International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN : 2278-800X, www.ijerd.com
Volume 4, Issue 9 (November 2012), PP. 36-42

Efficient Round Robin CPU Scheduling Algorithm

Sukumar Babu .B[1], Neelima Priyanka .N[2], Sunil Kumar .B[3]
[1]Vijaya Institute Of Technology For Women, JNTUK, Vijayawada, India.
[2]SRK Institute Of Technology ,JNTUK, Vijayawada.
[3]Sri Sarathi Institute of Engg & Tech, Nuzvid, India.

Abstract:- One of the main fundamental function of an operating system is scheduling. There are two types of uni-processor systems in general. Those are uni-programmed and multi-programmed operating systems. In a uni-programmed operating system only one process is executed at a time, while the remaining process must wait until the CPU is free. In a multi-programmed operating system, it is capable of executing multiple processes concurrently. CPU Scheduling is the basis of multi-programmed operating system. The scheduler is responsible for multiplexing processes on the CPU. There are many scheduling algorithms available for a multi-programmed operating system like FCFS, SJF, Priority, Round Robin etc. In this paper, we mainly focused on a new algorithm called Efficient Round Robin (ERR) algorithm for multi-programmed operating system. In this paper we developed a tool which gives output in the form of experimental results with respect to some standard scheduling algorithms like FCFS, SJF, Priority, Round Robin etc.

Keywords:- Operating System, uni-processor, uni-programming, multi-programming, Resource utilization, Scheduling, FCFS, SJF, Priority, Round Robin, ERR etc.

I. INTRODUCTION

Modern Operating Systems are moving towards multitasking environments which mainly depends on the CPU scheduling. Scheduling is the heart of any computer system since it contains decision of giving resources between possible processes. Sharing of resources between multiple processes is also called as scheduling. All the resources of computer are scheduled before use; as CPU is one of the major computer resources therefore its scheduling is vital for operating system [1]. When more than one process is ready to take control of CPU, the operating system must decide which process will take control of CPU first. The component of the operating system that is responsible for making this decision is called scheduler and the algorithm used by it is called scheduling algorithm [2,3]. In computer system, all processes execute by alternating their states between two burst cycles: CPU burst cycle and I/O burst cycle. Generally, a process starts its execution with a CPU burst then performs I/O burst, again another CPU burst then another I/O burst and so on, and finally this process ends with the last CPU burst with a system request to terminate execution [1]. CPU bound process is that which performs a lot of computational tasks and do little I/O, while I/O bound process is that which performs a lot of I/O operations [1]. The typical task performed by the scheduler is to give the control of CPU to another process when one process is doing the I/O operations. The reason behind it is that I/O takes long time to complete its operation and CPU has to remain idle [4, 5]. Scheduling algorithms can be classified into pre-emptive and non-pre-emptive scheduling. The algorithm proposed in this article is pre-emptive in nature and attempts to give fair CPU execution time by focusing on average waiting time and average turnaround time of a process. This article comprises of the following sections: Section 2 gives a brief introduction to different types of schedulers and its functionality. Section 3 presents scheduling parameters, which will decide against which parameters the new CPU algorithm will be tested. Section 4 introduces existing scheduling algorithms. Section 5 explains the Efficient Round robin (ERR) algorithm. Section 6 contains pseudo code of the ERR algorithm. Section 7 explains the two basic elements that make up the simulation and provide an interactive user interface. Section 8 presents a graphical comparison of the new algorithm with existing algorithms. Section 9 will provide conclusion of the work.

II. TYPES OF SCHEDULERS

The Operating system has three different types of schedulers namely – 'long term scheduler', 'medium term scheduler' and 'short term scheduler' [6]. Ready Queue: A ready-queue is a queue data structure, consisting of two ends one is front and another is rear. The jobs are loaded into the ready queue through the front, and the jobs are deleted from the ready queue through the rear [6].

A. Long-term Scheduler :

It is also called as high level scheduler, admission scheduler or job scheduler. The long term scheduler selects the jobs from the pool of jobs and loaded these jobs into main memory ready queue for execution [1, 6].

B. Medium-term Scheduler :

It is also called mid-term scheduler. The medium-term scheduler performs the swap-in and swap-out operations between the main memory ready queue and I/O, i.e., If a process request an I/O in the middle of the execution, then the process removed from the main memory and loaded into waiting queue (swap out). When the I/O operation completed, then the job moved from waiting queue to ready queue (swap-in) [1, 6].

C. Short-term Scheduler :

It is also called CPU scheduler. The function of the short-term scheduler is, selects a job from the ready queue and gives the control of the CPU to that process with the help of dispatcher. The dispatcher performs the switching between the CPU from one process to another process. The method of selecting a process from the ready queue is based on the CPU scheduling algorithm [1, 6].

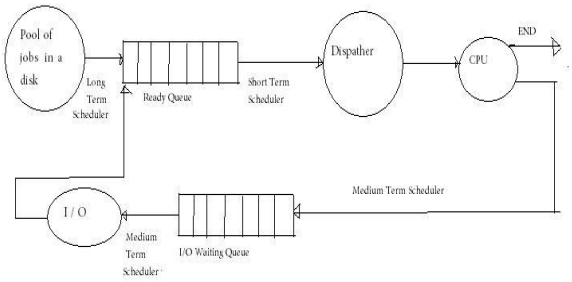


Fig: Queuing Diagram for the CPU Scheduling

III. SCHEDULING PARAMETERS

There are different scheduling algorithms with different characteristics which decides selection of process using different criteria for execution by CPU. The criteria for a good scheduling algorithm depend, among others, on the following measures [1].

A. CPU Utilization: It is the average fraction of time, during which the processor is busy [7,8].
B. 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 [7].
C. Waiting Time: The average period of time a process spends waiting. Waiting time may be expressed as turnaround time less the actual execution time [7].
D. Turnaround time: The interval from the time of submission of a process to the time of completion is the turnaround time [7].
E. Response time: Response time is the time from submission of a request until the first response is produced [7].
F. Priority: give preferential treatment to processes with higher priorities [7].
G. Fairness: Avoid the process from starvation. All the processes must be given equal opportunity to execute [7].

IV. OVERVIEW OF EXISTING CPU SCHEDULING ALGORITHMS.

A. First Come First Served (FCFS) Scheduling.

It is the simplest CPU Scheduling algorithm. The criteria of this algorithm are 'the process that requests first are holds the CPU first' or which process enter the ready queue first is served first' [6]. The workload is processed in the order of arrival time, with no pre-emption [8]. Once a process has been submitted to the CPU, it runs into completion without being interrupted. Such a technique is fair in the case of smaller processes but is quite unfair for long an unimportant job [9]. Since FCFS does not involve context switching therefore it has minimal overhead. It has low throughput since long processes can keep processor occupied for a long time making small processes suffer. As a result waiting time, turnaround time and response time can be low [10].

B. Shortest Job First (SJF) Scheduling.

The criteria of this algorithm are which process having the smallest CPU burst, CPU is assigned to that process next. If two process having the same CPU burst time FCFS is used to break up the tie [6]. SJF can be worked as pre-emptive and non–pre-emptive in nature based on the arrival time and burst time of the processes. SJF reduces average waiting time of the processes as compared to FCFS. SJF favors shorter processes over longer ones which is an overhead as compared to FCFS. It selects the job with the smallest burst time ensuing CPU availability for other processes as soon as the current process reaches its completion. This prevents smaller processes from suffering behind larger processes in the ready queue for a longer time [9][11].

C. Priority Based Scheduling.

In this algorithm, priority is associated with each process and on the basis of that priority CPU is allocated to the processes. Higher priority processes are executed first and lower priority processes are executed at the end. If multiple processes having the same priorities are ready to execute, control of CPU is assigned to these processes on the basis of FCFS [1, 4]. Priority Scheduling can be preemptive and non-preemptive in nature.

D. Round Robin (RR) Scheduling.

It is a preemptive scheduling algorithm. It is designed especially for time sharing systems. In this algorithm, a small unit of time called time quantum or time slice is assigned to each process [6]. When the time quantum expired, the CPU is switched to another process. Performance of Round Robin totally depends on the size of the time quantum.

V. PROPOSED WORK: EFFICIENT ROUND ROBIN SCHEDULING.

The proposed algorithm ERR is pre-emptive in nature. In this algorithm all the processes are sorted in ascending order in the ready queue. ERR scheduling algorithm maintains a small unit of time called time quantum or time slice is assigned to each process. According to that time quantum processes are executed and if time quantum of any process expires before its complete execution, it is put at the end of the ready queue and control of the CPU is assigned to the next shortest incoming process. In this algorithm ready queue is a circular queue. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue sets of timer to interrupt after assigned time quantum, and dispatches the process. The average waiting time and average turnaround time obtained from ERR is better than existing CPU scheduling algorithms. ERR is fair in scheduling and effective in time sharing environment. In ERR scheduling, CPU is given to each process for equal time period, no process has to wait for long time for the CPU. Working Procedure of the Proposed Algorithm ERR is discussed below:

- First collect all the list of processes, their burst time, arrival time and time quantum.
- Arrange processes and their burst time in ascending order based on their arrival time.
- Repeat 1 and 2 until all process complete their execution

1. If the current time is equal to arrival time of the ready queue process And if burst time of current process is greater than time slice then execute for time slice period else execute for burst time period.
2. If current time equals to arrival time and burst time equals to zero, then remove the process from the ready queue.
   - During above process calculate the waiting time and turn around time of each process.

VI. PSEUDO CODE

```
 bursttime[n] ← 0
 arrivaltime[n] ← 0
numprocess[n] ← 0
turn[n] ← 0
wait[n] ← 0
temp ← 0
currenttime ← 0
tslice ← 0
waittime=0
turntime=0
avgwaititme=0.0
avgturntime=0.0

Read bursttime[n] and arrivaltime[n]

For i← 0 to n-1
  For j ← I to n
  If arrivaltime[i] > arrivaltime[j]
     temp ← bursttime[i]
    bursttime[i] ← bursttime[j]
    bursttime[j] ← temp
   temp ← arrivaltime[i]
   arrivaltime[i] ← arrivaltime[j]
   arrivaltime[j] ← temp
end for

For i← 0 to n-1
  For j ← I to n
  If( arrivaltime == currentime && bursttime[i] >bursttime[j])
   temp ← bursttime[i]
  bursttime[i] ← bursttime[j]
    bursttime[j] ← temp
   temp ← arrivaltime[i]
   arrivaltime[i] ← arrivaltime[j]
   arrivaltime[j] ← temp
end for j

if bursttime[i] != 0
   if currenttime == arrivaltime[i]
     if tslice < bursttime[i]
        arrivaltime[i]←arrivaltime[i]+tslice[i];
       bursttime[i]←bursttime[i]-tslice[i]
       currenttime=currenttime+tslice
  else
 arrivaltime[i]←arrivaltime[i]+bursttime[i];
```

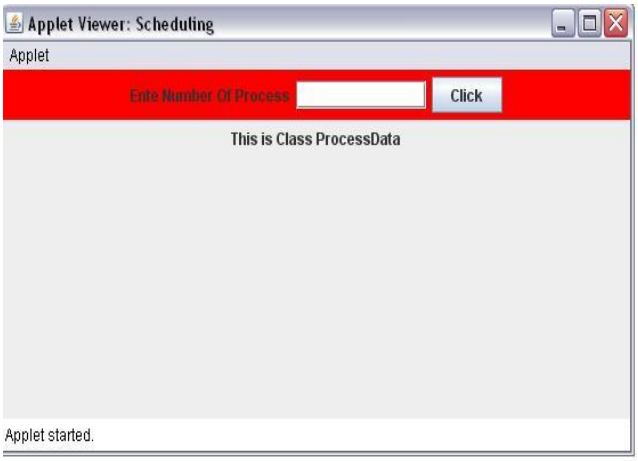
```
 currenttime=currenttime+ bursttime[i]
bursttime[i]←bursttime[i]- bursttime[i]
end else
if bursttime[i] == 0
   turn[i] = currenttime
end if
end if
else
currenttime++
end else

for i← 0 to n
wait[i]← turn[i]-burst[i]-arrivaltime[i]
turn[i]← turn[i]-arrivaltime[i]
waittime=waittime+wait[i]
turnrime=turntime+turn[i]
end of for

avgwaittime=waittime/n
avgturntime = turntime/n
```

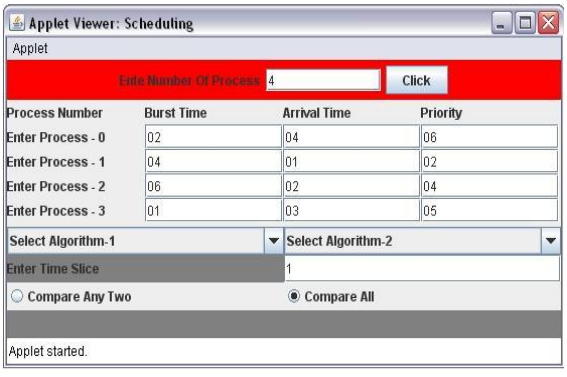
VII. SIMULATION DESIGN

The simulation provides an interactive GUI interface to the user through which a user can input data related to different processes then applying different algorithms based on the algorithm choices given in the simulator. The simulator employs JAVA swings. The front end user interfaces are developed using java awt's and swings. The parent screen allows the user to give number of processes to be in execution.



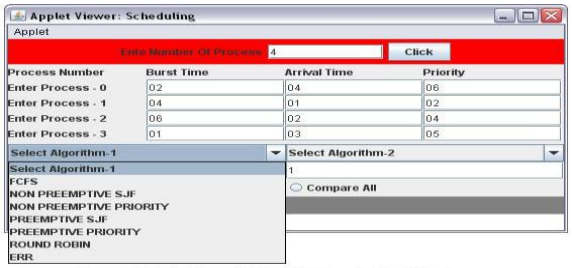
Screen1 : Enter Number of Process

Screen2 allows the use give the details of the processes like Burst time, Arrival Time, Priority and Time slice.



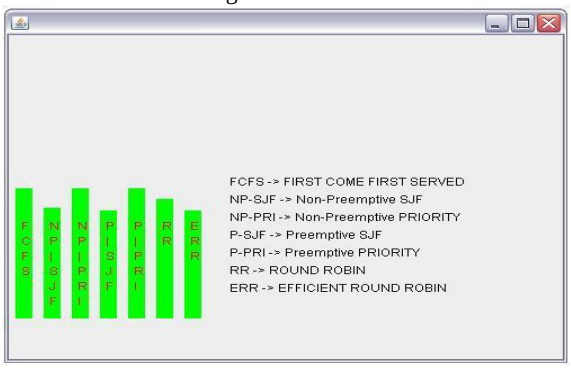
Screen 2: Enter Burst , Arrival and Priority of processes

Screen 3 allows the user to compare any two algorithms of his choice or allows the user to compare all algorithms.



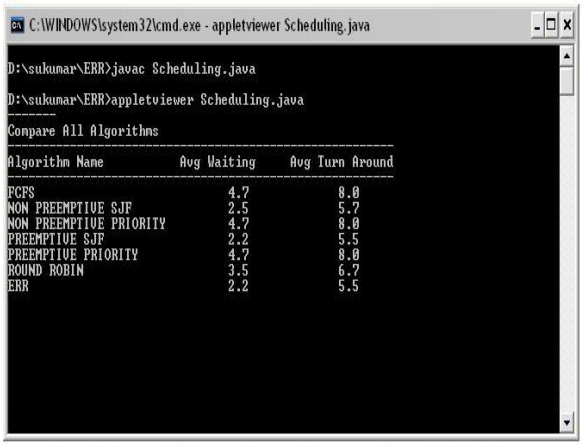
Screen 3: Selection of Algorithm1 and Algorithm2

Screen 4 shows the comparison results of the selected algorithms.



Screen 4: Comparing All Algorithms

Screen 5 gives the average waiting time and average turnaround time of selected algorithms.

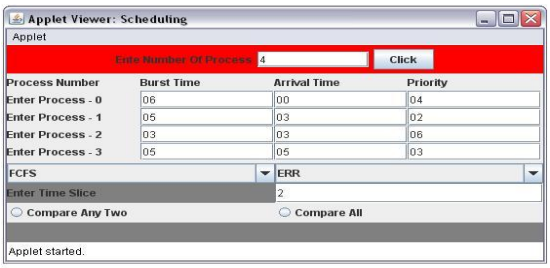


Screen 5: Average Waiting and Average Turnaround time of all Algorithms

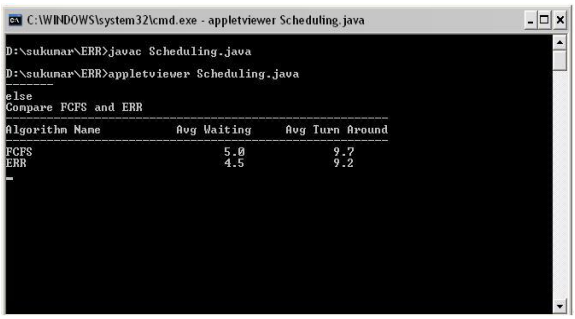
VIII. COMPARISON OF ERR WITH OTHER CPU SCHEDULING ALGORITHMS.

To compare the performance of ERR, it was implemented with some other existing CPU scheduling algorithms. By taking list of processes, processes are scheduled using different CPU scheduling algorithms. Average waiting time and average turnaround time was noted for each. For example the processes are having the following burst time, arrival time and priority.

Process	Burst Time	Arrival Time	Priority
P1	6	0	4
P2	5	3	2
P3	3	3	6
P4	5	5	3

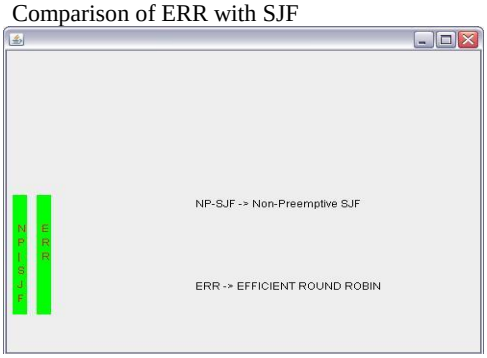


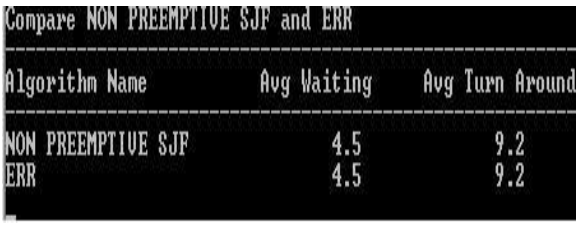
A. Comparison of ERR with FCFS



From the above figures it is observed that ERR scheduling algorithm is better than the FCFS scheduling algorithm.

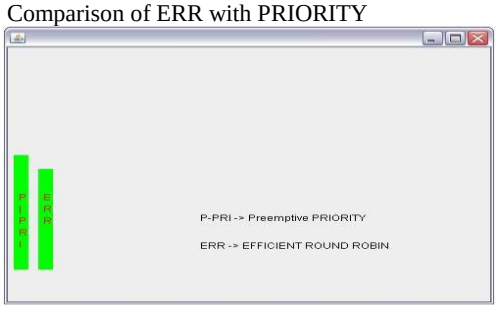
B. Comparison of ERR with SJF

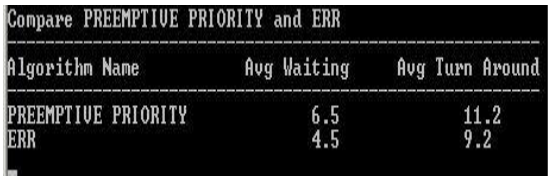




From the above figures it is observed that ERR scheduling algorithm and the SJF scheduling algorithm are equal. But when time slice is increasing then ERR algorithm gives much better average waiting time and average turnaround time than the SJF algorithm.

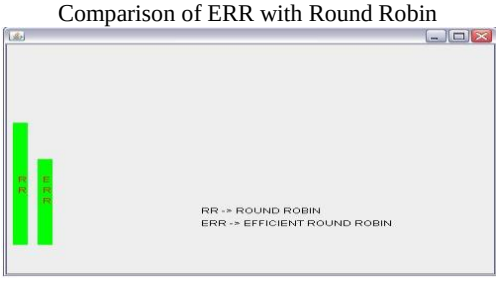
C. Comparison of ERR with PRIORITY





From the above figures it is observed that ERR scheduling algorithm is better than the PRIORITY scheduling algorithm.

D. Comparison of ERR with Round Robin

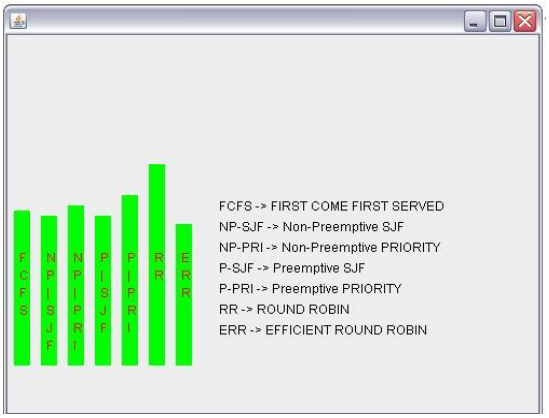



```
Compare ROUND ROBIN and ERR
-------------------------------------------------------
Algorithm Name          Avg Waiting     Avg Turn Around
-------------------------------------------------------
ROUND ROBIN                 9.5             14.2
ERR                         3.7             8.5
```

From the above figures it is observed that ERR scheduling algorithm is better than the ROUND ROBIN scheduling algorithm.

E. Comparison ERR with all Algorithms

From the below figures it is observed that ERR Scheduling algorithm works much better than the existing CPU Scheduling algorithms




```
C:\WINDOWS\system32\CMD.exe - appletviewer Scheduling.java

D:\sukumar\ERR>javac Scheduling.java

D:\sukumar\ERR>appletviewer Scheduling.java
--------
Compare All Algorithms
-------------------------------------------------------
Algorithm Name          Avg Waiting     Avg Turn Around
-------------------------------------------------------
FCFS                        5.0             9.7
NON PREEMPTIVE SJF          4.5             9.2
NON PREEMPTIVE PRIORITY     5.5             10.2
PREEMPTIVE SJF              4.5             9.2
PREEMPTIVE PRIORITY         6.5             11.2
ROUND ROBIN                 9.5             14.2
ERR                         3.7             8.5
```

IX. CONCLUSION

The paper presents a new CPU scheduling algorithm called ERR CPU Scheduling Algorithm. Paper also contains simulation interface and its working, which interactively takes input from the user and compares the process set against different algorithm pairs. The result of the simulation for different process sets using different scheduling algorithms has been presented graphically in this piece of work. The last half of the paper provides analytical result with each set of graph. From the above graphs and results, it is clear that ERR is more efficient than FCFS, SJF Pre-emptive and SJF Non Pre-emptive, Pre-emptive Priority and Non Pre-emptive Priority, Round Robin.

X. REFERENCES

[1]. Abraham Silberschatz , Peter Baer Galvin, Greg Gagne, "Operating System Concepts", Sixth Edition.
[2]. Andrew S. Tanenbaum, Albert S. Woodhill, "Operating Systems Design and Implementation", Second Edition.
[3]. Lingyun Yang Jennifer M. Schopf and Ian Foster, "Conservative Scheduling : Using predictive variance to improve scheduling decisions in Dynamic Environments", Super Computing 2003, November 15-21, Pheonix, AZ, USA.
[4]. Mohammed A.F. Al-Husainy, 2007. Best-Job-First CPU Scheduling Algorithm. Information Technology Journal, 6: 288-293.
[5]. Sindhu M , Rajkamal R, Vigneshwaran P, "An Optimum Multilevel CPU Scheduling Algorithm", ACE, pp.90-94, 2010 IEEE International Conference on Advances in Computer Engineering, 2010.
[6]. P. Balakrishna Prasad "Operating Systems" second edition.
[7]. I. A. Dhotre "Operating Systems"
[8]. Milan Milenkovic, "Operating systems Concepts and Design", McGRAW-HILL, Computer Science Series, second edition.
[9]. H. M. Dietel, "Operating Systems", Pearson Education, Second Edition.
[10]. http://en.wikipedia.org/wiki/Scheduling
[11]. M Gary Nutt, "Operating Systems –A Modern Perspective:, Second Edition, Pearson Education, 2000.