Implementation of Alternating Median Based Round Robin Scheduling Algorithm

Conference Paper · December 2016

DOI: 10.1109/CIT.2016.25

3 authors, including:

Salman Arif
University of Nottingham
2 PUBLICATIONS **13** CITATIONS

SEE PROFILE

Naveed Ghaffaf
National University of Sciences and Technology
1 PUBLICATION **6** CITATIONS

SEE PROFILE

Implementation of Alternating Median Based Round Robin Scheduling Algorithm

Salman Arif,Naveed Ghaffar,Ali Javed
Department of Computer Engineering
University of Sciences and Technology(NUST)
Islamabad, Pakistan
salmanarif38@gmail.com
naveed35@ce.ceme.edu.pk
alijaved389@yahoo.com

Abstract

The scheduling of tasks run by CPU directly affects the performance of our computer system. Different scheduling algorithms satisfy a limited spectrum of the scheduling criteria that focus on reducing either of response time or number of context switches and waiting time. This paper introduces a novel scheduling algorithm that meets the scheduling criteria at a broader spectrum by giving first priority to the reduction of the number of context switches and waiting time and second priority to reduction of response time. The proposed algorithm also offers a lesser overhead of time quantum calculation. The paper provides the comparison between the proposed and similar algorithms through the numerical analysis and verification of the results. The paper also empirically evaluates the working and behavior of the proposed algorithm.

Keywords—multitasking; starvation; scheduling criteria; prioirty scheduling; scheduling algorithms; time quantum; ready queue

I. INTRODUCTION

There are several processes of our computer system seeking attention of the CPU for execution. These processes or tasks are executed by the CPU undergoing a phenomenon known as multitasking. The execution of these processes determines the performance of our computer system. Appropriate allocation of processes to the CPU is required to avoid starvation which decreases the performance of the system. Different techniques and algorithms had been introduced for the allocation of processes to the CPU for the optimization of the system. A suitable criterion of scheduling was also defined for the scheduling algorithms in order to effectively meet the optimization demands. The scheduling criteria is constituted of waiting time, response time, context switches, turnaround time, throughput and CPU utilization [1].

Initially algorithms were designed using a simpler approach of first come first serve by the First Come First Serve (FCFS) algorithm or focusing the early execution of the processes having smaller burst times by the Shortest Job First (SJF) algorithm. Similarly, the Round Robin (RR) algorithm was implemented giving each process an equal and fixed share of execution until its completion irrespective of the attributes of the process. But there was a tradeoff that the scheduling algorithms had to face while addressing the scheduling criteria. The algorithms were destined to offer compromise by focusing on some factors of the scheduling criteria and neglecting the other factors. For example, one cannot simultaneously reduce waiting time and response time effectively. Some algorithms like RR effectively reduce the response time using smaller time quantum consequently increasing the waiting time. On contrary to RR, SJF effectively reduces waiting time and increases response time as a result. Algorithms like RR implemented the concept of static time quantum using a fixed value of time quantum throughout scheduling. This approach lacked the flexibility in choice of time quantum for the execution of processes. In Priority Scheduling, the processes were assigned priority and the execution of the processes took place based on their priority. In order to cater the limitation in choice of time quantum, the dynamic time quantum was implemented that calculated the time quantum on run-time while the calculation was based on the burst times of processes. The pertinent concept is basically a complicated approach to achieve a greater solution for better performance. It requires a great cost of calculation. Hence the benefit offered by the implementation of dynamic time quantum must be larger than burden of complex and computationally intense calculation it offers.

Some algorithms dynamically calculate time quantum to reduce the waiting time, response time and context switches like AN algorithm [2], SRBRR algorithm [3] and MB algorithm [4]. These algorithms were implemented to offer the least compromise targeting the reduction of all the mentioned factors of scheduling criteria. The increased size of the time quantum reduces the waiting time and the context switches but increases the response time while the decreased size of the time quantum reduces the response time but compromises on the reduction of the waiting time and the context switches [4]. Hence a medium sized time quantum can reduce all the factors to some extent if not to a greater extent instead of effectively targeting some factors. These algorithms use a time quantum that is computationally

978-1-5090-4314-9/16 $25.00 © 2016 IEEE
DOI 10.1109/CIT.2016.25

intensive for calculation and offer a greater overhead for this purpose in order to efficiently meet the scheduling criteria.

This paper proposes a novel scheduling that reduces the three major factors of the scheduling criteria giving by focusing more on reducing waiting time and context switches giving them first priority and second priority to reduction of response time. The proposed algorithm uses a set of two time quanta used alternately in iterations or cycles of scheduling. The two-time quanta are calculated only once in the first cycle of scheduling and remain fixed for the next cycles of scheduling. Hence the set of two time quanta is calculated on run-time only once which requires less overhead of calculation as compared to the other algorithms like AN, SRBRR and MB that also use the dynamic time quantum but calculate it on run-time separately for each cycle of scheduling. The article also illustrates the comparison between the proposed and the mentioned algorithms through numerical verification of results and empirical evaluation on the basis of trend that is carried out by waiting and response time.

II. Alternating Median Based round robin Scgeduling algorithm

The proposed algorithm uses a set of two time quanta for scheduling the ready queue. These time quanta are used alternately in cycles of scheduling. Before the first cycle of scheduling starts, one of the time quanta is set equal to the median of burst time of processes present in the ready queue while the other one is calculated as the difference between the highest value of burst times and the median of burst times of processes present in the ready queue. The two-time quanta remain fixed throughout scheduling and are not recalculated until the architecture of the ready queue gets changed by the arrival of a new process. The larger time quantum is used if even number of cycles of scheduling are completed while the smaller time quantum is used if the odd number of cycles of scheduling are completed. Therefore, the two-time quanta are used alternately in cycles of scheduling. However, at the start, larger time quantum is used as the number of cycles completed is "0" which is an even number. In order to keep the check of the number of cycles completed, we use the cycle completion count denoted by "*c*" representing the number of cycles completed. Hence if "*c*" is even, then the larger time quantum is used for scheduling and if "*c*" is odd, then the smaller time quantum is used for scheduling.

Denoting the greater time quantum by "*g*" and smaller time quantum by "*s*", the formula for the calculation of time quantum "*T*" is shown in (1):

$$T = \begin{cases} g, & c \text{ is even} \\ s, & c \text{ is odd} \end{cases} \quad (1)$$

If the median is denoted by "*m*" and the difference between the highest value of burst times of processes and median is denoted by "*d*", then the formulae for the calculation of "*g*" and "*s*" and are shown in (2) and (3) respectively:

$$g = \begin{cases} m, & m > d \\ d, & m < d \end{cases} \quad (2)$$

$$s = \begin{cases} d, & m > d \\ m, & m < d \end{cases} \quad (3)$$

The flow chart of the AMBRR is shown in Fig. 1.

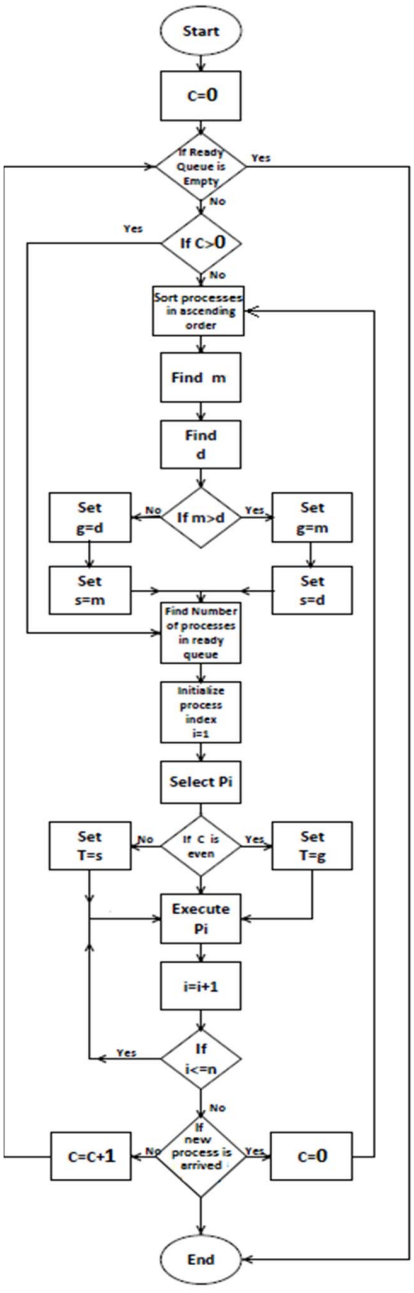

Figure 1: The Flow Chart of AMBRR Algorithm

It must be noted that the following steps of the AMBRR are computationally intensive and they must be executed only if some conditions are satisfied:

- Sorting processes in ascending order of their execution times
- Calculating median of burst time of processes
- Calculation of the difference between the highest value and the median of burst times of processes
- Finding the total no. of processes ready to be executed

We need to focus upon some conditions for the execution of these steps if it is to be occurred more than once. All of these steps need to be executed again if a new process arrives in the ready queue starting from sorting of processes to the recalculation of median and the difference between the highest value of burst times and median leading to finding of number of processes in ready queue. Secondly, if a cycle is completed but the processes in the ready queue do not finish their execution and the ready queue remains non-empty, then the first three steps that represent calculation of two-time quanta must not be executed. This is because the time quanta are calculated only once before the first cycle of scheduling occurs. But, there is a chance that some processes might have finished their execution after the completion of a cycle, so we need to update the number of processes present in the ready queue represented by "*n*" when the value of "c" becomes greater than zero.

It must be noted that the application of algorithm requires an extra memory for using two time quanta instead of one. However, we need to calculate the two-time quanta on run-time only once. This is also a very big advantage as compared to the similar algorithms like AN, SRBRR and MB that face the computational cost of calculation of time quantum for every new cycle of scheduling on run-time. Also, the minimum value of time quantum required to execute all the processes of ready queue in the first cycle of scheduling is equal to the highest value of burst times of processes in the ready. Moreover, the sum of two time quanta "*g*" and "*s*", is always equal to the highest value of burst times of processes in a ready queue. Consequently, the time of execution of processes in a ready queue done by the CPU becomes equal to the highest value of burst times of processes in two cycles. Hence the algorithm completes the scheduling of all the processes in two cycles and the ready queue becomes empty. This phenomenon is clearly evident from the examples in Section II.

III. NUMERICAL COMPARISON AND VERIFICATION OF RESULTS

This section numerically implements the proposed algorithm and compares the results with similar algorithms like AN, SRBRR and MB. The results are then analyzed and verified on the basis of numerical comparisons.

A. First Comparison

Suppose the ready queue contains the following series of processes with their corresponding execution times as given in TABLE I:

TABLE I: Processes for First Comparison

Processes	Execution Time (ms)
Process 1	5
Process 2	8
Process 3	16
Process 4	18
Process 5	24

1) Results

The TABLE II gives the results for the TABLE I processes using algorithms of RR, AN, SRBRR, MB and AMBRR.

TABLE II: Results for First Comparison

Algorithms	Time Quantum (ms)	Waiting Time (ms)	No. of Context Switches	Response Time (ms)
RR	8	25.2	10	13.6
AN	14,5,5	27.2	8	17.2
SRBRR	16,5,3	22	7	18.4
MB	15,4,5	27.8	8	17.8
AMBRR	16,8	17.2	7	18.4

The bar charts for the results shown in TABLE II are shown in Fig. 2.

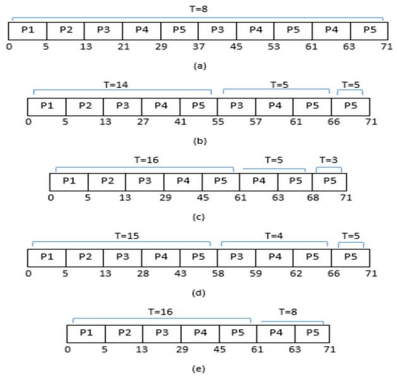

Figure 2: (a) RR, (b) AN, (c) SRBRR, (d) MB, (e) AMBRR

2) Analysis

The In this comparison, the RR algorithm has given the second highest value of waiting time and highest value of context switches but the least value of response time due to the implementation of the time quantum smallest in size which is "8".

Waiting time given by AN is larger than SRBRR, RR and AMBRR because its first-time quantum is smaller than first time quantum of SRBRR and AMBRR due to which P3, P4 and P5 in Fig. 2 (c) and Fig.2 (d) continue to exist in the

second cycle. In case of RR, waiting time of AN is larger because its second and third time quanta i.e. “5” and “5” are larger than the single fixed time quantum of RR which is “8” although first time quantum of AN which is “14” is larger than “8”. AN give waiting time slightly smaller than MB because first time quantum of AN is smaller than first time quantum of MB which is “15”. The reason is that P3, P4 and P5 continue to exit in case of both AN and MB and in case of AN, they have to wait less for processing due to smaller time quantum in first cycle as compared to the larger time quantum of MB in first cycle. Considering context switches, AN produces larger context switches than SRBRR and AMBRR due to having smaller time quanta than SRBRR and AMBRR in every cycle. Both AM and MB have given equal number of context switches due to very small differences in time quanta of all cycles. As compared to RR, first larger time quantum of AN has finished the execution in less number of cycles as shown in Fig. 2 (a) and Fig. 2 (b). AN has first time quantum smaller than SRBRR, MB and AMBRR and larger than RR. By this reason, it gives response time lesser than response times of SRBRR, MB and AMBRR but larger than RR.

SRBRR has waiting time smaller than MB and larger response time and context switches by using larger first and second time quanta than MB. As compared to AMBRR, although first time quantum of both the algorithms are same, SRBRR gives larger waiting time by having smaller second time quantum. However, due to small differences in time quanta considering all cycles of scheduling, both SRBRR and AMBRR have same response time and the number of context switches.

AMBRR has given minimum waiting time and no. of context switches as compared to other algorithms due to implementation largest time quanta in all cycles. However, it has given the largest response time but not with a bigger difference as compared to response times of AN, SRBRR and MB as it gives second priority to its reduction.

B. Second Comparison

Suppose the following series of processes in TABLE III that are ready to be executed:

TABLE III: Processes for Second Comparison

Processes	Execution Time (ms)
Process 1	9
Process 2	12
Process 3	14
Process 4	16
Process 5	20
Process 6	27
Process 7	30

2) Results

The TABLE IV gives results for the processes in TABLE III for algorithms of RR, AN, SRBRR, MB and AMBRR:

TABLE IV: Results for Second Comparison

Algorithms	Time Quantum (ms)	Waiting Time (ms)	No. of Context Switches	Response Time (ms)
RR	14	52.71	11	36.28
AN	18,8,3,1	49.57	12	38.86
SRBRR	16,11,3	47.57	10	38
MB	17,9,3,1	49.2	12	38.4
AMBRR	16,14	47.57	10	38

The bar charts for the results in TABLE IV are shown in Fig. 3.

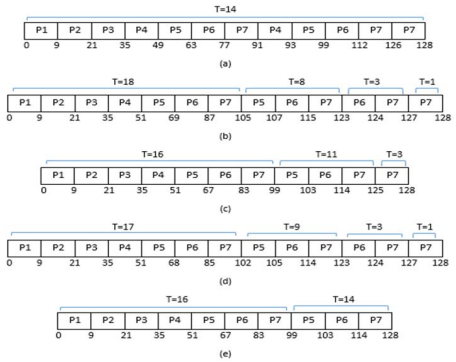

Figure 3: (a) RR, (b) AN, (c) SRBRR, (d) MB, (e) AMBRR

3) Analysis

RR has given the minimum response time and maximum waiting time by using the smallest time quantum i.e. “14” as compared to the first time quanta of AN, SRBRR, MB and AMBRR. However, it has given intermediate number of context switches as compared to the rest of algorithm because its time quantum is largest as compared to the second time quanta of the rest algorithms.

AN has given waiting time larger than SRBRR and AMBRR because it uses largest first time quantum i.e. “18” than the rest of algorithms and smallest second time quantum i.e. “8” due to which processes P5, P6 and P7 in AN wait and tend to exist longer in ready queue as shown in Fig. 3 (b) as compared to SRBRR in Fig. 3 (c) and AMBRR in Fig. 3 (e). However, due to minute difference in second time quanta of MB i.e. “9” and AN i.e. “8” that are smaller than the rest of second time quanta, P5, P6 and P7 also tend to exist longer in case of MB as shown in Fig. 3 (d) similar to AN in Fig. 3 (b) but wait slightly lesser than in case of AN. AN also gives the largest response time response time by using the largest first time quantum which affect has been catered in case of context switches by using smallest second time quantum resulting in maximum number of context switches.

Considering SRBRR, its second-time quantum i.e. “11” is larger than the second and third time quanta of MB that are “9” and “3” and this difference is greater than difference

between the first-time quanta of the two algorithms i.e. between "16" belonging to SRBRR and "17" to MB. Hence the effect of second time quantum dominates and having greater second time quantum, SRBRR gives lesser waiting time and number of context switches than MB. Since SRBRR has smaller first time quantum than MB, hence it gives lesser response time than MB. Due to no difference in first time quanta of SRBRR and AMBRR i.e. "16" and a small difference in second time quanta of SRBRR i.e. "11" and AMBRR i.e. "14", both give equal waiting time, response time and context switches. However, AMBRR having larger second time quantum than SRBRR takes less number of cycles to execute processes and calculates less number of time quanta than SRBRR as shown in Fig. 3 (c) and Fig. 3 (e).

MB has first time quantum "17" slightly larger than the first-time quantum of AMBRR i.e. "16", AMBRR has given lesser response time. Despite of this, AMBRR also gives lesser waiting time and number of context switches because it has a second-time quantum i.e. "14" considerably larger than the second and third time quanta of MB.

It is also clearly depicted that AMBRR has given its first priority to the reduction of waiting time and context switches giving least waiting time and context switches and second priority to the reduction of response time giving second smallest response time. It has also executed all processes in two cycles.

C. Third Copmarison

Consider the series of processes with their execution times in TABLE V.

TABLE V: Processes for Third Comparison

Processes	Execution Time (ms)
Process 1	2
Process 2	4
Process 3	7
Process 4	10
Process 5	12
Process 6	20

2) Results

The algorithms of RR, AN, SRBRR, MB and AMBRR are implemented on the processes of TABLE V. The results for the pertinent implementation of mentioned algorithms are given in TABLE VI.

TABLE VI: Results for Third Comparison

Algorithms	Time Quantum (ms)	Waiting Time (ms)	No. of Context Switches	Response Time (ms)
RR	10	15.5	8	12.83
AN	9,5,6	17.66	9	12.33
SRBRR	9,3,8	17.66	9	12.33
MB	9,4,7	17.66	9	12.33
AMBRR	11,9	15	8	13

The bar charts for the results shown in TABLE VI are shown in Fig. 4.

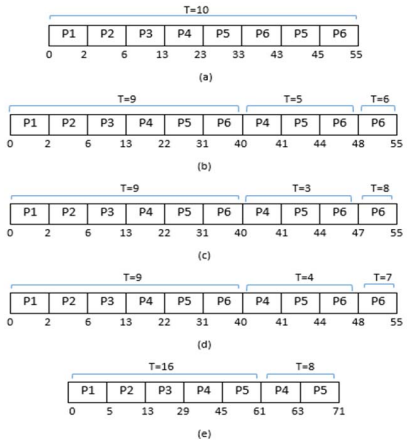

Figure 4: (a) RR, (b) AN, (c) SRBRR, (d) MB, (e) AMBRR

3) Analysis

The RR has given second smallest waiting time, smallest no. of context switches and second largest response time than the rest of algorithms because it uses second largest time quantum i.e. "10".

AN, SRBRR and MB have surprisingly given equal waiting time response and number of context switches because first time quanta of all three algorithms i.e. "9" are same and second time quanta of these algorithms i.e. "5" used by AN, "3" used by SRBRR and "4" used by MB have a very small difference between them. Secondly, all three have given smallest response time and largest waiting time and no. of context switches. This has happened because all three algorithms use the of time quanta smaller than the time quanta used by the other two algorithms in every cycle. Considering first cycle, time quantum "9" used by these three algorithms is smaller than time quantum "10" used by RR and time quantum "11" used by AMBRR. In second cycle time quanta "5", "3" and "4" used by AN, SRBRR and MB respectively are smaller than time quanta "10" and "9" used by RR and AMBRR respectively. Although AMBRR has completed execution in two cycles, but RR still uses a time quantum i.e. "10" greater than the time quanta "6", "8" and "7" used by AN, SRBRR and MB respectively.

AMBRR like in previous comparisons gives minimum waiting time and no. of context switches giving first priority to their reduction and largest response time giving its reduction a second priority by using largest time quanta in every cycle.

IV. EMPIRICAL ANALYSIS AND EVALUATION

Suppose some processes ready to be executed in an ascending order of burst time from [4] shown in TABLE VII:

TABLE VII: Tasks present in ready queue

Tasks	Burst Time (ms)
P1	1
P2	2
P3	3
P4	4
P5	5
P6	6
P7	7
P8	8
P9	9
P10	10

These processes are executed using the pairs of time quanta where time quantum of each pair is used alternately for scheduling. The waiting time and response time are calculated for each pair of time quanta. The values of resultant waiting time and response time for corresponding pairs of time quanta are shown in TABLE VIII. It can be observed that the waiting time reduces with the increase in the value of first time quantum present in each pair of time quanta. This is because the processes having relatively smaller burst time get executed in the first cycle of scheduling and don't have to wait for execution in the second cycle of scheduling. On contrary, the increase in the waiting time from first pair of time quanta i.e. "(1,10)" to the third pair i.e. "(3,8)" with the increase in the value of first time quantum in each pair can be contributed to the very small values of the first time quanta in each pair because the majority of the processes will remain in the ready queue in the second cycle of scheduling and due to unfinished execution will have to wait more.

The response time increases with the increase in the value of the first time quantum present in each pair because the processes with relatively larger burst times due to unfinished execution tend to remain in the ready queue in the second cycle and have to wait longer for response thus increasing the response time.

The trends of the two pertinent factors can be easily analyzed by plotting their normalized graphs against the corresponding pairs of time quanta. The graphs are normalized by dividing each value of waiting time and response time by the maximum waiting time and response time respectively. By the help of normalization, we can easily observe the effect induced by each pair of time quanta more effectively. These trends are shown in Fig. 5.

TABLE VIII: Pairs of Time Quanta used for scheduling and their Results

Pairs of time Quanta	Waiting Time (ms)	Response Time (ms)	Normalized Waiting Time (ms)	Normalized Response Time (ms)
(1,10)	20.1	4.5	0.8816	0.2727
(2,9)	22.1	8.1	0.9693	0.4909
(3,8)	22.8	10.9	1	0.6606
(4,7)	22.5	13	0.9868	0.7879
(5,6)	21.5	14.5	0.9430	0.8788
(6,5)	20.1	15.5	0.8816	0.9394
(7,4)	18.6	16.1	0.8158	0.9758
(8,3)	17.3	16.4	0.7588	0.9939
(9,2)	16.5	16.5	0.7237	1
(10,1)	16.5	16.5	0.7237	1

It is clear from the Fig. 5 that the pairs having intermediate values of time quantum tend to reduce waiting time and response time like two pairs that are "(6,5)" and "(5,6)". However, the pair "(6,5)" focusses less on reducing response time than waiting time by using first larger time quantum i.e. "6" and then smaller i.e. "5". The pair "(5,6)" does exactly the opposite by using smaller time quantum i.e. "5" first and then larger one i.e. "6". Our proposed algorithm works like the former case of using pair "(6,5)" because it first finds the intermediate value by calculating median to target both, the waiting time and the response time and gives the first priority to the reduction of waiting time by using first larger time quantum. Also when we apply the proposed algorithm on the series of processes in TABLE VII, then the resultant pair of time quantum we get is "(6,4)" which is very close to the pair "(6,5)".

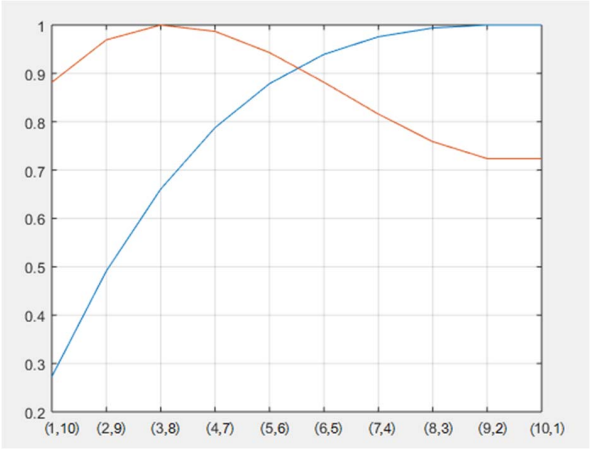

Figure 5: Resultant Waiting Time (Red) and Response Time (Blue)

V. CONCLUSION

form It is concluded from numerical comparisons and the empirical evaluation of results that the proposed algorithm addresses the scheduling criteria at a wider spectrum reducing no. of context switches and waiting time giving them first priority and focusing relatively less on response

time giving it second priority by using larger time quantum first and then second time quantum in alternation. Since the two-time quanta sum up to the highest burst time a process has in a ready queue, the algorithm executes all the processes in two cycles by using both time quanta in each cycle and only once. So, there is no need to calculate time quantum on run time for every cycle which reduces the overhead of calculation of time quantum which is a computationally intensive task. Hence the proposed algorithm saves resources of system for time quantum calculation and has also given accurate results without any aberration.

REFERENCES

[1] A. Silberschatz, P.B. Galvin and G. Gagne (2012), Operating System Concepts, 8th ed. Wiley Publishing, 2008

[2] Abbas Noon, Ali Kalakech, Seifedine Kadry, "A New Round Robin Based Scheduling Algorithm for Operating Systems: Dynamic Quantum Using the Mean Average", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 1, May 2011.

[3] Rakesh Mohanty, H. S. Behera Khusbu Patwari, Manas Ranjan Das, Monisha Dash, Sudhashree, "Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm", Vol.2, No.2, pp. 23-93, 2011

[4] Salman Arif, Saad Rehman, Farhan Riaz, "Design of Modulus Based Round Robin Scheduling Algorithm", 9th Malaysian Software Engineering Conference (MySEC2015), Kuala Lampur, 2015, pp. 230-235

[5] W. Stallings, Operating Systems, Internals and Design Principles, 4th ed. California, Prentice Hall 2000, 2000

[6] C. Yaashuwanth, Dr. R. Ramesh, "A new Scheduling Algorithm for Real Time Tasks", International Journal of Computer Science and Information Security, Vol.6, No.2, 2009

[7] Manish Kumar Mishra, "Improved Round Robin Scheduling Algorithm" Journal of Global Research in Computer Science, ISSN: 2229-371X, Vol.3, No.6, June 2012

[8] H.S. Behera, "Enhancing the CPU Performance using a modified mean-deviation round robin scheduling algorithm for real time systems", journal of global research in Computer science, ISSN: 2229-371X, Vol.3, No.3, March 2012

[9] NeetuGoel, R.B/Garg, "A comparative study of CPU Scheduling Algorithms",IJGIP Journal

[10] Abbas Noon, Ali Kalakech, Seifedine Kadry, "A new Round Robin Based Scheduling Algorithm for Operating system:Dynamic quantum using the mean average", International Journal of Computer Science, ISSN:1694-0814, Vol.8,ISSUE 3, NO.1, May 2011

[11] Debashree Nayak, Sanjeev Kumar Malla, "Improved Round Robin Scheduling using Dynamic time quantum", International Journal of computer Applications, Vol 38, No.5, January 2012

[12] Ali jbaeer Dawood, "Improved Efficiency of Round Robin Scheduling using Ascending Quantum and Minimum-Maximum Brust Time", Journal of university of anbar for pure science, ISSN:1991-8994, Vol.6,No.2 , 2012

[13] Manish Kumar, Abdul Kadir Khan, "An improved Round Robin CPC Scheduling Algorithm", Journal of Global Research in Computer Science, Vol.3, No.6, 2012

[14] Avi Silberschatz, Peter Baer Galvin and Greg Gagne, "Operating System Concepts Essentials", 7th ed. USA:John Wiley and Sons Inc, 2011

[15] Ishwari Singh Rajput, "A priority based Round Robin CPU Scheduling Algorithm for Rea Time Systems", (IJIET)International Journal of Innovations in Engineering and Technology, Vol.1, issue 3, October 2012

[16] Lalit Kishor,Rajendra Singh, "Optimized Scheduling Algorithms", International Journal of Computer Applications, International Conference on computer communication and networks, CSI-COMNET-2011

[17] 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 Theoratical and Applied Information Technology, Vol.37, No.1, PP 1-16

[18] Abdulrazak Abdulrahim, Saleh E.Abdulaahi & Junaidu B.Sahalu, (2014) "A New Improved Round Robin (NIRR) CPU Scheduling Algorithm", International Journal of Computer Applications, Vol.90, No.4, pp 27-33

[19] H.S. Behera, "Weighted Mean Priority Based Scheduling for Interactive Systems", Journal of global Research in computer Science, ISSN: 2229-371X, Vol.2, No.5, May 2011

[20] Pallab Banerjee, Probal Banerjee & Sonali Dhal, "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, ISSN:2278-3075, Vol.1, ISSUE 3, Aug 2012.

[21] Manish Kumar Mishra, Dr.Faizur Rashid, "An improved Round Robi CPU Scheduling Algorithm with varying time quantum," IJCSEA Journal, Vol.4, No.4, Aug 2014