EXTENSIONS TO ROUND ROBIN SCHEDULING: COMPARISON OF ALGORITHMS

By

RUWANTHINI SIYAMBALAPITIYA

Lecturer, Department of Statistics & Computer Science, University of Peradeniya, Sri Lanka.

ABSTRACT

Round Robin (RR) scheduling algorithm is a widely used scheduling algorithm in timesharing systems, as it is fair to all processes and is free of starvation. The performance of the Round Robin algorithm depends very much on the size of the time quantum selected. If the time quantum is too large, the performance of the algorithm would be similar to that of FCFS (First Come First Serve) scheduling. On the other hand, if the time quantum is too small, the number of context switches will be large. Therefore, it is necessary to have some idea about the optimum level of time quantum, so that the average waiting time and turnaround times, and the number of context switches are not too large. Several extensions to the round robin algorithm have been proposed in the literature to overcome these difficulties. In this study, the author has picked some of these extensions and tried comparing their effectiveness by means of some examples.

Keywords: Scheduling, Round Robin, Algorithms, Time Quantum, Burst Time.

INTRODUCTION

Round Robin (RR) is a scheduling algorithm designed especially for time sharing systems. It is somewhat similar to FCFS scheduling. The difference is that in RR, preemption of processes is allowed while there is no preemption in FCFS scheduling. When preemption of processes is allowed, a process is interrupted before completion, and the balance part is implemented in later stages. The idea is to give a fair share of processing to each available process.

In RR scheduling, a small time unit, known as a time quantum is defined, which is considered as the time limit allowed for each process at a time. The ready queue of processes is considered as a circular queue. Once a process spent its time quantum, it is preempted, which means that the process is interrupted and the remaining part is moved to the end of the queue using a context switch. Then, the next process takes over. This procedure continues until all the processes are completed. However, when a process completes its execution before its allocated time quantum is over, the next process in the ready queue continues execution.

Even though this idea may be attractive and looks fair for all the processes, it has its drawbacks as well. In fact, the performance of the RR algorithm depends very much on the size of the time quantum selected. If the time quantum is too large, the performance of the algorithm would be similar to that of FCFS scheduling. On the other hand, if the time quantum is too small, the number of context switches will be large. Therefore, it is necessary to have some idea about the optimum level of time quantum so that, the number of context switches would be limited and at the same time, average waiting time and turnaround times are not too large. Waiting time is the amount of time, a process spends waiting in the ready queue. The interval from the time of submission of a process at the time of completion is the turnaround time.

The paper is organized as follows : In section 1, the author presents some related work which extends the Round Robin scheduling concept. In section 2, the author discuss the computational experience by illustrating the detailed calculations. The author compares the performance of the algorithms in section 3, and the last section concludes the paper.

1. Related Work

A number of algorithms has been proposed to improve the outcome of the Round Robin scheduling. It is assumed that, the burst time of each process in the queue is known. Burst time is the actual time that is required to complete execution of a particular process or task in the computer. Instead of the static time quantum, a dynamic time quantum has been proposed by some of these approaches. However, in most of these, only examples of small sizes have been used to illustrate these procedures. The other problem is that, in most of these approaches, sorting the burst times of the given processes is required before hand, which might affect the speed of computation when a large number of processes are present in a given list. Therefore, the validity of these claims could not be verified for larger size problems as theoretical proofs have not been provided.

Here, a summary of each algorithm has been presented that the author has compared in this study. Certain algorithms were not mentioned as they give rise to very similar results.

- Round Robin Algorithm: The author has considered the Round Robin algorithm as the basis for comparing the performance of the other algorithms in this study.
- Ahad (2012) has observed that in many cases, jobs are preempted even if a negligible amount of execution time is left for a job. Therefore, a process by which the time quantum should be modified has been proposed. The time quantum of a process is modified based on some threshold value, which is calculated by taking average of the left out time of all processes in its last turn.
- Barman (2013) has proposed an algorithm (DTQRR - Dynamic Time in Round Robin Algorithm) in which, if arrival time of all the processes is zero, the time quantum is set to be the average of burst times of all the processes. If arrival time is not zero, the time quantum is changed dynamically depending upon arrival time and burst time of the processes.
- Behera et. al. (2011) have proposed a method (MTDQRR - Multi Dynamic Time Quantum Round Robin Algorithm), where the time quantum is calculated twice in a single round robin cycle. First, median of the burst times is taken as the time quantum up to the process considered as the median. For the succeeding processes, the time quantum is determined by taking the burst time of Upper Quartile of all processes. This whole operation occurs in a single scheduling cycle of the processes sorted in ascending order of the burst time of all the processes.
- In the algorithm proposed by Matanech (2009), the time quantum is repeatedly adjusted according to the burst time of the currently running processes. The time quantum is made equal to the median of the burst times of the remaining processes. If the median is less than 25, it will be made equal to 25.
- Negi (2013): In this paper, some minor changes to the conventional Round Robin algorithm, so that, the time quantum of those processes is increased to some extent whose remaining time in its last turn is less than or equal to an assigned threshold value. In this approach, this threshold value is assumed to be one fourth of the time quantum. If the remaining time of a process in its last turn is found out to be less than this threshold value, then the process is not preempted in its second last turn unless it completely finished its entire remaining execution time.
- Noon et al. (2011) demonstrated that this algorithm uses the idea of the dynamic time quantum. Initially, the time quantum is considered as the burst time of the first process in the queue. Then the time quantum is taken to be the average of the remaining burst times of the processes waiting in the queue.
- In the algorithm proposed by Rao et al., (2015) time slices of only those processes which require a slightly greater time slice than the allotted time cycles are only modified. If the remaining burst time is less than or equal to the one time slice, then execute the same process otherwise go for the next process.
- Vijaya Lakshmi (2015) has proposed an algorithm which arranges the processes in an ascending order of the burst times. Time quantum is calculated by multiplying the median of burst times by the

difference between maximum and minimum values of burst times and then dividing by the average of the burst times.

In addition, several other algorithms have also been proposed to solve this problem.

An algorithm presented by Kundargi and Emmi (2014) attempts to improve the disadvantage of Round Robin algorithm by using a dynamic time quantum.

Jaiswal et al. (2013) have proposed an approach on a dynamic time quantum, which is repeatedly changed to the immediate greater value of the previous time quantum.

Mishra and Rashid (2014) has presented an algorithm in which the dynamic time quantum is repeatedly adjusted to the minimum value of retaining burst time.

Datta et.al. (2015) have proposed an algorithm based on Round Robin and the shortest job first scheduling. Khankasikam (2013) has proposed an algorithm in which the time quantum is repeatedly adjusted according to the burst time of the running processes.

Helmy and Dekdouk (2007) has proposed an algorithm based on a weighting technique as an attempt to combine the low scheduling overhead of round robin algorithms and favor short jobs.

2. Computational Experience

Most of the papers mentioned above have illustrated their algorithms using small size examples such as problems with only five processes. Therefore, the author has randomly generated problems containing more processes (eg. using 7 and 12 processes) in order to study the behavior of these algorithms. The programs for the basic Round Robin algorithm and the eight extensions of the algorithm were tested using the same set of examples. The programs were implemented with a simulator constructed using a Pascal compiler.

The detailed computational steps are presented with respect to all the algorithms considered using the problem with 7 processes as given below. A summary of results for a 12 process problem is also presented in the next section.

Process	1	2	3	4	5	6	7	3	4	6	4
Burst-time	30	15	45	85	20	32	18	15	55	2	25
Run-time	30	15	30	30	20	30	18	15	30	2	25
Cycle	1	1	1	1	1	1	1	2	2	2	3
Context switch	-	1	2	3	4	5	6	7	8	9	10
Time-spent	30	45	75	105	125	155	173	188	218	220	245
Completed	Y	Y	N	N	Y	N	Y	Y	N	Y	Y

Table 1. Implementation of Round Robin Algorithm (Time Quantum = 30)

Example 1:

Problem with 7 processes

Burst time: 30, 15, 45, 85, 20, 32, 18

Total burst time = 245

Average burst time = 245/7 = 35

2.1 Round Robin Algorithm

An arbitrary time quantum of 30 has been used in the round robin algorithm. Implementation of Round Robin algorithm is shown in Table 1.

Total turnaround time = 30+45+ 188 + 245 + 125 + 220 + 173 = 1026

Average turnaround time = 1026/7 = 146.6

Total waiting time = 1026 – 245 = 781

Average waiting time = 781/7 = 111.6

Context switches = 10

2.2 Improved RR Algorithm

An arbitrary time quantum of 30 has been used for the improved RR algorithm (Ahad, 2012). A threshold value k is added to the time quantum whenever it is possible to finish a process without preemption. K is the ceiling (x), where x is the average left out time of uncompleted processes. Left out time is the remainder, when the burst time of each process is divided by the time quantum where the burst time is greater than the time quantum, Therefore, the left out times for the given processes are 0, 0, 15, 25, 0, 2, 0.

Process	1	2	3	4	5	6	7	3	4	4
Burst-time	30	15	45	85	20	32	18	15	55	25
Run-time	30	15	30	30	20	32	18	15	30	25
Cycle	1	1	1	1	1	1	1	2	2	3
Context switch	-	1	2	3	4	5	6	7	8	-
Time-spent	30	45	75	105	125	157	175	190	220	245
Completed	Y	Y	N	N	Y	Y	Y	Y	N	Y

Table 2. Implementation of Improved RR Algorithm (Time Quantum = 30, k = 6)

Hence, k = ceiling (15+25+2/7) = ceiling (42/7) = ceiling (6) = 6

Therefore, even though the time quantum is 30, whenever it is possible to complete a process, a time quantum of 36 could be used. Table 2 shows the implementation of improved RR algorithm.

Total turnaround time = 30+45+190+245 +125+157+ 175 = 967

Average turnaround time = 967/7 = 138.1

Total waiting time = 967 – 245 = 722

Average waiting time = 722/7 = 103.1

Context switches = 8

2.3 DTQRR Algorithm

Time quantum = average of burst times of all processes = 245/7 = 35

Implementation of DTQRR algorithm is shown in Table 3 (Barman, 2013).

Total turnaround time = 30+45+195+245+135+167+ 185 = 1002

Average turnaround time = 1002/7 = 143.1

Total waiting time = 1002 – 245 = 757

Average waiting time = 757/7 = 108.1

Context switches = 8

2.4 MTDQRR Algorithm

Two time quanta are used according to this algorithm. First, the processes are arranged according to the ascending order of burst times. Accordingly, the median of burst times is 30, which is the fourth burst time in the ascending order. This time quantum is used up to the fourth burst time in the list. The time quantum taken for the remaining processes from the next process in the queue is the upper quartile, which is 45 (Behera 2011). Implementation of MTDQRR algorithm is shown in Table 4.

Total turnaround time =15+33+53+83+115+160+245 = 704

Average turnaround time = 704/7 = 100.6

Total waiting time = 704 – 245 = 459

Average waiting time = 459/7 = 65.6

Context switches = 6

2.5 Self Adjustment RR (SARR) Algorithm

In this algorithm, the median of remaining burst times are adjusted repeatedly. Here, the time quantum is dynamic and taken as the median burst time. However, if the median is less than 25, it is taken equal to 25. Initially, the time quantum is set at 30, which is the median of burst times of the processes in the queue. After the first cycle, processes 6, 3 , and 4 still remain to be completed. The median for these processes is 15, which is less than 25. Therefore, according to the algorithm, time quantum is taken as 25 for the remaining processes. After that, only the process 4 goes to the third cycle (Matanech 2009). Table 5 shows the SARR algorithm implementation.

Total turnaround time =15+33+53+83+175+190+245 = 794

Process	1	2	3	4	5	6	7	3	4	4
Burst-time	30	15	45	85	20	32	18	10	50	15
Run-time	30	15	35	35	20	32	18	10	35	15
Cycle	1	1	1	1	1	1	1	2	2	3
Context switch		1	2	3	4	5	6	7	8	-
Time-spent	30	45	80	115	135	167	185	195	230	245
Completed	Y	Y	N	N	Y	Y	Y	Y	N	Y

Table 3. Implementation of DTQRR Algorithm

Process	2	7	5	1	6	3	4	4
Burst-time	15	18	20	30	32	45	85	40
Run-time	15	18	20	30	32	45	45	40
Cycle	1	1	1	1	1	1	1	2
Context switch	-	1	2	3	4	5	6	-
Time-spent	15	33	53	83	115	160	205	245
Completed	Y	Y	Y	Y	Y	Y	N	Y

Table 4. Implementation of MTDQRR Algorithm

Process	2	7	5	1	6	3	4	6	3	4	4
Burst-time	15	18	20	30	32	45	85	2	15	55	30
Run-time	15	18	20	30	30	30	30	2	15	25	30
Cycle	1	1	1	1	1	1	1	2	2	2	3
Context switch	-	1	2	3	4	5	6	7	8	9	-
Time-spent	15	33	53	83	113	143	173	175	190	215	245
Completed	Y	Y	Y	Y	N	N	N	Y	Y	N	Y

Table 5. Implementation of SARR Algorithm

Process	1	2	3	4	5	6	7	3	4	4
Burst-time	30	15	45	85	20	32	18	15	55	25
Run-time	30	15	30	30	20	32	18	15	30	25
Cycle	1	1	1	1	1	1	1	2	2	3
Context switch	-	1	2	3	4	5	6	7	8	-
Time-spent	30	45	75	105	125	157	175	190	220	245
Completed	Y	Y	N	N	Y	Y	Y	Y	N	Y

Table 6. Implementation of Conventional RR Algorithm

Average turnaround time = 794/7 = 113.4

Total waiting time = 794 – 245 = 549

Average waiting time = 549/7 = 78.4

Context switches = 9

2.6 Conventional RR (CRR) Algorithm

An arbitrary time quantum of 30 is used in this example. A threshold value is equal to one-fourth of the time quantum, which is equal to 8 is considered here. This threshold value is added to the time quantum, whenever it is possible to complete a particular process without preemption (Negi 2013). CRR algorithm is shown in Table 6.

Total turnaround time = 30+45+190+245+125+157+175 = 967

Average turnaround time = 967/7 = 138.1

Total waiting time = 967 – 245 = 722

Average waiting time = 722/7 = 103.1

Context switches = 8

2.7 AN Algorithm

Initially, the time quantum is taken as the burst time of the first process in the queue. Afterwards, the time quantum is modified and taken as the average of remaining burst times in the queue. Hence, the initial time quantum is 30. Subsequent time quanta are 24 and 31 (Noon 2011). Implementation of AN algorithm is shown in Table 7.

Total turnaround time = 30+45+188+245+125+214+173 = 1020

Average turnaround time = 1020/7 = 145.7

Process	1	2	3	4	5	6	7	3	4	6	4
Burst-time	30	15	45	85	20	32	18	15	55	2	31
Run-time	30	15	30	30	20	30	18	15	24	2	31
Cycle	1	1	1	1	1	1	1	2	2	2	3
Context switch	-	1	2	3	4	5	6	7	8	9	10
Time-spent	30	45	75	105	125	155	173	188	212	214	245
Completed	Y	Y	N	N	Y	N	Y	Y	N	Y	Y

Table 7. implementation of AN Algorithm

Process	1	2	5	6	7	3	4	4
Burst-time	30	15	20	32	18	45	85	50
Run-time	30	15	20	32	18	45	35	50
Cycle	1	1	1	1	1	1	1	2
Context switch	-	1	2	3	4	5	6	-
Time-spent	30	45	65	97	115	160	195	245
Completed	Y	Y	Y	Y	Y	Y	N	Y

Table 8. implementation of Improved CRR Algorithm

Total waiting time = 1020 – 245 = 775

Average waiting time = 775/7 = 110.7

Context switches = 10

2.8 Improved Conventional RR (CRR) Algorithm

Initially, the time quantum is taken as the floor (x), where x is the average of the burst times of the processes in the queue which is 35 in this example. Processes with burst times less than or equal to the time quantum are executed according to the basic round robin algorithm. Then, the processes with burst times exceeding the time quantum are arranged according to the remaining burst times and the number of cycles and allocate CPU. If the remaining burst time of current process is less than one time quantum, allocate CPU again to the current process. Otherwise, go to the next process (Rao 2015). Table 8 shows the Implementation of CRR algorithm.

Total turnaround time = 30+45+65+97+115+160+245 = 757

Average turnaround time = 757/7 = 108.1

Total waiting time = 757 – 245 = 512

Average waiting time = 512/7 = 73.1

Context switches = 6

2.9 RR Scheduling Algorithm

In this algorithm, Vijaya Lakshmi (2015) calculated the

Process	2	7	5	1	6	3	4	4
Burst-time	15	18	20	30	32	45	85	25
Run-time	15	18	20	30	32	45	60	25
Cycle	1	1	1	1	1	1	1	2
Context switches	-	1	2	3	4	5	6	-
Time-spent	15	33	53	83	115	160	220	245
Completed	Y	Y	Y	Y	Y	Y	N	Y

Table 9. Implementation of RR Scheduling Algorithm

Algorithm	Average Turnaround Time	Average Waiting Time	Context Switches
RR (TQ=30)	146.6	111.6	10
Improved RR	138.1	103.1	8
DTQRRD	143.1	108.1	8
MTQRR	100.6	65.6	6
SARR	113.4	78.4	9
CRR	138.1	103.1	8
AN	145.7	110.71	10
Improved CRR	108.1	73.1	6
RR Scheduling	100.6	65.6	6

Table 10. Comparison of Algorithms for Example Problem with 7 Processes

Algorithm	Average Turnaround Time	Average Waiting Time	Context Switches
RR (TQ=20)	233	203	22
Improved RR	205.6	175.6	18
DTQRRD	217.4	187.4	17
MTQRR	200.1	170.1	13
SARR	180.5	150.5	17
CRR	205.6	175.6	18
AN	224.8	194.8	22
Improved CRR	153.6	123.6	11
RR Scheduling	145.1	115.1	11

Table 11. Comparison of Algorithm for Example Problem with 12 Processes

time quantum as follows (Table 9).

Time quantum = (max. burst time-min. burst time)x median/average burst time.

= (85-15) x 30/35 = 60

Total turnaround time = 15+33+53+83+115+160+ 245 = 704

Average turnaround time = 704/7 = 100.6

Total waiting time = 704 – 245 = 459

Average waiting time = 459/7 = 65.6

Context switches = 6.

3. Comparison of Results

The author has compared the performance of the above algorithms based on the standard criteria: average turn around time, average waiting time and the number of context switches. In order to compare them, the author has used two numerical examples, one containing 7 processes and another problem with 12 processes.

Results are presented in Tables 10 and 11. RR indicate the results obtained for the standard Round Robin algorithm.

Example 1: problem with 7 processes

Burst times: 30, 15, 45, 85, 20, 32, 18

Example 2: problem with 12 processes.

Burst times:15,40,18,25,68, 30,22, 42,10,16,35,39

From the above results, it can be seen that, out of the eight algorithms compared with the Round Robin algorithm, results obtained from the four algorithms appeared to be superior than those of the others. These are the algorithms proposed by Behera (2011), Matanech (2009), Rao et. al. (2015), and Vijaya Lakshmi (2015). Algorithms proposed by Negi (2013), Ahad (2012), Barman (2013), and Noon (2011) are placed lower down the order according to the results obtained from the example problems. However, it is not possible to rank them according to the superiority without considering further examples.

An advantage of the algorithm proposed by Behera et. al. (2011) was that, the time quantum is calculated twice in a single Round Robin cycle. First, the median of the burst time is taken as the time quantum up to the process considered as the median. For the succeeding processes, the time quantum is determined by taking the burst time of Upper Quartile of all the processes. This helps to reduce the overall processing time.

In the algorithm proposed by Mataneh (2009), the median of remaining burst time is adjusted repeatedly. This can be considered as a strength of this algorithm. Here, the time quantum is dynamic and is taken as the median burst time.

The benefit of the algorithm proposed by Rao et. al. (2015) is that, time slices of only those processes which require a slightly greater time slice than the allotted time cycles are only modified.

A strength of the algorithm presented by Vijaya Lakshmi (2015), is that a formula has been used to determine the time quantum. It attempts to define the finest time quantum using a formula which incorporate maximum, minimum and median burst times.

An advantage of the conventional RR (Negi 2013) algorithm is that, a threshold value is added to the time quantum, whenever it is possible to complete a particular process without preemption. However, a drawback of this algorithm is that, an arbitrary time quantum has to be used in solving a given problem.

A special feature of the improved RR (Ahad 2012) algorithm is that, the processes waiting in the ready queue are divided into two categories. The processes in the first category are the one for which the time quantum is modified and the processes in the second category will be processed as per the classical Round Robin algorithm. This algorithm also suffers from the necessity to use an arbitrary time quantum.

Noon et al. (2011) proposed an algorithm in which, initially, the time quantum is taken as the burst time of the first process in the queue. However, no justification was made for this selection: whether this approach would improve the performance of the algorithm.

The time quantum is set to be the average of burst time of all the processes in the algorithm proposed by Barman (2013). However, it was not mentioned whether this method would improve the performance of the algorithm.

Conclusion

In the absence of any theoretical results to establish the superiority of a particular algorithm, it is difficult to generalize the results that the author has obtained. However, in order to further generalize these results, problems with a much larger number of processes need to be solved.

But, based on the results that the author has already obtained, out of the eight algorithms tested, four algorithms stand clearly above the other algorithms in the list. These are the algorithms proposed by Vijaya Lakshmi (2015), Rao et al. (2015), Matanech (2009), and Behera (2011). However, all of these papers report their results and analyze them based on the examples of small size containing few processes only. Hence, it is not possible to rank them according to the level of performance. Therefore, it is necessary that, more comprehensive analysis of the performance of these and other algorithms available for the extensions to the Round Robin algorithm has to be carried out in order to claim the superiority of these algorithms.

Considering the importance of this problem in operating system design, this can be considered as an extremely useful extension to this study.

References

[1]. **Ahad M.A, (2012).** "Modifying Round Robin Algorithm for Process Scheduling using Dynamic Time Quantum Precision". *Special Issue of International Journal of Computer Applications*, pp. 5-10.

[2]. **Barman. D, (2013).** "Dynamic Time Quantum in Round Robin Algorithm Depending on Burst and Arrival Time of the Processes". *International Journal of Innovative Technology and Exploring Engineering*, Vol.2, No.4, pp.60-64.

[3]. **Behera H.S. et.al, (2011).** "Design and Performance Evaluation of Multi Cyclic Round Robin Algorithm using Dynamic Time Quantum". *Journal of Global Research in Computer Science*, Vol.2, No. 2, pp. 48-53.

[4]. **Datta K., Jana M., and Mazumdar A, (2015).** "An Effective Dynamic Quantum Round Robin CPU Scheduling Algorithm". *International Journal of Computer Applications*, Vol.130, No.6, pp.1-5.

[5]. **Jaiswal R.R., Geetha K., and Mohan R, (2013).** "An Intelligent Adaptive Round Robin (IARR) Scheduling Algorithm for Performance Improvement in Real Time Systems". *Proceedings of International Conference on Advances in Mechanical Engineering*.

[6]. **Helmy T. and Dekdouk A, (2007).** "Burst Round Robin: As A Proportional-Share Scheduling Algorithm". in *Proceedings of the 4th IEEE-GCC Conference on Towards Techno-Industrial Innovations*, pp.424-428.

[7]. **Khankasikam K, (2013).** "An Adaptive Round Robin Scheduling Algorithm: A Dynamic Time Quantum Approach". *International Journal of Advancements in Computing Technology*, Vol.5, No.1, pp.595-603.

[8]. **Kundargi N. and Emmi M.S, (2014).** "Job Scheduling Algorithm using Finest Time Quantum for Real Systems". *International Journal of Latest Trends in Engineering and Technology*, Vol.4, No.1, pp.120-123.

[9]. **Matanech, R.J, (2009).** "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.

[10]. **Mishra M.K. and Rashid F, (2014).** "An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum". *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, Vol.4, No.4.

[11]. **Negi S, (2013).** "An Improved Round Robin Approach using Dynamic Time Quantum for Improving Average Waiting Time". *International Journal of Computer Applications*, Vol.69, No.14, pp.12-16.

[12]. **Noon, A, Kalakech, A. and Kadry, S. (2011).** "A New

Round Robin based Scheduling Algorithm for Operating Systems: Dynamic Quantum using the Mean Average". *IJCSI International Journal of Computer Science*, Vol.8, No.3, pp.224-229.

[13]. Rao, G.S.N., Srinivasu, N. and Rao, G.R.K. (2015). "Dynamic Time Slice Calculation for Round Robin Process Scheduling Using NOC". *International Journal of Electrical and Computer Engineering*, Vol.5, No.6, pp.1480-1485.

[14]. Vijaya Lakshmi, G. (2015). "Determining a Finest Time Quantum to Improve the Performance of Round Robin Scheduling Algorithm". *International Journal of Innovative Research in Computer and Communication Engineering*, Vol.3, No.7, pp.6913-6918.

ABOUT THE AUTHOR

Dr. Ruwanthini Siyambalapitiya is a Lecturer attached to the Department of Statistics & Computer Science at University of Peradeniya, Sri Lanka. Her main research interests are Algorithms and Operating Systems.