Adaptive Round Robin CPU Scheduling using AMS Time Quantum

Palak Bhavsar[1] Mrs. Shilpa Serasiya[2]
[1]M.E Student [2]Professor
[1,2]Department of Computer Science & Engineering
[1,2]KITRC College, Kalol, Gujarat, India

Abstract— Operating system scheduling is designed for allocation of available processes to the CPU resources. Efficiency of CPU Scheduler depends on the design of the high quality scheduling algorithm which suits the scheduling goals. The main goal of scheduling algorithm is to maximize CPU utilization and throughput along with minimizing turnaround time, waiting time and number of context switching for a set of requests. There are some popular CPU scheduling algorithms available which are FCFS, SFJ, Priority scheduling and RR. Among all of them Round Robin being the most popular choice, but it is not able to give us the better reading of turnaround time and waiting time. So, this article is about improving performance of scheduling algorithm in terms of turnaround time, waiting time and number of context switching by adapting the method of Round Robin algorithm. The performance of Adaptive Round Robin algorithm is compared with RR, using CPU Scheduling Simulator which is developed using Java and Java Swing.

Key words: CPU, FCFS

I. INTRODUCTION

Central Processing Unit (CPU) Scheduling plays an important role by switching the CPU among various processes. Any process contains the program code and its current activity, which is an instance of a computer program that is being executed by the CPU [4].

The purpose of the operating system is that to allow the process as many as possible running at all the time in order to make the best use of CPU [2]. In CPU when a new process is created a decision needs to be made whether to run the parent process or the child process at that time scheduling concept will be used [1]. To allocate CPU time slice efficiently and mean while offering nice user experience is the main purpose of Scheduler [6].

The success of a scheduler depends on the design of high quality scheduling algorithm [2]. The idea of the process scheduling is, under the reasonable circumstances, to maximize the utilization of the CPU as great as possible. There are some factors, that should be taken into the consideration while choosing scheduling algorithm, usually from two aspects: on one hand, for the user side, efficient, fair, fast response time, short turnaround time; on the other hand, for the system side, the selected scheduling algorithm should increase the system throughput, improve the resources utilization and make a balanced use of all types of resources [4].

II. THEORY

There are many types of CPU Scheduling algorithms available, from them First Come First Serve (FCFS) is most simple technique for Scheduling, it allow the first process submitted to run first and processes are inserted into the tail of a queue when they are entered [2]. In Shortest Job First who have two categories which is preempted and non-preempted where processes are entered into ready queue according to their burst time but in some cases this method creates starvation for remaining processes [1].

Another one is Priority scheduling in which each process has its priority with the value between 0-9, with that value they will use the resource with higher to lower priority. So, starvation could be occurs with lower priority [1]. The basic idea behind the Round Robin Scheduling is to separate the CPU execution time into several time slices or time quantum, where time quantum is fixed amount of time, each process will execute according to that time quantum. The CPU provides service for each process alternatively. This algorithm chooses the processes from the ready queue according to FCFS manner [4]. Round Robin is considered the most widely used scheduling algorithm in CPU scheduling [5].

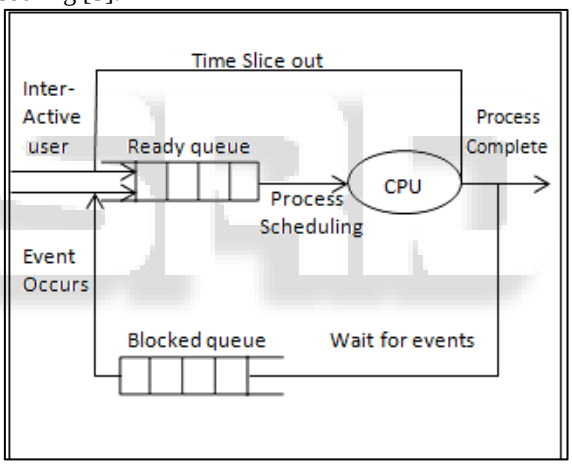


Fig. 1: Process Scheduling Queue Model

The figure 2.1 shows following three conditions that may occur while process in ready queue is executed orderly.

1) The process ends before it runs out of time slice, then the system should advance scheduling.
2) The process of execution put forward I/O request and is blocked, which enters into the corresponding blocked queue.
3) The process is not end until it runs out of a time slice, which should be remove to end of the ready queue by the system, waiting the next scheduling [4].

All those algorithms have their own advantages and disadvantages, which is detailed explained in teaching material [1][2]. Performance of all algorithms is explained in material [2], where qualitative analysis of Round Robin algorithm according to their service and response time is proved in material [4] with mathematical functions.

In the field of scheduler, Linux also achieves continuous innovation and developments. Early Linux

scheduler used minimal designs Linux 1.2 scheduler used a circular queue for runnable task management that operated with round robin Scheduling policy, Linux version 2.2 introduced the idea of scheduling classes, the 2.4 kernel used O(N) scheduler [6]. Linux 2.4, used multilevel feedback queue. In version 2.6.0 to 2.6.22 Linux kernel used "Fair Scheduling" named "Rotating Staircase Deadline ", and from that Linux 2.6.23 used "Completely Fair Scheduler" (CFS) is the first implementation of a fair queuing process scheduler widely used in a general purpose operating system [6].

III. RELATED WORK

In recent years a number of CPU Scheduling mechanisms have been developed for predictable allocation of processors so as a part of it. An Improved Round Robin Scheduling algorithm allocates the time quantum to all the processes once. After that they use SJF to select the next process from ready queue [3].

In [5], a new algorithm for the CPU Scheduling is presented using FFGA (Fonseca and Fleming's Genetic Algorithm) multiobjective optimization, and also compared the results of this algorithm with FCFS, Priority, SJF and RR from that they conclude that FFGA is more optimized then the other algorithms.

The concept of completely fair in scheduling named SD (Staircase Scheduler) in Linux has its advanced version called RSDL (The Rotating Staircase Deadline Scheduler). The most important characteristic of CFS is Fairness; it did a lot work to keep scheduling fair, including group scheduling [6].

In [8] for the hierarchical and random CH selection protocol of WSN (Wireless Sensor Network) is discussed, they have been developed an energy efficient "Round Robin" mechanism.

In Bluetooth technology, the important role of the scheduling policy in determine network throughput, many researchers in recent year try to develop more efficient scheduling algorithm for Bluetooth. In [9] they propose a new scheduling policy based on Round Robin algorithm which used widely today in Bluetooth technology.

IV. PROPOSED WORK

Here, a new approach for CPU Scheduling algorithm is explained, which can be used to improve the performance of CPU in Waiting Time, Turnaround Time and context Switches. This new approach of Scheduling algorithm called Adaptive Round Robin using AMS (Average Max Static) Time Quantum is based on the integration of Round Robin (RR) and Shortest Job First (SJF). Priority of a process is based on SJF and execution of a process is based on RR with Static Time Quantum. So, after all it gives the advantage of simple RR in reducing starvation and also integrates the advantage of SJF scheduling.

The performance of Round robin scheduling algorithm has a great concern with the size of the Time Slice or Time Quantum [3]. The selection of the size of time quantum is quite difficult, if the time quantum or time slice is too big, the algorithm turns into FCFS algorithm, or if the time quantum is small than many context switches occurs [4].

So, to solve that problem of time quantum we firstly count the Time quantum from our available processes with the use of these two equations shown below.

$$TQ = \frac{AVG+MAXBT}{2}$$

Where,

$$AVG = \frac{\text{(Sum of BT of all the process)}}{\text{Number of processes}}$$

TQ=Time Quantum
AVG=Average of all the processes
MAXBT=Maximum Burst Time

By taking this Static Time quantum (Fixed TQ) from all those processes which are available in ready queue. In this algorithm Firstly Maximum Burst Time will find out from all processes which are present, than calculate TQ according to above equation. Here in this method we will take static TQ so through all the process TQ will never be changed.

Now we set our TQ then we can take one by one process according to their burst time, which means that a process with smallest bust time will come fist for execution form the ready queue and apply our TQ on them, after execution of that process if remaining Burst time of that process is less than our fixed TQ then algorithm will execute that process again and finish its execution or if it is greater than fixed TQ than it will directly go to the end of ready queue and choose the next process from ready queue. Again process with smallest Burst time will be chosen.

So, in this algorithm with the use of combination of RR and SJF with the static Time Quantum we tried to solve the problem of execution of processes along with its Average turnaround time, Average waiting time and Context switches.

Following is the Proposed Adaptive Round Robin Algorithm steps:

1) Step 1. START
2) Step 2. Calculate TQ , form all available processes with the help of MAXBT and AVG.
3) Step 3. Make a Ready Queue of the processes say RQ.
4) Step 4. Do step 5, 6, 7 and 8 WHILE Ready Queue becomes empty.
5) Step 5. Pick process from RQ according to minimum burst time (BT) and allocate the CPU to it for aTQ Time.
6) Step 6. Calculate Remaining Burst Time (RBT) of process.
7) Step 7. If the RBT of the current running process is less than TQ then allocate CPU again to the currently running process for the RBT and go to step 5.
8) Step 8. Remove the currently running process from execution and put it at the tail of the RQ.
9) Step 9. Calculate ATT, AWT and CS.
10) Step 10. END

V. EXPERIMENTS PERFORMED

A. CASE 1 – Zero Arrival Time:

Consider four processes P1, P2, P3 and P4 arriving at Time 0 with Burst Time 15, 7, 28 and 20 respectively. Time

Quantum (TQ) has been calculated 23 milliseconds (ms) as per the equation give before.

Our proposed ARR CPU Scheduling picks the first process from the RQ according to the minimal value of burst time which is P2 and allocate the CPU to it for a time interval 23 ms. After executing P2 there is no burst time remain for P2. So, P2 has finished execution, th will be removed from the RQ. Next process from ready queue according to its minimal burst time is P1 with 15 ms. P1 will finish execution in one round and it will be removed from RQ. P4 is the next for execution with 20 ms. P4 also will finish execution in one round and will be removed from RQ.

At last P3 will arrive for execution because it has maximum amount of burst time. TQ will apply on P3 which is 23 ms and calculate RBT of P3. Now, RBT of P3 is less than the TQ, so CPU will be allocated again to P3 for a Remaining Burst Time. So P3 also has finish execution and it will be removed from the ready queue. The waiting time is 0 for P2, 7 for P1, 22 for P4 and 42 for P3. Average waiting time is 17.75 ms and average turnaround time is 35.25 ms for above processes.

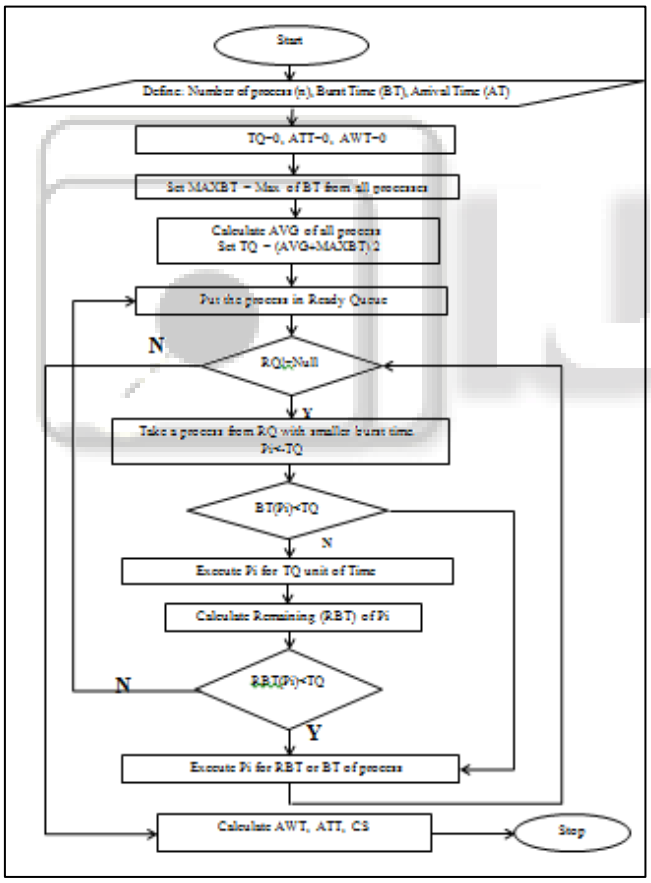


Fig. 2: Flowchart of Adaptive Round Robin Using AMS Time Quantum Algorithm

Let's compare performance of Adaptive Round Robin Algorithm with Simple Round Robin. Gantt chart for both the algorithms are shown below with the table valued ATT, AWT and CS.

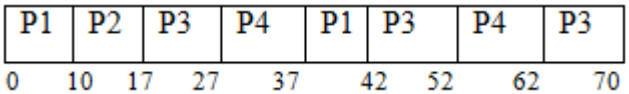


Fig. 3: Gantt chart For Simple RR (TQ = 10 Ms)

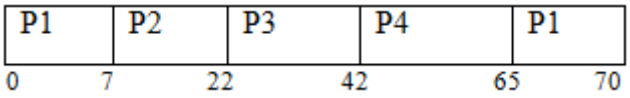


Fig. 4: Gantt chart for Adaptive RR (TQ = 23 ms)

	ATT	AWT	CS
Simple RR (TQ = 10)	47.75	30.25	4
Adaptive RR (TQ = 23)	35.25	17.75	1

Table 1: Comparison of RR and ARR for case – 1

B. CASE 2 –Non-Zero Arrival Time:

Consider some processes Shown in Table 5.2 with different Arrival Time and Burst Time where TQ is given for RR is 100 from [10].

Process no.	Arrival Time	Burst Time
P0	0	250
P1	50	170
P2	130	75
P3	190	100
P4	210	130
P5	350	50

Table 2: Process no, Arrival Time and Burst Time

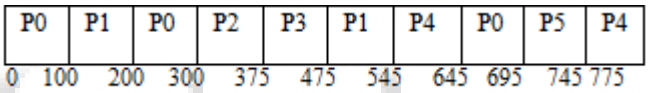


Fig. 5: Gantt chart for Simple RR (TQ = 100 ms)

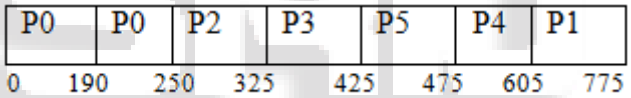


Fig. 6: Gantt chart for Adaptive RR (TQ = 190 ms)

	ATT	AWT	CS
Simple RR (TQ = 100)	446.66	317.5	4
Adaptive RR (TQ = 190)	320.83	191.66	1

Table 3: Comparison of RR and ARR for case – 2

VI. SIMULATION

In this paper we also introduce you to our simulator named "CPU Scheduling Simulator", which we developed using Java and Java Swing. This simulator has been designed to study the behavior of different algorithm according to their waiting time and turnaround time. So, let's execute Table 5.2 into simulator and results shows in below snapshots.

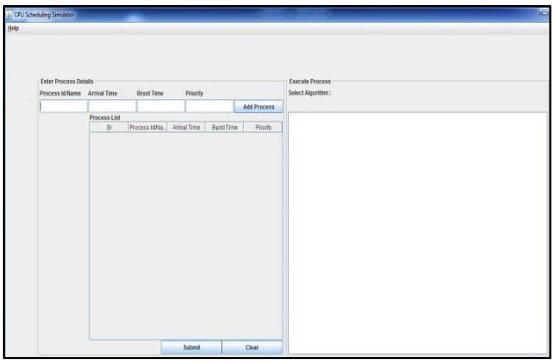


Fig. 7: CPU Scheduling Simulator

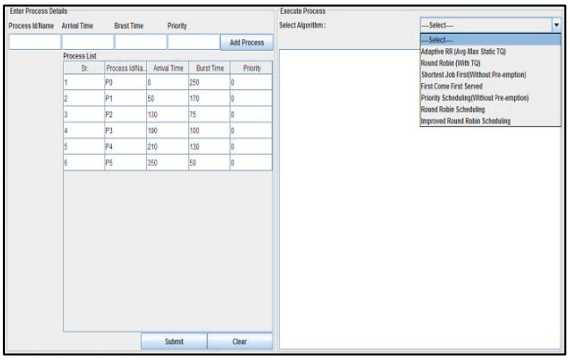
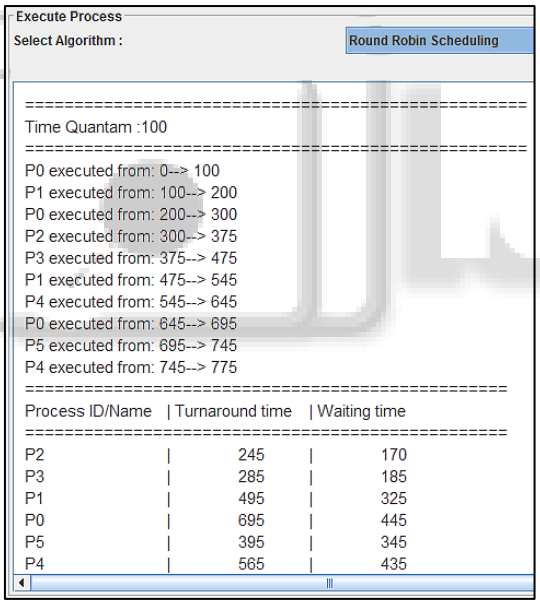


Fig. 8: Add Processes and List of Algorithms



Execute Process

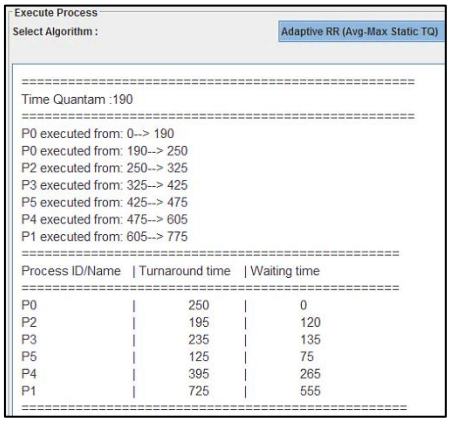
Select Algorithm : Round Robin Scheduling

Time Quantam :100

P0 executed from: 0--> 100
P1 executed from: 100--> 200
P0 executed from: 200--> 300
P2 executed from: 300--> 375
P3 executed from: 375--> 475
P1 executed from: 475--> 545
P4 executed from: 545--> 645
P0 executed from: 645--> 695
P5 executed from: 695--> 745
P4 executed from: 745--> 775

Process ID/Name	Turnaround time	Waiting time
P2	245	170
P3	285	185
P1	495	325
P0	695	445
P5	395	345
P4	565	435

Fig. 9: Execution of Round Robin Algorithm



Execute Process

Select Algorithm : Adaptive RR (Avg-Max Static TQ)

Time Quantam :190

P0 executed from: 0--> 190
P0 executed from: 190--> 250
P2 executed from: 250--> 325
P3 executed from: 325--> 425
P5 executed from: 425--> 475
P4 executed from: 475--> 605
P1 executed from: 605--> 775

Process ID/Name	Turnaround time	Waiting time
P0	250	0
P2	195	120
P3	235	135
P5	125	75
P4	395	265
P1	725	555

Fig. 10: Execution of Adaptive Round Robin

VII. RESULTS ANALYSIS

Using Simulator we took around 60 processes at ones for execution and note down the results. Graph shows the performance of Adaptive Round Robin algorithm in terms of ATT, AWT and CS below where Case – 1 is the analysis for processes with 0 arrival time and Case – 2 is the analysis of those processes which have different arrival time. We also took Improved Round Robin algorithm from [3] and Simple Round Robin algorithm for comparison purpose. We compare them by taking same amount of TQ.

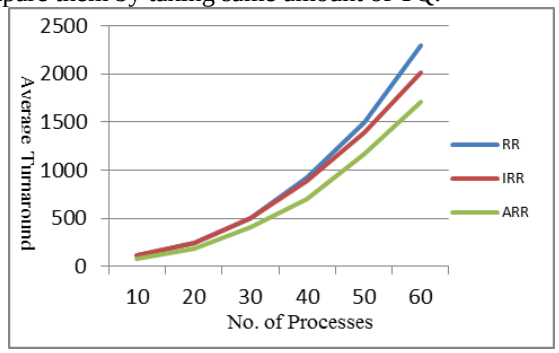


Fig. 11: Average Turnaround Time for Case -1

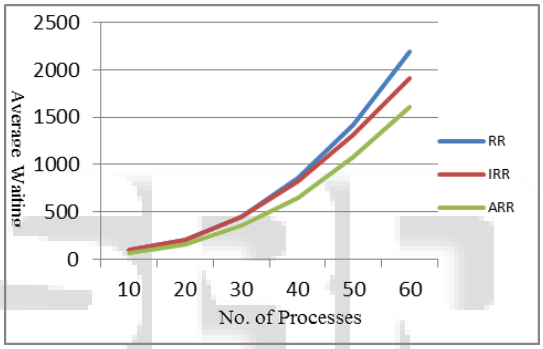


Fig. 12: Average Waiting Time for Case – 1

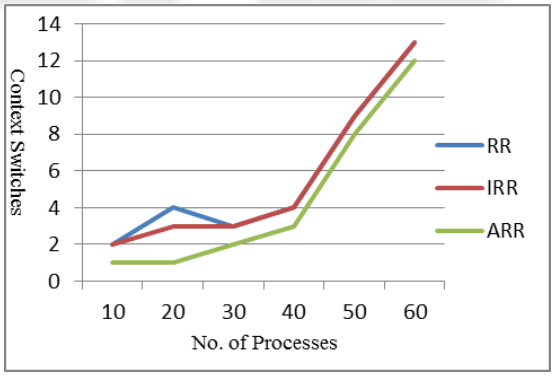


Fig. 13: Context Switches for Case -1

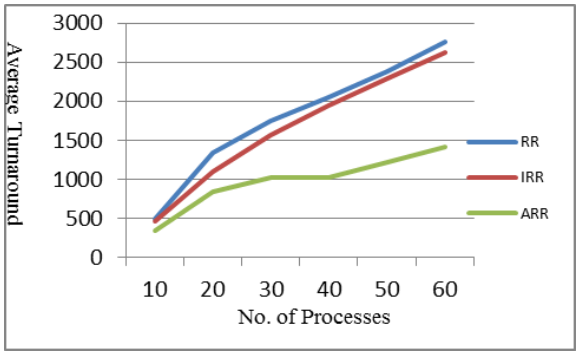


Fig. 14: Average Turnaround Time for Case – 2

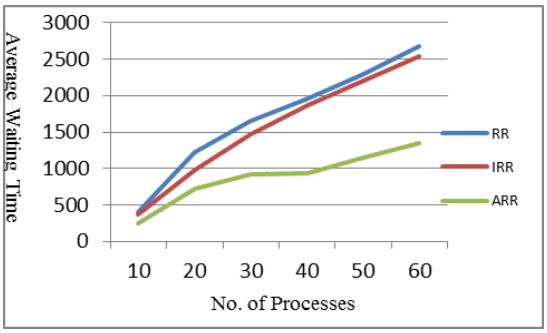


Fig. 15: Average Waiting Time for Case – 2

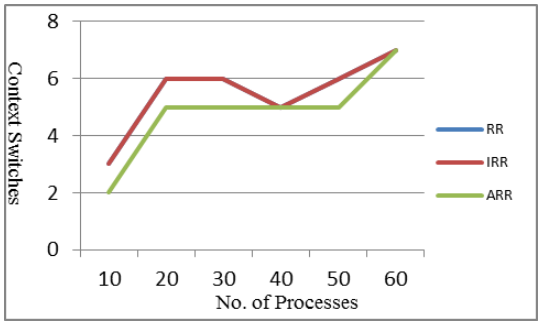


Fig. 16: Context Switches for Case – 2

VIII. CONCLUSION

Scheduler is the main part of any Operating System. Efficiency of scheduler depends on the design of the scheduling algorithm which fulfills scheduling goals. Different types of algorithms mention in section II so as we all know that performance of the Round Robin Scheduling has a very great concern with the size of Time Quantum. The proposed algorithm selects the optimum TQ for available processes by which the algorithm neither turn into FCFS nor generated more numbers of context switching and also perform best in terms of Average Turnaround Time (ATT) and Average Waiting Time (AWT). As mention into the results analysis section all the six graphs shows that Adaptive Round Robin (ARR) algorithm reduces the value of ATT, AWT and CS also as compare to the Round Robin algorithm. So with this article we can say that our algorithm is able to improve the performance of CPU at some level. You can also make some change with this algorithm and use other method to improve the performance of the operating system and make more advanced Scheduling algorithm.

REFERENCES

[1] Silberschartz, Galvin and Gagne, Operating System concept, 8th Edition, Wiley, 2009.
[2] Pushpraj Singh, Vinod Singh, Anjani Pandey, "Analysis and Comparison of CPU Scheduling Algorithms", International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 1, January 2014.
[3] Manish Kumar Mishra, Abdul Kadir Khan, "AN IMPROVED ROUND ROBIN CPU SCHEDULING ALGORITHM", Journal of Global Research in Computer Science, Volume 3, ISSN-2229-371X, June 2012, pg.no.64-69.
[4] SHEN Xue–qin, " The Realization And Performance Analysis of RR Process Scheduling Algorithm", 978-0-7695-5011-4/13 $26.00 © 2013 IEEE.
[5] MEHDI NESHAT, MEHDI SARGOLZAEI, ADEL NAJARAN, ALI ADELI, " THE NEW METHOD OF ADAPTIVE CPU SCHEDULING USING FONSECA AND FLEMING'S GENETIC ALGORITHM", Journal of Theoretical and Applied Information Technology, 15th March 2012. Vol. 37 No.1
[6] Wenbo Wu, Xinyu Yao, Wei Feng, Yong Chen, "Research on Improving Fairness of Linux Scheduler", 978-1-4799-1334-3/13/$31.00 ©2013 IEEE.
[7] Neetu Jain and P. V. Suresh, "A Java Based Visual Tool to Learn CPU Scheduling Algorithms", 978-93-80544-12-0/14/$31.00_c 2014 IEEE.
[8] Rabia Noor Enam, Syed Misbahuddin, Mumtazul Imam, " Energy Efficient Round Rotation Method for a Random Cluster Based WSN", 978-1-4673-1382-7/12/$31.00 ©2012 IEEE.
[9] Daqing Yang, Gouri Nair, Balaji Sivaramakrishnan, Harishkumar Jayakumar and Arunabha Sen," Round Robin with Look Ahead: A New Scheduling Algorithm for Bluetooth", 1530-2016/02 $17.00 © 2002 IEEE.
[10] Scheduling:http://en.wikipedia.org/wiki/Scheduling http://en.wikipedia.org/wiki/Round_robin_scheduling

