Improvised Optimum Multilevel Dynamic Round Robin Algorithm for Optimizing CPU Scheduling

Neetu Goel
Research Scholar,
Teerthanker Mahaveer University, U.P

R.B. Garg
Ex- Professor
Delhi University

ABSTRACT

CPU scheduling has strong effect on resource utilization as well as overall performance of the system. In order to simulate the behavior of multiple jobs in a multiprogramming computer system needs to be specified. The most important aspect of job scheduling is the ability to create a multi-tasking environment. The intention should be allowed as many as possible running processes at all time in order to make best use of CPU. Round Robin algorithm performs optimally in timeshared systems, but it is not suitable for soft real time systems, because it gives more number of context switches, larger waiting time and larger response time. The main objective of this paper is to improve the previous OMDRRS with calculates intelligent time slice and warps after every round of execution and assumed that all the processes were come at randomly as well as all the processes have priority. In order to simulate the behavior of various CPU scheduling algorithms and to improve Round Robin scheduling algorithm using dynamic time slice concept, we purpose new improved CPU scheduling algorithm called "Optimum Dynamic Round Robin Scheduling" (OMDRR). Our experimental results show that our proposed algorithm performs better in terms of reducing the number of context switch, average waiting time and average turnaround time.

Keywords

CPU Scheduling, Round Robin Scheduling, OMDRR, Context Switch, Turn Around Time, Waiting Time

1. INTRODUCTION

CPU scheduling is somewhat similar to other types of scheduling which have been studied over the years. CPU scheduling refers to the decision of allocating a single resource among multiple clients, the order of allocation and its duration. The primary objective of scheduling is to optimize system performance in accordance with the criteria deemed most important by the system designers [1]. There are a number of such algorithms with each having its respective advantages and drawbacks. In order to determine the comparative and competitive advantages and disadvantages of these algorithms, they need to be simulated and their performance indices studied and used for better understanding of operating system principles. The data that drives the simulation can be generated in several ways. The most common method uses a random number generator, which is programmed to generate processes; CPU burst times, arrivals, departures, and so on, by using probability distributions. The scheduling simulator illustrates the behavior of scheduling algorithms in opposition to a simulated mix of process loads. In Round Robin (RR) every process has its equal priority and is given a time quantum or time slice after which the process is preempted. Although RR gives improved response time and uses shared resources efficiently its limitations are larger waiting time, undesirable overhead and larger turnaround time for processes with variable CPU bursts due to the use of static time quantum that motivates, to implement RR algorithm with dynamic burst time concept.

To properly illustrate the functionality of various CPU scheduling algorithm as well as the improvement of RR scheduling called "An Optimum Multilevel Dynamic Round Robin Scheduling Algorithm" (OMDRRS) and the effect of each algorithm which it has on the execution of processes was written using VB6.0 and the results of all algorithms were collected and compared for the Turnaround time, Waiting time, Context Switch & Gantt Chart. This paper is divided into four sections. Section I gives a brief introduction on the various aspects of the scheduling algorithms, the approach to the current paper and the motivational factors leading to this improvement. Section II presents an overview of some of the simulators that are available and their respective drawbacks. Section III presents the proposed algorithm and illustration of our proposed new algorithm (OMDRRS). In Section IV, an experimental analysis and Result of our algorithm (OMDRRS) and its comparison with the static RR algorithm. Conclusion is presented in Section V followed up by the references used.

2. RELATED WORK

OMDRRS[2] is visual basic simulator which calculates intelligent time slice and warps after every round of execution and assumed that all the processes were come at same arrival time as well as all the processes have no priority. Process Scheduling Simulator [3] is a java-based web application that implements FCFS, SJF, Priority SJF and Round Robin. It requires a high-speed internet connection to load the applet, and also requires that Java software be either installed or updated. Each input in the system is characterized by its arrival time, CPU burst and I/O bursts. It claims to be very efficient but a sample run disclosed that it is very slow. Another simulator "CPU Scheduling Simulator (CPUSS)" [4], CPUSS is a framework that allows users to quickly and easily design and gather metrics for custom CPU scheduling strategies including FCFS, Round Robin, SJF, Priority First, and SJF with Priority Elevation rule. The long list of the capabilities it can handle makes it too complex and complicated for simple academic demonstrations and use by non-computer geeks such as fresh students that are just taking their first course in Computer Science. Above all, it runs in the windows-DOS environment which is characterized by unattractive user interface and hence, lacks user-friendliness. A project that is very close to our work is a simulator presented by [5]. However, this simulator was designed for a software project scheduling rather than CPU process scheduling, hence not relevant for our consideration in this study. MOSS, Modern Operating Systems Simulators, was found in [6]. It is a collection of Java-based simulation programs which illustrate key operating system concepts presented in a textbook by Tanenbaum (2001) for university students using the text. This does not fit in to independent software that can be used freely without any such constraint. The best simulator we could find, so far, during our survey of previous related work was presented by [9]. It shows the implementation of a lightweight, simple, robust and flexible tool for the comparative and experimental study of two existing as well as an innovative probabilistic CPU process scheduling algorithm, using Average Waiting Time (AWT) and Average Turn-around Time (ATT) as the criteria for performance evaluation. The tool was used to simulate the three algorithms using eight datasets representing different scenarios of processes with their burst times and respective locations on a virtual queue. The limitation of the proposed software is that it is meant for non- preemptive processes, hence not suitable for real-time applications. However, it is not as robust as ours in the sense that we implemented a Dynamic Round Robin algorithm in addition to FCFS, SJF and ROUND ROBIN algorithms. Our major objective is to simulate the behavior of various CPU scheduling algorithms and to improve Round Robin scheduling algorithm using dynamic time slice concept, called improvised OMDRRS, which calculates intelligent time slice and changes after every round of execution. So, we introduce the new dynamic scheduling algorithm while using its excellent comparison with FCFS, SJF, Priority and Round Robin as a proof of the new algorithm's efficiency of Turnaround Time, Waiting Time, Context Switch and desirability for academic demonstrations and possible implementation in real-life systems.

3. PROPOSED ALOGRITHM: OMDRRS

OMDRRS scheduling algorithm simulator is developed using the Microsoft Visual Basic 6.0 Professional Edition's Integrated Developed Environment (IDE) under windows operating system which can be used for study and for evaluation of CPU scheduling algorithms in real time operating systems. Based on the Processes data input and selected scheduling algorithm FCFS, SJF, Round Robin, Preemptive SJF, Priority Scheduling and the Dynamic algorithms were computed which display the average turnaround time, average waiting time, context switch and Gantt chart were automatically generated & displayed at runtime. The user interfaces are simple, concise, unambiguous and easy to use but complete only with the relevant information. The inputs of burst time, Arrival Time & Priority are re- useable for comparing with all other algorithms as well as simulator has the facility to add new process at run time. The innovative Dynamic algorithm is well implemented and its mode of operation was clearly shown and presented in the simulator.

3.1 Our Revised Proposed Algorithm

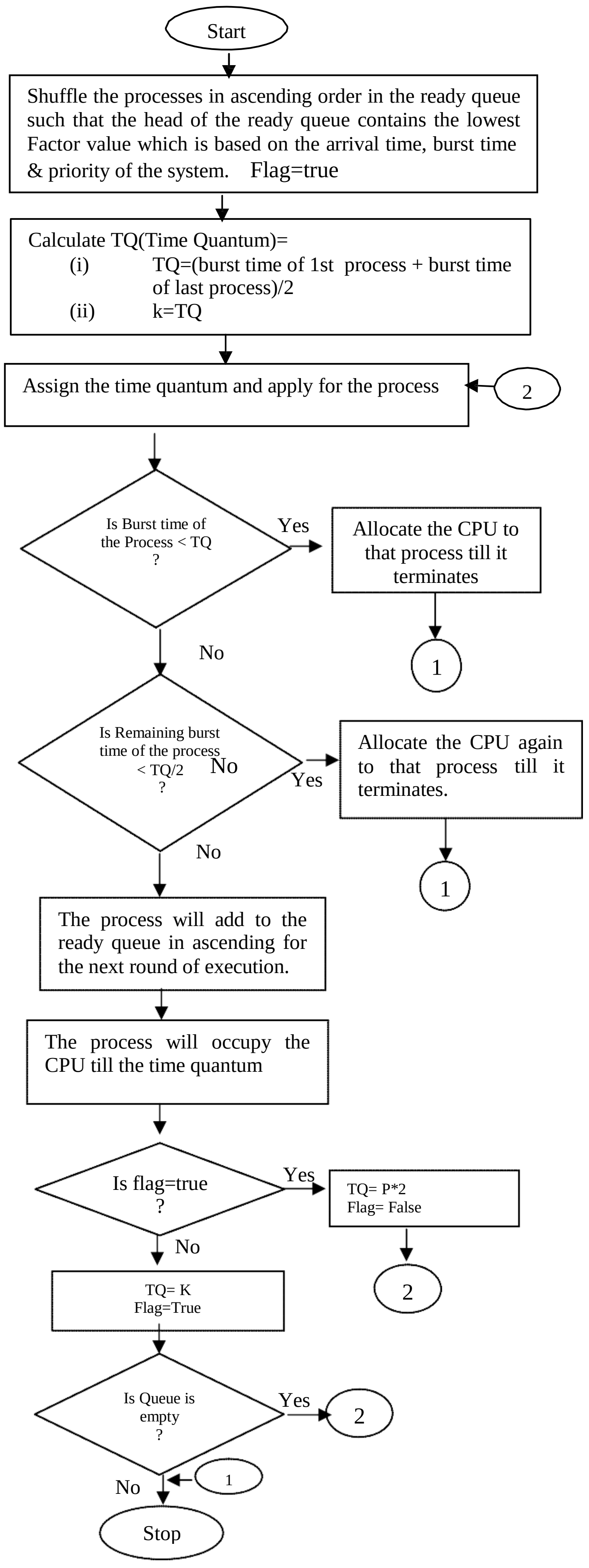
Algorithm

When Arrival Time, Priority and burst time are given

In our algorithm, combines the working principle of fundamental scheduling algorithms. Dynamically Time Slice (DTS) is calculated which allocates different time quantum to each process based on priority, shortest CPU burst time and context switch avoidance time.

```
Step 1: Compute the factor analysis F= Burst time * 0.2
+ Arrival time * 0.3 + Priority of the process * 0.5
Step 2: Shuffle the processes in ascending order according to the factor of each process in the ready queue (RQ) such that the head of the ready queue contains the lowest factor process based on the burst time, arrival time & priority of the process.
Step 3:
        (i) low= RQ(burst value of the first process),
            high=RQ(burst value of the last process)
        (ii) TQ=(low + high) / 2
Step 4: Assign the time quantum and apply for each process say k=TQ.
Step 5: IF (burst time of the process < k)
{
    Allocate the CPU to that process till it terminates.
}
ELSE IF (Remaining burst time of the process <k/2)
{
    Allocate the CPU again to that process till it terminates.
}
ELSE
{
        (i) The process will occupy the CPU till the time quantum and it is added to the ready queue in ascending order according to the remaining burst time for the next round of execution.
        (ii) TQ= TQ *2 or TQ=TQ/2
        (iii) K=TQ
        (iv) Goto Step 4
}
```

3.2 Logic Diagram: When Arrival Time, Priority and burst time are given


Start
Shuffle the processes in ascending order in the ready queue such that the head of the ready queue contains the lowest Factor value which is based on the arrival time, burst time & priority of the system. Flag=true
Calculate TQ(Time Quantum)=
(i) TQ=(burst time of 1st process + burst time of last process)/2
(ii) k=TQ
Assign the time quantum and apply for the process
2
Is Burst time of the Process < TQ ?
Yes
Allocate the CPU to that process till it terminates
1
No
Is Remaining burst time of the process < TQ/2 ?
No
Yes
Allocate the CPU again to that process till it terminates.
1
No
The process will add to the ready queue in ascending for the next round of execution.
The process will occupy the CPU till the time quantum
Is flag=true ?
Yes
TQ= P*2
Flag= False
2
No
TQ= K
Flag=True
Is Queue is empty ?
Yes
2
1
No
Stop

3.3 Simulation of OMDRRS Algorithm

In designing the simulator, it is important for each of the processes to be as similar as possible, and include only those variations which were specific to the algorithm being implemented. The software was implemented to simulate the operations of FCFS, SJF(Non Preemptive & Preemptive), Highest Priority, Round Robin and Improving of Round Robin scheduling algorithm. These algorithms were implemented in order to establish a valid premise for effective comparison. The simulation was run several times to ensure fairness to all datasets and presented for each algorithm using Average Turn-around Time, Average Waiting Time, Context Switch and Gantt chart as the performance evaluation indices.

3.4 Comparison of OMDRSS Algorithm with existing Algorithms

Table 1 shows the datasets started with a set of predefined burst Time, Arrival Time & priority for each process in the simulation. Graph1 depicts result of six scheduling algorithms based on the Turnaround time, Waiting time & context Switch. The number of processes can be extended to any length as desired. For demonstration purpose, a maximum of 10 jobs were implemented and reported in this research. However, the maximum attainable number of jobs was not determined because it totally depends on the Memory size.

Table 1: Processes with its Id, Burst Time, Arrival Time and Priority

Process ID	Burst Time(ms)	Arrival Time(ms)	Priority
P0	23	0	3
P1	34	5	1
P2	34	3	3
P3	12	6	4
P4	8	8	2
P5	10	4	5
P6	31	1	1
P7	23	2	4
P8	9	3	5
P9	16	6	1

3.5 Results and Discussions

Fig. 1 to 6 shows the main output screen of the simulator of all the algorithms to perform FCFS, SJF(NP), SJF, Piroity, Round Robin & OMDRRS automatically once we enter the burst time, arrival time and priority of the processes based on the Table1. The user can select by clicking the required Algorithm button and automatically ATA, AWT, CS, RT and gantt chart will disappear.

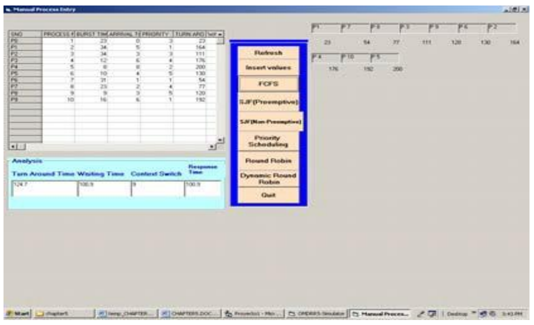
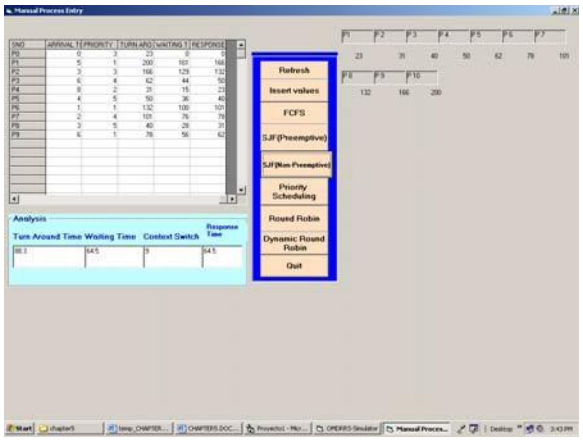


Fig 1: The Result Window for FCFS Algorithm



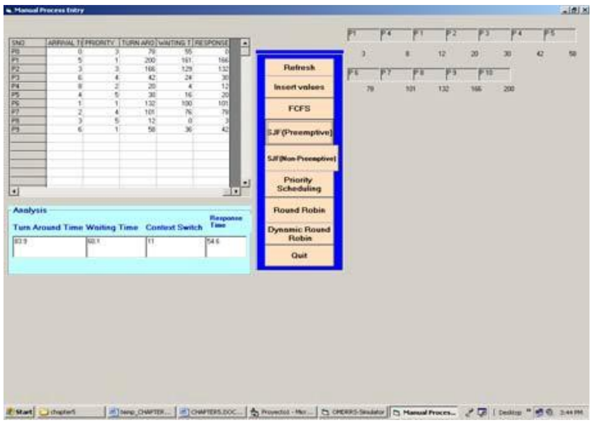
SNO	ARRIVAL T	PRIORITY	TURN ARO	WAITING T	RESPONSE
P0	0	3	23	0	0
P1	5	1	200	161	166
P2	3	3	166	129	132
P3	6	4	62	44	50
P4	8	2	31	15	23
P5	4	5	50	36	40
P6	1	1	132	100	101
P7	2	4	101	76	78
P8	3	5	40	28	31
P9	6	1	78	56	62

Analysis

Turn Around Time	Waiting Time	Context Switch	Response Time
88.3	64.5	9	64.5

```
P1   P2   P3   P4   P5   P6   P7
23   31   40   50   62   78   101
P8   P9   P10
132  166  200
```

Fig 2: The Result Window for SJF(NP) Algorithm



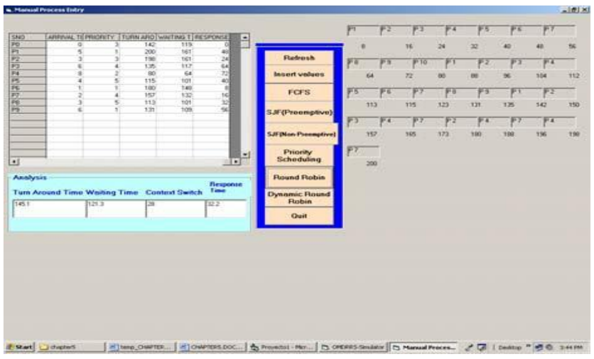
SNO	ARRIVAL T	PRIORITY	TURN ARO	WAITING T	RESPONSE
P0	0	3	79	55	0
P1	5	1	200	161	166
P2	3	3	166	129	132
P3	6	4	42	24	30
P4	8	2	20	4	12
P5	4	5	30	16	20
P6	1	1	132	100	101
P7	2	4	101	76	79
P8	3	5	12	0	3
P9	6	1	58	36	42

Analysis

Turn Around Time	Waiting Time	Context Switch	Response Time
83.9	60.1	11	54.6

```
P1   P4   P1   P2   P3   P4   P5
3    8    12   20   30   42   58
P6   P7   P8   P9   P10
79   101  132  166  200
```

Fig 3: The Result Window for SJF(P) Algorithm



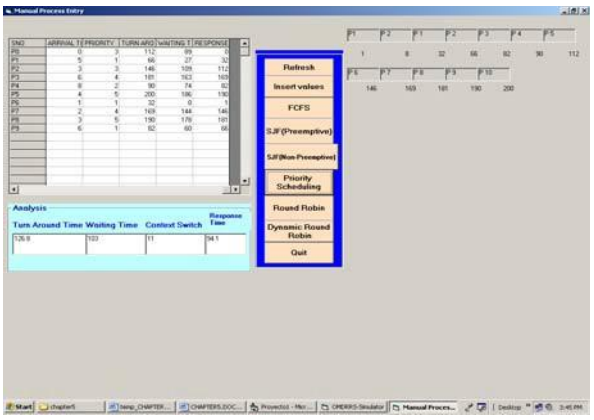
SNO	ARRIVAL T	PRIORITY	TURN ARO	WAITING T	RESPONSE
P0	0	3	142	119	0
P1	5	1	200	161	48
P2	3	3	198	161	24
P3	6	4	135	117	64
P4	8	2	80	64	72
P5	4	5	115	101	40
P6	1	1	180	140	8
P7	2	4	157	132	16
P8	3	5	113	101	32
P9	6	1	131	109	56

Analysis

Turn Around Time	Waiting Time	Context Switch	Response Time
145.1	121.3	28	32.2

```
P1   P2   P3   P4   P5   P6   P7
8    16   24   32   40   48   56
P8   P9   P10  P1   P2   P3   P4
64   72   80   88   96   104  112
P5   P6   P7   P8   P9   P1   P2
113  115  123  131  135  142  150
P3   P4   P7   P2   P4   P7   P4
157  165  173  180  188  196  198
P7
200
```

Fig 4:The Result Window for Round Robin Algorithm(TQ=8)



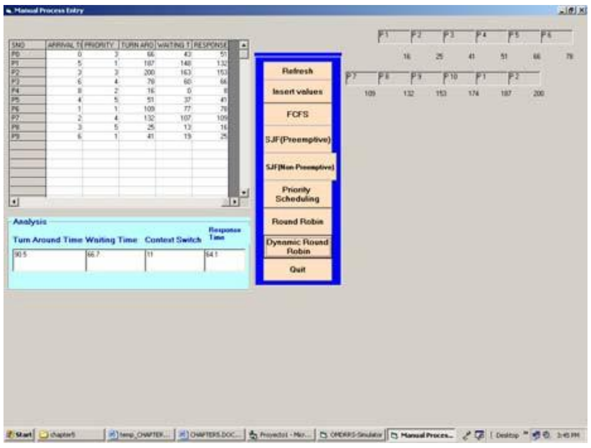
SNO	ARRIVAL T	PRIORITY	TURN ARO	WAITING T	RESPONSE
P0	0	3	112	89	0
P1	5	1	66	27	32
P2	3	3	146	109	112
P3	6	4	181	163	169
P4	8	2	90	74	82
P5	4	5	200	186	190
P6	1	1	32	0	1
P7	2	4	169	144	146
P8	3	5	190	178	181
P9	6	1	82	60	66

Analysis

Turn Around Time	Waiting Time	Context Switch	Response Time
126.9	103	11	94.1

```
P1   P2   P1   P2   P3   P4   P5
1    8    32   66   82   90   112
P6   P7   P8   P9   P10
146  169  181  190  200
```

Fig 5: The Result Window for Priority Scheduling Algorithm(TQ=8)



SNO	ARRIVAL T	PRIORITY	TURN ARO	WAITING T	RESPONSE
P0	0	3	66	43	51
P1	5	1	187	148	132
P2	3	3	200	163	153
P3	6	4	79	60	66
P4	8	2	16	0	8
P5	4	5	51	37	41
P6	1	1	109	77	78
P7	2	4	132	107	109
P8	3	5	25	13	16
P9	6	1	41	19	25

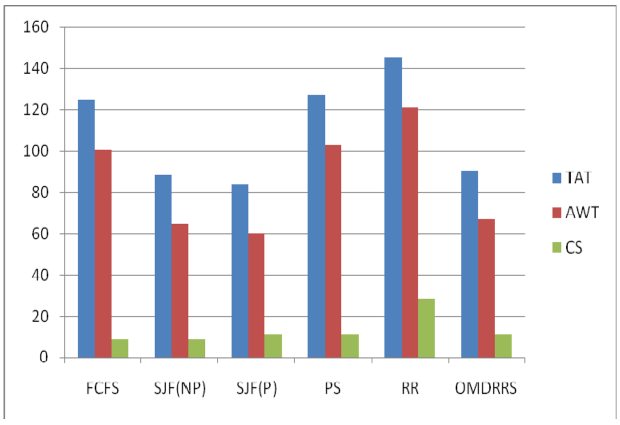
Analysis

Turn Around Time	Waiting Time	Context Switch	Response Time
90.5	66.7	11	64.1

```
P1   P2   P3   P4   P5   P6
16   25   41   51   66   79
P7   P8   P9   P10  P1   P2
109  132  153  174  187  200
```

Fig 6:The Result Window for Dynamic Round Robin Algorithm

Graph 1 shows the bar graph of comparison between of FCFS, SJF(NP), SJf(P), PS, RR and OMDRRS based on the result generated by the designed simulator. We plot the bar diagram of processes using Turnaround Time, Waiting Time and Context Switch criteria



Graph1: Comparison of CPU Scheduling Algorithms

We can see from the above experiment context switch, average waiting time and average turnaround time both are reduced by using our proposed algorithm. The reduction of context switch, average waiting time and average turnaround time shows maximum CPU utilization and minimum response time. We observed that proposed algorithm much more efficient as compared to simple RR algorithm.

4. CONCLUSION AND FUTURE WORK

A simulation has been described and used for comparative analysis of various job scheduling methods with regards to CPU efficiency, job turnaround time, context switch and the job waiting time. Validation results showed that the simulation runs were accurate. Round Robin has the highest average waiting time and lowest CPU efficiency; therefore this scheduling method also performs weakly with job throughput. The waiting time is reduced as the time-slice size decreases in the RR scheduling method, since the probability of an arrival in any single time slice is very small. It is concluded that the proposed algorithm is superior as it has less waiting response time, usually less pre-emption and context switching thereby reducing the overhead and saving of memory space. Future work can be based on this simulator can be test on various operating systems like Linux, Solaris, open BSD.

5. REFERENCES

[1] Sukanya Suranauwarat, “A CPU Scheduling Algorithm Simulator”, October 10-13, 2007, Milwaukee, WI 37th ASEE/IEEE Frontiers in Education Conference.

[2] Goel N., Garg R. B., “A Simulation of an Optimum Multilevel Dynamic Round Robin Scheduling Algorithm”, *International Journal of Computer Applications , ISSN 0975 – 8887*, Volume 76– No.7, August 2013

[3] http://vip.cs.utsa.edu/classes/cs3733s2004/notes/ps/runps.html

[4] http://www.codeplex.com/cpuss

[5] F. Padberg, “A Software Process Scheduling Simulator”, in Proc. of the 25th *IEEE International Conference on Software Engineering (ICSE’03)*, 0270-5257/03,2003

[6] http://www.ontko.com/moss

[7] D.A. Cardella, “A Simulator of Operating system Job Scheduling”, Visual Basic 6. Available for download http://www.freevbcode.com/ShowCode.asp?ID=407 9,2002

[8] S. H. Nazleeni, H. M. A. Anang, M. H. Hasan, A. A.Izzatdin, and M. W. Wirdhayu, ”Time comparative simulator for distributed process scheduling algorithms,” World Academy of Science, Engineering and Technology 19, pp.84-89,2006

[9] Anifowose F. A., “MySIM: A Light-Weight Tool for the simulation of Probabilistic CPU Process Scheduling Algorithm”, International Journal of Computer and Electrical Engineering, Vol. 4, No. 1, February 2012

[10] Umar Saleem and Muhammad Younus Javed, “Simulation of CPU Scheduling Alogrithm”, 0- 7803-6355-8/00/$10.00@2000 IEEE

[11] Sun Huajin’, Gao Deyuan, Zhang Shengbing, Wang Danghui; “ Design fast Round Robin Scheduler in FPGA”, 0-7803-7547-5/021/$17.00@2002 IEEE

[12] Md. Mamunur Rashid and Md. Nasim Adhtar, “ A New Multilevel CPU Scheduling Algorithm”, Journals of Applied Sciences 6 (9): 2036-2039,2009

[13] Goel N., Garg R. B., “A Comparative Study of CPU Scheduling Algorithms”, International Journal of Graphics Image Processing (IJGIP), Vol 2 issue 4, Dec-2012

[14] Silberschatz, A. P.B. Galvin and G. Gagne (2012), Operating System Concepts, 8th edition, Wiley India.

[15] Sabrian, F., C.D. Nguyen, S. Jha, D. Platt and F. Safaei, (2005), “Processing resource scheduling in programmable networks”, Computer communication, 28:676-687 Seltzer, M P. Chen and J outerhout, 1990, “Disk scheduling revisited in USENIX”, Winter technical conference.