A Novel Amended Dynamic Round Robin Scheduling Algorithm for Timeshared Systems

Uferah Shafi[1], Munam Shah[1], Abdul Wahid[1], Kamran Abbasi[2], Qaisar Javaid[3], Muhammad Asghar[4], and Muhammad Haider[1]

[1]Department of Computer Science, COMSATS University Islamabad, Pakistan
[2]Department of Distance Continuing and Computer Education, University of Sindh, Pakistan
[3]Department of Computer Science, International Islamic University, Pakistan
[4]Department of Computer Science, Bahuddin Zikriya University, Pakistan

Abstract: *Central Processing Unit (CPU) is the most significant resource and its scheduling is one of the main functions of an operating system. In timeshared systems, Round Robin (RR) is most widely used scheduling algorithm. The efficiency of RR algorithm is influenced by the quantum time, if quantum is small, there will be overheads of more context switches and if quantum time is large, then given algorithm will perform as First Come First Served (FCFS) in which there is more risk of starvation. In this paper, a new CPU scheduling algorithm is proposed named as Amended Dynamic Round Robin (ADRR) based on CPU burst time. The primary goal of ADRR is to improve the conventional RR scheduling algorithm using the active quantum time notion. Quantum time is cyclically adjusted based on CPU burst time. We evaluate and compare the performance of our proposed ADRR algorithm based on certain parameters such as, waiting time, turnaround time etc. and compare the performance of our proposed algorithm. Our numerical analysis and simulation results in MATLAB reveals that ADRR outperforms other well-known algorithms such as conventional Round Robin, Improved Round Robin (IRR), Optimum Multilevel Dynamic Round Robin (OMDRR) and Priority Based Round Robin (PRR).*

Keywords: *CPU, scheduling algorithm, round robin scheduling FCFS, ADRR.*

Received August 10, 2015; accepted February 15, 2017
https://doi.org/10.34028/iajit/17/1/11

1. Introduction

Central Processing Unit (CPU) scheduling is the most important component which affects the efficiency of the system in a computing environment. In a single processor system, one process can execute at one time, other processes are delayed until the CPU gets free. CPU scheduling algorithms provide base for multiprogramming. In multiprogramming [6], there are several processes in the memory, CPU always have one process to execute and it switches between different processes to make itself busy all the time. It is the duty of operating system to make decision that which process should be executed. If CPU scheduling is efficient then high computation can be done correctly and accurately and system can retain stable. The scheduling algorithm needs to be optimized to make the system more productive and to increase the system throughput.

There are several methods that an operating system uses to select a process and then assign it to the CPU for execution. Every algorithm has some advantages and some limitations [8]. First Come and First Serve (FCFS) Scheduling Algorithm [16] is an algorithm in which new processes are popped in at the tail of the queue and from head processes are assigned to the CPU for execution. The process that arrives first are assigned first to the CPU based on their arrival time. In this algorithm, the average waiting is high. As if first arriving process that is served first has a large CPU burst time then the remaining processes will wait until that process finishes its execution. Starvation is a major problem in FCFS. Shortest Job First (SJF) Scheduling Algorithm [1] is another scheduling algorithm, in which the process with minimum CPU burst time is assigned first to the CPU. The short-term scheduler adds the processes at the head of the queue with smallest CPU burst time and inserts the processes at the tail of the queue with high CPU burst time. The main problem in this scheduling is to find next CPU burst time. The average waiting time in this algorithm is smaller when compared to Round Robin (RR) Scheduling algorithm. In the Priority Scheduling Algorithm [15], each process is assigned a priority by the operating system or the user. The process that has highest priority will be assigned to CPU first. This is preemptive scheduling as running process will be preempted by the incoming process with highest priority and will be assigned to CPU first. In Round Robin Scheduling Algorithm [17], each process is assigned to CPU for a specific time interval (time quantum or time slice). This results in no starvation because every process gets same amount of time for execution. Context switch is the big problem in this algorithm. If quantum is small, more number of context switches will occur which result in low efficiency of CPU and if quantum is large, then the algorithm behaves as FCFS. The conventional time quantum based CPU scheduling algorithms do not offer fair allocation of the CPU and result in starvation. We propose a novel Amended Dynamic Round Robin (ADRR) algorithm. In ADRR, the processes which have lowest CPU burst time must wait for lesser time. High throughput, small waiting time and small number of context switches make the proposed algorithm perform better when compared with similar approaches. The rest of the paper is organized as follow: Section 2 reviews the existing literature. Section 3 describes the design of the proposed algorithm. Section 4 provides the mathematical analysis and simulation results of ADRR algorithm. The paper is concluded in section 5.

2. Related Work

RR scheduling algorithm is widely implemented in most of the operating systems for better CPU performance. A lot of research has been carried out to improve and optimize the performance of RR scheduling by maximizing CPU utilization, throughput and minimizing average waiting time, response time, and turnaround time. The conventional RR scheduling algorithm has following versions.

a) *Improved Round Robin* [23]: this is a hybrid approach of conventional RR and SJF algorithms. Using this technique all the processes are allocated to CPU according to RR in first cycle and after the first iteration, all the processes are entertained as SJF. This methodology gives small average waiting time and average turnaround time when compared with conventional RR.
b) *Optimum Multilevel Dynamic Round Robin* [4]: in this algorithm, all processes are arranged in increasing order and are assigned to the CPU. An intelligent Quantum Time (QT) is calculated. After each cycle, value of QT is doubled which results in decreased number of context switches. Before preempting a process, a condition is checked based on remaining CPU burst time to reduce waiting, this is the reason, this algorithm gives improved results as matched to conventional RR.
c) *A Priority Based Round Robin* [15]: in this algorithm, all processes are treated according to the priority of each process in first round. In second round, priority of each process is set based on the remaining CPU burst time of the process. All processes are treated according to their new priorities which results in lesser average waiting time.

Many other flavors in RR CPU scheduling algorithm have been presented. In year 2010, a new technique was proposed as optimized RR scheduling algorithm [18] for CPU Scheduling. This algorithm is consisted of three rounds. In first round, all processes are treated according to conventional RR algorithm. In second round, QT is doubled and follow the SJF algorithm to select processes. In last round, the first two rounds are repeated until all the processes finish execution.

Self-Adjustment-Round-Robin (SARR), a new technique was proposed by Matarneh [10]. On the basis of which quantum time is adjusted dynamically. In each iteration, quantum time is assigned the value dynamically according to the burst time of currently executing process. A new methodology named as Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice was presented by Hiranwal [7]. In the proposed algorithm, all processes are first arranged in ascending order, and time slice is set as the value of CPU burst time of mid process. If number of processes are even, the average burst time value is set and vice versa. Another technique is recently presented by Behera [2], in which all the processes in ready queue are sorted in ascending order. Firstly, median is found and then QT is set according to median. Another new algorithm is proposed in [12]. This algorithm adjusts QT by calculating mean and average of the processes in the ready queue.

A better version of RR scheduling algorithm is presented by Varma [21]. The authors have used the idea of shortest remaining time in RR manner and possibly the best values of QT by calculating the median and highest burst time. In [14], another scheduling algorithm is developed which sets the value of QT equal to the difference of highest CPU burst and lowest CPU burst time. In [5], another CPU scheduling algorithm is designed. Here, maximum and minimum CPU burst time are calculated and QT is adjusted by multiplying the sum of maximum and minimum CPU burst time.

In year 2012, an improved version of RR was proposed [3]. A new criterion is introduced to adjust the value of QT. Firstly, average of CPU burst time of all processes is calculated and QT is set as the sum of the average and maximum CPU burst times. During the same year, Varma *et al*. [22] presented another flavor of RR in. The square root function is used to calculate the value of QT. Mishra [11] proposed a new algorithm in 2014 which sets the value of QT by using the features of RR and SJF. Shyam and Nandal [19] presented another algorithm in. They set QT by calculating values of the mean and the highest burst time. In [13], a different RR algorithm is developed on the basis of group of processes using the values of minimum and maximum CPU burst times. Most of the existing solutions result in more waiting time taking complex mechanism taking into consideration the complex QT calculation. We propose a simple yet efficient CPU scheduling algorithm with better results. In year 2016, Khan *et al*. [9] presented a new group based technique to schedule resources based on different parameters of CPU to improve the system efficiency and turnaround time.

3. Proposed Amended Dynamic Round Robin (ADRR)

In this section, we discuss in detail the design of our proposed ADRR algorithm. ADRR makes use of dynamic QT as an alternate of fixed QT. In traditional RR algorithm, a fixed QT is assigned to each process. In ADRR, CPU burst time is measured dynamically and QT is set equal to the value of the lowest CPU burst time. After each cycle, QT is readjusted. In our proposed ADRR, all the processes located in the ready queue are first arranged in increasing order based on the CPU burst time so that the process having lowest CPU burst time will be at the head of the ready queue and the process having highest CPU burst time will be at the tail of the ready queue. Processes are assigned to the CPU in such a way that the process which has shortest CPU burst time, waits for minimum time interval.

In ADRR, the QT, which is core factor in the performance of RR, is set equal to the value of the least CPU burst time. We set a threshold value of QT as 20 and then check a condition, i.e., if QT is less than the threshold (20), then the condition is true and QT is set as 20. We check this condition to avoid the Value of QT being very small which will result in more number of context switches. All processes are assigned to the CPU for the specific time interval. Processes will be preempted if their remaining CPU burst time is greater than the half of the QT. Preempted processes are again inserted into ready queue in ascending order. After the first cycle, QT is readjusted and set as the value equal to the lowest value of CPU burst time. Same rule is repeated until all the processes finish their execution. This means that in each cycle of ADRR, QT is dynamically set as the value equal to the lowest CPU burst time of the process. The block diagram of the proposed ADRR algorithm is provided in Figure 1.

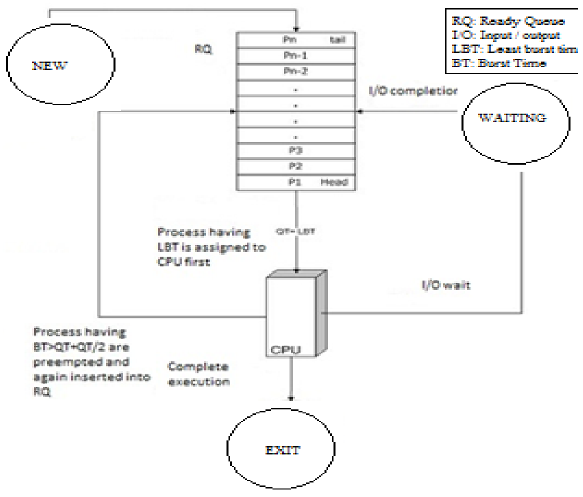

Figure 1. Block diagram of ADRR.

We can see that all the states of the processes are represented. A process changes its state during its life cycle. Newly created processes are in new state, when loaded into memory then these are in ready state, if processes need I/O then processes go into waiting state, when processes are assigned to CPU then these processes are in running state and finally processes finish execution are in exit or terminated state. Processes are inserted into ready queue in ascending order and assigned to CPU from head of the queue, quantum time is adjusted as the minimum CPU burst time in each cycle and minimum one process finishes its execution. The process which have burst time greater than quantum time are preempted and before preemption if the remaining burst time is less than or equal to quantum time then this process is not preempted and completes its execution. The pseudo code of the ADRR is provided in Algorithm1.

Algorithm 1: Amended Dynamic Round Robin

```
//Input: Number of processes
//Burst time of processes,
//loop variable i;
 // QT: Quantum Time
// RBT: Remaining burst time
// RQ: Ready Queue
// BT: Burst Time
// LBT: Least burst time
while(RQ! =null) {
    sort processes (Ascending);
    QT= LBT;
      if(QT<20){
        QT=20;}
      else{ QT=QT; }
    for i=0 to Number of Processes {
      pi-> QT;  if(BTi<QT)
    {Process} BT=0;// Exit                    else if(BTi>QT){
         RBTi= BTi-QT;
                              if(RBTi<=QT/2){ BTi=0; } //Process Exit
         let pi to     finish   its execution don't preempt
         insert pi into RQ
         }
        //Calculate TurnaroundTime(); waitingTime(); Average
        Time();
insert pi into RQ
}
//Calculate TurnaroundTime(); waitingTime(); Average Time();
}New Processes are inserted into RQ
END
```

The flow chart of ADRR Scheduling algorithm is shown below in Figure 2. It could be observed in Figure 2 that the processes entering the system are added into ready queue and are organized in increasing order, quantum time is set as the minimum CPU burst time. To minimize the number of context switches, the minimum value of quantum time is set as 20. All processes can execute for that time slice. Before preempting the process, a condition is checked based on remaining CPU burst time. In each round minimum one process finishes its execution.

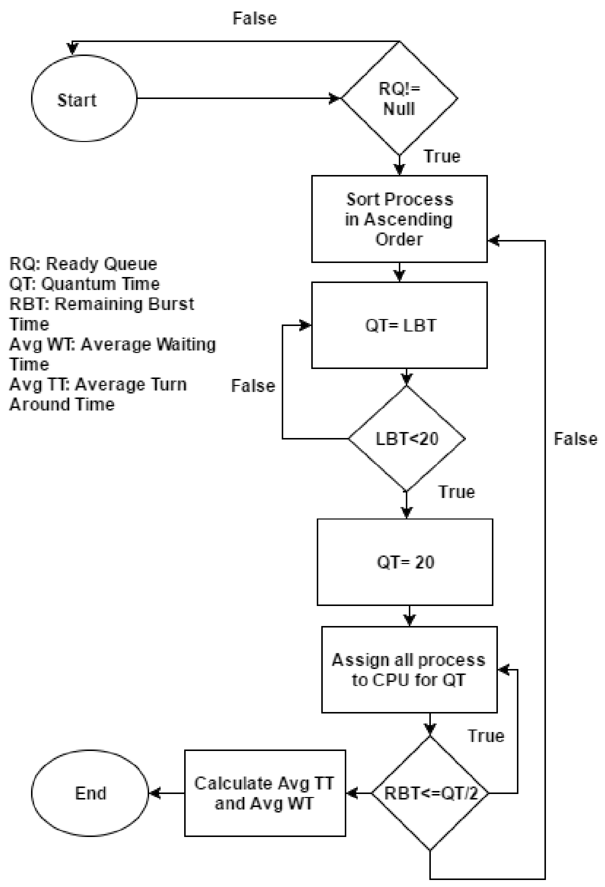
False
Start
RQ!= Null
True
Sort Process in Ascending Order
RQ: Ready Queue
QT: Quantum Time
RBT: Remaining Burst Time
Avg WT: Average Waiting Time
Avg TT: Average Turn Around Time
QT= LBT
False
LBT<20
False
True
QT= 20
Assign all process to CPU for QT
True
End
Calculate Avg TT and Avg WT
RBT<=QT/2

Figure 2. Flow chart of amended dynamic round robin.

4. Numerical Analysis and Simulation Results

To compare the performance of proposed ADRR scheduling algorithm, we perform mathematical analysis and computer simulation for different parameters such as number of context switches, average waiting time and average turnaround time etc. The simulation is performed to analyze the performance of proposed ADRR in MATLAB 2013, on Window 7 operating system. Numerical and simulation results are discussed below. We compare the performance of proposed ADRR with Round Robin (RR), Priority based Round Robin (PRR), Optimum Multilevel Dynamic Round Robin (OMDRR), and Improved Round Robin (IRR).

We consider two examples, i.e., Example A and Example B. as shown in Table 1. The average waiting time, average turnaround time and number of context switches are calculated for each algorithm. Five processes have been considered with distinct CPU burst time and arrival time. In example A, we take burst time of each process randomly and assume quantum time as 5ms. The processes arrive in the order as: P1, P2, P4, P3 and P5. Here, the number of context switches is 4 and the average waiting time is 17ms. The average turnaround time is 29.8ms.

Table 1. Set of input.

Process_id	**Arrival time(ms)**	**Example A CPU burst time (ms)**	**Example B CPU burst time (ms)**	**Priority**
P1	0.5	22	60	4
P2	1	18	20	2
P3	2	9	25	1
P4	3	10	10	3
P5	4	5	40	5

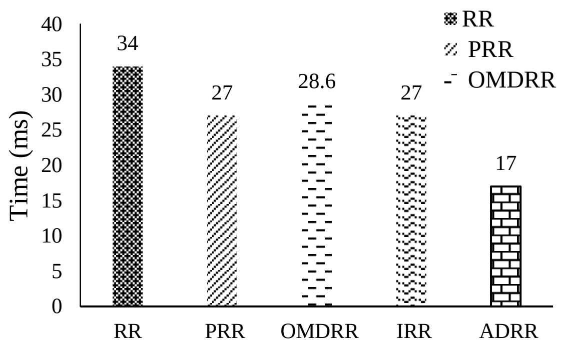

Figure 3. Comparison of average waiting time in each algorithm.

The numerical results obtained are plotted in Figure 3. It could be clearly observed from the graph that ADRR gives the smallest waiting time. While, IRR and PRR give equal average waiting times. We discuss another important parameter in a scheduling algorithm, i.e., context switch which helps in the critical evaluation CPU scheduling algorithm. The CPU remains idle during switching the process, comparison of number of context switches is presented in Figure 4. It could be clearly observed that in conventional RR, there is maximum number of context switches and ADRR gives minimum number of context switches.

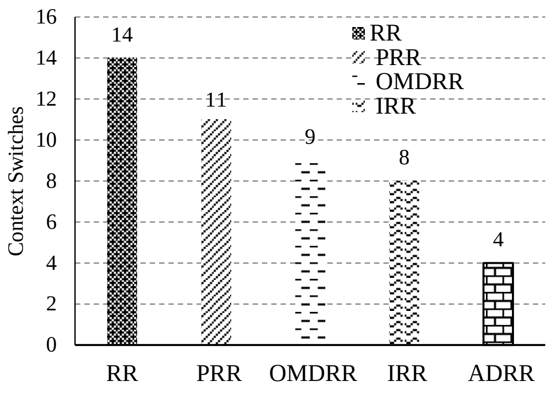

Figure 4. Comparison of number of context switches in each algorithm.

We proceed to the waiting time. A comparison of proposed algorithm against other selected algorithms is provided in Figure 5.

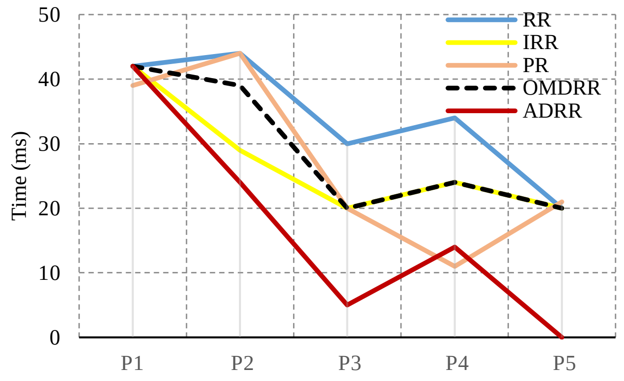

Figure 5. Simulation comparison of waiting time of each process.

Processes are plotted on x-axis and their waiting times are plotted n y-axis. It is obvious from the graph that in ADRR, P5 has smallest CPU burst time that's why waiting time of P5 is zero and P3 has minimum burst time among all remaining processes so its waiting time will be smaller than other processes. Hence processes with small CPU burst times have to wait for small interval. Since all processes are assigned to CPU for quantum time, no chance of starvation. ADRR gives better results as compared to other algorithms. Turnaround time is also an important factor in efficiency of an algorithm. Smaller value of turnaround time makes the algorithm more efficient. The comparative analysis of average turnaround time in competing algorithms is shown in Figure 6 which shows the result for average turnaround time for different processes having different burst times. ADRR gives least average turnaround time when compared with other scheduling algorithms. Turnaround time of each process in opposing algorithms is shown below in Figure 7. Graphic statistics reveals that P5 has the least turnaround time in ADRR due to its small waiting time. ADRR gives small turnaround time as compared to other algorithms.

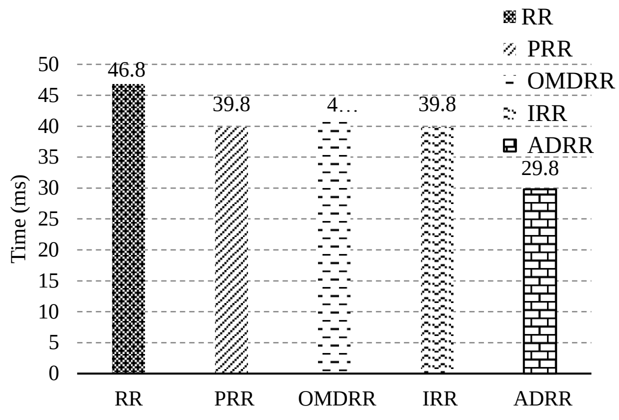

Figure 6. Comparison of average turnaround time in each algorithm.

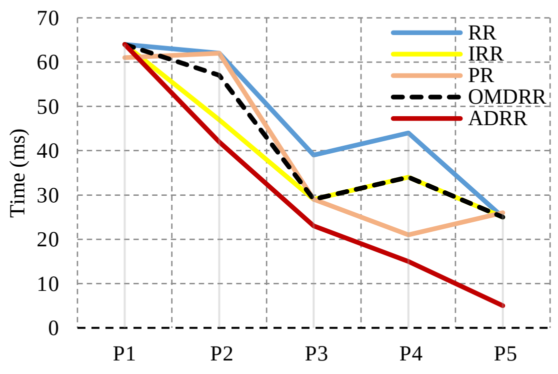

Figure 7. Simulation comparison of turnaround time of each process.

We discuss another example, i.e., Example B. We take burst time of each process randomly and assume QT as 10ms in this case. The order of the processes becomes: P1, P5, P1, P5, P3, P2, and P4. If we compute the values for this case, the total number of context switches is 7, the average waiting time is 42ms and the average turnaround time is 73ms. Figure 8 depicts the average waiting time in each algorithm, conventional RR gives highest value of average waiting time as compared to other competing round robin flavors. Proposed algorithm gives the smallest average waiting time. If average waiting time is small that's mean throughput of the processes will be high.

We discuss the waiting time of each process in each algorithm for Example B. The numerical results obtained in this case are plotted in Figure 8.

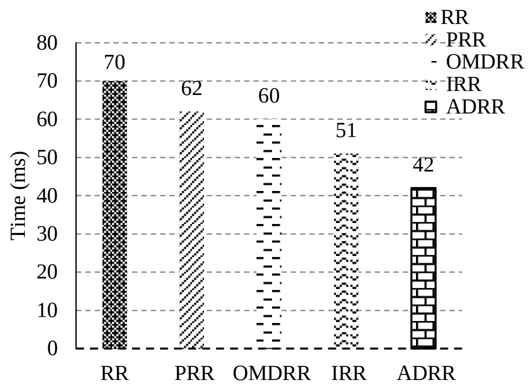

Figure 8. Comparison of average waiting time in each algorithm.

The simulation is run to compare the waiting time for each process for Example B. It could be observed in Figure 9 that P4 has least CPU burst time. Waiting time increases with the increase in the burst time. Results prove that waiting time of each process is lower in ADRR when compared with the waiting times of the processes in other algorithms.

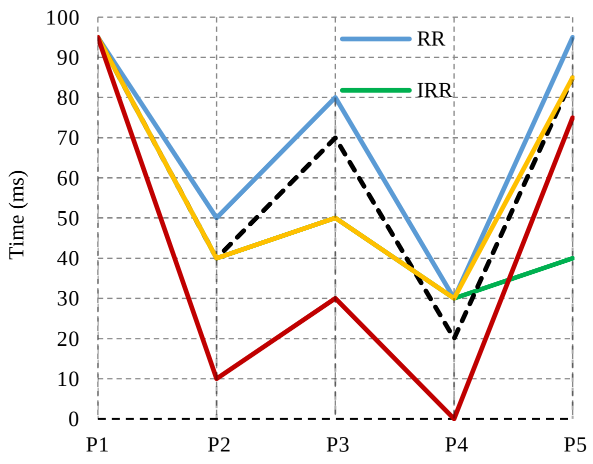

Figure 9. Simulation comparison of waiting time of each process.

In example B, context switch comparison is shown in Figure 10. It could be observed that the efficient design of ADRR results in lesser context switches. In ADRR, there is small number of context switches as compared to selected algorithms in the analysis.

The simulation comparison of turnaround time of each process is shown in Figure 11. We can see that P4 has least turnaround time in all algorithms in example B. Statistical status of Figure-12 shows that ADRR gives better result. Graph shows that among all selected algorithms, ADRR gives least turnaround time for all processes.

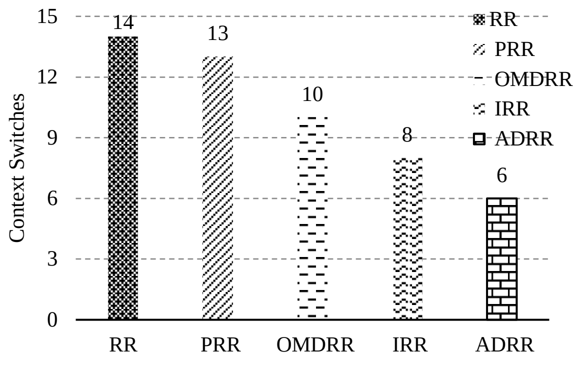

Figure 10. Comparison of number of context switches in each algorithm.

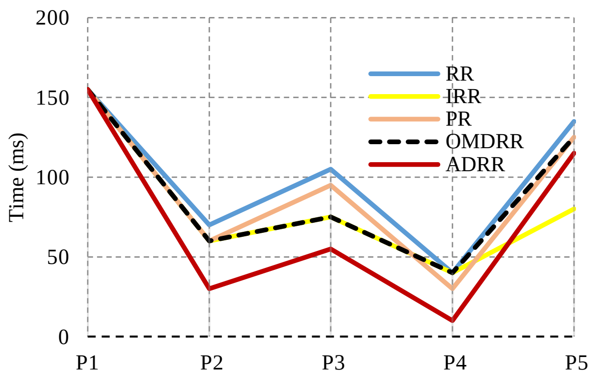

Figure 11. Simulation comparison of turnaround time.

The comparison between the average turnaround times of all algorithms being compared in example B is shown in Figure 13 Average waiting time in ADRR gives smallest value.

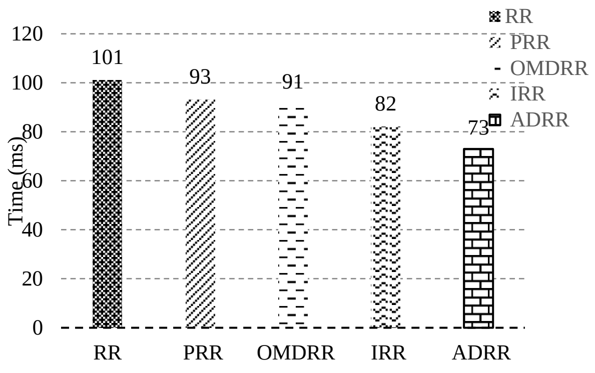

Figure 12. Comparison of average waiting time in each algorithm.

a) *Performance Comparison between RR, IRR, OMDRR, PRR, and ADRR*: Average waiting time and average turnaround time calculated from each algorithm for example A and B is provided in Table 2.

Table 2. Performance comparison of different algorithms.

Algorithm	Example A Average Waiting Time (ms)	Example B Average Waiting Time (ms)	Example A Average Turnaround Time (ms)	Example B Average Turnarond Time (ms)	Example A No. of Context Switches	Example B No. of Context Switches
RR [20]	34	73	46.8	101	13	14
IRR [23]	26	50	39.8	82	8	8
OMDRR [4]	28.6	60	41.5	91	9	10
PRR[15]	26.8	50	39.8	93	8	8
ADRR	17	42	29.8	73	4	7

b) *More Numerical Examples*: We consider three more examples, i.e., Example C, D and E. The set of input for these examples is provided in Table 3. Note that QT has been set to 5ms.

Table 3. Set of input for examples C, D, and E.

Process ID	Arrival time (ms)	Example C		Example D		Example E	
		Burst Time ms	Priority	Burst Time ms	Priority	Burst Time ms	Priority
P1	0.5	22	5	20	3	7	4
P2	1	34	2	12	5	22	5
P3	2	27	3	32	1	34	2
P4	3	50	1	10	4	5	1
P5	4	17	4	41	2	35	3

We calculate the average waiting time for five sets of input for each algorithm. The values obtained in this case are provides in Table 4.

Table 4. Performance comparison of different algorithms for waiting time (WT) for examples A-E.

Algorithm	Ex. A WT(ms)	Ex. B WT(ms)	Ex. C WT (ms)	Ex. D WT (ms)	Ex. E WT (ms)	Avg WT(ms)
RR [20]	34	70	89..4	55.59	43.79	58.55
IRR [23]	27	51	54.4	39.6	32.8	40.96
OMDRR [4]	28.6	60	56.8	45.6	35.2	45.24
PRR [15]	27	62	84.4	52.6	40.8	53.36
ADRR	17	42	48.4	33.6	27.8	33.76

c) *Summary and Findings*: The numerical results provided in Table 4 are plotted in Figure 13. It could be clearly observed that the proposed ADRR gives far more batter results for five different sets of inputs when compared with other well-known CPU scheduling algorithms. We proved with the help of many experiments that the proposed method gives less average waiting, turnaround time and small number of context switches as paralleled to conventional round robin [6], Optimum Multilevel Dynamic round robin [4], improved round robin [23] and priority based round robin [15].

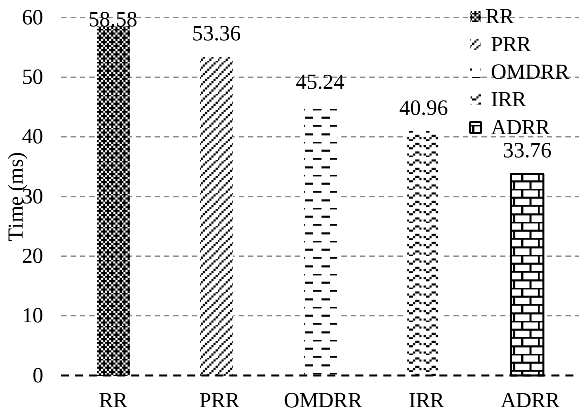

Figure 13. Average waiting Time for 5 examples.

5. Conclusions

In this paper, a novel CPU scheduling algorithm named as ADRR Scheduling algorithm is proposed. Some of the salient features of ADRR are dynamicity of QT and multiple number of rounds which yield optimized values of waiting time and number of context switches. We performed both numerical analysis and simulation experiments for the proposed ADRR algorithm. Different examples for different CPU burst times were run and then compared with other well-known scheduling algorithms. The results prove that the proposed ADRR algorithm has outperformed other algorithms in terms of less average waiting time, small number of context switches and less turnaround time. Potential of time sharing systems can be upgraded with the suggested procedure and can be amended in future to improve the working of real time system.

References

[1] Aggarwal H., “Comparative Performance Study of CPU Scheduling Algorithms,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 6, pp. 714-717, 2014.

[2] Behera H., Mohanty R., and Nayak D., “A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and its Performance Analysis,” *International Journal of Computer Science and Communication*, vol. 5, no. 5, pp. 10-15, 2010.

[3] Banerjee P., Banerjee P., and Dhal S., “Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm using Dynamic Time Quantum with Round Robin Scheduling Algorithm using Static Time Quantum,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 1, no. 3, pp. 56-62, 2012.

[4] Chavan S., and P., and Tikekar., “An Improved Optimum Multilevel Dynamic Round Robin Scheduling Algorithm,” *International Journal of*

Scientific and Engineering Research, vol. 4, no. 12, pp. 298-301, 2013.

[5] Dawood A., "Improving Efficiency of Round Robin Scheduling Using Ascending Quantum And Minumim-Maxumum Burst Time," *Journal of University of Anbar for Pure Science*, vol. 6, no. 2, 2012.

[6] Goel N. and Garg R., "A Comparative Study of CPU Scheduling Algorithms," *International Journal of Graphics and Image Processing*, vol. 2, no. 4, pp. 245-251, 2012.

[7] Hiranwal S. and Roy K.,"Adaptive Round Robin Scheduling using Shortest Burst Approach Based on Smart Time Slice," *International Journal of Computer Science and Communication*, vol. 2, no. 2, pp. 319-323, 2011.

[8] Kumar A., Rohal H., and Arya S., "Analysis of CPU Scheduling Policies through Simulation," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 5, pp. 1158-1162, 2013.

[9] Khan M., Hyder S., Ahmad G., and Begum S., "A Group Based Fault Tolerant Scheduling Mechanism to Improve the Application Turnaround Time on Desktop Grids," *The International Arab Journal of Information Technology*, vol. 13, no. 2, pp. 274-280, 2016.

[10] Matarneh R., "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes," *American Journal of Applied Sciences*, vol. 6, no. 10, pp. 1831-1837, 2009.

[11] Mishra M. and Rashid F., "An Improved Round Robin Cpu Scheduling Algorithm with Varying Time Quantum," *International Journal of Computer Science, Engineering and Applications*, vol. 4, no. 4, pp. 1-8, 2014.

[12] Noon A., Kalakech A., and Kadry S., "A New Round Robin Based Scheduling Algorithm for Operating Systems : Dynamic Quantum Using the Mean Average," *International Journal of Computer Science*, vol. 8, no. 3, pp. 224-229, 2011.

[13] Panda S., Dash D., and Rout J., "A Group based Time Quantum Round Robin Algorithm using Min-Max Spread Measure," *International Journal of Computer Applications*, vol. 64, no. 10, pp. 1-7, 2013.

[14] Panda S. and Bhoi S., "An Effective Round Robin Algorithm using Min-Max Dispersion Measure," *International Journal on Computer Science and Engineering*, vol. 4, no. 1, pp. 45-53, 2012.

[15] Rajput I. and Gupta D., "A Priority Based Round Robin CPU Scheduling Algorithm for Real Time Systems," *International Journal of Innovations in Engineering and Technology*, vol. 1, no. 3, pp. 1-11, 2012.

[16] Singh P., Singh V., and Pandey A., "Analysis and Comparison of CPU Scheduling Algorithms," *International Journal of Emerging Technology and Advanced Engineering*, vol. 4, no. 1, pp. 91-95, 2014.

[17] Singh V. and Gabba T., "Comparative Study of Processes Scheduling," *International Journal of Computing and Business Research*, vol. 4, no. 2, 2013.

[18] Singh A., Goyal P., and Batra S., "An Optimized Round Robin Scheduling Algorithm for CPU Scheduling," *International Journal on Computer Science and Engineering*, vol. 2, no. 7, pp. 2383-2385, 2010.

[19] Shyam R. and Nandal S., "Improved Mean Round Robin with Shortest Job First Scheduling," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 7, pp. 170-179, 2014.

[20] Somani J. and Chhatwani P., "Comparative Study of Different CPU Scheduling Algorithms," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 11, pp. 310-318, 2013

[21] Varma P., "A Best Possible Time Quantum for Improving Shortest Remaining Burst Round Robin Algorithm," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 11, pp. 228-237, 2012.

[22] Varma P., "Improved Shortest Remaining Burst Round Robin Using RMS as its Time Quantum," *International Journal of Advanced Research in Computer Engineering and Technology*, vol. 1, no. 8, pp. 60-64, 2012.

[23] Yadav R., Mishra A., Prakash N., and Sharma H., "An Improved Round Robin Scheduling Algorithm for CPU Scheduling," *International Journal on Computer Science and Engineering*, vol. 2, no. 4, pp. 1064-1066, 2010.

Uferah Shafi holds a BSc degree in Mathematics from Bahauddin Zikariya University, Multan, Pakistan and MSc degree in Information Technology from Quaid-i-Azam University, Islamabad, Pakistan and MS in Computer Science from COMSATS Institute of Information Technology, Islamabad, Pakistan. Her research interests include in pattern recognition and in field of AI. Currently she is doing research and development in Unicorn Black.

Munam Shah received B.Sc and M.Sc degrees, both in Computer Science from University of Peshawar, Pakistan, in 2001 and 2003 respectively. He completed his MS degree in Security Technologies and Applications from University of Surrey, UK, in 2010, and has passed his PhD from University of Bedfordshire, UK in 2013. Since July 2004, he has been a Lecturer, Department of Computer Science, COMSATS University Islamabad, Pakistan. His research interests include MAC protocol design, QoS and security issues in wireless communication systems. Dr. Shah received the Best Paper Award of the International Conference on Automation and Computing in 2012. Dr. Shah is the author of more than 120 research articles.

Abdul Wahid is Assistant Professor in the Department of Computer Science, CIIT. He has completed Ph.D from Kyungpook National University, Rep of Korea. His research interests include but are not limited to Vehicular Ad-hoc Network, Wireless Sensor Network, Underwater Wireless Sensor Network, Cyber Physical Systems, Software defined Networking, Information-centric Networking.

Kamran Abbasi holds a PhD degree in Computer Science from UK and currently serving as Associate Professor at University of Sindh Pakistan, his key research areas are Operating Systems, Computer Networks, Information Systems and Educational Technology.

Qaisar Javaid is working as an Assistant Professor in the Department of Computer Science & Software Engineering at IIUI. He is consistently doing research in the areas of Computer Networks, Information Security, Cloud Computing and IoT. He is also heading the CISCO Networking Academy of International Islamic University, Islamabad, Pakistan.

Muhammad Asghar is serving as an Assistant Professor at Bahauddin Zakariya University Multan. His research interests include video content retrieval and video processing.

Muhammad Haider graduated from Quaid-i-Azam University Islamabad, Pakistan. Currently, he is working as a software developer. His research interests are in reverse engineering and AI.