An Improved Dynamic Round Robin Scheduling Algorithm Based on a Variant Quantum Time

Ahmed Alsheikhy[1], Reda Ammar[1] and Raafat Elfouly[2]
[1]Computer Science and Engineering Dept. University of Connecticut
[2]Computer Engineering Dept. Faculty of Engineering, Cairo University
[1]{ahmed, reda}@engr.uconn.edu
[2]{relfouly@eng.cu.edu.eg}

Abstract—**In real-time embedded systems, scheduling policy is considered one of the main factors that affect their performance. It helps to choose which task should be selected first from ready queue to run. Round Robin (RR) scheduling algorithm is widely used and its performance highly depends on a Quantum size Q_t, which is a predefined amount of time assigned by CPU to every task to be executed. However, the performance degrades with respect to an average waiting time (AWT), an average Turn-Around time (ATT) and a number of Context Switches (NCS). This paper presents a new improved dynamic Round Robin scheduling algorithm to reduce the average waiting time, turn-around time and the number of context switches in order to improve the system overall performance. It also presents a comparative analysis between several existing Round Robin algorithms based on the average time for waiting and turn-around and number of context switches.**

Index Terms—**Round Robin (RR), scheduling algorithms, quantum time, average waiting time, system performance, turn-around time, context switch, real-time embedded systems**

I. Introduction

Operating system is the interface between a user and a machine and it has many features to deliver an excellent service to the user. Scheduling is one of that fundamental features and it is responsible about deciding which job is selected and run from ready queue [1]. Scheduling method affects CPU performance since it determines the CPU and resources utilizations [1, 2]. The main purpose of scheduling policy is to ensure completely fairness between different tasks in the ready queue, maximizing the throughput, minimizing the average waiting and turn-around times and the overhead occurs from context switches, and makes sure no starvation happens at all.

Two schemes of scheduling algorithms exist today which are 1. Preemptive algorithms: where a task is blocked by a higher priority process; and 2. Non preemptive algorithms: where the task completes its execution time even if a higher priority process has arrived [1]. Several factors are used to determine whether a scheduling policy is good or not and can be summarized as follows: A. Waiting time: the time between the task becomes available until the first time of its execution; B. CPU Utilization: the percentage of the CPU being busy; C. Turn-around time: the summation of waiting and execution time for each task and D. Fairness: which is dividing the CPU time equally among all available jobs [1,2,3].
Multiple algorithms exist which can be summarized as follows: 1) First-Come-First-Serve (FCFS): a process arrives first is immediately allocated to the CPU. The major disadvantage of this algorithm is that a process with small burst time takes long time waiting to be executed if another process with long burst is chosen first. 2) Shortest Job First (SJF): this approach allocates processes with short bursts first from their ready queue. It is more efficient than FCFS since it minimizes the average waiting time for small jobs. However, processes with long burst time wait longer which cause a starvation for CPU resources. The drawback of this method is that it requires advance knowledge about CPU burst time which is impractical and difficult in most cases. 3) Round Robin (RR): each process gets its turn to be executed in a fairly time slicing; this concept is known as Time Quantum and it is fixed for all processes. When the quantum time for any process expires, it is temporarily blocked and placed at the end of the ready queue. This procedure is applied on all available tasks until no more tasks exist in the ready queue. This algorithm's efficiency depends totally on the quantum time; if it is small amount then frequent context switch occurs which causes too much of overhead. On the other hand, too long quantum time increases the average waiting and turn-around times. 4) Earliest Deadline First (EDF): a process with shortest deadline time gets its turn first since it has a highest priority among all other processes. This algorithm is considered optimal can be used for both types of tasks (periodic and aperiodic). 5) Multilevel Feedback Queues (MFQ): a process moves between different queues; this action is characterized by the CPU burst time (Bt). If the process requires too much time then it moves to a queue where lower priorities processes are placed. However, if it waits long time then it moves to the queue of the higher priorities processes to prevent starvation from occurring [3,4,5,6].

Our contribution in this paper is done by developing a new improved dynamic Round Robin (RR) algorithm that works either on a uniprocessor or multiple processors systems. The main objective of this algorithm is to maximize throughput while minimizing the average waiting and turn-around times along with the overhead occurring from context switching between several tasks.

In the remainder of this paper, we present related work on scheduling schemes in Section 2, followed by a detailed discussion of the proposed approach in section 3. Section 4 includes two case studies to show the validation of the proposed approach. Section 5 includes a comparative analysis between different existing Round Robin schemes. Section 6 is the conclusion of the paper.

II. Related work

In today's technology, many operating systems perform multitasking operations which mainly depend on scheduling algorithms to ensure that all processes meet their deadline times and execute fairly [2]. Multitasking can be defined as a concept of performing multiple operations at the same time. However, it does not imply that all tasks, also known as processes, are executed in parallel.

In Round Robin, a concept called time slicing is used where each process gets same amount of time for its execution and this concept is known as quantum time.

In [1], A. R. Dash et al proposed an optimized Round Robin algorithm with dynamic quantum time. He claimed that their approach is the optimal one. However, our proposed scheme achieves better results in terms of the average waiting time, average turn-around time and number of context switches. They named their algorithm as DABRR which stands for Dynamic Average Burst time Round Robin. We conducted a comparison analysis between DABRR and our proposed method in section 5 and also several versions of Round Robin are included in the comparison. In [3], D. Maste et al proposed an intelligent dynamic Round robin algorithm for multilevel feedback queues. Each queue is assigned a time slice and it changes with each round of execution dynamically. They used Neural Networks (NN) to control the value of the quantum time to optimize the turn-around time. To find the dynamic quantum time, they considered burst time span as a method to obtain the quantum time with help from average priorities and highest priority of a queue. The overhead occurred from their approach was higher than expected whereas our scheme minimizes the overhead since the quantum size is bigger so each process gets enough time to complete its execution time if possible which implies that the overhead will be minimum. In [4], A. Noon et al proposed a new dynamic Round Robin scheduling algorithm using the mean average as a method to compute a new value for quantum time in each round. The operating system decides the value of quantum time based on the burst time of the existed set of tasks in the ready queue. Their algorithm gives a better result in terms of the average waiting and turn-around times compared to the static Round Robin scheme. However, those values are still high and more modification is needed in order to achieve better results. In addition, the number of context switches occurs in their scheme is still high and causes too much overhead; our approach achieves less number of context switches which implies that less overheard occurs.

I. S. Rajput and D. Gupta in [5] developed a priority based Round Robin scheduling algorithm for real-time systems. Their proposed architecture was implemented to gain a good performance in terms of context switch and the average waiting and turn-around times in the static version of Round Robin scheme. This algorithm did not provide more improvement in the context switches and the average waiting and turn-around times while our scheme gives a better performance since all previous parameters are minimized as much as possible. In [6], H. S. Behera et al proposed a new dynamic Round Robin algorithm to determine the quantum time value by considering number of tasks N in the ready queue. If it is odd, then the quantum value is just the middle burst time in the ready queue or the quantum will be the average value of the middle burst time plus its successor one for even number of tasks in the ready queue. Their method showed a good results in decreasing the performance parameters as shown in section 5. On the other hand, our scheme achieves better results as shown in section 5.

D. Nayek et al in [7] developed an improved Round Robin scheduling scheme using dynamic quantum time; they used a combination method of obtaining the median value M of burst time for all available tasks in the ready queue. Then, taking the average value of that median plus the highest burst time existed in the queue. Their approach gave the best results among all previous schemes mentioned earlier. They claimed that their quantum time estimation is optimal. However, our scheme achieves better results since we obtained the minimum values for all three performance parameters.

III. The Proposed Algorithm

Many real-time systems, such as servers, Android, industrial plant monitoring where an alert might be produced after a set of reading from sensors reached a certain level of hazard detection and anti-lock brake (ABS) in cars, have aperiodic tasks which need to be processed as fast as possible; which in turn means that their response time must be as minimum as possible. The proposed algorithm is suitable for any real-time system with aperiodic tasks; we have chosen Android as a case study since some versions of it still use Round Robin as scheduling policy beside other schemes. In Android platform, static Round Robin is used and the proposed algorithm fits and can produce a significant reduction in response time; the average expected improvement in delay is around 40% to 55% when compared to the static approach. Furthermore, we achieved the minimum number of context switches as proved by a simulation system we developed to test and show the validation of the proposed method.

In Round Robin (RR) algorithm, the performance mainly depends on the size of its quantum time (Q_t); so a small size gives poor performance while too large size tends the algorithm to be "FCFS". So choosing size of the quantum time is very critical to enhance the system performance and this reason was the motivation behind the proposed approach. The size of quantum was selected to be large enough in order to accommodate more processes to finish their execution times to minimize the overhead occurring from context switches. However, that size must not lead the proposed algorithm to act exactly like FCFS. So to choose large time slice "quantum", the average of two highest burst times was computed and then the average of two lowest arrival times was taken from that estimated value for one time only; later, we subtract the average of arrival time for only lowest process. The reason behind that is to keep the quantum as large as possible while maintaining its properties as Round Robin method.

The following pseudo code describes how the proposed scheme works.

Algorithm: Improved Dynamic Round Robin

```
1.  Burst Time $B_t$; Arrival Time $Ar_t$; Tasks T; Average
    Waiting Time AWT; Average Turn-Around Time
    ATT; Number of Context Switches NCS; Quantum
    Time $Q_t$
2.  Initialization: AWT = 0, ATT = 0 and NCS = 0
3.  While (ready queue != 0)
4.  If (all arrival times $Ar_t$ == 0), then
5.  Sort all tasks in ascending order based on their
    burst time values
6.  $Q_t = \sum$ of the two highest $B_t / 2$
7.  Assign $Q_t$ to every process in the ready queue to
    proceed
8.  If ([$B_t(i) - Q_t$] == 0), then
9.  Remove $B_i(t)$ from ready queue
10. else
11.  continue proceeding with remaining tasks
12. $Q_t = \sum$ of the highest remaining $B_t$ with old $Q_t / 2$
13. Continue until no more processes in the ready
    queue
14. If a new process arrives, insert it at its right place
15. If ($A_r(i)$ has different value), then
16. $Q_t$ = the first arrived task
17. Assign $Q_t$ to available process in the ready queue
    to proceed; if more processes arrive in the ready
    queue, then,
18. Sort all tasks in ascending order based on their
    burst time values
19. $Q_t = \sum$ of the highest remaining $B_t$ with old $Q_t / 2$
    $- \sum$ of the two lowest remaining $Ar_i / 2$; for once
    only.
20. If ([$B_t(i) - Q_t$] == 0), then
21. Remove $B_i(t)$ from ready queue
22. else
23. For next rounds;
24. $Q_t = \sum$ of the highest remaining $B_t$ with old $Q_t / 2$
    - the lowest remaining $Ar_i / 2$.
25. If new process arrives, just insert it at its right
    position
26. Until (the end of sorted processes is reached)
27. Calculate AWT, ATT and NCS
```

IV. CASE STUDY

Two examples are presented in this paper to demonstrate how the proposed scheme works.

Case 1 (in [4]): Same arrival time for four processes in the ready queue as shown in table 1.

Table 1: available processes

Tasks	Arrival Time (Ar_t)	Burst Time (B_t)
T_1	0	40
T_2	0	20
T_3	0	80
T_4	0	60

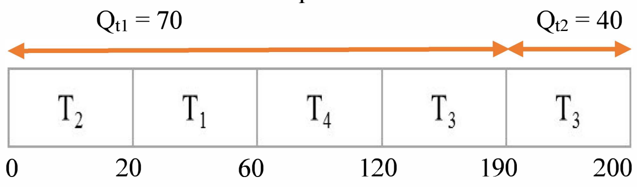
The procedures for the proposed scheme are as follows:

1. Sort all processes according to their burst times as shown in table 2.

Table 2: Sorting all processes

Tasks	Arrival Time (Ar_t)	Burst Time (B_t)
T_2	0	20
T_1	0	40
T_4	0	60
T_3	0	80

2. For round 1: $Q_t = [60 + 80] / 2 = 140 / 2 = 70$ time units
3. For round 2: $Q_t = [70 + 10] / 2 = 80 / 2 = 40$ time units
4. The Gantt chart for all processes is as follows:



5. Table 3 shows the waiting and turn-around times for every process and the average value for both parameters.

Table 3: waiting and turn-around times

Tasks	Waiting Time	Turn-around Time
T_1	20	40
T_2	0	40
T_3	120	200
T_4	60	120
Average	50	100

Case 2 (in [7]): Same arrival times; table 4 lists five tasks with their arrival and burst times.

Table 4: several tasks with same arrival times

Tasks	Arrival Time (A_{rt})	Burst Time (B_t)
T_1	0	80
T_2	0	45
T_3	0	62
T_4	0	34
T_5	0	78

The Gantt chart is as follows:

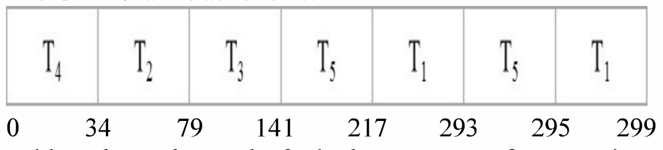


Table 5 shows the results for both parameters after repeating the previous procedures.

Table 5: waiting and turn-around times

Tasks	Waiting Time	Turn-around Time
T_1	219	299
T_2	34	79
T_3	79	141
T_4	0	34
T_5	141	217
Average	94.6	154

Case 3 (in [7]): Different arrival times; five tasks are listed in table 6 with their arrival and burst times. The procedures are the same except that the quantum time Q_t for Round 1 is determined to be the burst time value for the first arrived task.

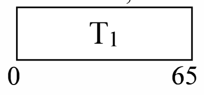
Table 6: several tasks with different arrival times

Tasks	Arrival Time (Ar_t)	Burst Time (B_t)
T_1	0	65
T_2	1	72
T_3	4	50
T_4	6	43
T_5	7	80

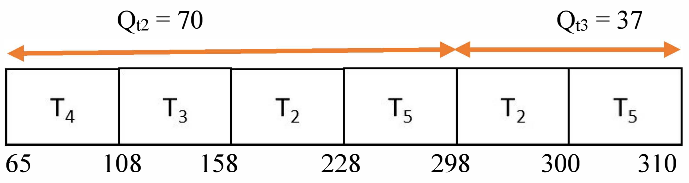
Procedures:

For round 1:

Q_t = 65 time units; the Gantt chart is as follows:



For round 2: several processes are in the ready queue; we repeat the same procedures we did in previous example. Table 6 shows the obtained results for all tasks. The Gantt chart is as follows:



The two performance parameters for Round Robin algorithm are shown in table 7.

Table 7: Waiting and turn-around times

Tasks	Waiting Time	Turn-around Time
T_1	0	65
T_2	227	299
T_3	104	154
T_4	65	102
T_5	223	303
Average	122.6	184.6

V. COMPARISON ANALYSIS

We used several versions from Round Robin algorithm to do a comparison analysis which include static and dynamic. For static version, the quantum time size was chosen to be 25 time units. All tasks with their arrival times and burst times were taken from [1]. For more information about several algorithms of Round Robin being used within this paper, the readers are referred to [1], [6] and [11] respectively. The comparison includes the algorithms DABRR in [1], static Round Robin S.R.R, DQRRR in [6] and SARR in [11] and lastly the proposed approach; it was conducted based on the three performance parameters which are Average Waiting Time (AWT), Average Turn-around Time (ATT) and Number of Context Switches (NCS); due to space limitation, only selected methods were included in the analysis. The goal of this comparison analysis is to show values of all three parameters.

Case 1: same arrival times; table 8 shows five tasks that are in the ready queue with the same arrival time

Table 8: tasks in the ready queue

Tasks	Arrival Time (A_{rt})	Burst Time (B_t)
T_1	0	40
T_2	0	55
T_3	0	60
T_4	0	90
T_5	0	102

The following charts show the comparison analysis between all several Round Robin algorithms mentioned earlier.

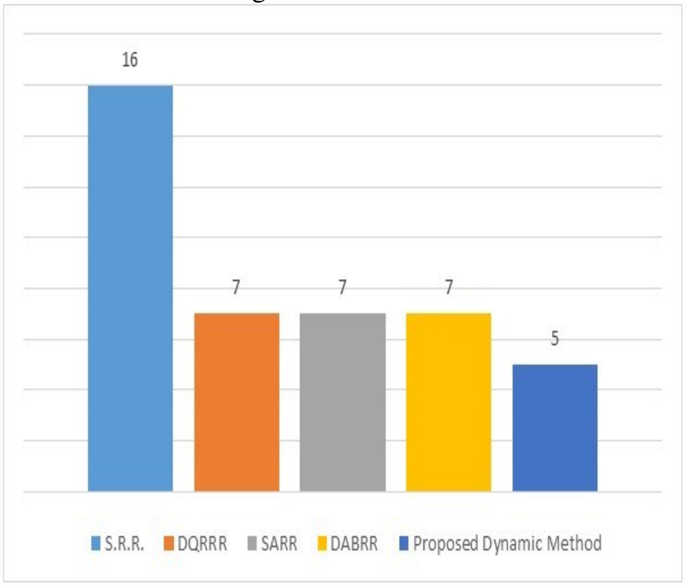


Fig.1: Number of context switches results

The proposed algorithm gave the minimum number of context switches among other four algorithms. However, the proposed scheme yielded a better result for both the average waiting time and the average turn-around time as shown in figures 2 and 3 respectively.

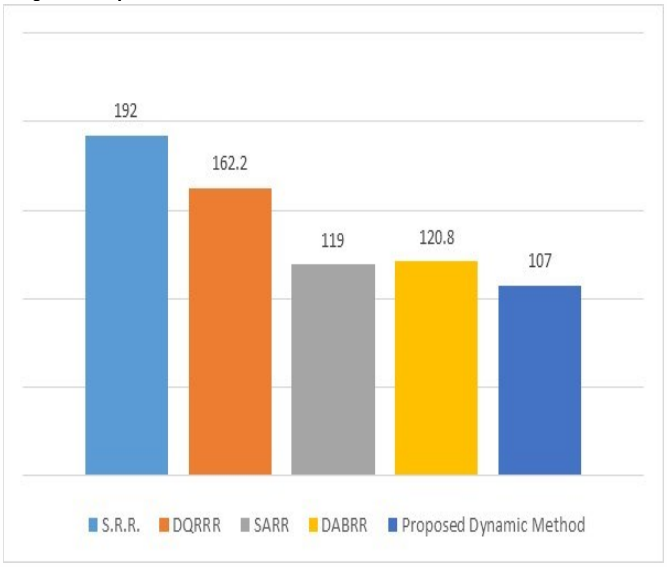


Fig. 2: Average waiting time results

Figure 2 shows that the minimum average waiting time was obtained by the proposed algorithm.

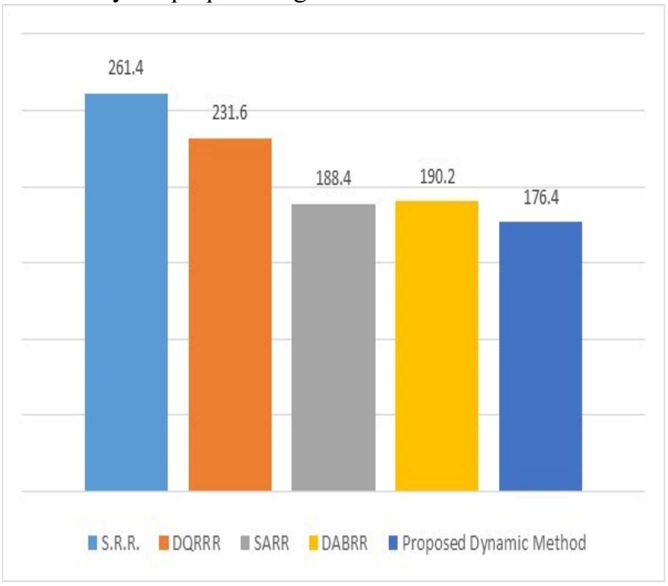


Fig. 3: Average turn-around time results

The proposed algorithm produced the minimum average turn-around time; by improving all three parameters, the throughput is improved too.

Case 2: five tasks with the same arrival times and different burst times are shown in table 9.

Table 9: Tasks in the ready queue

Tasks	Arrival Time (Ar_t)	Burst Time (B_t)
T_1	0	105
T_2	0	85
T_3	0	55
T_4	0	43
T_5	0	35

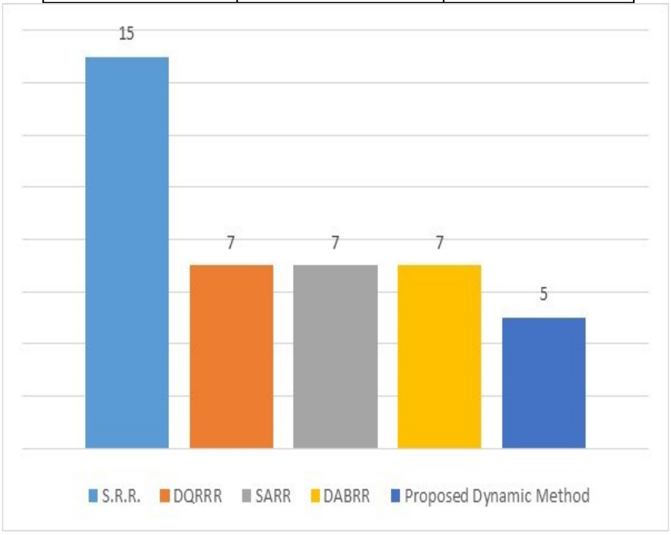


Fig. 4: Number of context switches results

The minimum number of context switches was achieved by the proposed algorithm which is considered the optimal result.

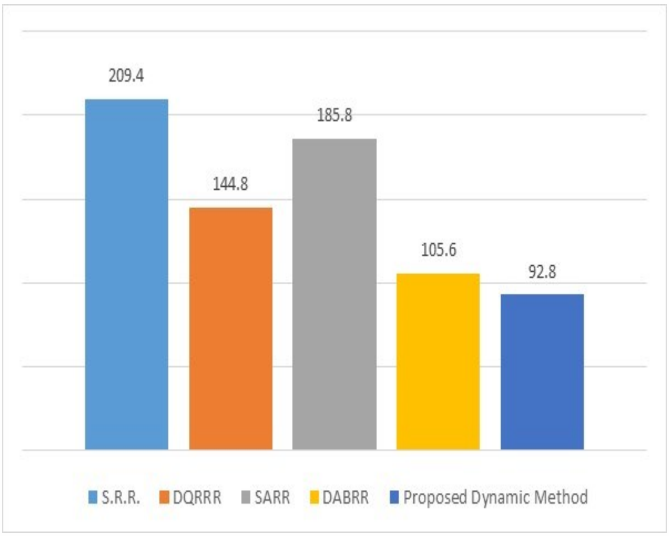


Fig. 5: Average waiting time results

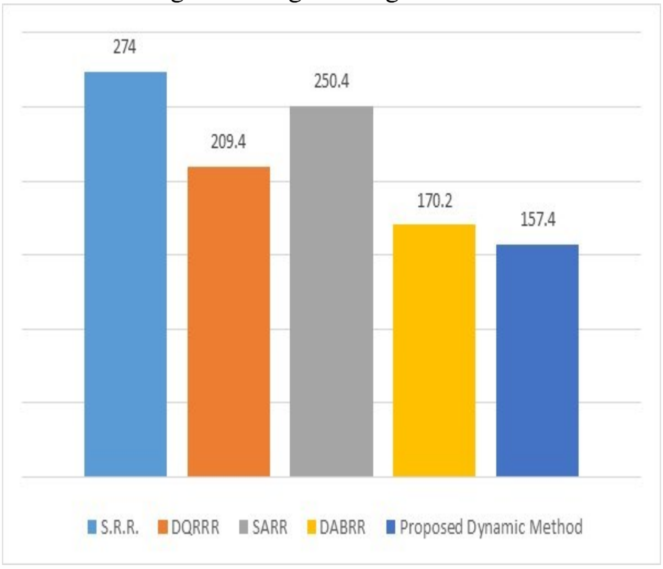


Fig. 6: Average turn-around time results

The proposed algorithm gave the minimum average waiting and turn-around times as shown in figures 5 and 6 respectively.

Case 3: five tasks with the different arrival times and different burst times are shown in table 10.

Table 10: Tasks in the ready queue

Tasks	Arrival Time (Ar_t)	Burst Time (B_t)
T_1	0	45
T_2	5	90
T_3	8	70
T_4	15	38
T_5	20	55

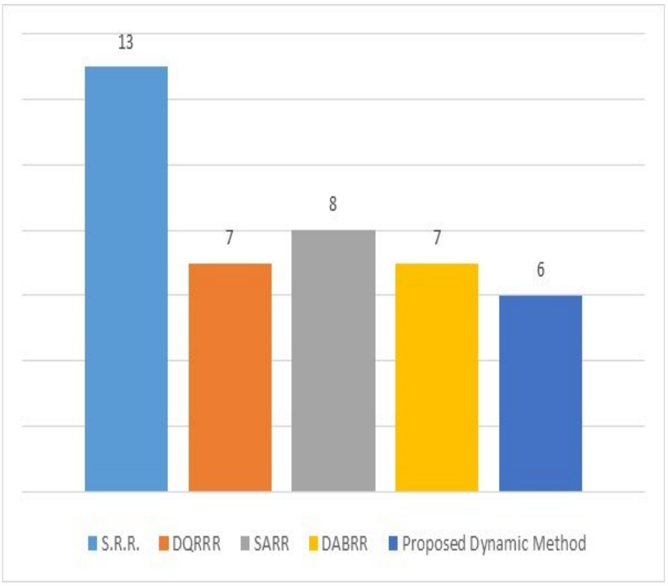


Fig. 7: Number of context switches results

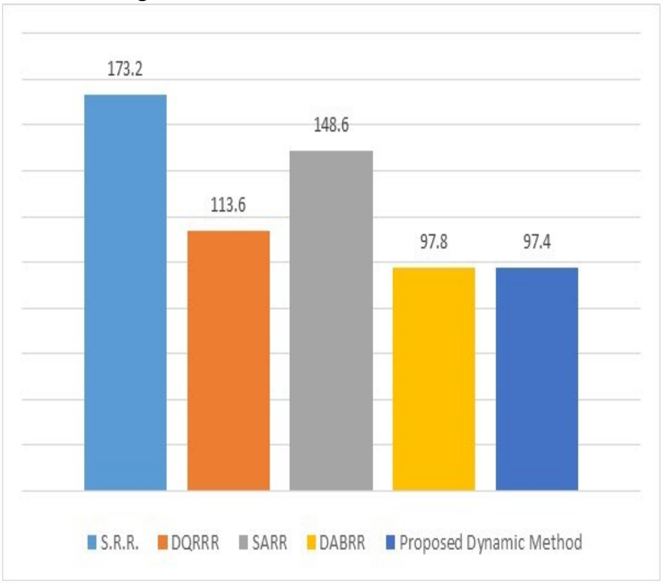


Fig. 8: Average waiting time results

The proposed scheme produced the minimum average waiting time compared to all other algorithms even though the difference between it and DABRR was very slight which around 0.4.

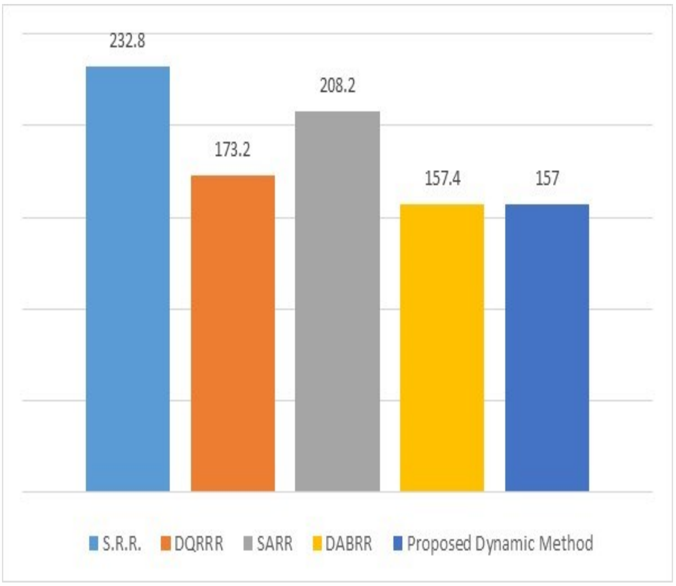


Fig. 9: Average turn-around time results

Our proposed algorithm yields better results for all three performance parameters of the Round Robin as shown in the previous comparison. The number of context switches (NCS) is dramatically decreased in our proposed scheme. All previous cases were given to demonstrate the usefulness of the proposed method; however, we performed more experiments using the developed simulation system. More than 100 tasks were tested more than 7000 times with random execution and arriving times. The maximum number of processes generated by the simulation was around 200 which took around an hour to complete. It is obvious that the elapsed time would be small if all processes had the same arrival times.

VI. CONCLUSION

This paper presented a method to minimize the average waiting time, the average turn-around time and number of context switches. Reducing all three factors implies that the throughput increases which is considered as one of our objectives. Three examples were given to demonstrate how the proposed scheme works. Furthermore, we conducted comparative analysis between five algorithms in terms of the three performance parameters; our scheme gave the minimum results in most all cases shown within this paper or have been done earlier.

REFERENCES

[1] 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 (IJCSEIT), Vol. 5, No. 1, February 2015.

[2] M. S. Iraji, “Time Sharing Algorithm with Dynamic Weighted Harmonic Round Robin”, Journal of Asian Scientific Research, Vol. 5, No. 3, pp. 131-142, 2015

[3] D. Maste, L. Ragha and N. Marathe, “Intelligent Dynamic Time Quantum Allocation in MLFQ Scheduling”, International Journal of Information and Computation Technology, ISSN 0974-2239,

Vol. 3, No. 4, pp. 311-322, 2013. International Research Publications House.

[4] 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 (IJCSI), Vol. 3, Issue 3, No. 1, pp. 224-229, May 2011.

[5] I. S. Rajput and D. Gupta, “A Priority Based Round Robin CPU Scheduling Algorithm for Real Time Systems”, International Journal of Innovations in Engineering and Technology (IJIET), Vol. 1, Issue 3, pp. 1-11, October 2012.

[6] H. S. Behera, R. Mohanty and D. Nayek, “A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis”, International Journal of Computer Applications (0097 – 8887), Vol. 5, No. 5, pp. 10-15, August 2010.

[7] D. Nayek, S. K. Malla and D. Debadarshini, “Improved Round Robin Scheduling using Dynamic Time Quantum”, International Journal of Computer Applications (0097 – 8887), Vol. 38, No. 5, pp. 34-38, January 2012.

[8] 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, August 2014.

[9] A. Singh, P. Goyal and S. Batra, “An Optimized Round Robin Scheduling Algorithm for CPU Scheduling”, International Journal on Computer Science and Engineering, pp. 2383-2385, 2010.

[10] C. Yaashuwanth and R. Ramesh, “A New Scheduling Algorithms for Real-Time Tasks”, International Journal of Computer Science and Information Security (IJCSIS), Vol. 6, No. 2, 2009.

[11] R. Matarneh, “Self-Adjusted 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

[12] S. Kuankid, A. Aurasopon and W. Sa-Ngiamvibool, “Effective Scheduling Algorithm and Scheduler Implementation for Use with Time-Triggered Co-operative Architecture”, Elektronika IR Elektrotechnika, ISSN 1392-1215, Vol. 20, No. 6, pp. 122-127, 2014.

[13] B. Miller, F. Vahid and T. Givargis, “RIOS: A Lightweight Task Scheduler for Embedded Systems”, WESE 12th Proceedings of the workshop on Embedded and Cyber-Physical Systems Education, ACM, New York, NY, USA, 2013.

[14] S. M. Mostafa, S. Z. Rida and S. H. Hamad, “Finding Time Quantum of Round Robin CPU Scheduling Algorithm in General Computing Systems Using Integer Programming”, International Journal of Research and Reviews in Applied Science (IJRRAS), Vol. 5, No. 1, pp. 64-71, October, 2010.

[15] C. J. Hwang, W. Mu and A. Sampat, “Effective Scheduling on Mobile Embedded System”.

[16] R. M. Das, M. L. Prasanna and Sudhashree, “Design and Performance Evaluation of A New Proposed Fittest Job First Dynamic Round Robin (FJFDRR) Scheduling Algorithm”, International Journal of Computer Information Systems (IJCIS), Vol. 2, No. 2, 2007