Efficient Process Scheduling Algorithm using RR and SRTF

Preeti Sinha
School of Information Technology and Engineering
Vellore Institute of Technology
Vellore, India
sinhapreeti128@gmail.com

B.Prabadevi
School of Information Technology and Engineering
Vellore Institute of Technology
Vellore, India
prabadevi.b@vit.ac.in

Sonia Dutta
School of Information Technology and Engineering
Vellore Institute of Technology
Vellore, India
duttasonia4321@gmail.com

Deepa N
School of Information Technology and Engineering
Vellore Institute of Technology
Vellore, India
deepa.rajesh@vit.ac.in

Neha Kumari
School of Information Technology ad Engineering
Vellore Institute of Technology
Vellore, India
neha.kumari2017@vitstudent.ac.in

2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE) 978-1-7281-4142-8/20/$31.00 ©2020 IEEE 10.1109/ic-ETITE47903.2020.101

***Abstract*— The performance of the multiprocessor system and time-sharing system rely upon CPU scheduling algorithm. Some of the well known efficient CPU scheduling algorithms are Round Robin scheduling (RR), Shortest job first(SJF), Preemptive version of SJF(SRTF), First Come First Serve(FCFS) etc. As of now, the Round Robin scheduling algorithm is considered as the efficient process scheduling algorithm among all the existing CPU scheduling algorithms. However, in RR the shortest one have to wait for a longer time and in SRTF longer process behaves as a suspended process as short tasks keep on executing. This paper proposes a new scheduling process algorithm ESRR (Efficient Shortest Remaining Time Round Robin) that consolidate two of the preemptive version of existing scheduling algorithms namely RR and SRTF. ESRR reduces total waiting time and turn around time in compare to RR and reduces the waiting time of the shorter process and it also provides a longer process to execute faster than SRTF.**

Keywords—Round Robin, First come First Serve, Shortest job First, Efficient Shortest Remaining time Round Robin, Preemptive, Multiprocessor, Timesharing

I. INTRODUCTION

The fundamental function of an operating system is SCHEDULING. Before the resources are used, they must be scheduled. Operating system focus on multitasking environment in which Round Robin scheduling plays its part. CPU scheduling needs that all the processes allocate the CPU fairly and processes do not get into starvation. Round Robin scheduling algorithm works on the principle of First come First Serve scheduling algorithm, but the preemption is the added functionality to switch between the processes. In SRTF, CPU scheduler selects the process which has the shortest amount of time remaining to complete execution. A major problem of RR is fixed time slice value.

An Operating System (OS) is a software component of the computer system which is responsible for proper management and coordination of tasks and the sharing of resources of the computer. The fundamental function of an operating system is SCHEDULING. Before the resources are used, they must be scheduled. The Operating system acts as a host for all the applications programs which run on the machine. It also handles all the details related to the operation of the hardware.

Some of the significant features of the Operating System are as follows:

- Processor Managing
- Memory Management
- File Handling
- Device Managing
- Security Handling
- System performance controlling
- Error detecting and handling
- Job accounting and handling
- Synchronisation with other software and users.

To manage these features CPU has many scheduling algorithms. These algorithms deal with the problem of the process in ready-queue which is to be allocated to the CPU. There are several scheduling algorithms based on which scheduler selects the processes. These are mainly of two types:

- The non-preemptive scheduling algorithm
- Pre-emptive scheduling algorithm

Non-Preemptive – Once a process is allocated to the CPU, the CPU can't be interrupted from that process until its completion. FCFS, SJF, Priority Scheduling are some of the examples of Non-Preemptive scheduling.

Pre-emptive - Once a process is allocated to CPU, CPU can be interrupted from that process whether its execution has been completed or not. RR, SRTF etc are examples of Pre-emptive scheduling algorithm.

FCFS : Oone of the meekest scheduling algorithm in order of process arrival. Its implementation is done by FIFO (First In First Out). As it is non pre-emptive, it runs to its completion. Hence there is a low rate of resource utilisation, due to which short jobs suffer the delay in the execution.

SJF , when scheduling of job is done on the basis of shortest execution time. It can be implemented either in pre-emptive and non-preemptive scheduler. When SJF scheduler invoked it searches in the ready queue to find the processes in its shortest execution time. It works optimally only when the

exact future of the process are known at the time of scheduling. It is sometimes difficult to estimate the process behaviour in future reliably. Hence effective implementation of SJF is somewhat complicated.

The RR scheduling algorithm is also called a Fair shared algorithm. It is used in time-sharing and user system where the main objective is to provide good response time and share the system equally among all users. Basically, the CPU time is divided in a time slice. Each process is allocated a one-time slice while running. No process can run for more than one time slice while there are other processes are waiting in the ready queue. If a process needs more than one-time slice to complete it goes at the end of the ready queue. If the running processes release control to the Operating system due to input-output request then other processes are scheduled for running. Hence this algorithm utilises the system resources inequitable manner.

S**RTF** scheduling algorithm selects the process for the execution, which has the shortest amount of time remaining to complete its execution. SRTF may lead to starvation. Means the short processes are added continuously to the CPU scheduler then currently running process will never be capable to execute.

Selection of a CPU scheduling algorithm depends upon the following selection criteria:

- **Throughput:** It is the total no. of processes that complete its execution in the given unit of time.
- **CPU Utilisation:** It includes the utilisation of a CPU in the best way and not to spare any CPU cycle.
- **Turnaround time:** Amount of time taken for the execution of a particulate process. It includes the interval time between submission of a process and the completion of its execution.
- **Waiting time:** It includes the amount of time expended in the ready queue.
- **Response time:** It includes the amount of time from when a process makes a request until the first response of the process is generated.

II. LITERATURE SURVEY

In paper [1], the authors presented the PFRR (Prioritized Fair Round Robin) scheduling algorithm. This technique is a combination of SJF, RR, and priority scheduling with dynamic time quantum. Their objective to set the time quantum value based on priority and execution time of the tasks.

The author has taken two cases to demonstrate PFRR. Case I. The author has considered five processes and their burst time is 15ms, 32ms, 10ms, 26ms, and 20ms. The author assumed that the priority of all of the processes is the same. All the processes arrived at the same time in the ready queue. The average waiting time(AWT) is 38.77, average turn around time(ATAT) is 59.37 and total no of context switches (CS) is 10. Case 2. The has taken the same processes but with different priorities. The arrival time of all the processes is identical. The AWT is 35.03ms, ATAT is 56.43 and number of CS is 9. By comparing AWT, ATAT and the total number of CS, it is concluded that PFRR is better than IRRVQ, Round robin and RR based on Priority.

In paper [2] the authors describe a GPA (generalized priority algorithm) to achieve efficient execution in compare to FCFS and RR scheduling. In the Generalised Priority Algorithm, the processes are prioritised on the basis of their size in such a manner that the process having the highest size gets the highest rank.Five VMs with 512 RAM having MIPS 1000, 500, 250, 250, 250. The proposed algorithm is proved to be better than the conventional algorithm like FCFS and RR.

In paper [3] the author represented a new scheduling algorithm Varying Response Ration Priority (VRRP) which is the combination of priority-algorithm and shortest remaining time has been proposed so that all the processes are executed with minimum AWT and no starvation.To demonstrate this algorithm the author took 5 processes, P1, P2, P3, P4, P5 with arrival time 0ms, 1ms, 3ms, 4ms, 10ms and burst time is 10ms, 3ms, 5ms, 4ms, 6ms.

In case VRRP, we get average waiting time is 7.8ms, average response time 6.4ms and no of context switches is 6. So the proposed algorithm never lead to starvation and is better than FCFS, SJF, RR, highest response ration next, adaptive RR and mid average round robin.

In paper [4] the author proposed an Improved RR CPU scheduling algorithm which is an extended version of RR where each process is allotted a priority level low, medium and high and on the basis of this priority level the time quantum for that process is assigned. By executing processes with smaller burst time the throughput of the algorithm is increased.

In this the author took five processes with burst time 550ms, 800ms, 200ms, 2600ms and 1600ms and arrival time 0ms, 200ms, 100ms, 400ms, 0ms and priority is 3, 1, 3, 2 and 2 respectively.

By conventional RR, the AWT is 2090ms, ATAT is 3240ms and by the proposed algorithm the AWT is 1640ms, ATAT is 2790ms. The Improved RR minimizes the no. of CS. In addition, it reduces the AWT and ATAT when compared to RR. In future, the author will modify the proposed algorithm for dynamic time quantum. It can also be implemented based on the priority-based algorithm.

In paper [5] the new CPU scheduling algorithm has been introduced known as Dynamic Average Burst Round Robin(DABRR). In this algorithm, dynamic time quantum has been used instead of static time quantum in RR. The author demonstrated this algorithm by considering five processes P1, P2, P3, P4, P5 in a ready queue. The arrival time of all processes is zero milliseconds. The burst time of P1, P2, P3, P4, P5 are 15ms, 32ms, 102ms, 48ms,29ms, respectively. Static time quantum is defined as total burst time / #processes in ready queue (i.e, TQ=TBT/n). By following DABRR, the turn around time of individual processes are 15ms, 76ms, 226ms, 169ms, 44ms respectively and an average turn around time is 106ms. The waiting time of individual processes is 0ms, 44ms ,124ms,121ms,15ms respectively and average waiting time 60.8ms. And the total context switch is 21.

This paper exhibits demonstration of better performance in terms of AWT,ATAT and CS. DABRR saves 41 percent of waiting time in comparison to round robin algorithm as well as it saves 31 percent of turn around time in comparison to

traditional RR. This algorithm does perform well in multiprocessor environment.

In paper[6] the author presented an enhanced Round Robin CPU scheduling algorithm Improved Round Robin CPU scheduling algorithm with varying time quantum (IRRVQ) which using the concept of Shortest Job First scheduling algorithm with varying time quantum of RR.

This algorithm is demonstrated using four processes p1, p2, p3, p4 arranged in a ready queue. The arrival time for all processes is zero and burst time is 12ms, 8ms,21ms, 15ms respectively. In the ready queue, all the processes are arranged in ascending order according to their burst time. And the CPU is assigned to the processes for a time quantum equal to the burst time of the first process in the ready queue. The waiting time for P1 is 24ms, P2 is 0ms, P3 is 35ms, P4 is 32ms. And average waiting time 22.75ms. The average turn around time is 41.75ms by following IRRVQ. When time quantum is 8 the avg. turn around time is 39.75ms, for time quantum 4 the avg. turn around time is 41.75ms, for time quantum is 3, the avg. turn around time is 43.25ms and when time quantum is 6 the avg. turn around time is 42.75ms in RR.

In this algorithm, the CPU has been effectively utilised as compared to traditional RR algorithm. In addition to this, the waiting time and turn around has also been reduced. More effort needs to be done to make the system more effective.

In paper[7] emphasised better scheduling in a real-time system. In non-real system, the deadline for scheduling algorithm is not considered whereas in the real-time system the deadline is the backbone for scheduling the processes. The RRP (Round Robin with Priority) been implemented to minimize the AWT so that it can produce an efficient output in a minimal amount of time.

To explain this algorithm, the author took seven processes 0, 1, 2, 3, 4, 5 and 6 whose burst time is 9ms, 5ms, 8ms, 2ms, 7ms, 3ms and 4ms having priority 1, 3, 4, 0, 5, 6 and 0 respectively. And the time quantum is set to 3ms. The average time waiting for this algorithm is 13.4ms and the average turn around time is 18.85ms.

The avg waiting is minimum than the traditional RR and all other algorithms excluding SRTF. Turn around time is also minimum in comparison to RR. CPU is effectively utitlised. The performance of the system needs to be enhanced.

In paper [8] An algorithm OMDRRS (An Optimimum Multilevel dynamic Round Robin Scheduling algorithm) is introduced to enhance the performance of Round Robin and priority scheduling algorithm. OMDRRS combines all the traditional scheduling algorithms with dynamic time quantum. The Time quantum of each process is calculated using priorities, CPU execution time and context switch.

To demonstrate this algorithm the author considered 50 processes with different arrival time. Following the algorithm, set the priority for all the processes. The WT and TAT is executed on simulator and evaluation are done on ANOVA and t-test.

Enhanced RR scheduling algorithm had been proposed by [9] where the time quantum is decided based on the burst time of the processes. They have implemented it using java and fount ERRBTQ is better than other RR algorithms.

Statistics depict that the enhanced confidence index of OMDRRS. OMDRRS gives better result than all the conventional algorithms in terms of waiting time, turn around time, response time and context switching. This algorithm also decreases the overhead and saves memory space. More improved can be done to increase latency..

III. PROPOSED ALGORITHM

Flow diagram of ESRR algorithm is shown in Fig. 1.

Our proposed algorithm ESRR presents a solution to reduces total waiting time and turn around time in compare to RR and reduces the waiting time of shorter process and it also provides longer process to execute faster than SRTF. Our proposed approach gives better result than RR and SRTF for longer and shorter processes. RR works on the principle of FFCS algorithm but the preemption is the added functionality to switch between the processes . In SRTF, CPU scheduler select the process whichhas shortest amount of time remaining to complete execution . Our proposed algorithm consolidates SRTF and time sharing concept of RR Following is the proposed ESRR algorithm:

Input: Let us take five processes named P1, P2, P3, P4, P5 and we take the time quantum as 4 millisecond. The execution time of the processes are 8,10,5,7,3 respectively.

Output: According to ESRR average waiting time is 8.8ms and average turnaround time is 15.4ms Variables: P[n], arr_time[n], exe_time, TQ, Ready queue.

Step1: First take input of no. of processes into n.
Take input of exe_time and arr_time
Allocate CPU to front process
Step2:
If(new process arrives)
If(no process exists in ready queue)
Allocate cpu to new process
Else
Compare exe_time of process with the existing processes from front to rear
If(exe_time is less)
Put the new process before comparing process in the ready queue
Else
insert in the queue at (++rear) position.
Endif
Endif
Endif
If (process exists)
If(remexe_time<=TQ)
Allocate CPU to front process for remexe_time Remove the process from ready queue Else
Allocate for the TQ time interval
Endif
Endif
Repeat this block till all the process get executed.
Step3:
If (new process added)
Input exe_time and repeat step2
Endif

Here we demonstrate an example to depict the working principle of ESRR.

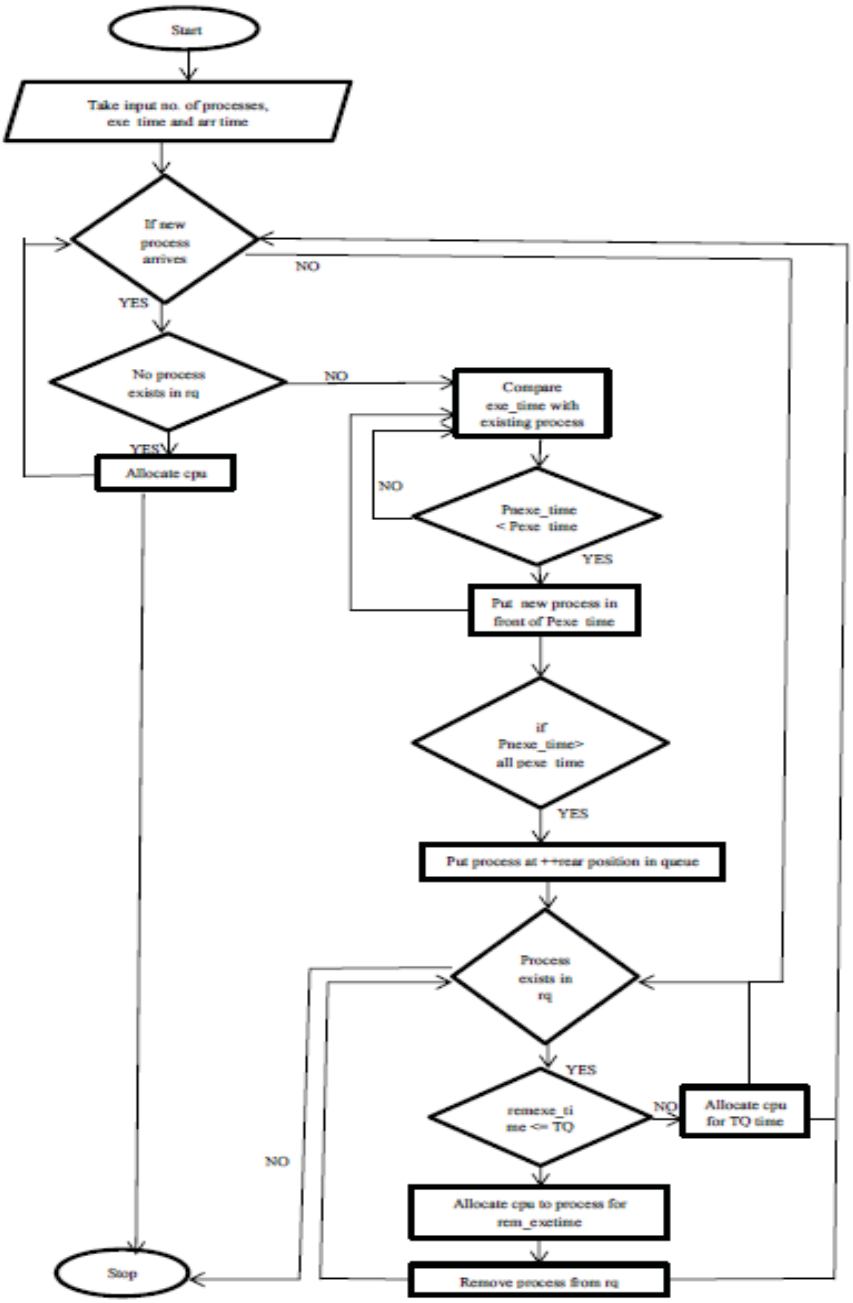

Fig. 1. Flow of the proposed system

TABLE I. EXECUTION TIME AND ARRIVAL TIME OF ALL PROCESSES

Arrival time(ms)	Process name	Execution Time
0	P1	8
0	P2	10
2	P3	5
5	P4	7
7	P5	3

Let the Time Quantum be 4 millisecond.

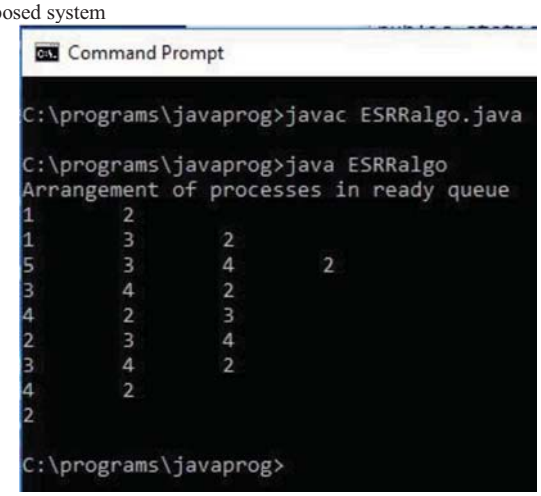

Fig. 2. The output of the source code of ESRR

Our proposed algorithm ESRR will work according to the following technique.
At 0ms P1(8ms) and P2(10 ms) arrives as shown in gnat chart Fig.3.

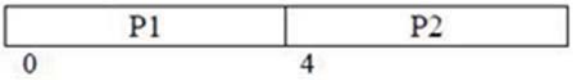

Fig. 3. Gantt chart at 0s

At 2 ms P3(5 ms) arrives. After execution of P1 for 4ms processes arranged as in Fig. 4.

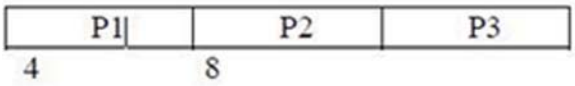

Fig. 4. Gantt chart after 4ms

At 5ms P4(7ms) arrives, at 7ms P5(3ms) arrives. After execution of P1 for 8ms processes are arranged as in Fig. 5.

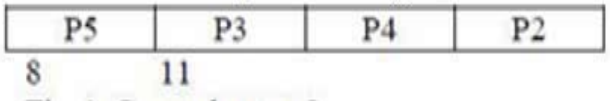

Fig. 5. Gantt chart after 8ms

After 3ms execution of P5 finishes. Now no new processes are arriving, So the execution will be as in Fig. 6.

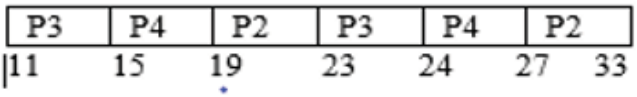

Fig. 6. Gantt chart till the end of execution of all processes

IV. CALCULATION OF CPU PERFORMANCE

Turnaround time of individual processes

P1= (8-0) ms = 8 ms
P2 = (33-0) ms = 33 ms
P3 = (24-2) ms = 22 ms
P4 = (27-5) ms = 22 ms
P5 = (11-7) ms = 4 ms

Waiting time of individual processes

P1= 0 ms
P2 = (33-10) ms = 23 ms
P3 = (22-5) ms = 17 ms
P4 = (22-7) ms = 15 ms
P5 = (4-3) ms=1ms

Average turnaround time = (8+33+22+22+4)/5 = 17.8
Average waiting time = (0+23+17+15+1)/5 = 11.2
In round robin scheduling algorithm ,
average waiting time = 16.8 ms
average turnaround time = 23.4 ms
In SRTF ,
average waiting time = 8.4 ms
average turnaround time = 15 ms

In ESRR the average waiting time and turn around time is comparatively lesser than RR but slightly greater than SRTF. This is depicted in Fig. 7. And it gives a turn to longer processes on a time-sharing basis which is not possible in SRTF.

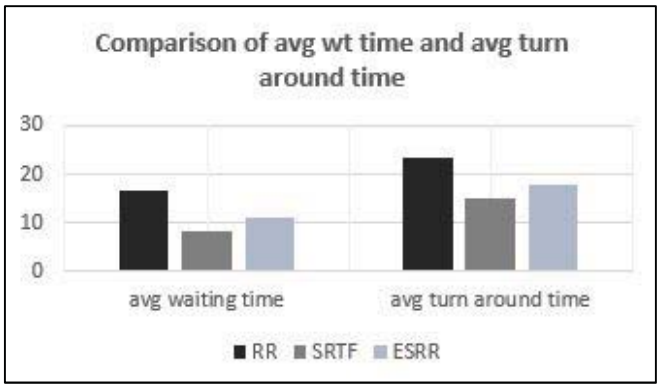

Fig. 7. Bar Graph depicting the comparison between average waiting time and average total turn around time of RR, SRTR, ESRR algorithm.

V. CONCLUSION

In this paper, ESRR algorithm is proposed for eliminating the drawbacks of RR and SRTF CPU scheduling algorithms. The performance of a CPU scheduling algorithm depends on the Turnaround time, the total number of context switches, waiting time, response time. Therefore ESRR improves the performance of the CPU scheduling algorithm by reducing waiting time for longer process and also reducing the Turnaround time. ESRR is compared to various CPU scheduling algorithms including RR, SRTF, SJF and prioritized fair RR algorithm. From the results and observations, we have proved that our proposed CPU scheduling algorithm ESRR is better than RR, PFRR, SRTF. Thus, ESRR reduces total waiting time and turnaround time in comparison to RR and reduces the waiting time of the shorter process and it also provides a longer process to execute faster than SRTF.

REFERENCES

[1] Yasin, A., Faraz, A., & Rehman, S. (2015, December). Prioritized Fair Round Robin Algorithm with Variable Time Quantum. In Frontiers of Information Technology (FIT), 13th International Conference on (pp. 314-319). IEEE.

[2] Agarwal, D., & Jain, S. (2014). Efficient optimal algorithm of task scheduling in cloud computing environment. arXiv preprint arXiv:1404.2076.

[3] Singh, P., Pandey, A., & Mekonnen, A. (2015). Varying response ratio priority: a preemptive CPU scheduling algorithm (VRRP). Journal of Computer and Communications, 3(04), 40.

[4] Mishra, M. K. (2012). An improved round robin cpu scheduling algorithm. International Journal of Global Research in Computer Science (UGC Approved Journal), 3(6), 64-69.

[5] Dash, A. R., & Samantra, S. K. (2016). An optimized round Robin CPU scheduling algorithm with dynamic time quantum. arXiv preprint arXiv:1605.00362.

[6] Mishra, M. K. (2012). An Improved Round Robin CPU Scheduling Algorithm. Journal of Global Research in Computer Science, 3(6), 64-69.

[7] Charu Rani, Mrs. Manju Bala (2015). Comparison of Round Robin based on CPU Scheduling Algorithms. International Journal of Advanced Research In Computer and Communication Engineering. Vol. 4, Issue 9,

[8] Goel, N., & Garg, R. B. (2016). Performance analysis of cpu scheduling algorithms with novel omdrrs algorithm. International Journal of Advanced Computer Science and Application, 7(1), 216-221.

[9] Indusree, J. R., & Prabadevi, B. (2017, November). Enhanced round robin CPU scheduling with burst time based time quantum. In Materials Science and Engineering Conference Series (Vol. 263, No. 4, p. 042038).M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.