A Performance Evaluation of the New Improved Round Robin (NIRR) CPU Scheduling Algorithm Using Normal and Exponential Statistical Distributions

Abdulrazaq Abdulrahim[1,*], Saleh E. Abdullahi[+], Sahalu B. Junaidu[*], Mohammed Y. Tanko[*]
[*]Department of Computer Science, Ahmadu Bello University, Zaria, Nigeria
[+]Department of Computer Science, Nile University of Nigeria
[1]aabdulrahim@abu.edu.ng

Abstract—*The objective of this paper is to evaluate the performance of New Improved Round Robin (NIRR) against First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), Improved Round Robin (IRR) and Longest Job First with Combinational Burst Time (LJF+CBT) scheduling algorithms by using normal and exponential statistical distributions to generate the burst time of the processes to be used. These algorithms were evaluated and benchmarked based on Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Number of Context Switches (NCS). NIRR compared with other RR Scheduling algorithms, produces minimal AWT, ATAT and NCS in both statistical distributions. Based on these results, it should be preferred over other scheduling algorithms for systems that adopt RR CPU scheduling.*

Keywords— ***Round Robin, New Improved Round Robin, Improved Round Robin, Longest Job First with Combinational Burst Time, Normal statistical distribution, Exponential statistical distribution***

I. INTRODUCTION

The time sharing operating system is an operating system in which available CPU is divided into equal slots. These slots are assigned to all the users connected to the system, any user can use the system only for the specified time slot, if the user finishes its work within the specified time slot, that's fine but if the user still has some work pending, then the user has to wait for its turn to complete the remaining work. In the simplest possible system, it can be achieved in a polled loop with a Round-Robin scheme, but it is difficult to ensure fairness if no form of time-slice allocation is created.

The Round Robin (RR) CPU scheduling algorithm is an algorithm that is specially designed for real time systems and time sharing systems [3][4][5][10][12]. It is a fair scheduling algorithm because it gives equal time to all processes [2]. It is both simple and easy to implement, and starvation-free [11]. It gives each process a small unit of CPU time (time quantum), that allows the first process in the ready queue to run until its time quantum expires, and then run the next process in the ready queue. In a situation where the process needs more time, the process runs for the full length of the time quantum and then it is preempted and then added to the tail of the queue. The main problem of RR CPU scheduling algorithm is that; its performance is sensitive to time quantum selection, because if time quantum is very large then it will be the same as the FCFS scheduling. If the time quantum is extremely too small then it will be the same as processor sharing algorithm and number of context switches will be very high. Each value of time quantum will lead to a specific performance and will affect the algorithm's efficiency by affecting the processes waiting time, turnaround time, response time and number of context switches [2][10][11][12].

The content of this paper is organized as follows: other CPU scheduling algorithms will be discussed in section II. The scheduling criteria of CPU scheduling algorithms will be discussed in section III. Section IV will be discussing on the literature review while sections V and VI will be discussing on Normal and Exponential statistical distributions respectively. Section VII will focus on the simulation of the algorithms under study and finally, the conclusion of the findings will be summarized in section VIII.

II. OTHER CPU SCHEDULING ALGORITHMS

The other basic CPU scheduling algorithms are First Come First Serve (FCFS), Shortest Job First (SJF) and Priority Scheduling (PS).

A. FCFS

It is the simplest form of CPU scheduling algorithms, which allocates CPU to the processes on the basis of their arrival to the ready queue. Arriving processes are inserted into the tail (rear) of the ready queue and the process to be executed next is removed from the head (front) of the ready queue. A long CPU bound process may dominate the CPU and may force shorter CPU bound processes to wait prolonged periods and also it has minimal average CPU utilization or average throughput [9].

B. SJF

The scheduler arranges processes according to shortest burst times in the ready queue, so that the process with least burst time is scheduled first. If two processes have equal burst times, then FCFS procedure is followed. It is provably optimal, in that it gives the minimum average waiting time and minimum average turnaround time for a given set of processes [9]. Long running processes may wait for prolonged periods, because the CPU has a steady supply of short processes [13][14]. It has also been proven to be the fastest scheduling algorithm, but it suffers from one important problem: How does the scheduler know how long the next CPU burst is going to be? [13].

C. PS

It associates each process with a priority number. The CPU is allocated to the process with the highest priority. If there are multiple processes with same priority, then FCFS will be used to allocate the CPU. Lower priority processes may starve, because the CPU may have a steady supply of higher priority processes [13].

III. SCHEDULING CRITERIA

Many criteria have been suggested for comparing CPU scheduling algorithms. Those characteristics are used for comparison and to make a substantial difference in which algorithm is judged to be the best [2]. The criteria include the following:

Context Switch: This is the process of storing and restoring context (state) of a preempted process, so that execution can be resumed from same point at a later time.

Throughput: This is the number of processes completed per unit time.

CPU Utilization: This is a measure of how much busy the CPU is.

Turnaround Time: This refers to the time interval from the time of submission of a process to the time of its completion.

Waiting Time: This is the total time a process has been waiting in ready queue.

Response Time: It is approximately the time of submission of a process until its first access to the CPU.

So, a good scheduling algorithm should possess the following characteristics [3][5]:

Minimum context switches.

Maximum CPU utilization.

Maximum throughput.

Minimum turnaround time.

Minimum waiting time.

Minimum response time.

In this paper, more tests will be done on the algorithm presented in [2] (i.e. A New Improved Round Robin (NIRR)) against First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), Improved Round Robin (IRR) and Longest Job First with Combinational Burst Time (LJF+CBT) scheduling algorithms by using normal and exponential statistical distributions to generate the burst time of processes and also assessing these algorithms with the same scheduling criteria (i.e. average waiting time, average turnaround time, average response time and number of context switches) used in [2].

IV. LITERATURE REVIEW

Oyetunji and Oluleye [9] evaluated the performance of three basic CPU scheduling algorithms (namely First Come First Serve, Priority Scheduling and Shortest Job First) based on four CPU scheduling objectives (average waiting time, average turnaround time, average CPU utilization and average throughput) to determine which algorithm is most suitable for which objective. Maria, Aminu, Sani and Saleh [8] evaluated the performance of three CPU scheduling algorithms, namely FCFS, SJF and RR scheduling algorithms. The burst and arrival time of processes used in the simulation were generated using exponential distribution, it was observed from the results that the SJF produces the minimal average waiting time. Suri and Sumit [14] analyzed the impact of scalability on different CPU scheduling algorithms with reference to average waiting time, average turnaround time and average response time to determine which algorithm is most suitable for uniprocessor environment. The burst time, arrival time and priority of processes were randomly generated using exponential probability distribution and the performance of all algorithms has been evaluated with reference to arrival time or without arrival time. Manish and AbdulKadir [7] proposed an algorithm known as Improved Round Robin (IRR) CPU Scheduling algorithm. It works by allocating the CPU to processes in RR fashion. This algorithm reduces drastically the AWT, ATAT and NCS compared to the RR Scheduling algorithm.

Abdullahi and Junaidu [1] developed an algorithm that made improvement to the Longest Job First (LJF) CPU scheduling algorithm by reducing the waiting time of shorter processes. This algorithm known as Longest Job First with Combinational Burst Time (LJF+CBT) works by sorting the processes in descending order of their burst times and then it determines a threshold known as Combined Weighted Average (Cwa) which is the average of the processes. Abdulrazaq, Saleh and Junaidu [2] developed an algorithm known as A New Improved Round Robin (NIRR) CPU Scheduling Algorithm; that improved the algorithm by [7]. This algorithm introduced a new queue known as the ARRIVE queue; which holds processes according to their arrival times while there are other processes in the ready queue (say REQUEST) waiting for CPU allocation.

V. NORMAL DISTRIBUTION

A normal distribution in a variate X with mean μ>0 and variance $\sigma^2>0$ is a statistic distribution with probability density function (pdf)

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \quad (1)$$

On the domain $x \in (-\infty, \infty)$

The mean and variance of a normally distributed data is given by:

$$E(X) = \frac{a+b}{2} \text{ and } V(X) = \frac{(b-a)^2}{12} \quad (2)$$

Where a and b are the upper and lower bound of the data respectively [6].

VI. EXPONENTIAL DISTRIBUTION

According to [6] consider a random variable X that is exponentially distributed with a parameter rate of exponential distribution λ. The probability density function (pdf) is given by:

$$f(x) = \lambda e^{-\lambda x}$$

It has mean and variance is given by:

$$E(X) = \frac{1}{\lambda} \text{ and } V(X) = \frac{1}{\lambda^2}$$

VII. SIMULATION

FCFS, SJF, RR, IRR, LJF+CBT and NIRR were simulated and their performance on four performance criteria: AWT, ATAT, ART and NCS were observed. The simulations were carried out in a single processor environment with only CPU bound and no I/O bound processes. The system was assumed to have no context switching cost [2].

A process generator routine was built to generate the process sets. Each process in the process set is a tuple: <(process_id, CPU_time)> [2].

The Burst time (i.e. the CPU_time) was generated using both normal and exponential distributions. A process burst time generator was developed to take care of the random burst time of different processes in the system.

A. Experimental setup

Hardware

- Hewlett Packard (HP) laptop with a T2300 processor running at 1.66GHz
- 1.5GB of RAM and
- 75GB of hard disk

Software

- Window XP operating system
- NetBeans IDE 6.7.1 version and JDK1.7

B. RESULTS OF NORMAL DISTRIBUTION

The following figures show the relationships between the CPU scheduling algorithms under study. 3000 processes were generated using normal distribution with a mean of 50.5ms, standard deviation of 28.6ms and time quantum of 25ms used by RR and IRR.

Fig. 1 shows the overall graphical result of the Average Waiting Time for all cases of values taken. It was observed from the graph that the SJF has the best performance when number of processes is less than 1500 and NIRR came second, followed by LJF+CBT, FCFS, IRR and RR respectively. But, when the number of processes exceeds 1500, LJF+CBT tends to produce the best performance, SJF was second, then NIRR came third, followed by FCFS, IRR and RR respectively.

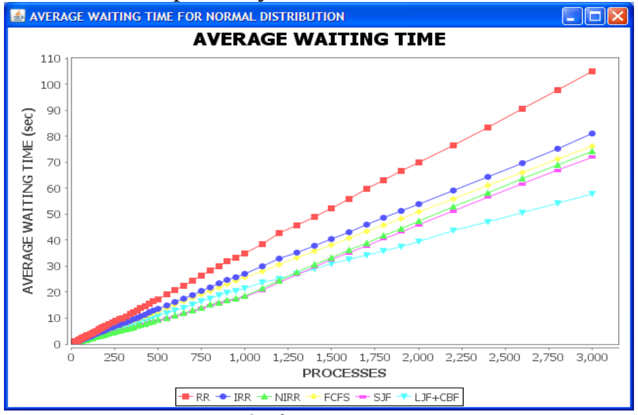

Fig. 1: Graph of Average Waiting Time

Fig. 2 shows the overall graphical result of the Average Turnaround Time for all cases of values taken. It was observed from the graph that the SJF has the best performance when number of processes is less than 1500 and NIRR came second, followed by LJF+CBT, FCFS, IRR and RR respectively. But, when the number of processes exceeds 1500, LJF+CBT tends to produce the best performance, SJF was second, then NIRR came third, followed by FCFS, IRR and RR respectively.

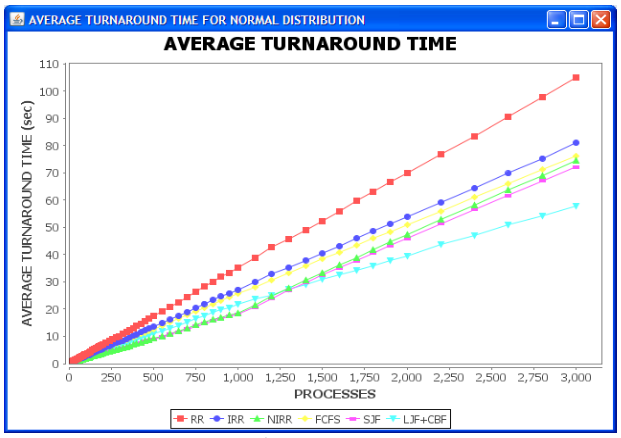

Fig. 2: Graph of Average Turnaround Time

Fig. 3 shows the overall graphical result of the Average Response Time for all cases of values taken. It was observed from the graph that the RR has the best performance, IRR came second, NIRR came third followed by SJF, LJF+CBT and FCFS respectively when the number of processes is less than 1700. But when the number of processes exceeds 1700, RR and IRR still maintain the first and second positions, while LJF+CBT came third followed by NIRR, SJF and FCFS respectively.

Fig. 4 shows the overall graphical result of the Number of Context Switches for the same processes. It was observed from the graph that LJF+CBT produced the best performance, FCFS and SJF came second producing the same performance followed by NIRR, IRR and RR respectively.

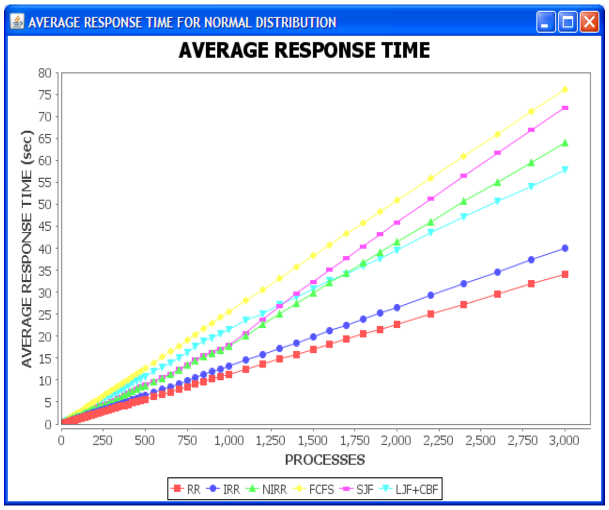

Fig. 3: Graph of Average Response Time

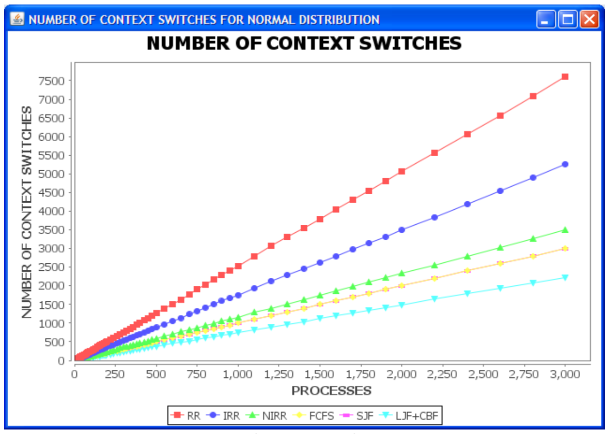

Fig. 4: Graph of Number of Context Switches

C. *RESULTS OF EXPONENTIAL DISTRIBUTION*

The following figures show the relationships between the CPU scheduling algorithms under study. 3000 processes were generated using exponential distribution with a rate (λ) of 0.02 per ms and time quantum of 25ms used by RR and IRR.

Fig. 5 shows the overall graphical result of the Average Waiting Time for all cases of values taken. It was observed from the graph that the SJF produced the best performance, and then NIRR, followed by the IRR, LJF+CBT, FCFS and RR respectively.

Fig. 6 shows the overall graphical result of the Average Turnaround Time for all cases of values taken. It was observed from the graph that the SJF produced the best performance, and then NIRR, followed by the IRR, LJF+CBT, FCFS and RR respectively.

Fig.7 shows the overall graphical result of the Average Response Time for all cases of values taken. It was observed from the graph that the RR has the best performance, IRR came second, then NIRR followed by the SJF, LJF+CBT and FCFS respectively.

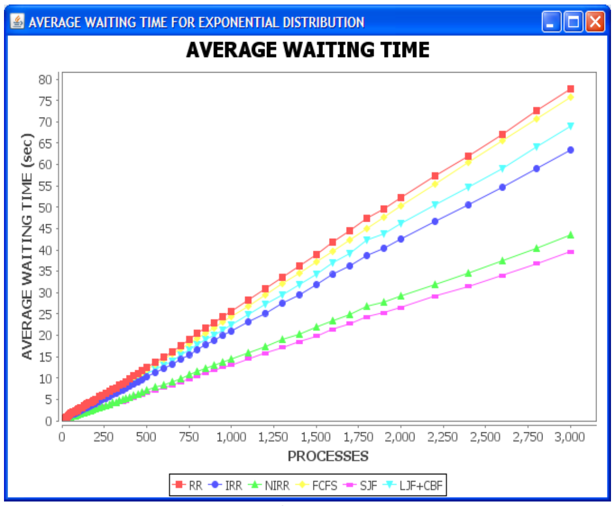

Fig. 5: Graph of Average Waiting Time

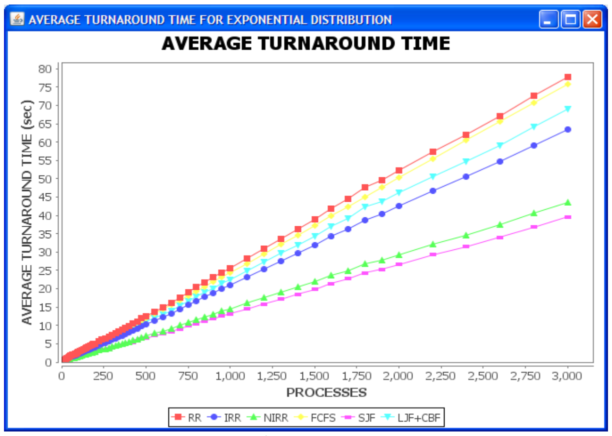

Fig. 6: Graph of Average Turnaround Time

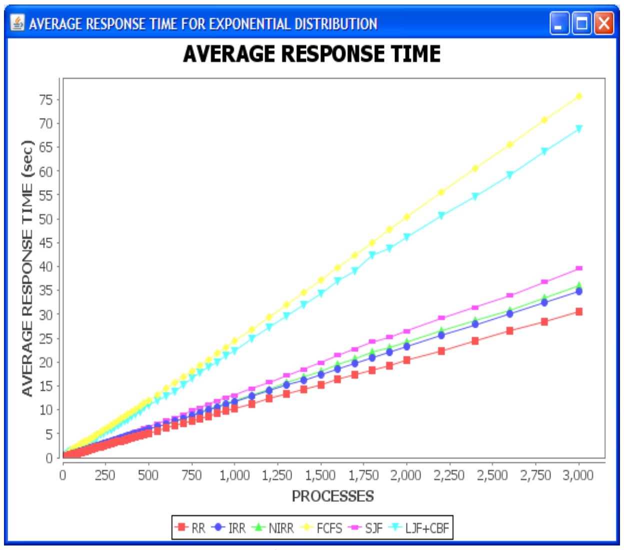

Fig. 7: Graph of Average Response Time

Fig. 8 shows the overall graphical result of the Number of Context Switches for the same processes. It was observed from the graph that LJF+CBT produced the best performance, FCFS and SJF came second producing the same performance followed by NIRR, IRR and RR respectively.

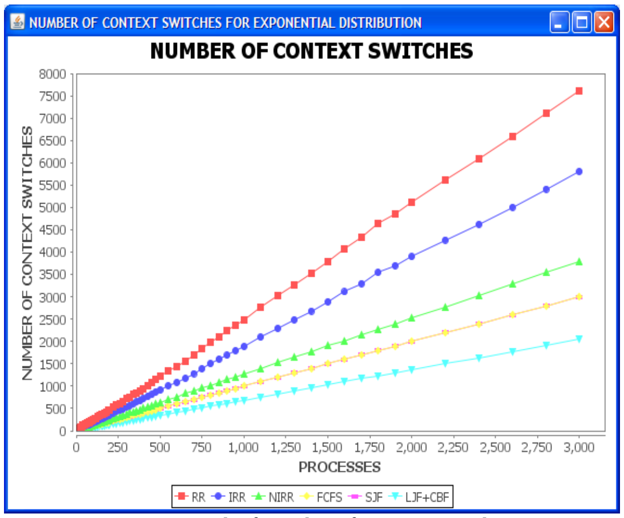

Fig. 8: Graph of Number of Context Switches

VIII. CONCLUSION

The algorithm presented in NIRR with FCFS, SJF, RR, IRR and LJF+CBT CPU scheduling algorithms were successfully implemented in Java. The results of burst time of processes that were generated using normal and exponential statistical distributions were compared based on four scheduling criteria namely, AWT, ATAT, ART and NCS.

Results from the simulation show that, in the Round Robin CPU scheduling category, NIRR will produce the best results in terms of minimizing Average Waiting Time (AWT), Average Turnaround Time (ATAT) and Number of Context Switches (NCS) in both statistical distributions. But with respect to minimizing Average Response Time (ART) in the Round Robin CPU scheduling category, NIRR will produce the worst result in both statistical distributions.

In general, from the simulation results of both normal and exponential statistical distributions, it shows that SJF will be the best scheduling algorithm followed by NIRR in terms of minimizing Average Waiting Time (AWT) and Average Turnaround Time (ATAT), this will be followed by IRR, LJF+CBT, FCFS and RR respectively in the exponential distribution. In both cases of statistical distributions, NIRR will come second when minimizing Average Waiting Time (AWT) and Average Turnaround Time (ATAT). But in the normal statistical distribution, as the number of processes exceeds 1500, LJF+CBT will be the best scheduling algorithm in terms of minimizing Average Waiting Time (AWT) and Average Turnaround Time (ATAT), followed by SJF, NIRR, LJF+CBT, FCFS, IRR and RR respectively. The NIRR will come third in term of minimizing Average Waiting Time (AWT) and Average Turnaround Time (ATAT) when the number of processes exceeds 1500 in the normal distribution. In the case of minimizing Average Response Time (ART), RR and IRR will come first and second respectively in both statistical distributions, this will be followed by NIRR, SJF, LJF+CBT and FCFS respectively. The NIRR will come third when minimizing Average Response Time (ART). But in the normal distribution, as the number of processes exceeds 1700, LJF+CBT will come third followed by NIRR, SJF and FCFS respectively. The NIRR will come fourth in terms of when minimizing Average Response Time (ART) when the number of processes exceeds 1700 in the normal distribution. The LJF+CBT will be the best scheduling algorithm in terms of minimizing Number of Context Switches (NCS) in both statistical distributions. FCFS and SJF will come second because they will produce the same performance, this will be followed by NIRR, IRR and RR respectively. The NIRR will come third when minimizing Number of Context Switches (NCS) in both statistical distributions.

Based on the results obtained, it was observed that the performance of NIRR is better than that of the Simple Round Robin CPU scheduling algorithm and the Improved Round Robin CPU scheduling algorithm by [7], in the sense that, it produces minimal average waiting time, average turnaround time and number of context switches in both statistical distributions. The recommendation in this work is to implement NIRR in the systems that adopt the Round Robin scheduling; so as to improve the performance of the systems.

REFERENCES

[1] Abdullahi, I and Junaidu, S. B, “Empirical Framework to Migrate Problems in Longer Job First Scheduling Algorithm (LJF+CBT),” International Journal of Computer Applications (0975 – 8887) , Vol. 75, No. 14, 2013, pp. 9-14.

[2] Abdulrazaq, A, Saleh, E. A and Junaidu B. S, “A New Improved Round Robin (NIRR) CPU Scheduling Algorithm,” International Journal of Computer Applications (0975 – 8887),Vol. 90, No. 4, 2014, pp. 27-33.

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

[4] Behera, H.S, Brajendra, K. S, Anmol, K. P and Gangadhar, S., “A New Proposed Round Robin with Highest Response Ratio Next (RRHRRN) Scheduling Algorithm for Soft Real Time Systems.” International Journal of Engineering and Advanced Technology , 1 (3), 2012, pp. 200-206.

[5] Ishwari, S. R and Deepa, G, “A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems,” International Journal of Innovations in Engineering and Technology (IJIET), Vol. 1 No. 3, 2012, pp 1-11.

[6] Jerry, B, John, S.C, Barry, L.N and David K, Discrete-Event System Simulation, Fourth ed., 2005, Pearson Education International,.

[7] Manish K. M. and Abdul Kadir K, “An Improved Round Robin CPU Scheduling Algorithm,” Journal of Global Research in Computer Science, ISSN: 2229-371X, Vol. 3, No. 6, 2012, pp 64-69.

[8] Maria Abur, Aminu Mohammed, Sani Danjuma and Saleh Abdullahi, “Critical Simulation of CPU Scheduling Algorithm using Exponential Distribution,” International Journal of Computer Science Issues, Vol. 8, No. 2,2011, pp. 201-206.

[9] Oyetunji E.O and Oluleye A. E., “Performance Assessment of Some CPU Scheduling Algorithms”. Research Journal of Information Technology , 1 (1), 2009, pp. 22-26.

[10] Saeidi, S and Hakimeh, A. B, “Determining the Optimum Time Quantum Value in Round Robin Process Scheduling Method,” I.J. Information Technology and Computer Science, Vol. 10, 2012, pp. 67-73.

[11] Samih, M. M, Rida, S. Z and Safwat H. H, “Finding Time Quantum Of Round Robin CPU Scheduling Algorithm In General Computing Systems Using Integer Programming.” IJRRAS ,Vol. 5. No. 1, 2010, pp. 64-71.

[12] Saroj Hiranwal and K. C. Roy, ”Adaptive Round Robin Scheduling using shortest burst approach based on smart time slice,” International Journal of Computer Science and Communication, Vol. 2, No. 2, 2011, pp. 319-323.

[13] Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, 7th Ed, 2005, John Wiley and Sons Inc.

[14] Suri, P.K and Sumit, M, “Design of Stochastic Simulator for Analyzing the Impact of Scalability on CPU Scheduling Algorithms,” International Journal of Computer Applications (0975 – 8887),Vol. 49, No. 17, 2012, pp. 4-9.