Optimization of Priority based CPU Scheduling Algorithms to Minimize Starvation of Processes using an Efficiency Factor

Muhammad A. Mustapha
Department of Mathematics,
Ahmadu Bello University,
Zaria, Nigeria

Saleh E. Abdullahi
Department of Computer Science,
Nigerian Turkish Nile University,
Abuja, Nigeria

Sahalu B. Junaidu
Iya Abubakar Computer Center,
Ahmadu Bello University,
Zaria,
Nigeria

ABSTRACT

The priority based CPU scheduling algorithm (i.e. Shortest Job First (SJF) or Priority Scheduling (PS)) is a kind of scheduling algorithm that assigns the CPU to processes based on the priority of each process. The shortcoming of both of these algorithms is starvation (i.e. starvation of processes with longer burst times in the case of SJF and starvation of processes with lower priorities in the case of PS). This paper proposes a new algorithm that introduces the concept of EFFICIENCY FACTOR to all processes. This proposed algorithm was implemented and benchmarked against SJF, PS and the Optimum Service Time Concept for Round Robin Algorithm (OSTRR) by [9] using Uniform distribution to generate the burst times, Exponential distribution to generate the priorities and Poisson distribution to generate the arrival times of processes. It is observed that in the SJF category, the traditional SJF produced better Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared with the proposed SJF. But they both produced the same Number of Context Switches (NCS). The proposed SJF produced better results compared with OSTRR with respect to AWT, ATAT, ART, NCS and WTVD. While in the PS category, the proposed priority produced better AWT, ATAT, ART and WTVD compared to the traditional Priority scheduling algorithm. But they both produced the same NCS. The proposed Priority algorithm produced better results compared with OSTRR with respect to NCS and WTVD also produced almost the same result in terms of AWT and ATAT in all categories of the statistical distributions used. Based on these results, the proposed priority algorithm should be preferred over the traditional priority algorithm.

Keywords

CPU scheduling algorithms, Efficiency factor, Shortest Job First Scheduling, Starvation, Priority Scheduling, Waiting Time Variance Deviation

1. INTRODUCTION

The Central Processing Unit (CPU) is an important component of the computer system; hence it must be utilized efficiently. This can be achieved through what is called CPU scheduling [7]. CPU Scheduling refers to a set of policies and mechanisms to control the order of work to be performed by a computer system. The CPU scheduling is one of the most important tasks of the operating system [5]. The need for a scheduling algorithm to achieve the efficiency of the CPU arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously). CPU Scheduling is the act of selecting the next process for the CPU to service, once the current process leaves the CPU idle. Some basic CPU scheduling algorithms are listed below:

1. First-Come First-Serve (FCFS)

By far the simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. The average waiting time under the FCFS policy, however, is often quite long [2].

2. Shortest-Job-First (SJF)

This is a priority based algorithm which associates with each process the length of the process's next CPU burst [4]. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If the next CPU bursts of two processes are the same, FCFS scheduling is used. The SJF scheduling algorithm gives the minimum average waiting time for a given set of processes. The real difficulty with the SJF algorithm is knowing the length of the next CPU request. The SJF algorithm an optimal algorithm because it produces minimum average waiting time, average turnaround time and number of context switches.

3. Priority Scheduling (PS)

A priority number is associated with each process. The CPU is allocated to the process with the highest priority. If there are multiple processes with same priority, then FCFS is used to allocate the CPU. A variation of this scheme allows preemption of the current process whenever a higher priority process arrives. Another variation of the policy adds an aging scheme, where the priority of a process increases as it remains in the ready queue.

4. Round-Robin Scheduling (RR)

This algorithm is especially designed for time-sharing systems; each process gets a small unit of CPU time. This algorithm allows the first process in the queue to run until it expires its time, and then runs the next process in the queue. In a situation where a 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.

1.1 Performance Criteria

The various CPU scheduling algorithms have different properties, and the choice of a particular algorithm may favor one class of processes over another. For selection of an algorithm for a particular situation, properties of various algorithms must be considered. The scheduling criteria include the following [10]:

1. Context Switch: A context switch is process of storing and restoring context (state) of a preempted process, so that execution can be resumed from same point at a later time.
2. Throughput: Throughput is defined as number of processes completed per unit time. Context switching and throughput are inversely proportional to each other.
3. CPU Utilization: This is a measure of how much busy the CPU is. Usually, the goal is to maximize the CPU utilization.
4. Turnaround Time: Turnaround time refers to the total time which is spent to complete the process.
5. Waiting Time: Waiting time is the total time a process has been waiting in ready queue.
6. Response Time: response time is the time from the submission of a request until the first response is produced

So, a good scheduling algorithm for real time and time sharing systems must possess the following characteristics:

1. Minimum context switches.
2. Maximum CPU utilization.
3. Maximum throughput.
4. Minimum turnaround time.
5. Minimum waiting time.
6. Minimum response time.

1.2 Waiting Time Variance

Variance is a statistical word, which is the measurement of the spread between numbers in a data set. It measures how far each number in the set is from the mean.

$$\sigma^2 = \frac{\sum(i-\mu)^2}{N} \qquad (1)$$

$$where\ \sigma^2\ is\ the\ variance, N\ is\ the\ number\ of\ processes,$$

$$i\ the\ i^{th}\ process\ \ and\ \mu\ is\ the\ mean\ of\ the\ processes$$

A variance value of zero indicates that all values within a set of numbers are identical; all variances that are non-zero are assigned positive numbers. A large variance indicates that numbers in the set are far from the mean and each other, while a small variance indicates the opposite [3]. The waiting time variance is the measure of how far the waiting time of processes are from the mean of the processes waiting times. This implies that waiting time variance of processes should be minimized [6].

Priority based CPU Scheduling Algorithms such as Shortest Job First Algorithm which assigns CPU to processes according to their burst times and Priority CPU Scheduling Algorithm which assigns CPU to processes according to their priorities suffer from the problem of starvation. They are not fair as they are biased to processes of high priorities [9].

Consequently the goal of this work is to minimize starvation in the priority based CPU scheduling algorithms that is shortest job first and priority scheduling algorithms thereby reducing the average waiting time, average turnaround time, average response time, number of context switches and waiting time variance deviation.

2. RELATED WORK

Brinch [1] proposed an algorithm named Highest Response Ratio Next (HRRN) CPU scheduling algorithm to address the shortcomings of the Shortest Job First (SJF) and Shortest Job Next (SJN) CPU Scheduling algorithms which are starvation and lack of fairness to longer burst time processes. It does this by using equation (2) to calculate the Response Ratio of processes

$$Response\ Ratio = \frac{Waiting\ Time + Service\ Time}{Service\ Time} \quad (2)$$

It allocates the CPU to the process with the Highest Response Ratio and it also takes care of the problem of determining the next burst time of processes in SJF. Its disadvantage is that it is computationally intensive and its limitation is that it does not address user priority.

Rakesh *et al* [8] proposed a new variant of Round Robin Scheduling algorithm by executing the processes according to a new calculated fit factor f and using the concept of dynamic time quantum. Generally with every process three factors are associated. These factors are user priority, burst time and arrival time. Above factors play an important role to decide in which sequence the processes will be executed. Sorting according to the importance of these factors, user priority comes first, then the burst time and at last the arrival time of the processes. If all the 3 factors are mixed up to calculate a new factor i.e. Fit Factor "f" which will decide the order of execution then average waiting time, average turnaround time and number of context switches will be decreased. So to increase the responsiveness of the system, RR algorithm should be used. Generally in RR algorithm, processes are taken from the ready queue in FCFS manner for execution. But in the proposed algorithm, "f" is calculated for each process. The process having the lowest "f" value will be scheduled first. The two important criteria that decide the early execution of processes are – higher user priority and shorter burst time. As user priority has higher importance than other factors, so it is given a weight age of 60% and burst time is given 40%, assuming that all the processes have same arrival time i.e. arrival time = 0. Let the User Priority = UP, User Priority Weight = UW, Shorter Burst time Priority = SP, Burst time Priority Weight = BW. Then Fit Factor "f" can be calculated as

$$f\ =\ UP\ *\ UW\ +\ SP * BW \qquad (3)$$

Dynamic time quantum is used in order to overcome the limitations of static RR. To get the optimal time quantum, median of the remaining burst time is taken as the time quantum.

Saxena and Agarwal [9] designed an algorithm known as Design and Performance Evaluation of Optimum Service Time Concept for Round Robin Algorithm (OSTRR). Generally, with every process, three factors are associated. These factors are user priority, burst time and arrival time. These factors play an important role to decide in which sequence the processes will be executed. Since the algorithm is a priority based system, the concept of Optimum Priority was employed, which combines user defined priority, effect of shorter burst time and effect of arrival time in a way so as to achieve better turnaround time and average waiting time. A weight of 0.5 is assigned to user or system defined priority, a weight of 0.3 to burst time and 0.2 to arrival time. This ensures that user defined priority; burst time; and arrival time get consideration while deciding order of execution of processes. A higher priority process gets a higher number; a shorter process also gets a higher number; and a process that arrived earlier also gets a higher number in the numbering scheme. An Optimum Priority 'OP' is given as:

$$OP = \lceil 0.5 * P + 0.3 * BT + 0.2 * AT \rceil \quad (4)$$

Where P is user or system defined priority; BT is priority number assigned according to shorter burst time; and AT is priority number assigned according to early arrival of the process. A time quantum was selected while taking into account the same considerations that is taken while selecting time quantum for RR algorithm. The Optimum service time 'OST' for each process is given by equation (5):

$$OST_i = OP_i * q \quad (5)$$

Where OST_i is the service time of process with Optimum Priority OP_i, q is the decided time quantum. All the processes are placed in a priority queue. After calculating the service time of each process, CPU is assigned to the process with highest optimum service time. In case of conflict, the process with shorter burst time is given preference. If conflict still persists the process that arrived earlier is given preference. The process decided executes for a period that is equal to optimum service time of the process OST_i or its burst time whichever is smallest. In case a process or the set of processes with same value of OP, which so ever is applicable, finish execution after a single round, the value of OST is redistributed removing the value of OST of finished process and redistributing the value of OST accordingly.

3. THE PROPOSED ALGORITHMS

The proposed CPU scheduling algorithms use Efficiency factor values to assign priority to processes which will be used for CPU allocation. The Efficiency factor value for the proposed SJF uses equation (6):

$$E = BTW + WW \quad (6)$$

Where BTW is Burst time Weight and WW is Waiting time Weight.

The first process to arrive the arrival queue is scheduled first since it is the only process in the system then and it is allocated the CPU to execute for the period of its burst time. By the time it has finished executing, some processes will have arrived the arrival queue their efficiency values is calculated by finding the Burst Tme Weight (BTW) for all the processes that arrived within this period so that the process with the least burst time is given a BTW of 1. All other processes are given BTW in ascending order of their burst times. Also each process is given Waiting Time Weight (WW), according to their arrival times so that the first process to arrive is given a WW of 1. All other processes under consideration are given WW of 2, 3… until all processes under consideration are given WW. Then the E-value of each process will be computed using equation (6) and the process with the lowest E-value is scheduled first followed by the process with next lower E-value until all processes that arrived the system have finished executing. This process of determination of E-values and scheduling of processes is performed until there is no process remaining in the arrival and ready queues.

The Efficiency factor value for the proposed Priority scheduling algorithm uses equation (7):

$$E = PW + \mathrm{BTW} + WW \quad (7)$$

where, PW is Priority Weight, BTW is Burst time Weight and WW is Waiting time Weight.

Similarly, for the priority (PS) category, the same method as in the case of SJF is employed but with the inclusion the Priority Weight (PW), which is calculated based on the priority values of the processes in which the process with the highest priority is given a priority weight of 1 and all other processes are given PW of 2, 3…until all processes under consideration are given PW. Then the E-value of each process will be computed using equation (7) and the process with the lowest E-value is scheduled first followed by the process with next lower E-value until all processes that arrived the system have finished executing. This process of determination of E-values and scheduling of processes is performed until there is no process remaining in the arrival and ready queues.

3.1 The pseudo code of the Proposed Priority Based CPU Scheduling Algorithms

This pseudo code illustrates how the priority based CPU scheduling algorithms work i.e. the priority and shortest job first CPU scheduling algorithms.

```
Step 1: Start

INPUT: Number of Processes (N), Burst Time (BT) of processes, Arrival Time (AT) of processes, Priority (PRIO) of processes, Queue READY, Queue ARRIVAL

OUTPUT: Number of Context Switches (NCS), Average Waiting Time (AWT), Average Turnaround Time (ATAT), Waiting Time Variance Deviation (WTVD) and Average Response Time (ART)

Step 2: $NCS = 0;$
            $AWT = 0;$
            $ATAT = 0;$
            $ART = 0;$
            $WTVD = 0;$
            $BT = \text{uniform (a, b)};$
            $AT = \text{Poisson (rate)};$
            $PRIO = \text{Exponential (rate)};$
            $Last\ Point = 0;$
        $n = N;$
Step 3: For $i = 1\ to\ n$
            $efficiency\ factor_i = 0;$   // efficiency factor
END For
Step 4: $WHILE\ (ARRIVAL != NULL)$
Step 5: $WHILE\ (READY != NULL)$
Step 6: For $i = 1\ to\ n$
        $REQUEST \leftarrow Process_i$   // Fill the ready queue
        //according to Arrival Time
            Assign priority weight, burst time weight and
            arrival weight using the generated values based on
            the proposed algorithm
END For
Step 7: For $i = 1\ to\ n$
        $if\ algorithm == SJF$
        $efficiency\ factor_i = BTW + WW$
        $else\ if\ algorithm == priority$
        $efficiency\ factor_i = PW + BTW + WW$
END if
END For
    $Min = efficiency\ factor[1]$
Step 8: For ($i = 1\ to\ n - 1$ )
If $efficiency\ factor_i < Min$ //sort processes according // to lowest efficiency factor
        $Min = efficiency\ factor_i$
        $efficiency\ factor_i = efficiency\ factor_{i+1}$
        $efficiency\ factor_{i+1} = Min$
END if
END For
Step 9: For ($i = 1\ to\ n$ )
        $Last\ Point = Last\ Point + BT_i$
        $TAT_i = Last\ Point - AT_i$
        $WT_i = TAT_i - BT_i$
        $RT = TAT_i - BT_i$
        $BT_i = 0$
        $NCS_i = NCS_i + 1$
END For
END WHILE
END WHILE
Step 10: uniform (a, b) { // generate random values using // uniform distribution
        $a + rand(0,1) * (b - a)$
}
Step 11: Poisson (rate) { // generate random values using // Poisson distribution
        $k = 0$
        $p = 1$
        $l = e^{-rate}$
        $do\ \{$
            $k = k + 1$
            $p = p * rand(0,1)$
        $\}while(p \geq l)$
        $k = k - 1$
        $return\ k$
}
Step 12: Exponential (rate) { // generate random values // using exponential distribution
        $return\ \frac{-\log(1 - rand(0,1))}{rate}$
}
}
Step 13: Calculate OUTPUT parameters
        $AWT = \frac{\sum_{i=1}^{n} WT_i}{n}$
        $ATAT = \frac{\sum_{i=1}^{n} TAT_i}{n}$
        $ART = \frac{\sum_{i=1}^{n} RT_i}{n}$
        $WTV = \frac{\sum_{i=1}^{n} (WT_i - AWT)^2}{n}$
For ($i = 1\ to\ k$ ) //k stands for the algorithm under consideration and opt is the algorithm with the least waiting time variance.
        $WTVD_k = \frac{WTV_k - WTV_{opt}}{WTV_{opt}} * 100\%$
END For
END
```

3.2 Illustrative Examples

To demonstrate the previous considerations, this example is considered, in which each process with its burst and arrival time as shown in Table 4.1, where the time quantum used in OSTRR is 10*ms*. The values were chosen randomly.

Table 1: Process Table

PR_ID	AT	BT	PRIO
P1	0	3	10
P2	1	2	6
P3	3	2	4
P4	5	4	3
P5	6	1	7
P6	7	5	11
P7	9	6	8
P8	12	4	9
P9	13	3	1
P10	15	2	5
P11	15	5	2
P12	15	6	13
P13	18	7	12

3.2.1 Shortest Job First (SJF)

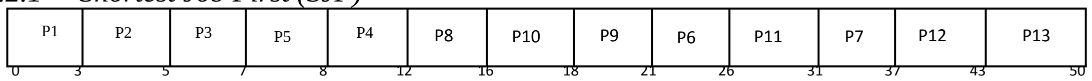

Figure 1: Gantt chart representation of SJF

3.2.2 Priority Scheduling (PS)

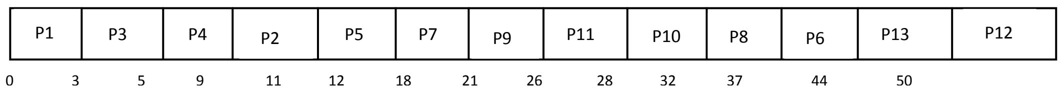

Figure 2: Gantt chart representation of Priority CPU Scheduling Algorithm

3.2.3 Proposed Shortest Job First

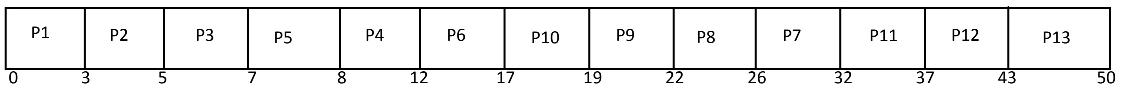

Figure 3: Gantt chart representation of Proposed Shortest Job First

3.2.4 Proposed Priority Scheduling

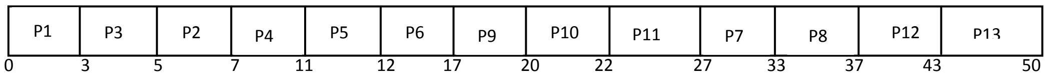

Figure 4: Gantt chart representation of Proposed Priority Scheduling

3.2.5 OSTRR

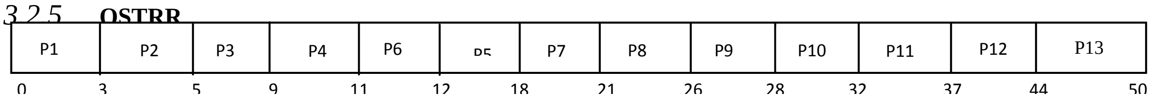

Figure 5: Gantt chart representation of OSTRR

Table 2: Comparative Table

Algorithms	AWT	ATAT	ART	NCS	WTVD (%)	WTVD (%)
SJF	8.31	13.31	8.3	0	16.76	
Priority	9.77	13.62	9.8	0		18.93
Pro. SJF	8.62	12.46	8.6	0	0	
Pro. Priority	9.15	13	9.15	0		0
OSTRR	9.46	13.21	9.5	0	35.15	29.62

Table 2 shows the comparative results of the algorithms under study. In the SJF category, the proposed algorithm produced the best result followed by the traditional SJF algorithm and the OSTRR. In the priority category the proposed algorithm produced the best result followed by the traditional priority algorithm followed by OSTRR.

4 SIMULATION

In this paper, the following five performance criteria are studied: AWT, ATAT, ART, NCS and WTVD. The evaluation is done using a simulation by generating 500 processes randomly with Uniform statistical distribution to generate burst times of the processes. The simulation is done in two different categories; in the first category (i.e. the Shortest Job First (SJF)) consisting of: SJF, the new SJF and Optimum Service Time Round Robin CPU Scheduling algorithms and in the second category (i.e. the Priority Scheduling (PS)) consisting of: PS, the new PS and Optimum Service Time Round Robin CPU Scheduling algorithms to observe these criteria. The simulation environment where all the experiments were performed was a single processor environment and all the processes are independent and CPU bound, no process was I/O bound and the system was also assumed to have no context switch cost.

A process generator routine to generate the process sets was built. Each process in the process set is a tuple: <(process*_id*, *arrival_time*, *CPU_time*, *priority*)>. The process arrival was modeled as a Poisson random process. Hence, the inter-arrival times are Poisson distributed. A process arrival generator was developed to take care of the random arrival of different processes to the system. The generator produces the inter-arrival times utilizing some specific mean (arrival intensity) of the distribution function. Burst time (i.e. the *CPU_time*) was generated using uniform distribution. Process priority (i.e. *priority*) was generated using exponential distribution. A process burst time generator was developed to take care of the random burst time of different processes in the system and also a process priority generator was developed to take care of the random priority of different processes in the system.

4.1 Results obtained using uniform distribution to generate burst time

The following shows the relationships between the CPU scheduling algorithms under study using 500 processes generated by uniform distribution with burst times ranging between 1 and 100*ms* and time quantum of 10*ms* used by OSTRR.

4.1.1 First category

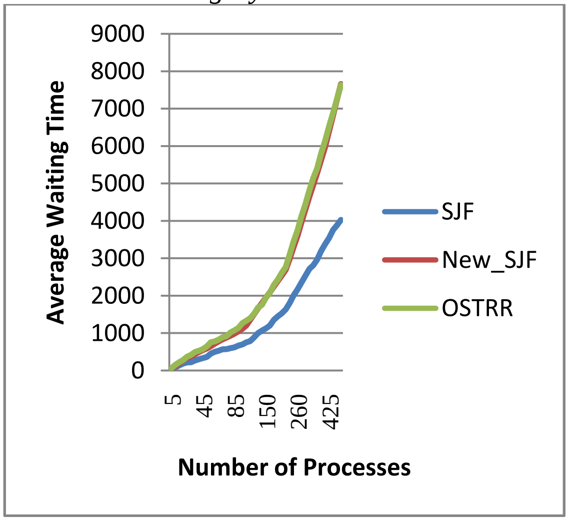

Figure 6: Graph of Average Waiting Time using Uniform distribution

Figure 6 shows the overall graphical result of the Average Waiting Time for all cases of values taken. It was observed from the graph that the traditional algorithm is better than the proposed algorithm while the proposed algorithm is better than OSTRR in terms of minimizing AWT.

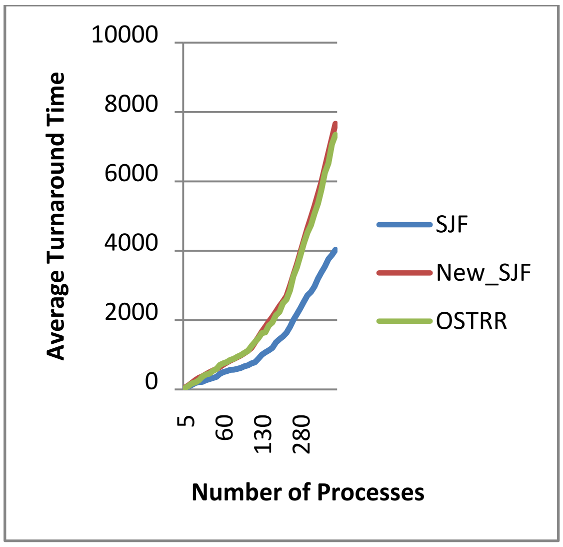

Figure 7: Graph of Average Turnaround Time using Uniform distribution

Figure 7 shows the overall graphical result of the Average Turnaround Time for all cases of values taken. It was observed from the graph that the traditional algorithm is better than the proposed algorithm while the proposed algorithm is better than OSTRR in terms of minimizing ATAT.

Figure 8shows the overall graphical result of the Average Response Time for all cases of values taken. It was observed from the graph that the traditional algorithm SJF is better than the proposed algorithm while the proposed algorithm is better than OSTRR in some cases in terms of minimizing ART.

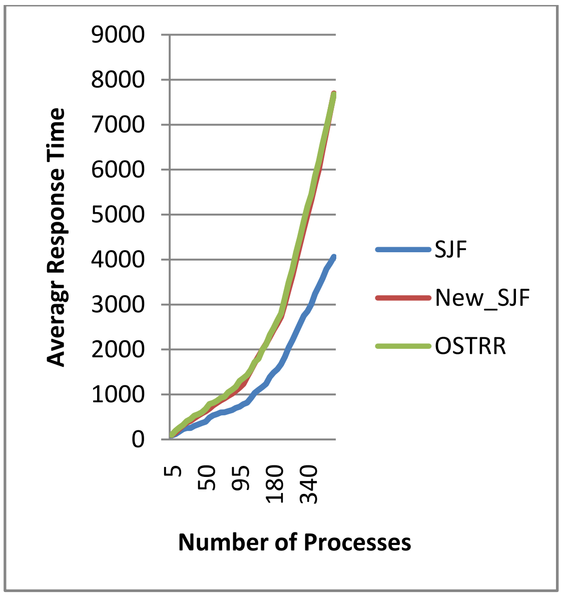

Figure 8: Graph of Average Response Time using Uniform distribution

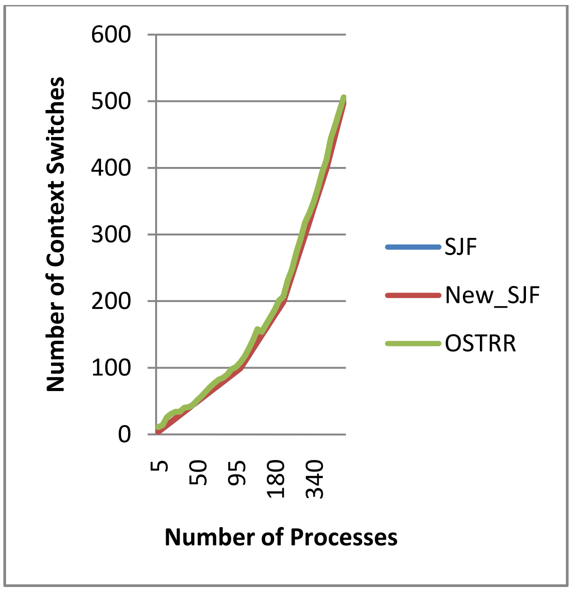

Figure 9: Graph of Number of Context Switches using Uniform distribution

Figure 9 shows the overall graphical result of the Number of Context Switches for the same processes. It was observed from the graph that the traditional algorithm and the proposed algorithm produce the same number of context switches which is better than that of OSTRR.

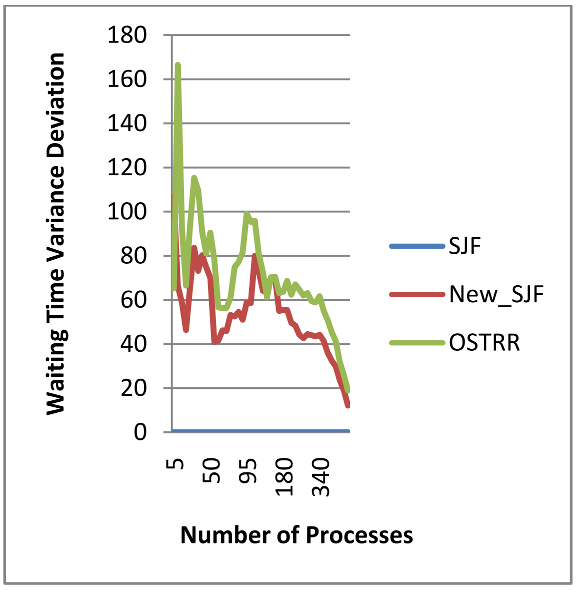

Figure 10: Graph of Waiting Time Variance Deviation using Uniform distribution

Figure 10 shows the overall graphical result of the Waiting Time Variance Deviation for the same processes. It was observed from the graph that the traditional algorithm is better than the proposed algorithm while the proposed algorithm is better than OSTRR in terms of minimizing WTVD.

From the results of the two scheduling algorithms (SJF and OSTRR) compared with the proposed algorithm (the new SJF), it has shown that the traditional SJF produced best Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared with the proposed new SJF. But they both produced the same

Number of Context Switches (NCS). The new proposed SJF produced better results compared with OSTRR with respect to Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART), Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD).

4.1.2 Second category

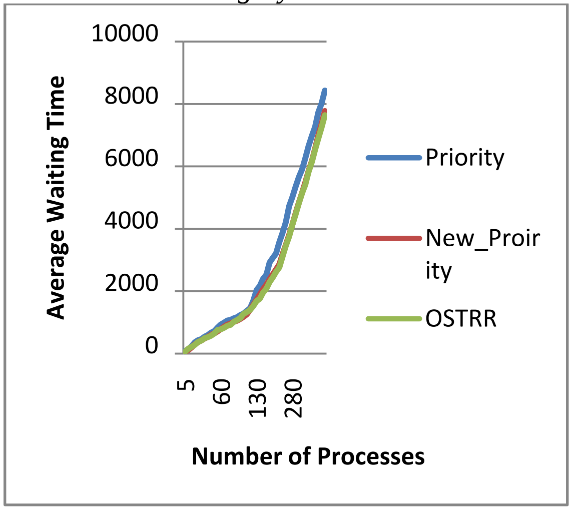

Figure 11: Graph of Average Waiting Time using Uniform distribution

Figure 11shows the overall graphical result of the Average Waiting Time for all cases of values taken. It was observed from the graph that OSTRR gives the least average waiting time while the proposed algorithm gives a better average waiting time than the traditional algorithm.

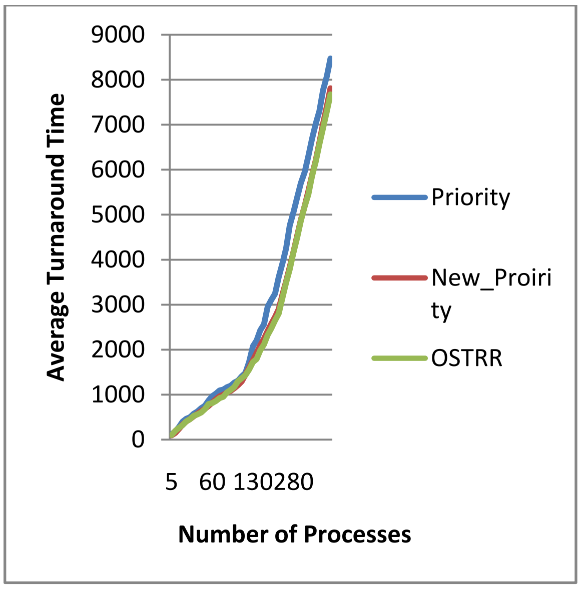

Figure 12: Graph of Average Turnaround Time using Uniform distribution

Figure 12 shows the overall graphical result of the Average Turnaround Time for all cases of values taken. It was observed from the graph that OSTRR gives the least average turnaround time while the proposed algorithm gives a better average turnaround time than the traditional algorithm.

Figure 13 shows the overall graphical result of the Average Response Time for all cases of values taken. It was observed from the graph that OSTRR gives the best average response time followed by the proposed algorithm and lastly the traditional algorithm.

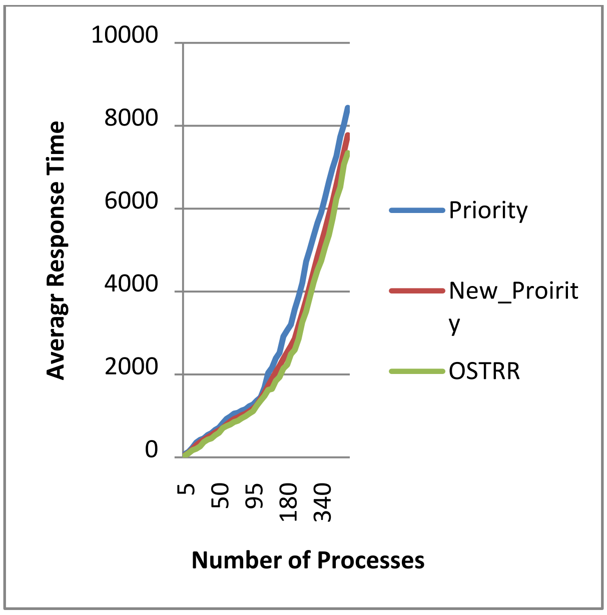

Figure 13: Graph of Average Response Time using Uniform distribution

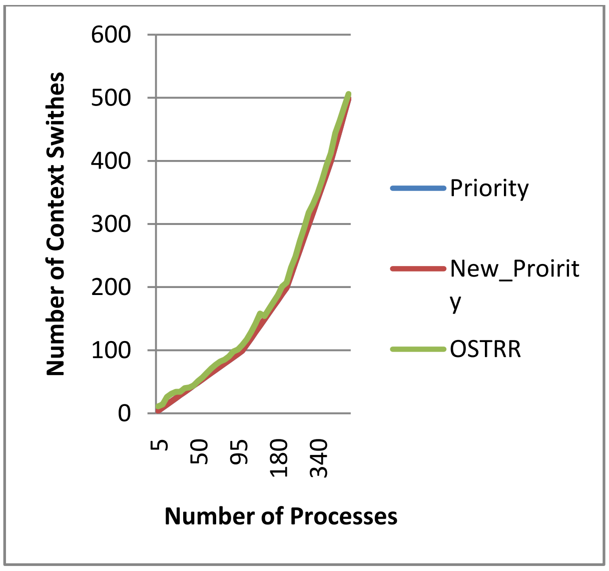

Figure 14: Graph of Number of Context Switches using Uniform distribution

Figure 14 shows the overall graphical result of the Number of Context Switches for the same processes. It was observed from the graph that the traditional algorithm and the proposed algorithm produce the same number of context switches which is better than that of OSTRR.

Figure 15 shows the overall graphical result of the Waiting Time Variance Deviation for the same processes. It was observed from the graph that the proposed algorithm is the best algorithm in terms of minimizing waiting time variance deviation followed by OSTRR and then the traditional algorithm. This shows that processes with low priorities will not starve in the proposed algorithm.

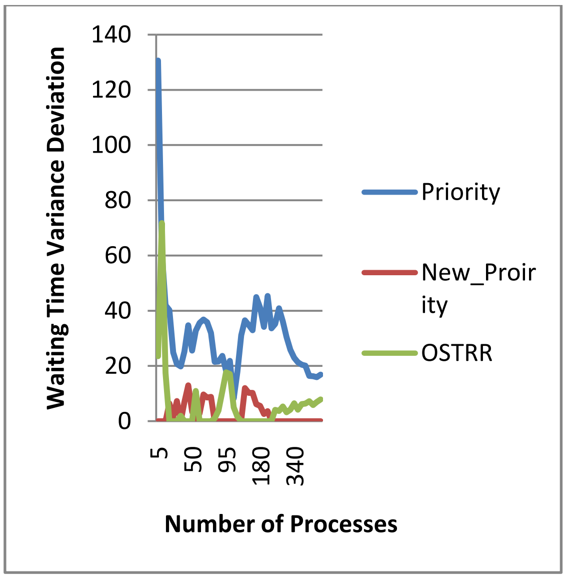

Figure 15: Graph of Waiting Time Variance Deviation using Uniform distribution

From the results of the two scheduling algorithms (Priority and OSTRR) compared with the proposed algorithm (the new Priority), it has shown that the new proposed priority produced best Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART) and Waiting Time Variance Deviation (WTVD) compared to the traditional Priority scheduling algorithm. But they both produced the same Number of Context Switches (NCS). The new proposed Priority algorithm produced better results compared with OSTRR with respect to Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD).

Table 3: Performance of Algorithms Based on Metrics Used in Comparing Them for First Category

Performance Metric	Traditional SJF (%)	OSTRR (%)
AWT	-47.86	-4.96
ATAT	-47.75	-1.96
ART	-47.86	-4.56
NCS	0	2.80
WTVD	-100	131.99

Table 4: Performance of Algorithms Based on Metrics Used in Comparing Them for Second Category

Performance Metric	Traditional Priority (%)	OSTRR (%)
AWT	7.36	-3.34
ATAT	7.34	-3.33
ART	7.36	-5.90
NCS	0	2.80
WTVD	16.69	12.30

5 CONCLUSION

This paper presents an algorithm that minimized starvation in priority based CPU scheduling algorithms. This has been done by modifying the optimum priority in Saxena and Agarwal (2012) by introducing the concept of EFFICIENCY FACTOR and applying it to the priority based CPU scheduling algorithms in order to reduce average waiting time, average turnaround time, average response time and waiting time variance deviation. The priority based CPU scheduling algorithms were grouped into two groups i.e. the shortest job first and the priority groups. In each group, the proposed algorithm with the traditional algorithm in that with OSTRR were implemented in Java and their results were compared based on Average Waiting Time (AWT), Average Turnaround Time (ATAT), Average Response Time (ART), Number of Context Switches (NCS) and Waiting Time Variance Deviation (WTVD) for different categories of processes that were generated randomly using either Exponential, Uniform or Normal distributions to generate the burst time; Uniform or Exponential distributions to generate priority and Poisson distribution to generate arrival time of processes.

The simulation results show that with 500 processes generated using Uniform distribution ranging between 1 and 100, priority using Exponential distribution time at a rate of 0.02, arrival time using Poisson distribution arriving at the rate of 3 milliseconds per process and time quantum of 10*ms* used by OSTRR.

In the SJF category, SJF and OSTRR compared with the proposed algorithm (the new SJF), it has shown that the traditional SJF produced better AWT, ATAT, ART and WTVD compared with the proposed new SJF. But they both produced the same NCS. The new proposed SJF produced better results compared with OSTRR with respect to AWT, ATAT, ART, NCS and WTVD. And in the PS category, Priority and OSTRR compared with the proposed algorithm (the new Priority), it has shown that the new proposed priority produced better AWT, ATAT, ART and WTVD compared to the traditional Priority scheduling algorithm. But they both produced the same NCS. The new proposed Priority algorithm produced better results compared with OSTRR with respect to NCS and WTVD.

The recommendation in this work is to implement the proposed priority algorithm instead of the traditional priority algorithm, so as to minimize starvation of processes. In the future, a more efficient algorithm should be developed to minimize starvation in Shortest Job First (SJF) algorithm.

6 REFERENCES

[1] Brinch, H. P. (1977). *The Architecture of Concurrent Programs.* Englewood Cli®s, NJ: Prentice Hall.

[2] *http://siber.cankaya.edu.tr/OperatingSystems/ceng328/node122.html.* (n.d.). Retrieved May 25, 2014, from http://siber.cankaya.edu.tr.

[3] *http://www.merriamwebster.com/dictionary/variance.* (n.d.). Retrieved June 28, 2014, from http://www.merriam-webster.com.

[4] *https://www.it.uu.se/edu/course/.../scheduling_algorithms/handout.pdf.* (n.d.). Retrieved May 25, 2014, fromhttps://www.it.uu.se/edu/course/.../scheduling_algorithms/handout.pdf.

[5] Mehdi, N., Mehdi, S., Adel, N., And Ali, A. (2012). The New Method Of Adaptive Cpu Scheduling Using Fonseca *And* Fleming's Genetic Algorithm. *Journal*

Of Theoretical And Applied Information Technology , 1-16.

[6] Nong, Y., Xueping, L., Toni, F., and Xiaoyun, X. (*2007*). Job scheduling methods for reducing waiting time variance. *Computers and Operations Research* , 3069 - 3083.

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

[8] Rakesh, M, Manas, D, Lakshmi, M. P and *Sudhashree*. (2011). Design and Performance Evaluation of A New Proposed Fittest Job First Dynamic Round Robin (FJFDRR) Scheduling Algorithm. *International Journal of Computer Information Systems* , *2* (2), 23-27.

[9] Saxena, H. F., and Agarwal, P. S. (2012). Design and Performance Evaluation of Optimum Service Time Concept for Round Robin Algorithm. *International Journal of Machine Learning and Computing* , *2* (2), 113-117.

[10] Silberschatz, P. B. Galvin and G. Gagne. (2006). *Operating System Concepts.* (7th, Ed.) John Wiley and Sons Inc. 111 River Street, Hoboken NJ, 07030, 153-168