Volume 5, Issue 7, July 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Smart Optimized Round Robin (SORR) CPU Scheduling Algorithm

[1]Rahul Joshi*, [2]Shashi Bhushan Tyagi
[1]M.Tech Scholar, [2]Asst. Professor
[1, 2] Department of Computer Science & Engineering, Tula's Institute of Engineering, Uttarakhand, India

Abstract- ***The essential and important aspect of operating system is multiprogramming. Multiprogramming is a process or method of executing multiple processes simultaneously in the memory. Its main aim to minimize the average waiting time, average turnaround time and maximize the CPU utilization. There are various CPU scheduling algorithms are used to performed multiprogramming like First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling (PS) and Round Robin(RR).The most widely used scheduling algorithm is Round robin scheduling among all of them. It is an optimal CPU scheduling algorithms in timeshared systems. In round robin algorithm the time quantum is static and the performance of these algorithms is totally depending upon the selection of time quantum. This static time quantum decrease the performance of CPU scheduling like Average waiting time ,average turnaround time and CPU utilization. There are various algorithms like Improved Round Robin (IRR) and An Additional Improvement Round Robin(AAIRR) had been proposed in past to improve the performance of RR schdeduling.The objective of this paper is to proposed a new CPU scheduling algorithm which will perform better than the simple Round Robin, Improved Round Robin(IRR) and An Additional Improvement Round Robin(AAIRR) algorithms in terms of minimizing average waiting time, average turnaround time and number of context switches.***

Keywords: Context switching, CPU scheduling, Gantt chart, Response time, Round Robin CPU scheduling algorithm, Turnaround time, Waiting time.

I. INTRODUCTION

In a single-processor system, processor can run only one process at a time and other processes have to wait until the CPU is free and these processes can be rescheduled again. The objective of multiprogramming is to run multiple processes at a time and to maximize the uses of CPU. Scheduling is a fundamental operating-system function. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating-system design. CPU scheduling determines which processes run when there are multiple run-able processes.

II. CPU SCHEDULING ALGORITHMS

There are various CPU scheduling algorithms. Some basic CPU scheduling algorithms are as follow**.**

A. FIRST-COME-FIRST-SERVED

This is a non- preemptive scheduling algorithm. FCFS as the name indicates assigns priority to the processes in the order in which they request the processes. The process which requests the CPU first, CPU is allocated to that processes first. FCFS completes the jobs in order their arrival.

B. SHORTEST-JOB-FIRST

In SJF CPU is assigned to the processes which have the smallest CPU burst or task. This method is quite same as the FCFS but the difference is that the job with the shortest computation time executed first. When a job comes in, insert it in the ready queue based on its length. It gives the minimum average waiting time and minimum average turnaround time for a given set of processes [3]
The two Schemas of SJF are:

- ***NON-PREEMPTIVE:***

Once the CPU assigned, processes not preempted until its CPU burst completes. In non- preemptive scheduling, a new process can only come when the current process terminates or it goes to waiting state. Non-preemptive algorithms are designed so that once a process enters the running state in the queue; it is not removed from the processor until it has completed its service time or the burst time.

- ***PREEMPTIVE:***

Preemptive SJF is sometimes called Shortest Remaining Time First scheduling (SRTF). With preemptive CPU scheduling, a new process can run when interrupt occurs. The preemptive scheduling is prioritized. The highest priority should always be the process that is currently utilized.

The process with the highest priority should always be the one that be currently using the processor or CPU. If a process or job is currently using the processor and a new process with a higher priority enters the CPU, or the ready list, the process or job on the processor should be removed and returned to the ready list until it is once again the highest-priority process in the system.

C. PRIORITY SCHEDULING

In Priority based scheduling firstly highest priority processes will execute. Priority scheduling associate a priority with each process, allocate the CPU to the process with the highest priority. Priority scheduling can be non-preemptive or preemptive. In Priority scheduling processes having low priority will suffer from the problem starvation.

Starvation A major problem with priority scheduling algorithm is indefinitely blocking or starvation. A priority scheduling algorithm can leave some low priority processes waiting indefinitely. In starvation a low priority task or process can wait forever because there are always some other jobs around that have higher priority due to which lower priority process suffers from the problem called starvation. One common solution to this problem is **aging.** Aging is a technique of increasing the priority of processes that wait in the system for a long time.

D. ROUND ROBIN

Round Robin is similar on FCFS with preemptive time slicing. A time slice or time quantum is a small unit of time. RR scheduling can give the effect of all processor sharing the CPU equally with its given time quantum. In RR scheduling each process gets equal time to execute. Round Robin (RR) is one of the simplest scheduling algorithms for processes in an operating system, which assigns time slices to each process in equal portions and in circular order [6], handling all processes without priority [7], arguably, the major issue in RR is the time slice ([23], [22], [5], [21]).

III. SCHEDULING CRITERIA

There are various criteria to check the performance of CPU scheduling algorithms. These various criteria are used for comparing the various CPU scheduling algorithms. The criteria include the following:

1. CPU Utilization: It is the average fraction of time, during which the processor is busy.

2. Throughput: It refers to the amount of work completed in a unit of time. The number of processes the system can execute in a period of time. The higher the number, the more work is done by the system.

3. Waiting Time: The average period of time a process spends waiting. Waiting time may be expressed as turnaround time less the actual execution time.

4. Turnaround time: The interval from the time of submission of a process to the time of completion is the turnaround time.

5. Response time: Response time is the time from submission of a request until the first response is produced.

6. Priority: give preferential treatment to processes with higher priorities.

7. Fairness: Avoid the process from starvation. All the processes must be given equal opportunity to execute.

So we can conclude that a good scheduling algorithm for real time and time sharing system must possess following characteristics [4]:

- Minimum context switches.
- Maximum CPU utilization.
- Maximum throughput.
- Minimum turnaround time.
- Minimum waiting time.

Due to a number of disadvantages these scheduling algorithms have, they are severely used except Round Robin scheduling in timesharing and real time operating system; and considered most widely used CPU scheduling algorithm [4] [5].

IV. RELATED WORKS

Over a period of time, researchers have developed a number of CPU scheduling mechanisms which have been used for predictable allocation of CPU. Some of the important works are listed below.

In [2] author proposed Improved Round Robin (IRR) CPU scheduling algorithm works similar to Round Robin (RR) with a small improvement. IRR picks the first process from the ready queue and allocate the CPU to it for a time interval of up to 1 time quantum. After completion of process's time quantum, it checks the remaining CPU burst time of the currently running process. If the remaining CPU burst time of the currently running process is less than 1 time quantum, the CPU again allocated to the currently running process for remaining CPU burst time. In this case this process will finish execution and it will be removed from the ready queue. The scheduler then proceeds to the next process in the ready queue. Otherwise, if the remaining CPU burst time of the currently running process is longer than 1 time quantum, the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.

In [1] author proposed algorithm (AAIRR) focuses on improving more on the improved Round Robin CPU scheduling algorithm. The algorithm by reduces the waiting time and turnaround time drastically compared to the simple Round Robin scheduling algorithm. This proposed algorithm works in a similar way as but with some modification. It works in three stages: Stage 1: It picks the first process that arrives to the ready queue and allocates the CPU to it for a time

interval of up to 1 time quantum. After completion of process's time quantum, it checks the remaining CPU burst time of the currently running process. If the remaining CPU burst time of the currently running process is less or equal to 1 time quantum, the CPU is again allocated to the currently running process for remaining CPU burst time. In this case this process will finish execution and it will be removed from the ready queue. The scheduler then proceeds to the next shortest process in the ready queue. Otherwise, if the remaining CPU burst time of the currently running process is longer than 1 time quantum, the process will be put at the tail of the ready queue. Stage 2: The CPU scheduler will then select the next shortest process in the ready queue, and do the process in stage 1. Stage 3: For the complete execution of all the processes, stage 1 and Stage 2 have to be repeated.

V. PROPOSED ALGORITHM

The proposed algorithm (SORR) focuses on the improvement on AAIRR CPU scheduling algorithm [1]. The algorithm by [1] reduces the no of context switch, waiting time and turnaround time drastically compared to the IRR Scheduling algorithm[2] and simple Round Robin scheduling algorithm. This proposed algorithm works in a similar way as [1] but with some modification. The proposed SRR CPU scheduling algorithm is based on the small improvement in AAIRR CPU scheduling. It executes the shortest job having minimum burst time first instead of FCFS during the AAIRR [1], IRR [2] and simple Round robin algorithm and it also uses Smart time quantum instead of static time quantum. Instead of giving static time quantum in the CPU scheduling algorithms, our algorithm calculates the Smart time quantum itself according to the burst time of all processes. The proposed algorithm eliminates the discrepancies of implementing simple round robin architecture. In the first stage SRR CPU scheduling algorithms all the processes are arranged in the increasing order of CPU burst time. It means it automatically assign the priority to the processes. Process having low burst time has high priority to the process have high burst time. Then in the second stage the algorithm calculates the mean of the CPU burst time of all the processes. After calculating the mean, it will set the time quantum dynamically i.e. average of mean and highest burst time .Then in the last stage algorithm pick the first process from the ready queue and allocate the CPU to the process for a time interval of up to 1 Smart time quantum. If the remaining burst time of the current running process is less than 1 Smart time quantum then algorithm again allocate the CPU to the Current process till it execution. After execution it will remove the terminated process from the ready queue and again go to the stage 3.
The flowchart for proposed algorithm is shown below in figure 1:

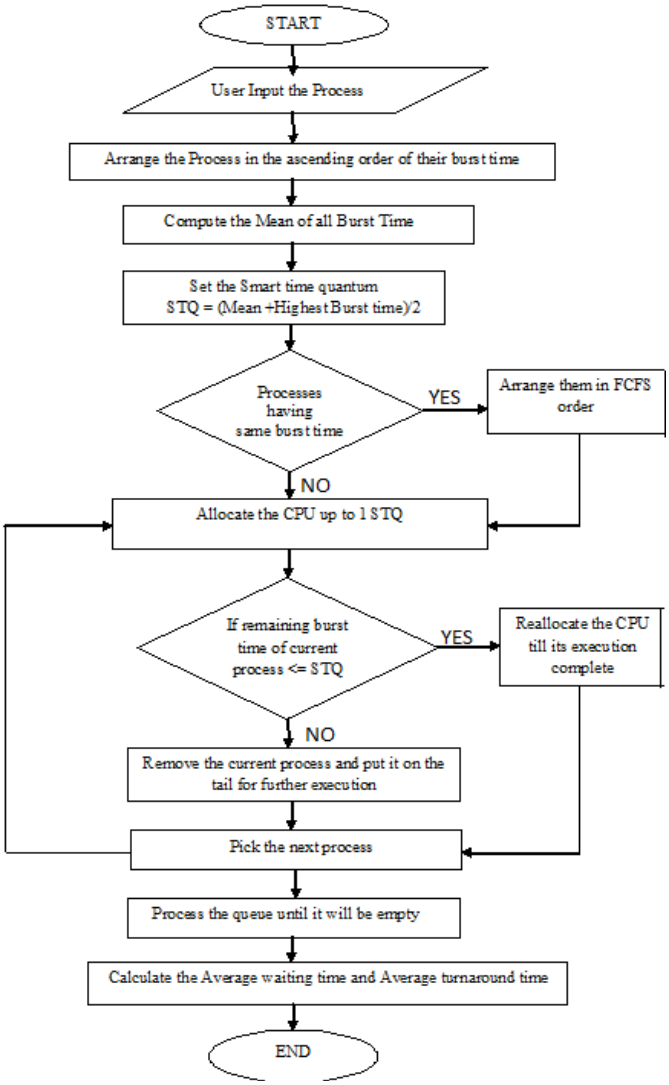


Figure 1 Proposed Algorithm Flowcharts

The steps of the proposed algorithm are as follow.

Step 1: START

Step 2: Arrange the processes in the increasing order of CPU burst time in the ready queue.

Step 3: Calculate the Mean of the CPU burst time of all the Processes

Mean (M)=(P1+P2+P3…….Pn)/n;

Step 4: Set the Smart time quantum (STQ) according to the following method

STQ=(Mean+ Highest burst time)/2;

Step 5: Allocate the CPU to the First process in the Ready Queue for the time period of 1Smart Time Quantum.

Step 6: If the remaining CPU burst time of the current process is less then or equal to 1 time quantum

- Reallocate the CPU to current process again for the remaining burst time. After the complete execution of the current process, remove it from the ready queue.
- Otherwise remove the current process from the ready queue and put it on the tail of the ready queue for further execution.

Step 7: Pick the next process from the ready queue and allocate the CPU to it up to the time period of 1 Smart time quantum and then again to the step 6.

Step 8: Process the queue until it will be empty.

Step 9: Calculate the No. of context switch, Average waiting time and Average Turnaround time of all process.

Step 10: END

VI. EXPERIMENTAL ANALYSIS

The experimental analysis that will be adopted in this paper uses all the assumptions and experiments performed in [1] and then compared the results in the original paper along with the results by the proposed algorithm.

1) CPU Burst Time in Increasing Order: The ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0 with burst time 5, 12, 20, 26 and 34*ms* respectively was considered. The time quantum in RR,IRR,AAIRR is static according to their algorithm i.e. 10 ms and in our proposed algorithm time quantum will be calculated dynamically(STQ).The comparative results of RR, IRR, AAIRR and proposed SORR are shown in Table 1. Figures 2-5 show the Gantt chart representation of RR, IRR, AAIRR and SORR respectively. Figure 5 shows the bar chart of the comparison.

.

Table 1: Comparative results of RR, IRR,AAIRR and SORR

Algorithm	Average Waiting Time(ms)	Average Turnaround Time (ms)	Number of Context Switch
RR	38.4	57.8	10
IRR	30.4	49.8	7
AAIRR	26.4	45.8	6
SORR	24.4	43.7	4

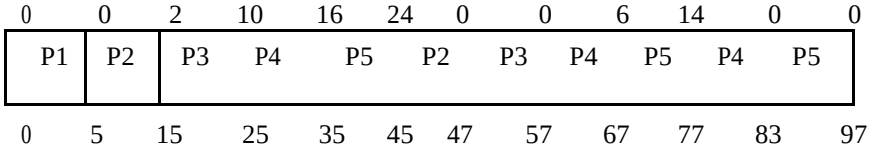


Figure 2: Gantt chart representation of RR

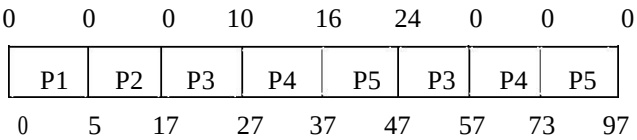


Figure 3: Gantt chart representation of IRR

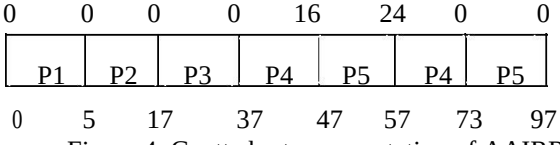


Figure 4: Gantt chart representation of AAIRR

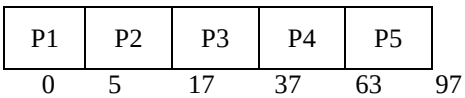


Figure 5: Gantt chart representation of SORR

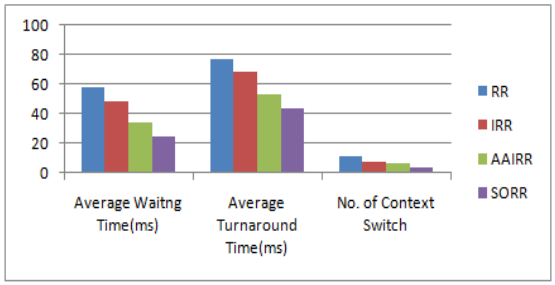


Figure 6: Bar chart of Case 1 (Increasing order)

2) CPU Burst Time in Decreasing Order: The ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0 with burst time 34, 26, 20, 12 and 5*ms* respectively was considered. The time quantum in RR, IRR, and AAIRR is static according to their algorithm i.e. 10 ms and in our proposed algorithm time quantum will be calculated dynamically (STQ).The comparative results RR, IRR and, AAIRR and SORR are shown in Table 2. Figures 7-10 show the Gantt chart representation of RR, IRR, AAIRR and SORR respectively. Figure 11 shows the bar chart of the comparison.

Table 2: Comparative results of RR, IRR, AAIRR and SORR

Algorithm	Average Waiting Time(ms)	Average Turnaround Time (ms)	Number of Context Switch
RR	58	77.4	11
IRR	49	68.4	8
AAIRR	34.4	53.8	7
SORR	24.4	43.79	4

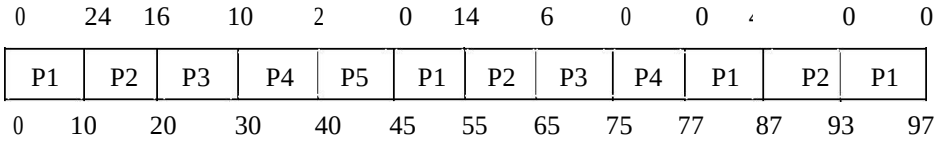


Figure 7: Gantt chart representation of RR

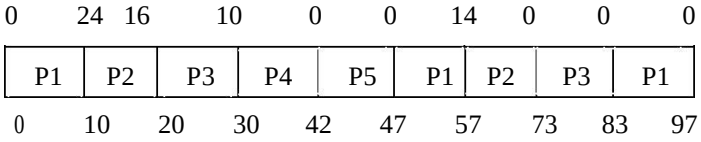


Figure 8: Gantt chart representation of IRR

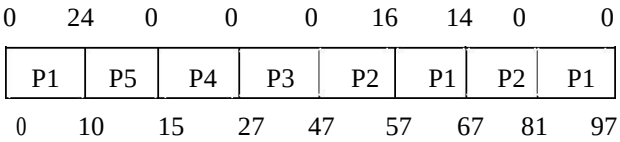


Figure 9: Gantt chart representation of AAIRR

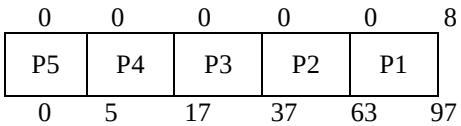


Figure 10: Gantt chart representation of SORR

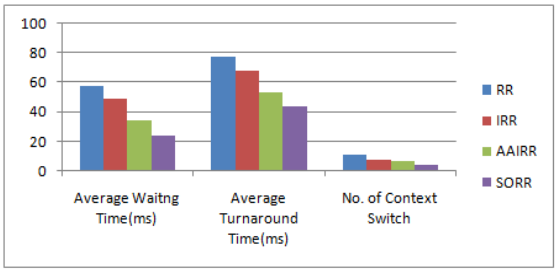


Figure 11: Bar chart of Case 2 (Decreasing order)

3) CPU Burst Time in Random Order: The ready queue with five processes P1, P2, P3, P4 and P5 arriving at time 0 with burst time 20, 34, 5, 12 and 26 respectively was considered. The time quantum in RR, IRR, AAIRR is static according to their algorithm i.e. 10 ms and in our proposed algorithm time quantum will be calculated dynamically (STQ). The comparative results of RR, IRR, AAIRR and SORR are shown in Table 3. Figures 12-15 show the Gantt chart representation of RR, IRR, AAIRR and SORR respectively. Figure 16 shows the bar chart of the comparison.

Table 3: Comparative results of RR, IRR and AAIRR

Algorithm	Average Waiting Time(ms)	Average Turnaround Time (ms)	Number of Context Switch
RR	48	67.4	11
IRR	40.4	59.8	8
AAIRR	31	50.4	6
SORR	24.4	43.7	4

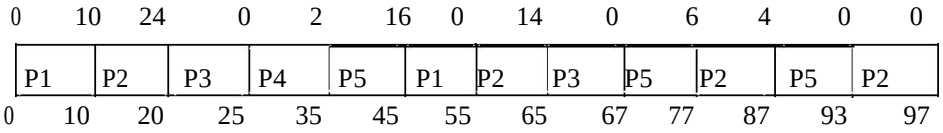


Figure 12: Gantt chart representation of RR

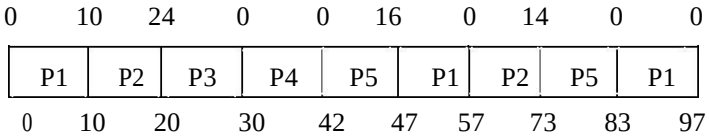


Figure 13: Gantt chart representation of IRR

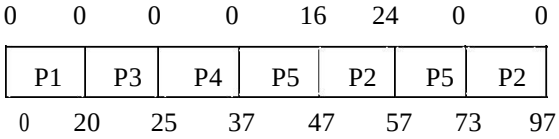


Figure 14: Gantt chart representation of AAIRR

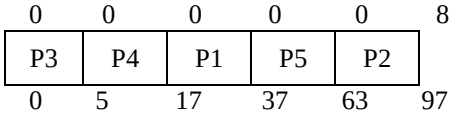
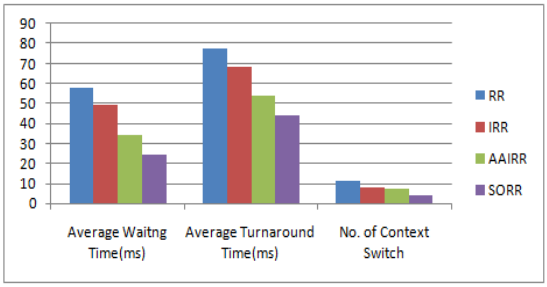


Figure 15: Gantt chart representation of SORR



b

Figure 16: Bar chart of Case (Random order)

VIII. CONCLUSION AND FUTURE SCOPE

In this paper, we have presented a Smart and Optimized Round Robin CPU scheduling algorithm which improved on the An Advanced Improved Round Robin Scheduling algorithm [1] and Improved Round Robin CPU scheduling algorithm [2]. Results have shown that the proposed algorithm gives better results in terms of average waiting time, average turnaround time and number of context switches in all cases of process categories than the simple Round Robin CPU scheduling algorithm, Improved Round Robin CPU scheduling algorithm and the An Advanced Improved Round Robin CPU Scheduling algorithm. The general problem in any form of Round Robin CPU scheduling algorithm, Improved Round Robin CPU scheduling algorithm and the An Advanced Improved Round Robin CPU Scheduling algorithm is the choice of time quantum. In all these proposed algorithms time quantum is static due to which in these cases the number of context switches, average waiting time and average turnaround time will be very high and in our proposed algorithm

(SORR) time quantum is calculated dynamically according to the burst time of all processes and it will find out a smart time quantum for all processes which gives good performance as compared to RR, IRR and AAIRR. So, a future scope is to determine more accurate dynamic time quantum or specific time quantum for each process so as to improve the performance.

REFERENCES

[1] Abdulraza, Abdulrahim,Salisu Aliyu, Ahmad M Mustapha, Saleh E Abdullahi, " An Additional Improvement in Round Robin (AAIRR) CPU Scheduling Algorithm," International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 2, February 2014 ISSN: 2277 128X.

[2] Manish kumar Mishra and Abdul kadir khan, "An Improved Round Robin CPU scheduling algorithm," Journal of Global Research in Computer Science Volume 3, No. 6, June 2012.Neetu

[3] Weiming Tong and Jing Zhao, "Quantum Varying Deficit Round Robin Scheduling Over Priority Queues", International Conference on Computational Intelligence and Security, pp. 252- 256, China, 2007

[4] Ajit, S, Priyanka, G and Sahil, B (2010): An Optimized Round Robin Scheduling Algorithm for CPU Scheduling, International Journal on Computer Science and Engineering (IJCSE), Vol. 02, No. 07, 2383-2385, pp 2382-2385.

[5] Ishwari, S. R and Deepa, G (2012): A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems, International Journal of Innovations in Engineering and Technology (IJIET), ISSN: 2319 – 1058, Vol. 1 Issue 3, pp 1-11.

[6] A. Bashir, M.N. Doja and R., Biswas, "Finding Time Quantum of Round Robin CPU Scheduling Algorithm Using Fuzzy Logic," The International Conference on Computer and Electrical Engineering, ICCEE 2008

[7] A., Silberschatz, J. L., Peterson, and P.B., Galvin, "Operating System Concepts," Addison Wesley, 7ED 2006

[8] T. Helmy and A. Dekdouk, "Burst round robin as a proportional-share scheduling algorithm," In Proceedings of The fourth IEEE-GCC Conference on Towards Techno-Industrial Innovations, pp. 424-428, 11-14, at the Gulf International Convention Center, Bahrain. 2007

.

