An Efficient Technique for Scheduling Algorithm In Real Time Environment For Optimizing the Operating System

M. Arvindhan[1], Reetu Singh[2],Kajol Kathuria[3]

[1]Assistant Professor, [2 3] PG Student,
[1 2 3]Department of CSE, [1 2 3] Galgotias University, Uttar Pradesh, India

Abstract-- The main intention of writing this paper is to introduce a new CPU algorithm called SJRTRR CPU Scheduling Algorithm. The proposed algorithm is based on Round Robin and helps to improve the average waiting time of Round Robin algorithm. Classic real-time scheduling algorithms RMS, EDF and LLF are discussed in uniprocessor systems. The scheduling thought and strategies are investigated in multiprocessor systems.

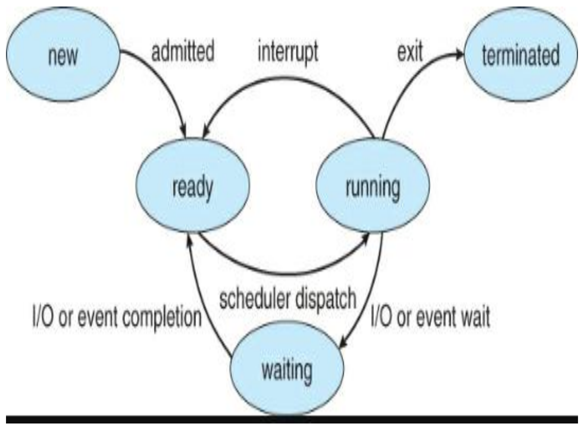
Keywords: *Scheduling Algorithms, SJRTRR, SJRT, RR, Turnaround time, waiting time ,centralized multiprocessscheduling***.**

I. INTRODUCTION

Various operations will be performed by the operating system Central Processing Unit scheduling is one of them. CPU scheduling is done by the short term scheduler. CPU scheduling is basically sharing of resources between multiple processes. Process is nothing but a program under execution, process should reside in the main memory, it occupied the CPU to execute the instructions. CPU scheduling is possible only in the multiprogramming operating systems. Any process will have basically five states that states are-

1. new
2. Ready
3. running
4. waiting
5. termination

Initially process will be in the new state it means the process under creation or being created. In the ready state there will be multiple no. of processes and one of the process will be selected and that will be dispatch onto the running state. In the running state there will be only one process at a time , it requires a input\output operation when it will get then process will done its execution and terminate otherwise it will be moved onto wait state.



Scheduling algorithms are divided into two parts:
1. Pre-emptive
2. Non pre-emptive

The proposed algorithm in this paper is preemptive in nature and focusing on average waiting time.

II. SCHEDULING PARAMETERS

The process is having different times or scheduling parameters:

1. **Arrival time:** time when the process is arrived into the ready state is called arrival time.
2. **Burst time:** time required by the process to complete its execution is called burst time.
3. **Completion time:** time when the process is done its complete execution is called completion time.
4. **Turnaround time:** time difference between the completion time and arrival time is called turnaround time.
5. **Waiting time:** time difference between turnaround time and burst time is called waiting time.
6. **Response time**: time difference between first response and arrival time of that process is called response time.
7. **Priority:** give preference to highest priority process.

III. EXISTING CPU SCHEDULING ALORITHMS

Goal of the CPU scheduling algorithm is to maximize the CPU utilization and throughput of the system and minimize the average waiting time, average turnaround time and average response time.

A. *FIRST COME FIRST SERVE:*

This algorithm is run on the basis of arrival time and its mode will be non pre-emptive. With this plan the process that demands the CPU initially is assigned the CPU first. The execution of the FCFS strategy is effectively keep up by the

FIFO queue. at the point when a process enters the ready queue its PCB is connected onto the tail of the queue. When the CPU is free it is allotted to the process at the leader of the queue. The running process is then expelled from the queue.

B. SHORTEST JOB FIRST

This algorithm associates with each process the length of the processes' next CPU burst. At the point when the CPU is accessible it is appointed to the process that has the lowest next CPU burst .if the CPU burst of the two processes are same FCFS algorithm is applied to break the tie.

C. PRIORITY ALGORITHM

A priority is connected with each process and the CPU is allocated to those processes which have highest priority. The processes have same priority are scheduled in FCFS order. Basically a SJF algorithm is same as priority algorithm where the priorities are inverse of the CPU burst time. Priority is lower CPU burst time is higher and vice versa.

D. ROUND ROBIN ALGORITHM

Round robin algorithm is specially introduced for the time sharing systems. it is behave like a FCFS scheduling but preemption is used to enable the processes for the switching purpose. A small unit of time known as time quantum is used. The ready queue is treated like a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval up to 1 time quantum.

IV. PROPOSED WORK: SJRTRR CPU SCHEDULING ALGORITHM

In the proposed algorithm we have tried to improve the average waiting time of the traditional Round robin algorithm by combining the two major concepts of SJF and SRTF (shortest job first and shortest job remaining first).

For example suppose we have three different processes

Process	**Burst time**
P1	**2**
P2	**4**
P3	**6**

Traditional Round Robin Algorithm (with Time Quantum 1)

P1	P2	P3	P1	P2	P3	P1	P3	P1	P3	P3	P3

0 1 2 3 4 5 6 7 8 9 10 11 12

P1 -> 0+2+2+1 = 5
P2-> 1+2 = 3
P3-> 2+2+1+1 = 6
Average waiting time = (5+3+6)/3 **= 4.66**

Proposed calculations (with Time Quantum Minimum burst time)

P2	P1	P1	P3

0 2 4
6 12

P1 ->2
P2-> 0
P3->6
Average waiting time = 8/3 = 2.6
Steps:

- The process having minimum burst time should be select first then second process will be selected.
- The time quantum should be equal to the shortest process.

In above example process P2 is the shortest process so we take time quantum according to the burst time of P2.

V. CONCLUSION

In this paper briefly discuses about the scheduling algorithm calling as shortest job remaining time round robin, In the past, the research of real-time scheduling mainly focuses on hard real-time and static scheduling. it combines the features of both SJRT and RR scheduling algorithms by using this algorithms we can improve the efficiency of turnaround time and waiting time of a process. The research of real-time system scheduling algorithms is turned in the direction of distributed, dynamic, soft real-time and mixed scheduling.

REFERENCES

[1]. T. Numanoglu, B. Tavli, and W. Heinzelman. An analysis of coordinated and non-coordinated medium access control protocols under channel noise. Military Communications Conference, 2005.MILCOM

[2]. K. Ramamritham and J. A. Stankovik, "Scheduling algorithms and operating support for real-time systems", Proceedings of the IEEE, vol. 82, January 1994.

[3]. C.L.Liu and L. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment", Journal of ACM, January 1973, 20(10): 46-61.

[4]. M.Dertouzos and K.Ogata, "Control robotics: The procedural control of physical process," Proc. IFIP Congress, 1974.

[5]. A. Mok, "Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment," Ph.d.thesis, MIT, Cambridge, Massachusetts, May 1983.

[6]. G.Saini, “Application of Fuzzy logic to Real-time scheduling”, Real-Time Conference, 14th IEEE-NPSS. 2005.

[7]. S. Baruah, G. Koren, B. Mishra, et al. “On the competitiveness of on-line real-time task scheduling”, In Proceedings of IEEE Real-Time Systems Symposium, December 1991, pp. 106-115.

[8]. J.W.S.Liu, “Real-Time Systems”, Pearson Education, India, 2001.

[9]. M. Dorigo and G. Caro, “The Ant Colony Optimization Metaheuristic in D.Corne, M. Dorigo and F.Glover(eds)”, New Ideas in Optimization, McGraw Hill, 1999.

[10]. V.Ramos, F.Muge, and P.Pina, “Self-organized data and image retrieval as a consequence of inter-dynamic synergistic relationships in artificial ant colonies”, In Second International Conference on Hybrid Intelligent System, IOS Press, Santiago, 2002.

[11]. C. D. Locke, “Best Effort Decision Making for Real-Time Scheduling”, Ph.d.thesis, Computer Science Department, Carnegie-Mellon University, 1986.

[12]. G.Koren and D.Shasha, “Dover: An optimal on-line scheduling algorithm for overloaded real-time systems”, SIAM Journal of Computing, April, 1995, 24(2): 318-339.