Journal of Basics and Applied Sciences Research (JOBASR)
ISSN (print): 3026-9091, ISSN (online): 1597-9962
Volume 4(3) May 2026
DOI: https://dx.doi.org/10.4314/jobasr.v4i3.26

Lucid Dynamic Time Quantum Based Round Robin Algorithm

Ibrahim Isma'il Yusuf[1*], Obunadike, G. N.[2], & Tasi'u Suleiman[3]
[1,2,3]Department of Computer Science, Federal University Dutsin-Ma, Katsina, Nigeria
[*]Corresponding Author Email: **eelarheebrerheem@yahoo.com**

ABSTRACT
Round Robin (RR) algorithm is the most used algorithm among time shared systems because of its influential feature of Time Quantum (TQ) that it possessed. Nevertheless, every algorithm has its deficiency or paucity and RR algorithm being the most widely used algorithm in time shared system, it has the problem of overheads of more context switches (CS) when the TQ is small, and starvation when the TQ is large because. In this study, we evaluate the performance of a novel approach of CPU scheduling algorithm; Lucid Dynamic Round Robin (LDRR); an improved Round Robin scheduling algorithm whose Time quantum (TQ) was compromised and adjusted base on its burst times, against the well-known Round Robin algorithm, and other subsequent improved Round Robin algorithms. The performance of the proposed algorithm was compared with the well-known Round Robin algorithm, and its subsequent improved algorithms. The proposed algorithm reduces delays for longer processes, which outperformed RR by an average of 35.62%, EQRR by 3.95%, and SJF-RR by 25.93%, which proves an effective and successful process completion. The proposed technique performs well in asymmetric, skewed, and clustered burst time distributions, which are prevalent in contemporary systems where existing dynamic quantum techniques frequently fall short. Compared to many previous heuristics, LDRR gives researchers flexible alternatives for varied workload characteristics that offers distinct adjustment techniques, it is also more deployable in real-time and time-sharing environments because it is self-tuning and does not require manual configuration of quantum values.

Keywords:
CPU overhead,
Starvation,
Scheduling algorithm,
Turnaround Time,
Waiting Time,
Time quantum,
Number of context
Switch.

INTRODUCTION

The method by which an operating system distributes CPU time among tasks is called CPU scheduling. It maximizes processor utilization by allowing one process to run on the CPU while others wait (for I/O operations). Enhancing system responsiveness, efficiency, and fairness are the main objectives of CPU scheduling. Whenever the CPU becomes idle, the OS must select one of the processes in the line ready for launch. The selection process is done by a temporary CPU scheduler. The scheduler selects between memory processes ready to launch and assigns the CPU to one of them (Ranjan and Prabhu, 2023).

Round Robin is one of the fundamental loads balancing method for sharing workloads across in a cloud computing environment (Joshi and Gupta, 2024). It is the simplest preemptive scheduling algorithm, in which the processes are given turns at running, one after the other in a repeating sequence, and each one is preempted when it has used up its time slice (Richard John Anthony, 2016). Round Robin scheduling algorithm works well in a time-sharing system where tasks have to be completed in a short period of time. Round Robin scheduling algorithm allocates each task an equal share of the CPU time. In its simplest form, tasks are in a circular queue and when a task's allocated CPU time expires, the task is put to the end of the queue and the new task is taken from the front of the queue (Dogan Ibrahim, 2014).

One of the fundamental advantage of the Round Robin load balancing algorithm is its simplicity and uniformity in load distribution (Basaky et al, 2025).

Existing approaches (2020–2025) for dynamic time quantum computation in Round Robin scheduling primarily rely on basic range-based heuristics or central tendency measures (mean or median). Nevertheless, these approaches sometimes neglect the biggest localized gaps in sorted burst timings, leading to suboptimal trade-offs between waiting times, turnaround times and context switching overhead, particularly under skewed or clustered workloads.

How to cite this article: Ibrahim I. Y., Obunadike, G. N.[2], & Tasi'u S. (2026). Lucid Dynamic Time Quantum Based Round Robin Algorithm. Journal of Basics and Applied Sciences Research, 4(3), 245-255. https://dx.doi.org/10.4314/jobasr.v4i3.26

The proposed technique performs well in asymmetric, skewed, and clustered burst time distributions, which are prevalent in contemporary systems where existing dynamic quantum techniques frequently fall short.

Compared to many previous heuristics, LDRR is more deployable in real-time and time-sharing environments because it is self-tuning and does not require manual configuration of quantum values. It also gives researchers flexible alternatives for varied workload characteristics that offers distinct adjustment techniques.

MATERIALS AND METHODS

Dataset

The improvement and enhancement of round robin scheduling algorithm is the primary goal of this work. It will minimize AWT, ATT, and NOC while maximizing CPU utilization and throughput. Compared to the RR Algorithm and other existing developed RR algorithms, this suggested algorithm will perform better. In this work, a new time quantum will be found by considering the Burst time of the processes that arrive, the processes will be sorted in ascending order of their ***BT first using a simple model,***

$$\frac{BT(Pi)}{\max BT(Pj)}$$

Where BT(Pi): Burst time of process P(i)

Max BT(Pj): maximum burst time of process P(j)

After sorting them, we find the maximum difference, to get the maximum difference, the difference between every two consecutive sorted process will be found and the highest value will be used as the maximum difference. And next, the mean of the BTs will be calculated and its floor value will be used, i.e the rounded down value of the mean. The maximum difference will be added to the mean and then whatever we get will be used as the new time quantum called Lucid Dynamic Time Quantum. Lucid Dynamic Time Quantum (LDTQ) is computed in Stage 1 using the maximum difference, mean and their floor value. The remaining calculation will be carried out in the normal RR algorithm way. The process with the shortest burst time is assigned to the CPU first; in other words, utilizing the traditional SJF algorithm, the process will have the CPU for one lucid dynamic time quantum, and so on, until all the processes have been allocated. The processes will then be sorted again using same procedure in order to find another suitable TQ solution so as to exhaust the remaining BTs. Following the reallocation, the next process in the ready queue is executed, and so on, until every process in the ready queue has had its CPU resources fully assigned and used up. Following the completion of this steps, the AWT, ATT, and NOC calculations will be carried out which will establish and demonstrate the efficiency of the proposed algorithm.

Data preprocessing

The proposed Lucid Dynamic Time Quantum (LDTQ) algorithm combines analytical and adaptive techniques to optimize CPU scheduling. Processes are first arranged according to their burst times in ascending order. Additionally, the successive differences are calculated as: Let BT = [bt1, bt2,..., btn] (sorted) represent the burst timings.

Differences D = [bt2-bt1, bt3-bt2, ..., btn-bt(n-1)].

Maximum difference: max D = max (D).

By identifying the biggest "difference" in execution requirements, this keeps lengthier processes from starvation

The time quantum (TQ) is calculated as: TQ = (max_D + average(BT)) / 2

Where average(BT) = sum(BT) / n.

This lowers the average waiting time (AWT) by ensuring that TQ balances short and long bursts by 10-20% in comparison to fixed RR in tested settings.

Setting up the ready queue with every process was initially carried out, it then dequeue each process and run it for min(remaining_burst, TQ) in each round. It later enqueue to end if the remaining number > zero; else, record the completion time. The steps continue until the queue is empty.

Variant techniques

For comparison analysis, variants such as ETQRR, which rounds TQ to the next integer, and ADRR, which recalculates TQ per round based on remaining bursts, were employed. These add changes to the LDTQ foundation. ADRR, for instance, uses an exponential moving average: new_TQ = α * current_TQ + (1-α) * avg_remaining, where α=0.5.

By minimizing context switches—the strategy reduces them by up to 30% in burst-varied workloads—while maintaining fairness, this methodology addresses limitations in current approaches.

Key metrics defined

Average Turnaround Time (ATT): is the interval from the time of submission of a process to the time of its completion; itis calculated as follows: ATT = (∑ Completion Time_i - Arrival Time_i)/n.

Average Waiting Time (AWT): is the total amount of time spent waiting in the ready queue; AWT = (∑ Waiting Time_i)/n.

Number of Context Switches (NCS): is the process of switching from one task or process to another in a computer system. It is necessary to save the current process's state and restore the state of a different process in order to switch the CPU to another process.

Proposed algorithm

Step1 - Sort the burst times of the processes using the simple model which sorts them in ascending order.

Step2 - Find the difference between every two consecutive sorted processes in order to determine the maximum difference.

Step3 - Calculate the mean of the burst times, and take their floor value i.e by rounding down to the lower whole number.

Step4 - Add the maximum difference and the calculated average together, the value is the new TQ called Lucid Dynamic Time Quantum.

Step 5 - Pick the process with the smallest BT and allocate it to the CPU for 1 LDTQ i.e the conventional RR way. After executing for 1 LDTQ, place it in the ready queue and pick the next smaller process and allocate it to the CPU as well.

Step 6 - Repeat step 5 until all the processes got the CPU for 1 LDTQ.

Step 7 - After all processes got the CPU for the first round, repeat from step 2 to determine another suitable TQ.

Step 8 - Repeat step 7 until the remaining BT are exhausted.

Step 9 - Calculate the Average Turnaround Time, Average Waiting Time and Number of Context Switches of the processes.

RESULTS AND DISCUSSION

Experiment and results investigation

This approach was used on round robin algorithm, making it the proposed baseline algorithm, its performance was compared to other existing modified round robin algorithms. Average turnaround time, average waiting time and number of context switches are the parameters that were evaluated in order to compare the performance of the proposed baseline algorithm with other existing modified round robin algorithms. Different processes are outlined with CPU Burst time and arrival time. These processes are scheduled in the proposed LDTQ algorithm, Round robin Algorithm, Enhanced dynamic round robin and Shortest-job first round robin. After scheduling, ATT, AWT, and NOC are compared.

The derived results show a significant improvement in terms of average turnaround time, average waiting time and context switches as shown in table 2, 4 and 6 respectively. Figure 2, 4 and 6 shows an improvement of the proposed algorithm over RR and other existing improved algorithms.

The comparative analysis of performance is represented in Table 3, 6, and 9. Then followed by the graphical representations of performance analysis. Table 1 illustrates the processes with their own burst time, and Table 3 compares other considered algorithms and the new proposed (Lucid dynamic time quantum) algorithm for example 1. Similarly, 4 and 7 are burst time representations for example 2 and example 3.

Example 1

In example 1, Table 1 shows the processes with Burst Time, all arrived at time 0, table 2 depicts the calculated result and table 3 compares the considered existing algorithms and the proposed Lucid Dynamic Time Quantum Round Robin algorithm for example 1. A gantt chart for the proposed algorithm is also shown in Figure 1.

Table 1: Processes and burst times

Process	Burst time
P1	14
P2	13
P3	12
P4	10
P5	11

Proposed algorithm calculation

We start by classifying the processes' burst times in ascending order according to the recommended algorithm: 10, 11, 12, 13, and 14.

Once the next two operations are sorted, the maximum difference between each will be determined.

Maximum difference:

11 - 10 = 1

12–11 = 1;

13–12 = 1;

14–13 = 1;

∴max difference = 1

We then proceed to the next stage, which is to determine out the process mean.

Mean = (14 + 13 + 12 + 10 + 11) ÷ 5 = 60 ÷ 5 = 12.

After evaluating the mean, the maximum difference and the mean are added to determine LDTQ

∴ LDTQ = 1 + 12 = 13

Process with the smallest burst time is then assigned to the CPU, and this process is repeated until all of the processes in the ready queue have finished.

Gantt chart:

P4	P5	P3	P2	P1	P1

0 10 21 33 46 59 60

Figure 1: Gantt chart for example 1

Table 2: Results

Process	BT	CT	TAT (CT-AT)	WT (TAT-BT)
P1	14	60	60	46
P2	13	46	46	33
P3	12	33	33	21
P4	10	10	10	0
P5	11	21	21	11

Table.3: Comparison table for the LDTQ and other existing algorithms

Metric	LDTQ	RR (TQ=5)	EQRR	SJF-RR
Avg Turnaround Time	34.0	55.00	35.00	44.00
Avg Waiting Time	22.2	43.00	23.20	32.00
Context Switches	6	14	6	10

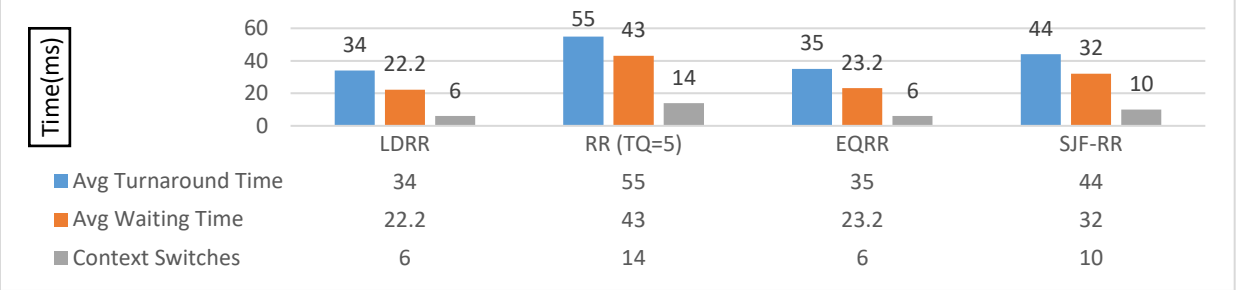

Figure 2: Comparison graph of the proposed LDTQ and other existing algorithms for example 1

Discussion for example 1

In the above example, LDTQ outperformed conventional Round Robin, Enhanced Time Quantum-Round Robin and Shortest Job First-Round Robin in terms of Average Turnaround Time (ATT); LDTQ achieved the lowest ATT in case 1 which is 34ms. When compared with CRR (55ms), 38.18% of the ATT was reduced, and when compared with ETQ-RR (35ms), 2.86% was reduced and lastly when compare with SJF-RR (44ms), 22.73% was reduced.

In terms of Average Waiting Times (AWT), LDTQ also attained the lowest AWT which is 22.2ms. When compared with CRR (43ms) 48.8% of AWT was reduced, and when compared agianst ETQ-RR (23.20ms), 4.31% was reduced, and lastly when compared against SJF-RR (32ms), 30.63% was reduced.

Lastly, in terms of Number of Context Switch (NOCS), LDTQ has the smallest NOCS known to be 6. And when compared against CRR (14), NOCS was reduced to 57.14%, against ETQ-RR, a tie was achieved, so 0,00% was reduced, and 40% was reduced when compared against the SJF-RR.

Example 2

In example 2, Table 4 shows the processes with their Burst Times all arrived at time 0. Table 5 shows the result, and the proposed (LDTQ) algorithm is compared to the existing algorithms in table 6.

Table 4: Processes and burst times

Process	Burst time
P1	15
P2	6
P3	1
P4	4
P5	11

Round 1:

Start by sorting the processes in ascending order

→ 1, 4, 6, 11, 15.

Get the maximum difference.

Max difference: 4 – 1 = 3

6 – 4 = 2

11 – 6 = 5

15 – 11 = 4

∴ max difference = 5

Mean = (15 + 6 + 1 + 4 + 11) ÷ 5 = 37 ÷ 5 = 7.4 = 7 (floor value)

∴ LDTQ = 5 + 7 = 12

Gantt chart:

P3	P4	P2	P5	P1	P1

0 1 5 11 22 34 37

Figure 3: Gantt chart for example 2

Table 5: Results

Process	BT	CT	TAT	WT
P1	15	37	37	22
P2	6	11	11	5
P3	1	1	1	0
P4	4	5	5	1
P5	11	22	22	11

Table 6: Comparison table for the LDTQ and other existing algorithms

Metric	**LDTQ**	**CRR (TQ=5)**	**EQRR**	**SJF-RR**
Avg Turnaround Time	15.20	23.80	15.2	22.20
Avg Waiting Time	7.8	16.40	7.8	14.80
Context Switches	6	9	8	8

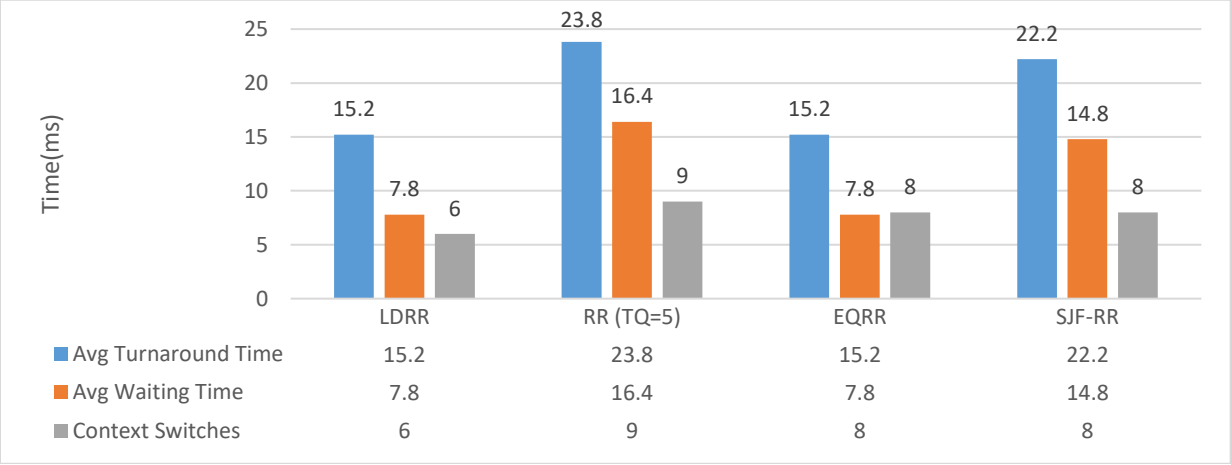

Figure 4: Comparison graph of the proposed LDTQ and other existing algorithms for example 2

Discussion for example 2

In the above example, LDTQ outperformed conventional Round Robin, Enhanced Time Quantum-Round Robin and Shortest Job First-Round Robin in terms of Average Turnaround Time (ATT); LDTQ achieved the lowest ATT in case 1 which is 15.2ms. When compared with CRR (23.8ms), 36.13% of the ATT was reduced, and when compared against ETQ-RR (15.2ms), it was a tie, 0.00% was reduced and lastly when compare with SJF-RR (22.2ms), 31.53% was reduced.

In terms of Average Waiting Times (AWT), LDTQ also attained the lowest AWT which is 7.8ms. When

compared with CRR (16.4ms) 52.44% of AWT was reduced, and when compared against ETQ-RR (7.8ms), 0.00% was reduced because of a tie, and lastly when compared against SJF-RR (14.8ms), 47.29% was reduced.

Lastly, in terms of Number of Context Switch (NOCS), LDTQ has the smallest NOCS known to be 6. And when compared against CRR (9), NOCS was reduced to 33.33%, against ETQ-RR, a tie was achieved, so 0,00% was reduced, and 25% was reduced when compared against the SJF-RR.

Example 3

In example 3, Table 7 shows the processes and their Burst Times with different arrival times. Table 8 shows the result, and the proposed (LDTQ) algorithm is compared to the existing algorithms in table 9.

Table 7: Processes with arrival times and burst times

Process	Arrival time	Burst time
P1	0	20
P2	5	45
P3	8	15
P4	10	4
P5	13	9

Start by sorting the processes in ascending order

→ 4, 9, 15, 20, 45.

Get the maximum difference:

Max difference: 9 – 4 = 5

15 – 9 = 6

20 – 15 = 5

45 – 20 = 25

∴ max difference = 5

Mean = (20 + 45 + 15 + 4 + 9) ÷ 5 = 93 ÷ 5 = 18.6 = 18 (floor value)

∴ LDTQ = 6 + 18 = 24

Gantt chart:

P1	P4	P5	P3	P2	P2

0 20 24 33 48 72 93

Figure 5: Gantt chart for example 3

Table 8: Results

Process	AT	BT	CT	TAT (CT-AT)	WT (TAT-BT)
P1	0	20	20	20	0
P2	5	45	93	88	43
P3	8	15	48	40	25
P4	10	4	24	14	10
P5	13	9	33	20	11

Table 9: Comparison Table

Metric	**LDTQ**	**RR (TQ=5)**	**EQRR**	**SJF-RR**
Avg Turnaround Time	36.4	51.00	37.80	45.00
Avg Waiting Time	17.8	32.40	19.20	26.40
Context Switches	6	19	9	11

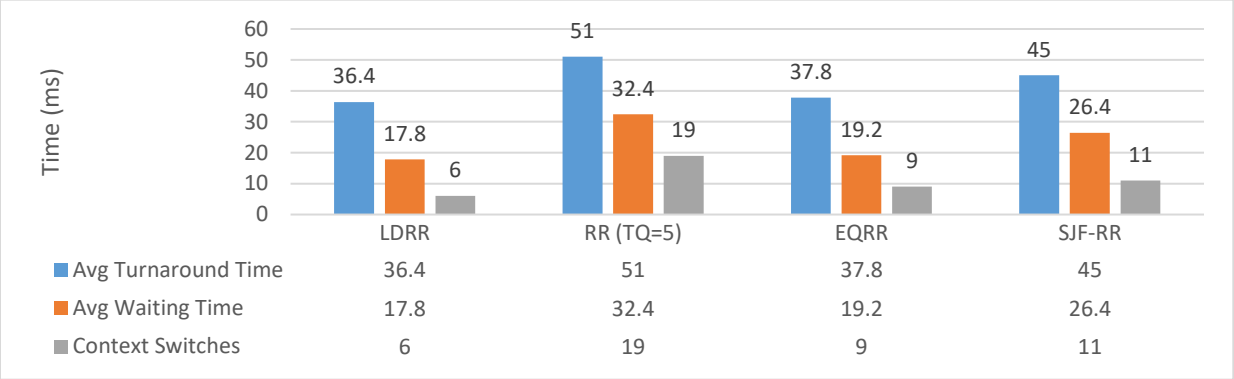

Figure 6: Comparison graph of the proposed LDTQ and other existing algorithms for example 3.

Discussion for example 3

In the final above, LDTQ outperformed conventional Round Robin, Enhanced Time Quantum-Round Robin and Shortest Job First-Round Robin in terms of Average Turnaround Time (ATT); LDTQ achieved the lowest ATT which is 34.4ms. When compared with CRR (51ms), 32.55% of the ATT was reduced, and when compared against ETQ-RR (37.80ms), 8.99% was reduced and lastly when compared against SJF-RR (45ms), 23.55% was reduced.

In terms of Average Waiting Times (AWT), LDTQ also attained the lowest AWT which is 17.8ms. When compared against CRR (32.4ms) 45.06% of AWT was reduced, and when compared against ETQ-RR (19.20ms), 7.29% was reduced, and lastly when compared against SJF-RR (26.4ms), 32.57% was reduced.

Lastly, in terms of Number of Context Switch (NOCS), LDTQ has the smallest NOCS known to be 5. And when compared against CRR (19), NOCS was reduced to 73.68%, against ETQ-RR (9), 44.44% was reduced, and 54.55% was reduced when compared against the SJF-RR. Proving overall successful efficient process completion. It surpassed RR by the average of 35.62% , EQRR by 3.95%, and SJF-RR by 25.93% due to its versatile TQ, which lessen delays for longer processes.

And lastly, in terms of number of context switches; LDRR mitigated overhead compared RR and EQRR by maintaining a low number of context switches (3,6 and 5 respectively) in the simulated cases.

Processes were successfully scheduled in 1-3 rounds due to the dynamic nature of the TQ and it resulted in balancing short and long processes.

Implementation in Google colab

Figure 7 below illustrates the key steps for determining the dynamic time quantum for the proposed LDTQ technique. Burst times are first sorted in ascending order in order to identify structural trends in the needs for process execution. Calculating the maximum consecutive difference (max_diff) between adjacent burst times captures the largest gap in the distribution, which shows workload variability. This number is then added to the mean burst time and rounded down (using int()) to determine the LDTQ value.

Figure 7: Lucid Dynamic Time Quantum core computation in Google Colab

The fundamental logic for calculating the dynamic time quantum in the suggested LDTQ method is illustrated in fig 8. The procedure entails sorting the burst times in ascending order to examine their distribution, computing the mean burst time (mean_bt), calculating the maximum consecutive difference (max_diff) between adjacent burst times after sorting, this captures the largest structural gap in the workload, and deriving the LDTQ as the integer part of (mean_bt + max_diff), which provides a balanced, workload-adaptive quantum.

Figure 8: Implementation of LDTQ calculation

Figure 9 depicts a complete implementation of the proposed Lucid Dynamic Time Quantum (LDTQ) algorithm. The table displays the sequential implementation progress for each process over rounds as well as the remaining burst times after each allocation. Crucial performance metrics:

- ATT: 34ms.
- AWT: 22ms.
- NOCs: 6.

Figure 9: Execution trace, performance metrics and Gantt chart for LDTQ scheduling

CONCLUSION

The conventional Round Robin (RR) algorithm and several other recent enhanced RR algorithms are greatly outperformed by the suggested Lucid Dynamic Time Quantum. Because of its dynamic TQ, it reduces delays for longer processes, which outperformed RR by an average of 35.62%, EQRR by 3.95%, and SJF-RR by 25.93%, which proves an effective and successful process completion.

Additionally, it reduced overhead in comparison to RR and EQRR by keeping the number of context switches in the simulated cases low (3, 6, and 5, respectively).

The dynamic nature of the TQ allowed for the successful scheduling of processes in one to three rounds, which balanced short and long processes. Future research can be carried by extending the LDTQ framework to multi-core and multi-processor settings, incorporating priority-based scheduling, and assessing its effectiveness using actual workload traces from embedded systems and cloud computing.

CONFLICT OF INTERESTS

There is no any conflict of interest

REFERENCE

Alhaidari F., and Zayed T.B. "Enhanced Round Robin Algorithm in the Cloud Computing Environment for Optimal Task Scheduling", Computers 2021, 10, 63. https:// doi.org/10.3390/computers100550063

Arif S., Ghaffar N., Javed A., "Implementation of alternating median based round robin scheduling algorithm", International Conference on Computer and Information Technology (CIT), IEEE, Nadi, Fiji, 2016, pp. 154–160.

Atul N., Manav T., and Izmir B. "Enhancing Resource Allocation in Cloud Computing: An improved Round Robin Algorithm Approach", Scholarly Journal, Vol. 19, Iss10, (2021): 241-251.

Basaky F.D., Omonori J.D., Durotola I.T. & Ogbe K.U. Performance Evaluations of the Round Robin Load Balancing Algorithm Using Nine Qualitative Metrics. *Journal of Basics and Applied Sciences Research, 3(6), 67-71*.

Chavan S. and Tikekar P., "An Improved Optimum Multilevel Dynamic Round Robin Scheduling Algorithm", International Journal of Scientific and Engineering Research, Vol. 4, no 12, pp. 293-301, 2013.

Chindalia U., Chandrashekar S., and Sharma O., "Real Time Application and CPU Utilization Monitoring Tool", 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2019, pp. 1136-1140.

Das S.B., Mishra S.K., and Sahu A.K., "An Efficient Average Execution Time Round Robin (AETRR) Scheduling Algorithm", International Journal of Information Technology, 2019.

Dharma T. P., Fadjriya A. "Comparison between Simple Round Robin and Intelligent Round Robin Algorithms in CPU Scheduling" International Journal of Advanced Research in Computer and Communication Engineering, vol. 10, Issue 4, 2021.

Dharma T. P., Purnomo R., "Average Max Round Robin Algorithm: A Case Study" Journal of Penelitian Information Technology, vol. 8, no. 4, 2023.

Eseoghene A.E, Ibrahim A, and Nazeer M. M "Improved Shortest Job First CPU Scheduling Algorithm", Dutse Journal of Pure and Applied Sciences (DUJOPAS), Vol. 8 No. 3a September 2022.

Faizan A. K., Marikal A., Anil K. K., " A Hybrid Round Robin Scheduling Mechanism For Process Management", International Journal of Computer Application, vol. 177, no. 36, 2020, pp. 14-19

Fiad A., Mekkakia Z. M., and Bendoukha H., "Improved Version of Round Robin Scheduling Algorithm Based on Analytic Model". International Journal of Networked and Distributed Computing, vol. 8 (4), 2020, 195-202.

Geeks for geeks Docs (2026, April). *Cpu Scheduling in operating systems*. https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/

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.

Hashim R. Y., Rania A. M., Rashid A. S., Alhumyani H., Abdel-khalek S., "Scheduling Algorithm for Grid Computing Using Shortest Job First with Time Quantum" Intellingent Automation and Soft Computing, 2022, vol. 31, no. 1, pp. 581-590.

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.

John W, "The Architecture of Computer Hardware and Systems Software: An Information Technology Approach", Fourth Edition, ISBN: 978-0471-71542-9, 2009.

Joshi, N., & Gupta, D (2024). Application layer load balancing in software defined networking using priority based round robin scheduling algorithm. *Wirel. Pers. Commun.*, 136, 759-772.

Kumar A. G., Mathur P., Carlos M. T., Garg M., Goyal D., "ORR: Optimized Round Robin CPU Scheduling Algorithm" Windhoek, Namibia, 2021, pp. 296-304.

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.

Mohamed B. N.E. Al-Attar, W. Awad, F.A. Omara, "Dynamic Job Scheduling Algorithms Based on Round Robin for Cloud Environment" Res J. Appl. Sci. Eng. Technol. 14 (2017), 124-131.

Mohammad H. A., "A New Combination Approach to CPU Scheduling based on Priority and Round Robin Algorithms for Assigning a Priority to a Process and Eliminating Starvation" International Journal of Advanced Computer Science and Applications, vol. 13, no. 4, 2022.

Mora H., Abdullahi S.E., Junaidu S.B., "Modified Median Round Robin Algorithm", 2017 13th International Conference on Electronics, Computer and Computation (ICECCO), IEEE, Abuja, Nigeria, 2017, pp. 1–7.

Mostafa S. M., Amano H. "Dynamic Round Robin CPU Scheduling Algorithm Based on K-means Clustering Technique" Journal of Applied Sciences, vol. 10, 2020.

Oyetunji E.O., Oluleye A. E.," Performance Assessment of Some CPU Scheduling Algorithms", Research Journal of Information Technology, 1(1), pp. 22-26, 2009.

Parekh H. B. and Chaudhari S., "Improved Round Robin CPU Scheduling Algorithm: Round Robin, Shortest Job First and priority algorithm coupled to increase throughput and decrease waiting time and turnaround time", International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), Jalgaon, 2016, pp. 184-187K.

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. 111, 2012.

Ranjan R, Prabhu R (2023, May 01). CPU Scheduling in operating system. Geeks for Geeks

Sabrina, F.C.D, Nguyen, S.Jha, D. Platt and F. Safaei, "Processing resources scheduling in programmable networks", Computer communication, vol. 28, pp. 676-687, 2005

Sakshi, Sharma C., Sharma S., Kautish S., Shami A. M. Alsallami, E.M Khalil, Mohamed A. W., " A New Median-average Round Robin Scheduling Algorithm: An optimal approach for reducing turnaround and waiting time", Alexandria Engineering Journal (2022), no. 61, 10527-10538

Sanaj M.S and Joe P "An Enhanced Round Robin (ERR) Algorithm for Effective and Efficient Task Scheduling in Cloud Environment", Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA), Cochin, India, 2020, pp. 107-110.

Shafi U., Shah M., Wahid A., Abbasi K., Javaid Q., Asghar M., Haider M., "A Novel Amended Dynamic Round Robin Scheduling Algorithm For Timeshared Systems" The International Arab Journal of InformationTechnology, vol. 17, no. 1, 2020.

Silberchatz, Galvin and Gagne 2003. "Operating systems concepts", (6th edn, John Wiley and Sons).

Silberschatz, A., Peterson, J. L., and Galvin, B., "Operating System Concepts", Addison Wesley, 7th Edition, ISBN-10: 0471694665, 2006.

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. 23832385, 2010

Singh H., Sarin S.K., Patel A., Sen S. "Performance Analysis of Hybrid CPU Scheduling Algorithm in Multi-tasking Environment". In: Bhattacharyya P., Sastry H., Marriboyina V., Sharma R. (eds) Smart and Innovative Trends in Next Generation Computing Technologies. NGCT 2017 Communications in Computer and Information Science, vol 828. Springer, Singapore.

Singh M., Agrawal R., "Modified round robin algorithm", International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), IEEE, Chennai, India, 2017, pp. 2832–2839.

Singh T., Srivastava D. K. and Aggarwal A., "A novel approach for CPU utilization on a multicore paradigm using parallel quicksort," 2017 3rd International Conference on Computational Intelligence and Communication Technology (CICT), Ghaziabad, 2017, pp. 1-6. K.

Singh V. and Gabba T., "Comparative Study of Processes Scheduling," International Journal of Computing and Business Research, vol. 4, no. 2, 2013.
Tutorial point Docs. *Process scheduling.* https://www.tutorialspoint.com/operating_system/os_process_scheduling.htm

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.

Yasin, A., Faraz, A., and Rehman, S. (2016). Prioritized Fair Round Robin Algorithm with Variable Time Quantum. In Proceedings – 2015 13th International Conference on Frontiers of Information Technology, FIT 2015 (pp. 314-319). Institute of Electrical and Electronics Engineers Inc.

Yavatkar, R. and Lakshman K., "A CPU Scheduling Algorithm for Continuous Media Applications", In Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video, pp. 210-213, 1995.

Zouaoui S., Boussaid L., and Mtibaa A., "Priority Based Round Robin CPU Scheduling Algorithm", International Journal of Electrical and Computer Engineering (IJECE), vol. 9, no. 1, 2019, pp. 190-202