RR). FCFS is the simplest form of CPU scheduling algorithm. This algorithm is simple to implement, but it generally does not provide the fastest service. Round Robin being the most popular choice in time shared system, but it may not be suitable for real time systems because of larger waiting time, turnaround time and more number of context switches. This paper describes an improvement in RR. A simulator program has been designed and tested the Improved Round Robin (IRR). After improvement in RR it has been found that the waiting time and turnaround time have been reduced drastically.

Keywords: CPU Scheduling, Round Robin Scheduling, Burst Time, Turnaround Time, Waiting Time, Response Time, Context Switch, Time Quantum.

INTRODUCTION

In multiprogrammed computing systems, inefficiency is often caused by improper use of CPU. In multiprogramming systems, multiple processes are being kept in memory for maximum utilization of CPU [1]. CPU utilization can be maximized by switching CPU among waiting processes in the memory and running some process all the time [2]. Which process should be selected next for service, is an important question, because it affects the effectiveness of the service. The main aim of the CPU scheduling algorithms is to maximizing CPU utilization and throughput and minimizing turnaround time, waiting time, response time and number of context switching for a set of requests. This study focuses on improving the effectiveness of Round Robin CPU scheduling algorithm.

Performance Parameters:

A currently running program is called a process. The processes waiting to be assigned to a processor are put in a queue called ready queue. To use CPU effectively, it should be busy as much as possible. Short Term Scheduler is the operating system component that selects a waiting process from the ready queue and allocates CPU to that process whenever CPU becomes idle [2]. The time for which a process uses the CPU is known as burst time. Arrival Time is the time at which a process joins the ready queue. Throughput is the number of processes that are completed per unit of time. Turnaround time is the total time taken by a process from the time of submission to the time of completion of the process. Waiting time of a process is the total time spent by the process waiting in the ready queue. The number of times CPU switches from one process to

another is known as context switch. The CPU scheduling algorithms focus on reducing the waiting time by scheduling the processes in an effective manner.

CPU Scheduling Algorithms:

In multi-programmed operating systems CPU scheduling plays a fundamental role by switching the CPU among various processes [2]. CPU scheduling algorithms are used to allocate the CPU to the processes waiting in the ready queue. Some of the popular CPU scheduling algorithms are First-Come-First-Served (FCFS), Shortest Job First (SJF), Priority Scheduling and Round Robin (RR). FCFS is the simplest form of CPU scheduling algorithm. In this scheduling algorithm, the process that arrives first in ready queue served first, so the name First-Come-First-Served. The average waiting time in FCFS is quite long [2]. In Shortest Job First (SJF) algorithm, process from the ready queue that has shortest CPU burst time will execute first. If two processes are having same CPU burst time and arrival time, then FCFS procedure is followed. In SJF average waiting time decreases. Priority scheduling algorithm allocates the CPU to the higher priority process from the ready queue. In Round Robin (RR), a small unit of time quantum is given to each process present in the ready queue which maintains the fairness factor. In this paper we have proposed an improvement in RR to reduce the waiting time and turnaround time.

RELATED WORK

In the recent years, a number of CPU scheduling mechanisms have been developed for predictable allocation of processor. An Improved Round Robin Scheduling Algorithm for CPU Scheduling [1] allocates the time quantum to all the process

only in first cycle. After executing all the processes once, they use SJF to select next process from the ready queue. Self-Adjustment Time Quantum in Round Robin Algorithm [3] is based on a new approach called dynamic time quantum, in which time quantum is repeatedly adjusted according to the burst time of the running processes. Dynamic Quantum with Readjusted Round Robin Scheduling Algorithm [4] uses the job mix order for the algorithm in [3]. Mixed Scheduling (A New Scheduling Policy) [5], uses the job mix order for non preemptive scheduling FCFS and SJF. According to job mix order, from a list of N processes, the process which needs minimum CPU time is executed first and then the highest from the list and so on till the nth process. A new weighting technique is introduced for CPU Schedulers in Burst Round Robin (BRR) [6]. Here shorter jobs are given more time, so that processes having shorter jobs are cleared from the ready queue in a short time span. Debashree Nayak et. al. [7] did the similar work as [3] [4]. They assign optimal time quantum to each process after every cycle of execution. Optimal time quantum is the average of highest CPU burst time and the median. In [8] a new CPU scheduling algorithm is presented.

This algorithm schedules the running of processes according to three parameters of CPU burst time, I/O service time, and priority of processes. This algorithm selects and runs the desired processes through adaptation. In this algorithm, the priority of processes increases with time, and no process encounters starvation. A new fare-share scheduling with weighted time slice [9] assigns a weight to each process and the process having the least burst time is assigned the largest weight. The time quantum is calculated dynamically, using weighted time slice method and then the processes are executed. Algorithm in [10] 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]. Algorithm in [11] is improved by using dynamic time quantum and multi cyclic time quantum.

A New Proposed Two Processor Based CPU Scheduling Algorithm with Varying Time Quantum for Real Time Systems [12] uses two processors, one is solely dedicated to execute CPU-intensive processes and the other CPU is dedicated to executed I/O-intensive process. This gives better result in a two processor environment than [4]. In [13] the average of the burst time of the processes is calculated after every cycle and allocated as dynamic time quantum. In Fair Priority Round Robin with Dynamic Time Quantum [14], the processes have been scheduled by giving importance to both the user priority and shortest burst time priority. A new calculated factor based on both user priority and the burst time priority, decides the individual time quantum for each process.

IRR CPU SCHEDULING ALGORITHM

The improved Round Robin (IRR) CPU scheduling algorithm works similar to Round Robin (RR) with a small improvement. IRR picks the first process from the ready

queue and allocate the CPU to it for a time interval of up to 1 time quantum. After completion of process's time quantum, it checks the remaining CPU burst time of the currently running process. If the remaining CPU burst time of the currently running process is less than 1 time quantum, the CPU again allocated to the currently running process for remaining CPU burst time. In this case this process will finish execution and it will be removed from the ready queue. The scheduler then proceeds to the next process in the ready queue. Otherwise, if the remaining CPU burst time of the currently running process is longer than 1 time quantum, the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.

Following is the proposed IRR CPU scheduling algorithm

- Step 1. START
- Step 2. Make a ready queue of the Processes say REQUEST.
- Step 3. Do steps 4, 5 and 6 WHILE queue REQUEST becomes empty.
- Step 4. Pick the first process from the ready queue and allocate the CPU to it for a time interval of up to 1 time quantum.
- Step 5. 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. After completion of execution, removed it from the ready queue and go to step 3.
- Step 6. Remove the currently running process from the ready queue REQUEST and put it at the tail of the ready queue.
- Step 7. END

Illustration:

Considering a ready queue with four processes P1, P2, P3 and P4 arriving at time 0 with burst time 15, 7, 28 and 20 respectively. Time quantum (TQ) has been assumed 10 milliseconds (ms). Our proposed IRR CPU scheduling picks the first process P1 from the ready queue and allocate the CPU to it for a time interval of 10 ms. After executing P1 for 10 ms, the remaining CPU burst of P1 is 5 ms. Since the remaining CPU burst time of P1 is less than the TQ, CPU will be allocated again to P1 for a time interval of 5 ms. P1 has finished execution, it will be removed from the ready queue. Next process in the ready queue is P2 with 7 ms CPU burst time. CPU will be allocated to P2 for a time interval of 7 ms. P2 will finish execution and it will be removed from the ready queue. Next process in the ready queue is P3 with 28 ms CPU burst time. CPU will be allocated to P3 for a time interval of 10 ms. Since the remaining CPU burst time of P3 is not less than the TQ,

CPU will be allocated to P4 for a time interval of 10 ms. Remaining CPU burst of P4 is 10 and it is not less than the TQ. After first cycle the processes remaining in the ready queue are P3 and P4 with remaining burst time 18 and 10 respectively. The first process P3 will be selected from the

ready queue for execution for a time interval of 10 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 8 ms. P3 has finished execution, it will be removed from the ready queue. Next process in the ready queue is P4 with remaining CPU burst time 10 ms. CPU will be allocated to P4 for a time interval of 10 ms. P4 has finish execution and it will be removed from the ready queue. The waiting time is 0 ms for P1, 15 ms for P2, 32 ms for P3 and 50 ms for P4, The average waiting time is 24.25 ms. Using the same set of process with same arrival and CPU burst times, the average waiting time is 30.25 ms in RR. The average turnaround time is 41.75 in IRR and 47.75 in RR.

EXPERIMENTAL ANALYSIS

Assumptions:

To evaluate the performance, we assumed that the environment where all the experiments are performed is a single processor environment and all the processes are independent. All the processes have equal priority. All the attributes like burst time, number of processes and the time slice of all the processes are known before submitting the processes to the processor. The context switching time is equal to zero i.e. there is no context switch overhead incurred in switching from one process to another. All

processes are CPU bound. No processes are I/O bound. The time quantum is taken in milliseconds.

Experiments Performed:

For performance evaluation of our proposed IRR algorithm, we have taken two different cases. In first case arrival time has been considered zero and CPU burst time has been taken in increasing, decreasing and random orders. In second case arrival time has been considered non zero and CPU burst time has been taken in increasing, decreasing and random orders.

CASE 1 - Zero Arrival Time:

In this case arrival time has been considered zero and CPU burst time has been taken in increasing, decreasing and random orders. Time quantum is 10 milliseconds.

CPU Burst Time in Increasing Order: We consider the ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0 with burst time 5, 12, 20, 26 and 34 respectively. The comparison result of RR and proposed IRR are shown in Table 1. Fig. 1 and Fig. 2 show the Gantt chart representation of RR and IRR respectively.

Table 1. Comparison of RR and IRR

Algorithm	Average Waiting Time (ms)	Average Turnaround Time (ms)
RR	38.4	57.8
IRR	30.4	49.8

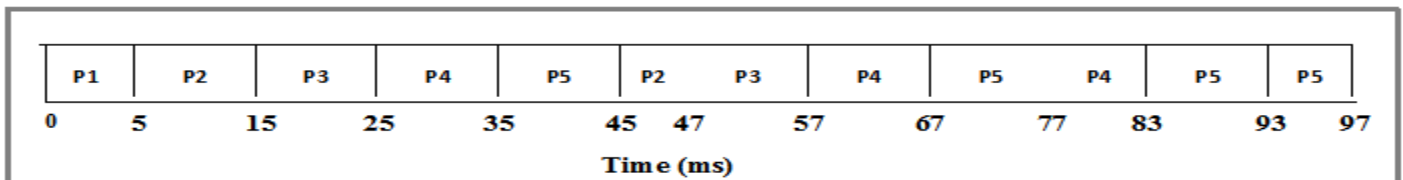


Figure 1. Gantt chart representation of RR

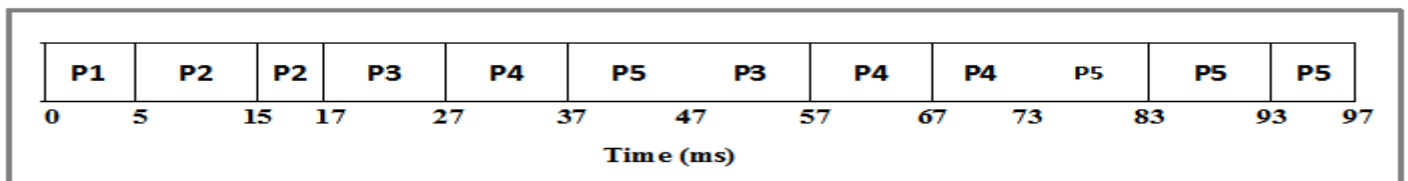


Figure 2. Gantt chart representation of IRR

CPU Burst Time in Decreasing Order: We consider the ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0 with burst time 34, 26, 20, 12 and 5 respectively. The comparison result of RR and proposed IRR are shown in Table 2. Fig. 3 and Fig. 4 show the Gantt chart representation of RR and IRR respectively.

Table 2. Comparison of RR and IRR

Algorithm	Average Waiting Time (ms)	Average Turnaround Time (ms)
RR	58	77.4
IRR	49	68.4

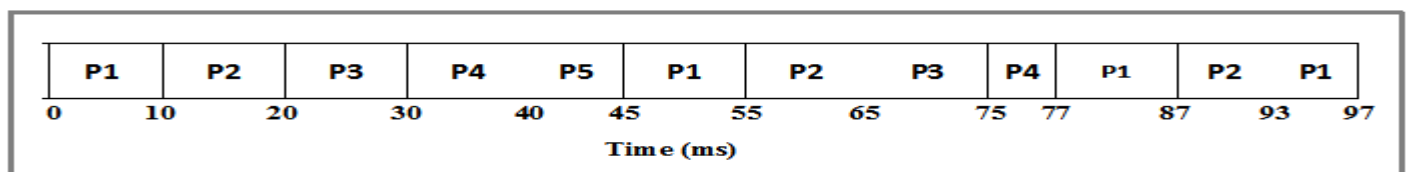


Figure 3. Gantt chart representation of RR

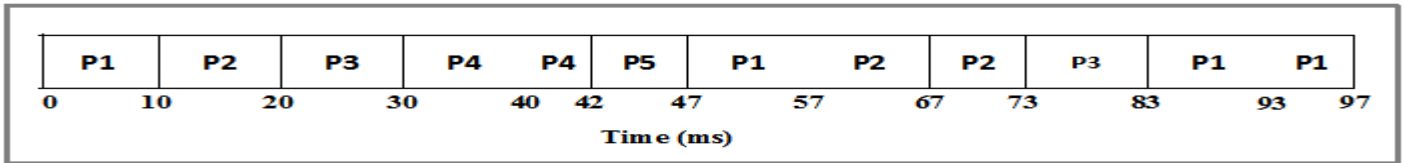


Figure 4. Gantt chart representation of IRR

CPU Burst Time in Random Order: We consider the ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0 with burst time 20, 34, 5, 12 and 26 respectively. The comparison result of RR and proposed IRR are shown in Table 3. Fig. 5 and Fig. 6 show the Gantt chart representation of RR and IRR respectively.

Table 3. Comparison of RR and IRR

Algorithm	Average Waiting Time (ms)	Average Turnaround Time (ms)
RR	47	67.4
IRR	40.4	59.4

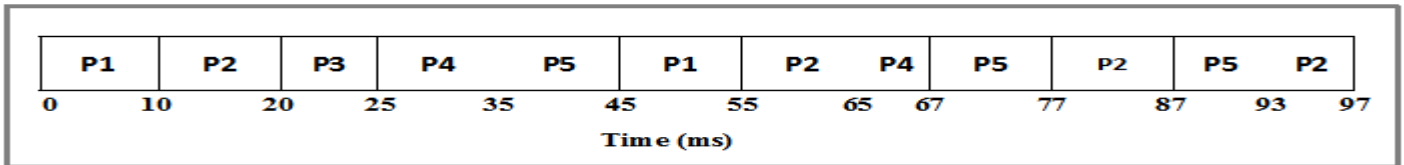


Figure 5. Gantt chart representation of RR

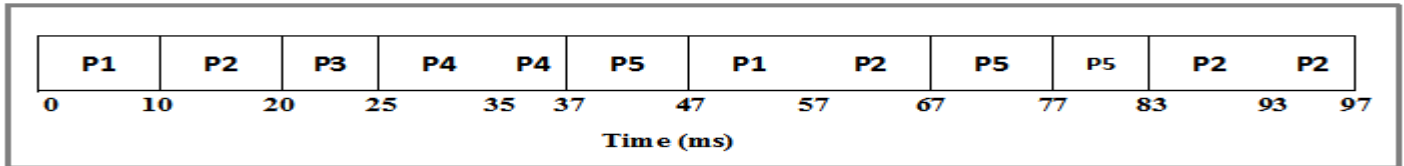


Figure 6. Gantt chart representation of IRR

CASE 1 – Non-Zero Arrival Time:

In this case arrival time has been considered non-zero and CPU burst time has been taken in increasing, decreasing and random orders. Time quantum is 10 milliseconds.

CPU Burst Time in Increasing Order: We consider the ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0, 4, 10, 15 and 17 with burst time 7, 18, 27, 30 and 36 respectively. The comparison result of RR and

proposed IRR are shown in Table 4. Fig. 7 and Fig. 8 show the Gantt chart representation of RR and IRR respectively.

Table 4. Comparison of RR and IRR

Algorithm	Average Waiting Time (ms)	Average Turnaround Time (ms)
RR	42	65.6
IRR	32	55.6

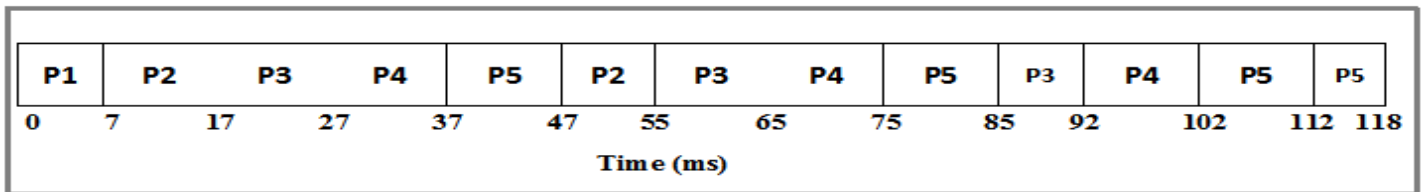


Figure 7. Gantt chart representation of RR

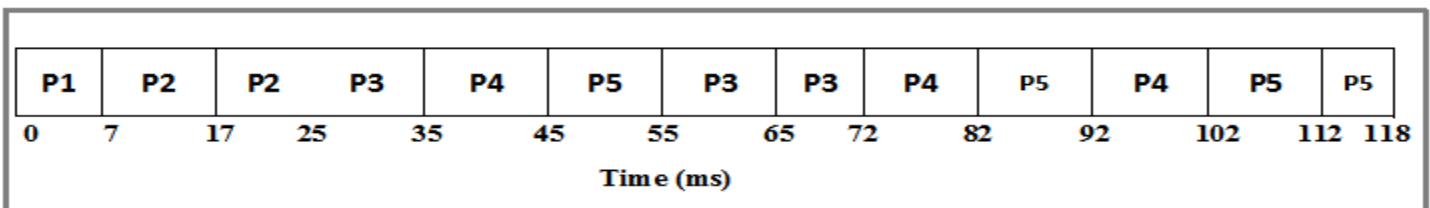


Figure 8. Gantt chart representation of IRR

CPU Burst Time in Decreasing Order: We consider the ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0, 4, 10, 15 and 17 with burst time 36, 30, 27, 18 and 7 respectively. The comparison result of RR and proposed IRR are shown in Table 5. Fig. 9 and Fig. 10 show the Gantt chart representation of RR and IRR respectively.

Table 5. Comparison of RR and IRR

Algorithm	Average Waiting Time (ms)	Average Turnaround Time (ms)
RR	60.6	84.2
IRR	51.4	75

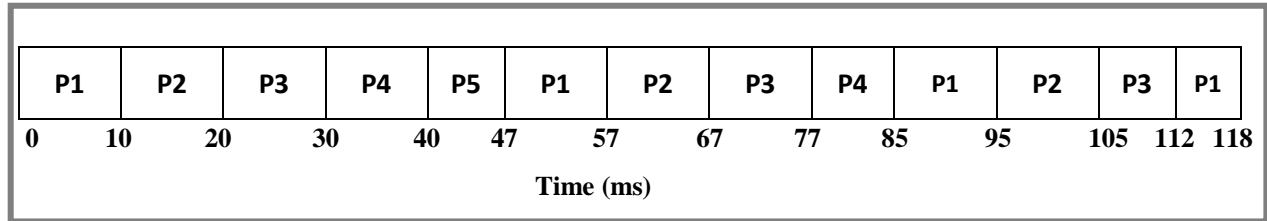


Figure 9. Gantt chart representation of RR

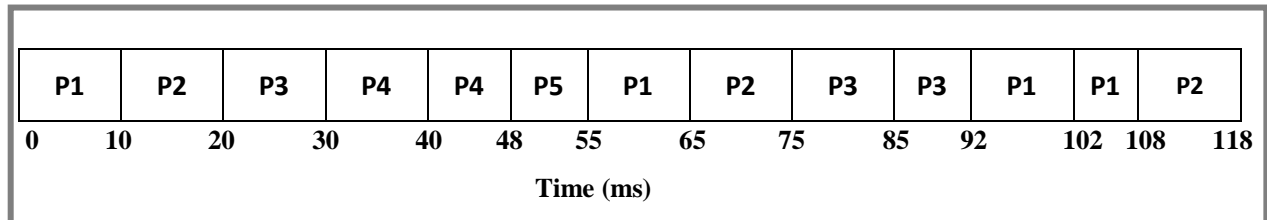


Figure 10. Gantt chart representation of IRR

CPU Burst Time in Random Order: We consider the ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0, 4, 10, 15 and 17 with burst time 27, 7, 30, 36 and 18 respectively. The comparison result of RR and proposed IRR are shown in Table 6. Fig. 11 and Fig. 12 show the Gantt chart representation of RR and IRR respectively.

Table 6. Comparison of RR and IRR

Algorithm	Average Waiting Time (ms)	Average Turnaround Time (ms)
RR	52	73.6
IRR	40	63.6

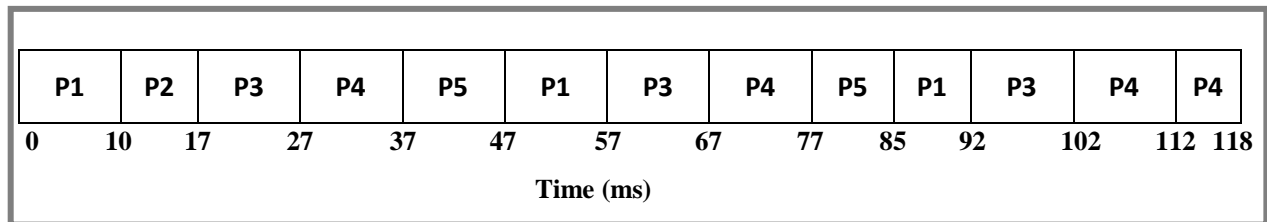


Figure 11. Gantt chart representation of RR

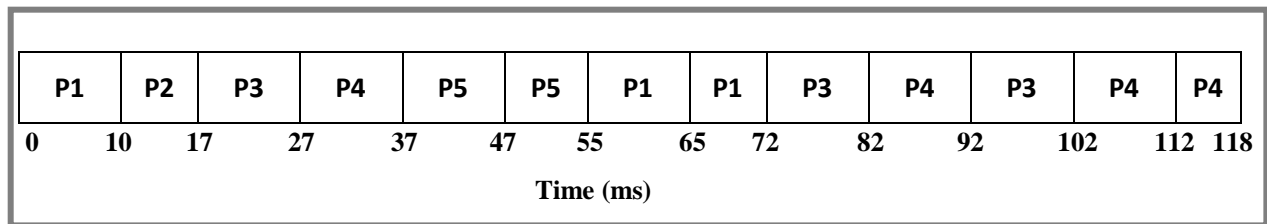


Figure 12. Gantt chart representation of IRR

CONCLUSION

One of the most important components of the computer resource is the CPU. CPU scheduling involves a careful examination of pending processes to determine the most efficient way to service the requests. Many CPU scheduling algorithms have been presented having some advantages and disadvantages. In this paper an improved round robin CPU

scheduling algorithm is proposed. Simulation results shows that the proposed IRR CPU scheduling algorithm is always giving better performance than RR. After improvement in RR it has been found that the waiting time and turnaround time have been reduced drastically. This algorithm can be implemented to improve the performance in the systems in which RR is a preferable choice.

REFERENCES

[1] Rakesh Kumar Yadav, Abhishek K Mishra, Navin Prakash and Himanshu Sharma, "An Improved Round Robin Scheduling Algorithm for CPU Scheduling", International Journal on Computer Science and Engineering, Vol. 02, No. 04, 2010, pp. 1064-1066.

[2] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts", 7th Edn., John Wiley and Sons Inc, 2005, ISBN 0-471-69466-5.

[3] Rami J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes", American J. of Applied Sciences 6(10):1831-1837, 2009.

[4] H.S. Behera, R. Mohanty, and Debashree 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, August 2010, pp. 10-15.

[5] Sunita Mohan, "Mixed Scheduling (A New Scheduling Policy)", Proceedings of Insight'09, 25-26 November 2009.

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

[7] Debashree Nayak, Sanjeev Kumar Malla, and Debashree Debadarshini, "Improved Round Robin Scheduling using Dynamic Time Quantum", International Journal of Computer Applications, Vol. 38, No. 5, January 2012, pp. 34-38.

[8] Mehdi Neshat, Mehdi Sargolzaei, Adel Najaran, and Ali 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, March 2012, pp. 1-16.

[9] H.S. Behera, Rakesh Mohanty, Jainaseni Panda, Dipanwita Thakur and Subasini Sahoo, "Experimental Analysis of a New Fair-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.

[10] H.S. Behera, Simpi Patel, Bijaylaxmi 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 (0975-8887), Volume 16, No.1, 54-60, February 2011.


[11] H.S. Behera, Rakesh Mohanty, Sabyasachi Sahu, Sourav Kumar Bhoi, "Design and performance evaluation of multi cyclic round robin(MCRR) algorithm using dynamic time quantum" Journal of global research in computer science(ISSN-2229-371X), volume 2, No.2, February 2011.


[12] H.S. Behera, Jainaseni Panda, Dipanwita Thakur and Subasini 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, April 2011, pp. 81-87.

[13] Abbas Noon, Ali Kalakech, and Seifedine 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, May 2011, pp. 224-229.

[14] M.K. Srivastav, Sanjay Pandey, Indresh Gahoi and Neelesh Kumar Namdev, "Fair Priority Round Robin with Dynamic Time Quantum", International Journal of Modern Engineering Research, Vol. 02, Issue. 03, May-June 2012, pp. 876-881.

Short Bio Data for the Author

 Manish Kumar Mishra is with the Department of Information Technology, Haramaya University, Ethiopia. He has published several research papers in national and international journals. His area of interest includes Operating system, Software Engineering, Database and OOP.

 Abdul Kadir Khan is with the Department of Computer Science, Haramaya University, Ethiopia. He has published several research papers in national and international journals. His areas of interest are Operating system, Software Engineering and Database.