SLUJST
Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Modified Round Robin with Highest Response Ratio Next CPU Scheduling Algorithm using Dynamic Time Quantum

Suleiman Ebaiya Abubakar[1], Sahabi Ali Yusuf[2], Afolabi A. Obiniyi[3] and Abdullahi Mohammed[4]
[1,2,3,4]Computer Science Department, Ahmadu Bello University, Zaria-Nigeria.
mall.sule10@gmail.com[1], sahabiali@yahoo.com[2], aaobiniyi@yahoo.com[3], abdullahilwafu@abu.edu.ng[4]

Abstract

Background: *The most popular time-sharing operating systems scheduling technique, whose efficiency heavily dependent on time slice selection, is the round robin CPU scheduling algorithm. The time slice works similar to the First-Come-First-Serve (FCFS) scheduling or processor sharing algorithm if it is large or extremely too small. Some of the existing research papers have an algorithm called Improved Round Robin with Highest Response Ratio Next (IRRHRRN) which made use of response ratio with a predefined time quantum of 10ms with the major aim of avoiding starvation. However, the IRRHRRN algorithm favors processes with shorter burst time than the ones with longer burst time, and gave no regard to the process arrival time, thus leading to starvation.* ***Aim:*** *this study tries to improve on the IRRHRRN algorithm by proposing the Modified Round Robin with Highest Response Ratio Next (MRRHRRN) CPU Scheduling Algorithm using Dynamic Time Quantum in order to reduce the problem of starvation.* ***Method:*** *Dynamic method of determining the time quantum was adopted.* ***Results:*** *The proposed algorithm was compared with four other existing algorisms such as Standard Round Robin (RR), Improved Round Robin (IRR), An Additional Improvement in Round Robin (AAIRR), and the Improved Round Robin with Highest Response Ratio Next (IRRHRRN) and it provided some promising results in terms of the Average Waiting Time of 35407.6 ms, Average Turnaround Time of 36117.6 ms, Average Response Time of 10894.8 ms and Number of Context Switch of 301 for the Non-Zero Arrival Times Processes.*

Keywords: *CPU Scheduling, Round Robin, Gantt chart, Time Quantum, and Throughput.*

1. Introduction

Computer hardware is managed by the operating system (OS) which provides a platform for executing programs in an efficient manner (Haldar, 2009). As tasks are assigned to available CPUs, modern operating systems use multitasking and multiprocessing environments for its CPU scheduling. It is the process of sharing the CPU across numerous processes (Silberschatz *et al.*, 2009). Improvement of throughput and utilization of CPU is the main purpose of CPU scheduling while reducing waiting time, context shifts, and response time (Singh *et al.*, 2010). Starvation will occur when some processes are deprived of access to their requested resources for very long time. Thus, allowing other processes to take the control of CPU time.

Scheduling explains how processes are distributed to use Central Processing Units (CPU) that are available (Galvin, 2004). Each process goes through a certain number of phases before being assigned to a Processor. A process began at arrival time which gets into the ready queue based on some scheduling algorithms and waits for the Job scheduler and dispatcher to decide on what scheduling algorithm to use.
In Round Robin algorithm processes execution depends on the selection of time slice. Should the time slice be very bulky or incredibly too small the algorithm will be the same as the FCFS scheduling or as Processor sharing algorithm for which there will be a large number of context shifts. (Saeidi & Hakimeh, 2012; Soraj & Roy, 2011). The efficiency of the algorithm will be impacted by the outputs that each value will produce in terms of waiting time, turnaround time, response time, and the frequency of context switches, throughput, and CPU utilization. (Abdulrahim *et al.*, 2014). The quantum time provision in Round Robin algorithm is done in two different ways:

SLUJST
Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

1. Static Time Quantum: - This is the method in which the time quantum to be used by the processes is predefined usually in 10-100*ms* range. Here the time quantum is provided without putting into consideration the process burst time or the processes properties to which it will be applied.
2. Dynamic Time Quantum: - This is the method in which the time quantum to be used on processes is determined by the properties of the processes on which it will be used. Properties like process burst time, arrival time, etc. The aim of dynamic determination of time slice is to improve the performance of Round Robin Scheduling algorithm.

CPU Scheduling is an integral function of the OS wherein processes are given CPU at certain time intervals. This time allocation to processes usually cause starvation on the part of some processes. Many studies were carried out to mitigate this problem in order to improve the CPU operation. However, some of these studies have achieved some considerable improvement in addressing starvation while leaving some aspect of it. For instance, the work of Baba Abu (2016) tries to reduce/mitigate the problem of starvation by introducing the principle of Highest Response Ratio Next (HRRN) in order to eliminate problem for which the shorter jobs gaining more priority than longer jobs which could lead to longer jobs being starved. Even though the principle seems promising, the proposed algorithm is still not fair enough to the jobs with higher burst times, as the calculated ratio gave more priority to the processes with smaller burst times. The algorithm is more of Shortest Jobs First Scheduling Algorithm because the shorter burst time determines the requested ratio and does not give regard to the arrival time, thus leading to the starvation problem. Another drawback of the algorithm is the use of static time quantum which elongate overall execution time by increasing the context switch and reducing the throughput.

This paper intends to overcome the problems associated with Baba'Abu (2016) algorithm by modifying the algorithm using dynamic time quantum. In this proposed algorithm each process will be allow access to the CPU based on the currently determine time slide in order of their arrival time

2. Related Works

The proposed algorithm in (Srinivasu *et al.*, 2015) enhanced the Improved Round Robin Algorithms by dynamically determining the quantum time with the available processes burst times in the ready queue. In this algorithm, different time quantum was determined using the average sum of the maximum and the minimum burst times (i.e., $Tq = \frac{Min(Bt)+Max(Bt)}{2}$) for which the processes were sorted in order of their ascending, descending and random orders while the arrival time were zero in each case. Comparison was made between the basic RR and the proposed algorithm. The results of the simulation showed that proposed algorithm ADRR performed better than the standard RR algorithm.

The proposed paper (Baba'Abu, 2016) adopted the principle of Highest Response Ratio Next (HRRN) method similar to Shortest Jobs Next in order to eliminate the problem of starvation. The algorithm, made used of response ratio i.e., $RR = \frac{Waiting\ Time+Burst\ Time}{Burst\ Time}$ in selecting process to be executed with a static time quantum of 10ms for every executing process. When each process under goes its first execution the system checks if its outstanding burst time is lower than or same as 10ms, then the active process is allowed to finish and terminate and the next process will be selected for execution. One of the problems of this algorithm is starving of the processes with longer burst time. Another problem is that, the algorithm is not fair in its selection process because it gave no regard to the processes arrival time and favor processes with shorter burst time.

The algorithm proposed by (Mody & Mirkar, 2019) known as Smart Round Robin CPU Scheduling algorithm focuses on the fundamental requirements for CPU Scheduling. The Smart Time Quantum (STQ) and Delta were two components of the approach used to dynamically assign CPU to the processes. The time quantum is calculated as the total sum of STQ and Delta, where STQ represents the typical difference between adjacent processes' burst times in the queue and Delta is equals to (STQ)/2. When a cycle began, the processes were sorted in the queue in order of their outstanding burst time, with the time quantum at the beginning believed to be predetermined or constant. For each cycle, a fresh Smart Time Quantum (STQ) and Delta are computed. Any process whose remaining burst time is less than the

Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Time Quantum (TQ) can be allow to run until it is finished and terminate. Instead of allowing processes with small burst time to be waiting for an entire round of execution to gain access to CPU time, this technique will enable smaller burst times processes complete their execution.

The proposed RRDTQ algorithm by (Sohrawordi *et al*., 2019) advocated the use of dynamic time quantum in CPU scheduling by taken the integer average value of remaining waiting processes the burst times in the ready queue as time quantum. The time quantum is recalculated as the integer average value of the remaining processes' burst times in the queue after each round of execution, and the remaining processes are sorted in order of their increasing burst times. When processes finished running, they are removed from the ready queue, and if not, they are place behind the list of the remaining processes in the ready queue.

The proposed technique by (Simon *et al*, 2022) is an improvement of Half Life Variable Quantum Time RR CPU scheduling technique. In this work, quantum time are allotted to processes in the ready queue using variable quantum, with the assumption that all processes are already waiting in the queue to be given access to CPU for execution. In order to determine if the result is less than, equal to, or more than the processes burst times, it first calculates the average of the available processes' burst times in the ready queue. In the event that it is less than or equal to, the QT will represent the average burst time for all of the tasks in the queue. Otherwise, the QT will be half of each process burst time. This technique is used to make sure that each of the processes execution is limited to two rounds. This system has two QT which are TQ1 and TQ2: TQ1 is the average burst of the tasks in the ready queue and TQ2 is half of each process burst time.

3. Materials and Methodology

The proposed MRRHRRN CPU scheduling technique modifies Baba'Abu (2016) Improved Round Robin with Highest Response Ratio Next (IRRHRRN) CPU Scheduling Algorithm by dynamically determining the time quantum to use in executing processes in the ready queue.

3.1 Implementation Phases

For the purpose of achieving the goal the proposed algorithm is divided into two phases such as:

3.1.1 For the 1st Phase

The system checks if the available processes arrival times are zero (i.e., $AT_i = 0$). If yes, then the system chooses the process that got into the queue first and allots the CPU to it for one TQ for the initial round of execution. It then calculates the TQ using Eq. (1) below for the available processes burst time in the ready queue. Any active task whose outstanding burst time is lower than or equal to one TQ during the execution will be allowed to finish and terminate, while the one whose remaining burst time is higher than one TQ should be place at the ready queue tail. The system will always check the ready queue for the remaining processes, recalculate new TQ and assign next process and/or reassign currently running process for the next round until the queue is empty. When the ready queue is empty and $BT_i = 0$ then, the system calculates and display the average waiting time, average turnaround time, and average response time as well as number of context switches.

$$TQ = \frac{\sum(Mean(Bt_i)+HRRN(Bt_i))}{2} \quad (1)$$

$$Mean(Bt) = \frac{\sum_{i=1}^{n}(Bt_i)}{N} \quad (2)$$

$$RR = \frac{Waiting\ time+Burst\ time}{Burst\ time} \quad (3)$$

Both Eq. (2) and Eq. (3) are components of Eq. (1) where Eq. (2) is the formula for mean of the processes burst times and Eq. (3) is the formula for response ratio (RR).

Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

3.1.2 For the 2nd Phase

The system checks if the available processes arrival time is non-zero (i.e., AT_i *!= 0*). The algorithm picks from the ready queue, process that first arrived the queue, and then allot CPU to run for its burst time (i.e., bt_1). Processes that move into the queue while the CPU is working on the first process will be added to the ready queue's tail in sequential order. The algorithm will check for the processes that arrived within the execution time of the first process, the next TQ will be determined using Eq. (1) for the burst time of the processes whose arrival time is less than or equal to their execution time. The next process in the queue will be assign CPU for execution. Any active process with a burst time remaining during execution that is less than or equal to one TQ will be permitted to finish and terminate, whereas any process with a burst time remaining greater than one TQ will be placed behind in the ready queue. The system will always check ready queue for the remaining processes, recalculate the new TQ and assign next process and/or reassign currently running process until arrival time of the processes is less than the execution time (i.e., $AT_i<ET_i$). When $AT_i<ET_i$, the new TQ will be recalculated using the available burst times of the remaining processes in queue and every task will be executed until the queue is empty. When the ready queue is empty and *BTi = 0* then, the system calculates and display the average waiting time, average turnaround time, and average response time as well as the number of context switches.

Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

3.2 Proposed Algorithm (MRRHRRN) Flowchart

The proposed algorithm (MRRHRRN) flowchart is shown in figure 1 below.

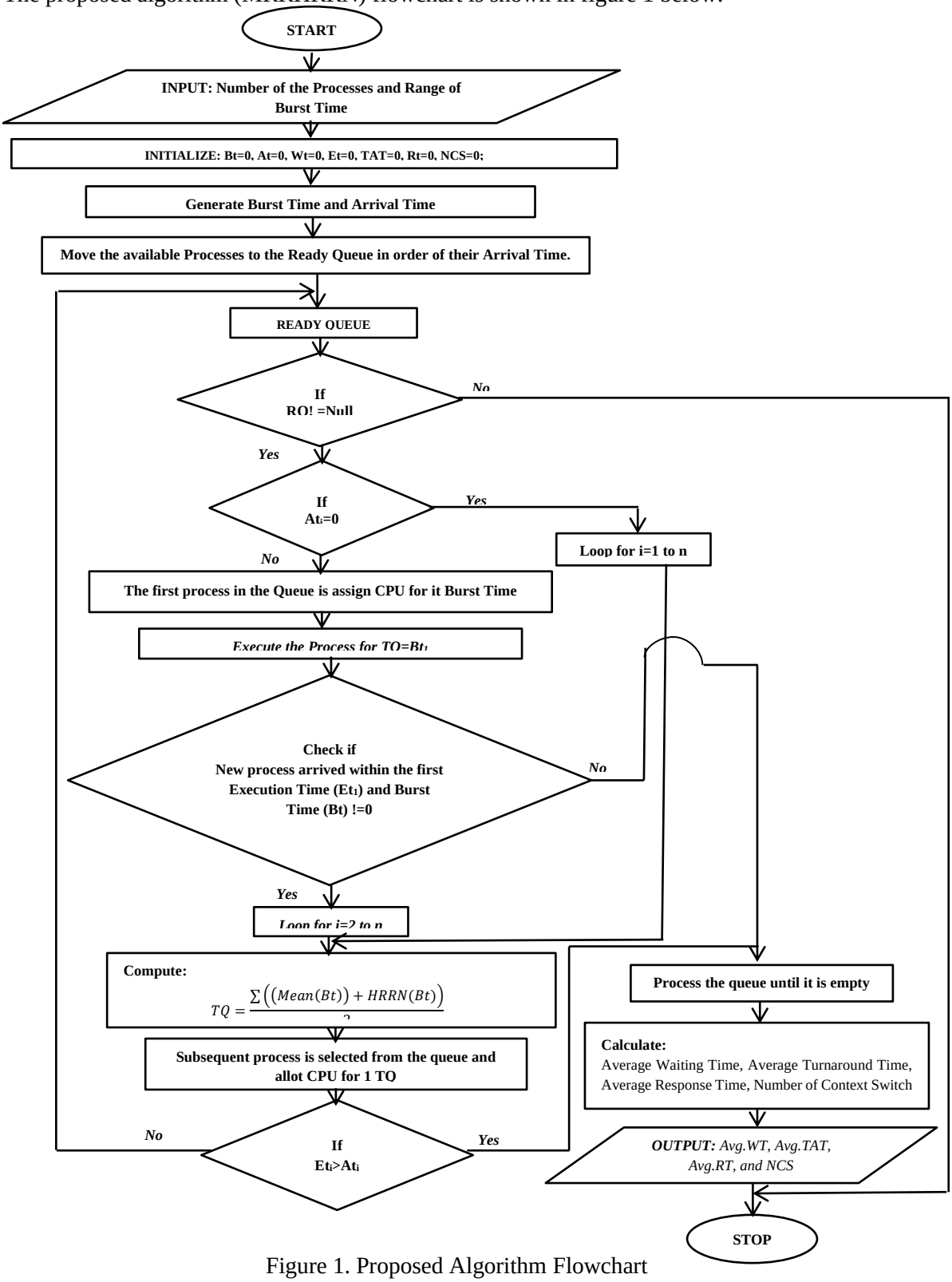


Figure 1. Proposed Algorithm Flowchart

Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

3.3 Interfaces

For the implementation of our proposed algorithm, process generator interfaces were constructed for generating set of Processes. In this sense, the processes are represented in a form of tuple: < (Process_*Id*, *Arrival_Time*, *Burst_Time*) >. The process arrival time was modeled in both Zero & Non-zero arrival times. The inter-arrival time for non-zero case were generated using Poisson distribution. To handle the random entrance of various processes into the system, a process arrival time generator class was created. The generator uses a certain mean (arrival intensity) of the distribution function to generate the inter-arrival times.

Let *F*(x) be a Poisson process with a mean $\propto = 0$. So, using the Inverse Transformation method:

$$F(x) = \sum_{i=0}^{x} \frac{e^{-\propto} \propto^{x}}{x!} \quad (4)$$

The Poisson distribution's significant characteristic of having an equal mean and variance to α, that is;

$$E(X) = \propto = V(X) \quad (5)$$

The Poison distribution was used to generate the arrival time for the non-zero AT. The development of a process arrival time generator is to manage the systems numerous processes' unpredictable arrival times.

Uniform Distribution: Consider a random variable X that is uniformly distributed on the interval (a, b). So, using Inverse Transformation method: The Probability Density Function (pdf) is given as:

$$f(x) = \frac{1}{b-a} \quad (6)$$

Where $a \leq x \leq b$

The cumulative density function (CDF) is given as:

$$F(x) = \frac{x-a}{b-a} = R \quad (7)$$

Or

$$x = a + (b-a)R \quad (8)$$

Which is a reasonable guess for generating X and R is always a random number on (0, 1) (Jerry *et al.*, 2005).

The mean and variance of uniform distribution is given as:

$$F(X) = \frac{a+b}{2} \quad (9)$$

and

$$V(X) = \frac{(b-a)^2}{12} \quad (10)$$

For the purpose of this research work, two major interfaces were designed to easy navigation around the different Java Classes in our simulation program such as Zero & Non-zero arrival times Interfaces.

3.3.1 Zero Arrival Time Interface

The user interface that the system uses to construct a series of processes is depicted in Figure 2 below. The system takes in the number of processes, time quantum and range of burst time as inputs and generate set of processes each with their burst time using uniform distribution and zero arrival time. Our proposed algorithm dynamically determines the time quantum using Eq. 1 while RR, IRR, AIRR and IRRHRRN use predefined time quantum of 10 *ms*. The interface displays text file outputs through the show output button after execution.

3.3.2 Non-Zero Arrival Time Interface

An interface that the user uses to communicate with the system is shown in Figure 3 below. With the input taken from the interface system interface, the number of processes, arrival time as well as the processes burst times are generated using Poison and Uniform distributions. The compute button enables the execution of the available processes based on the specified algorithm and the outputs are display in the text area. The show output button enables the user to view the text file output.

Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

3.3.3 Non-Zero Arrival Time Overall Interface

An interface that provides overall summary of the average waiting time, turnaround time, response time and number of context switch is shown in Figure 4 below. With the input taken from the interface, the number of processes, arrival time as well as the processes burst times are generated using Poison and Uniform distributions. The compute button enables the execution of the available processes based on the specified algorithm and the outputs are display in the provided area.

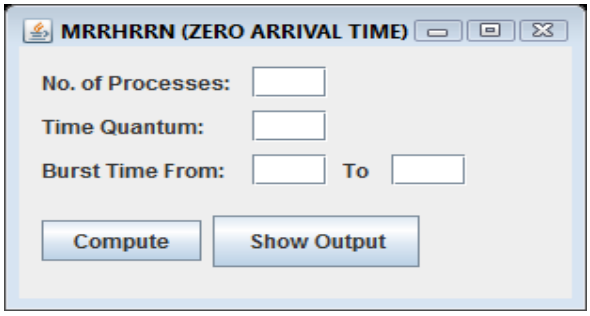


Figure 2. Zero Arrival Time Interface

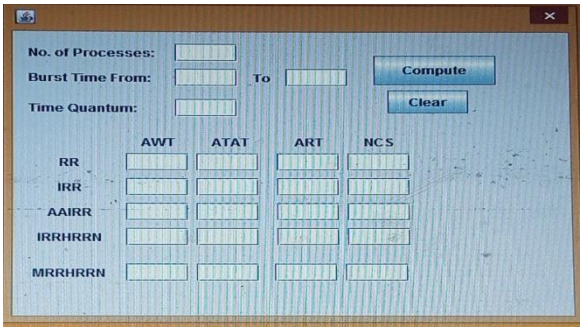


Figure 4. Summary Interface

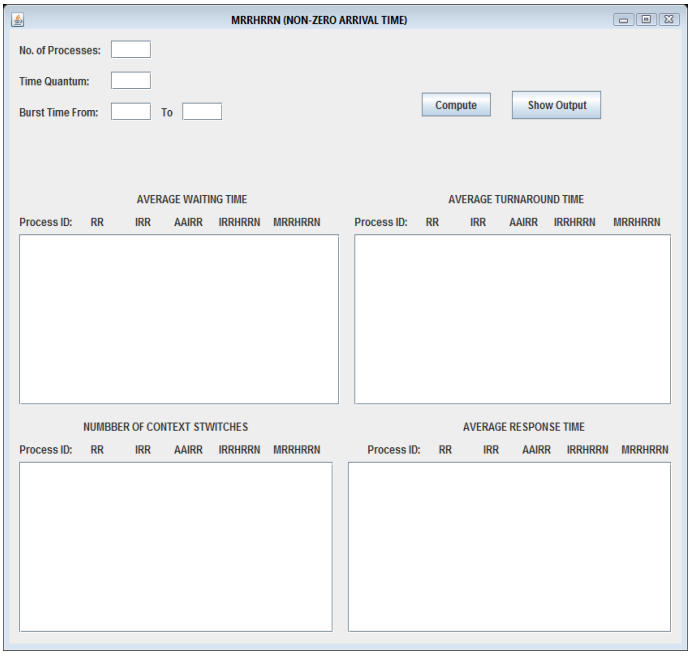


Figure 3. Non-Zero Arrival Time Interface

4. Results

The simulation results and each of their graphs for our proposed algorithm (MRRHRRN) are hereby discussed below.

4.1 For Processes with Zero Arrival Time

The correlations between the CPU scheduling strategies being investigated utilizing 150 processes and their burst time using the uniform distribution with lower and upper limits of 1 and 60*ms*, the mean, standard deviation, and time quantum of 29.9*ms*, 17.7*ms* and 10*ms* were used by RR, IRR, AIRR and IRRHRRN algorithms for zero AT processes are shown in figure 4, 5, 6, and 7 below.

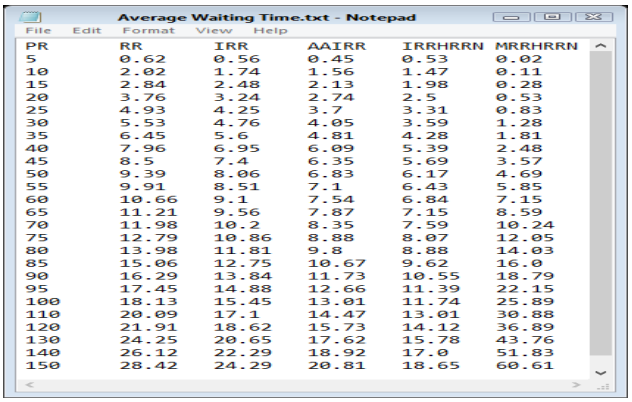
Figures 5 (A and B) represent the simulation results and graph of Average Waiting Time obtained for 150 processes generated and executed. The result indicates that our proposed algorithm MRRHRRN performed better than RR, IRR, AIRR and IRRHRRN algorithms in some instances such as: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, and 55 processes IDs as shown in the below.

In Figures 6 (A and B) represent the simulation results and graph of Average Turnaround Time obtained for 150 processes generated and executed. The result indicates that our proposed algorithm MRRHRRN performed better than RR, IRR, AIRR and IRRHRRN algorithms except in some instances such as: 130, 140 and 150 processes IDs.

Figures 7 (A and B) represent the simulation results and graph of Average Response Time obtained for 150 processes generated and executed. The results indicates that our proposed algorithm MRRHRRN performed less than RR, IRR, AIRR and IRRHRRN algorithms except in 5, 10, 15, 20 and 25 instances of processes IDs.

Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Whereas Figures 8 (A and B) represent the simulation results and graph of Number of Context Switches obtained for 150 processes generated and executed. The results show that our proposed algorithm MRRHRRN has less numbers of context switches when compare with RR, IRR, AIRR and IRRHRRN algorithms in all instances of processes IDs.



Average Waiting Time.txt - Notepad

File Edit Format View Help

PR	RR	IRR	AAIRR	IRRHRRN	MRRHRRN
5	0.62	0.56	0.45	0.53	0.02
10	2.02	1.74	1.56	1.47	0.11
15	2.84	2.48	2.13	1.98	0.28
20	3.76	3.24	2.74	2.5	0.53
25	4.93	4.25	3.7	3.31	0.83
30	5.53	4.76	4.05	3.59	1.28
35	6.45	5.6	4.81	4.28	1.81
40	7.96	6.95	6.09	5.39	2.48
45	8.5	7.4	6.35	5.69	3.57
50	9.39	8.06	6.83	6.17	4.69
55	9.91	8.51	7.1	6.43	5.85
60	10.66	9.1	7.54	6.84	7.15
65	11.21	9.56	7.87	7.15	8.59
70	11.98	10.2	8.35	7.59	10.24
75	12.79	10.86	8.88	8.07	12.05
80	13.98	11.81	9.8	8.88	14.03
85	15.06	12.75	10.67	9.62	16.0
90	16.29	13.84	11.73	10.55	18.79
95	17.45	14.88	12.66	11.39	22.15
100	18.13	15.45	13.01	11.74	25.89
110	20.09	17.1	14.47	13.01	30.88
120	21.91	18.62	15.73	14.12	36.89
130	24.25	20.65	17.62	15.78	43.76
140	26.12	22.29	18.92	17.0	51.83
150	28.42	24.29	20.81	18.65	60.61

Figure 5A: Output Results

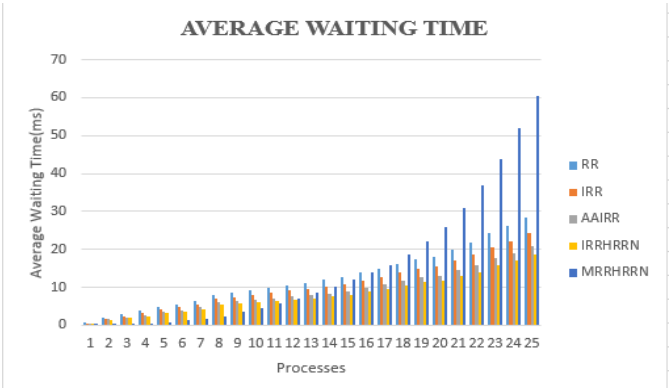
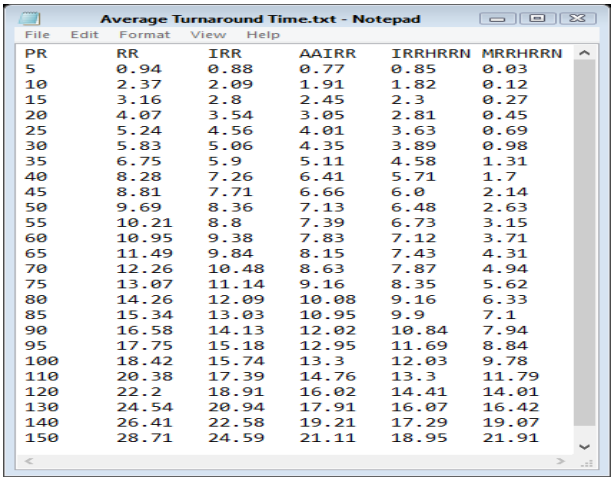


Figure 5B: Graph

Figure 5. Output Results and Graph of AWT for Zero Arrival Time Processes



Average Turnaround Time.txt - Notepad

File Edit Format View Help

PR	RR	IRR	AAIRR	IRRHRRN	MRRHRRN
5	0.94	0.88	0.77	0.85	0.03
10	2.37	2.09	1.91	1.82	0.12
15	3.16	2.8	2.45	2.3	0.27
20	4.07	3.54	3.05	2.81	0.45
25	5.24	4.56	4.01	3.63	0.69
30	5.83	5.06	4.35	3.89	0.98
35	6.75	5.9	5.11	4.58	1.31
40	8.28	7.26	6.41	5.71	1.7
45	8.81	7.71	6.66	6.0	2.14
50	9.69	8.36	7.13	6.48	2.63
55	10.21	8.8	7.39	6.73	3.15
60	10.95	9.38	7.83	7.12	3.71
65	11.49	9.84	8.15	7.43	4.31
70	12.26	10.48	8.63	7.87	4.94
75	13.07	11.14	9.16	8.35	5.62
80	14.26	12.09	10.08	9.16	6.33
85	15.34	13.03	10.95	9.9	7.1
90	16.58	14.13	12.02	10.84	7.94
95	17.75	15.18	12.95	11.69	8.84
100	18.42	15.74	13.3	12.03	9.78
110	20.38	17.39	14.76	13.3	11.79
120	22.2	18.91	16.02	14.41	14.01
130	24.54	20.94	17.91	16.07	16.42
140	26.41	22.58	19.21	17.29	19.07
150	28.71	24.59	21.11	18.95	21.91

Figure 6A: Output Results

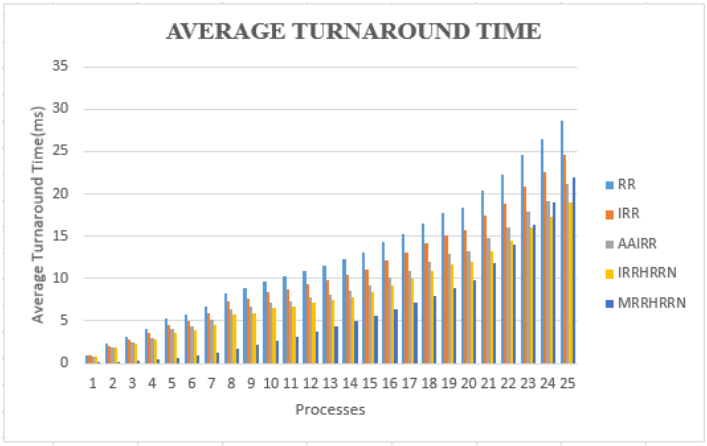


Figure 6B: Graph

Figure 6. Output Results and Graph of ATAT for Zero Arrival Time

Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Average Response Time.txt - Notepad

File Edit Format View Help

PR	RR	IRR	AAIRR	IRRHRRN	MRRHRRN
5	0.18	0.18	0.17	0.17	0.02
10	0.41	0.41	0.48	0.41	0.1
15	0.65	0.68	0.65	0.48	0.24
20	0.88	0.92	0.93	0.56	0.41
25	1.1	1.15	1.14	0.81	0.64
30	1.32	1.37	1.25	0.88	0.92
35	1.53	1.59	1.64	1.07	1.24
40	1.75	1.81	2.14	1.39	1.62
45	1.97	2.04	2.0	1.39	2.05
50	2.19	2.27	2.73	1.44	2.53
55	2.41	2.51	2.63	1.5	3.05
60	2.63	2.76	2.98	1.59	3.6
65	2.85	3.0	3.26	1.67	4.19
70	3.07	3.25	3.8	1.78	4.81
75	3.3	3.49	3.53	1.9	5.48
80	3.52	3.74	4.48	2.15	6.18
85	3.74	3.99	4.32	2.39	6.94
90	3.97	4.24	4.73	2.69	7.76
95	4.2	4.48	5.2	2.93	8.66
100	4.43	4.73	4.88	3.0	9.59
110	4.88	5.22	5.5	3.34	11.58
120	5.34	5.71	5.54	3.63	13.78
130	5.79	6.18	6.3	4.04	16.17
140	6.24	6.66	6.87	4.31	18.8
150	6.68	7.14	7.13	4.85	21.62

Figure 7A: Output Results

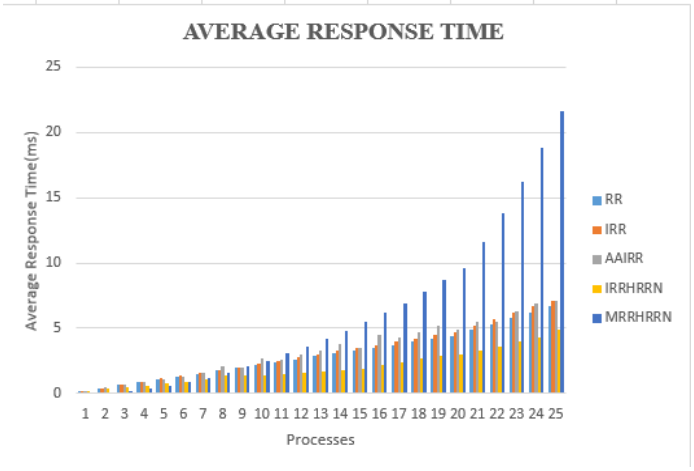


Figure 7B: Graph

Figure 7. Output Results and Graph of ART for Zero Arrival Time

Number of Context Switches.txt - Notepad

File Edit Format View Help

PR	RR	IRR	AAIRR	IRRHRRN	MRRHRRN
5	17.0	14.0	9.0	14.0	4.0
10	38.0	30.0	30.0	30.0	9.0
15	53.0	42.0	42.0	42.0	14.0
20	68.0	53.0	53.0	53.0	19.0
25	88.0	69.0	69.0	69.0	24.0
30	102.0	80.0	80.0	80.0	29.0
35	120.0	95.0	95.0	95.0	34.0
40	144.0	115.0	114.0	114.0	39.0
45	156.0	124.0	123.0	123.0	44.0
50	170.0	133.0	132.0	132.0	49.0
55	182.0	143.0	141.0	141.0	54.0
60	196.0	153.0	151.0	151.0	59.0
65	209.0	163.0	161.0	161.0	64.0
70	223.0	174.0	171.0	171.0	69.0
75	238.0	185.0	182.0	182.0	74.0
80	258.0	200.0	197.0	197.0	79.0
85	277.0	216.0	212.0	212.0	84.0
90	299.0	235.0	230.0	230.0	89.0
95	318.0	251.0	245.0	245.0	94.0
100	332.0	262.0	255.0	255.0	99.0
110	367.0	289.0	282.0	282.0	109.0
120	400.0	314.0	307.0	307.0	119.0
130	439.0	345.0	338.0	338.0	129.0
140	471.0	371.0	362.0	362.0	139.0
150	514.0	407.0	397.0	397.0	149.0

Figure 8A: Output Results

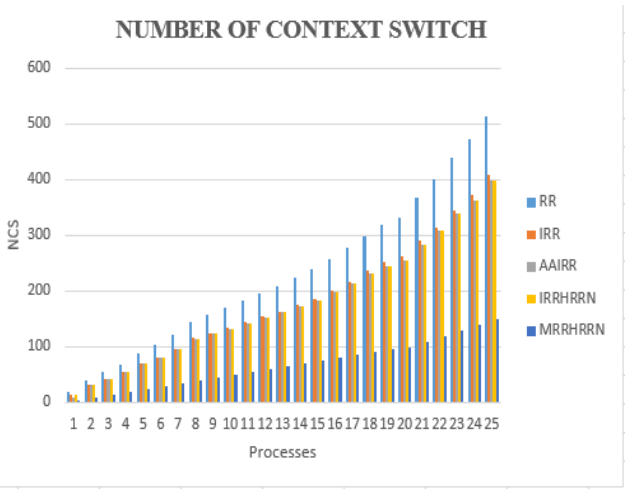


Figure 8A: Graph

Figure 8. Output Results and Graph of NCS for Zero Arrival Time Processes

4.2 For Processes with Non-Zero Arrival Time

The correlations between the CPU scheduling strategies being investigated utilizing 150 processes and their arrival and burst times using poison and uniform distributions with lower and upper limits of 1 and 60*ms*, the mean, standard deviation, and the time quantum of 32.1*ms*, 17.0*ms* and 10*ms* were used by RR, IRR, AIRR and IRRHRRN algorithms for non-zero AT are shown in figure 8, 9, 10, and 11 below.

Figures 9 (A and B) represent the simulation results and graph of Average Waiting Time obtained for 150 processes generated and executed. The output results indicate that our proposed algorithm MRRHRRN performed much better than RR, IRR, AIRR and IRRHRRN algorithms in all instances of the processes IDs except in 140 and 150 processes IDs where IRRHRRN performed better than our algorithm.

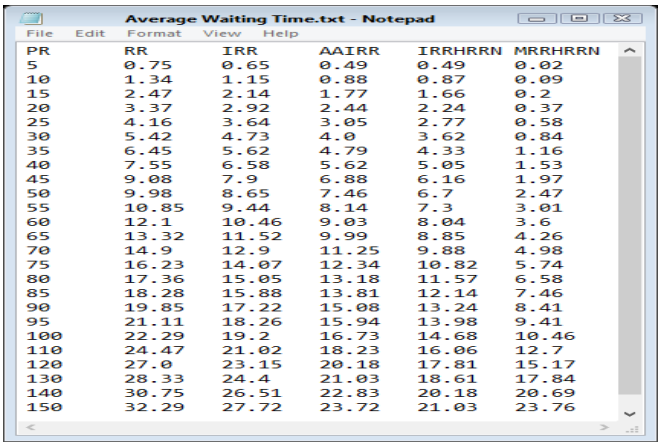
Figures 10 (A and B) represent the simulation results and graph of Average Turnaround Time obtained for 150 processes generated. The output results indicate that our proposed algorithm performed much better than RR, IRR, AIRR and IRRHRRN algorithms in terms of response time in all instances of the processes IDs except in 140 and 150 processes IDs where IRRHRRN performed better than our algorithm.

Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Figures 11 (A and B) represent the simulation results and graph of Average Response Time obtained. The output results indicate that our proposed algorithm MRRHRRN performed less than RR, IRR, AIRR and IRRHRRN algorithms in all instances except in some instances of 5, 10,15,20,25, and 30 processes IDs.

Figures 12 (A and B) represent the simulation results and graph of Number of Context Switches obtained for 150 processes generated. The output results indicate that our proposed algorithm performed much better than RR, IRR, AIRR and IRRHRRN algorithms in all instances of the processes IDs.

Figures 13 shows the overall simulation results obtained for 150 processes generated for the Non-Zero Arrival Times processes. The output results indicate that our proposed algorithm performed much better than RR, IRR, AIRR and IRRHRRN algorithms with Average Waiting Time of 35407.6 ms, Average Turnaround Time of 36117.6 ms, Average Response Time of 10894.8 ms and Number of Context Switch of 301.



PR	RR	IRR	AAIRR	IRRHRRN	MRRHRRN
5	0.75	0.65	0.49	0.49	0.02
10	1.34	1.15	0.88	0.87	0.09
15	2.47	2.14	1.77	1.66	0.2
20	3.37	2.92	2.44	2.24	0.37
25	4.16	3.64	3.05	2.77	0.58
30	5.42	4.73	4.0	3.62	0.84
35	6.45	5.62	4.79	4.33	1.16
40	7.55	6.58	5.62	5.05	1.53
45	9.08	7.9	6.88	6.16	1.97
50	9.98	8.65	7.46	6.7	2.47
55	10.85	9.44	8.14	7.3	3.01
60	12.1	10.46	9.03	8.04	3.6
65	13.32	11.52	9.99	8.85	4.26
70	14.9	12.9	11.25	9.88	4.98
75	16.23	14.07	12.34	10.82	5.74
80	17.36	15.05	13.18	11.57	6.58
85	18.28	15.88	13.81	12.14	7.46
90	19.85	17.22	15.08	13.24	8.41
95	21.11	18.26	15.94	13.98	9.41
100	22.29	19.2	16.73	14.68	10.46
110	24.47	21.02	18.23	16.06	12.7
120	27.0	23.15	20.18	17.81	15.17
130	28.33	24.4	21.03	18.61	17.84
140	30.75	26.51	22.83	20.18	20.69
150	32.29	27.72	23.72	21.03	23.76

Figure 9A: Output Results

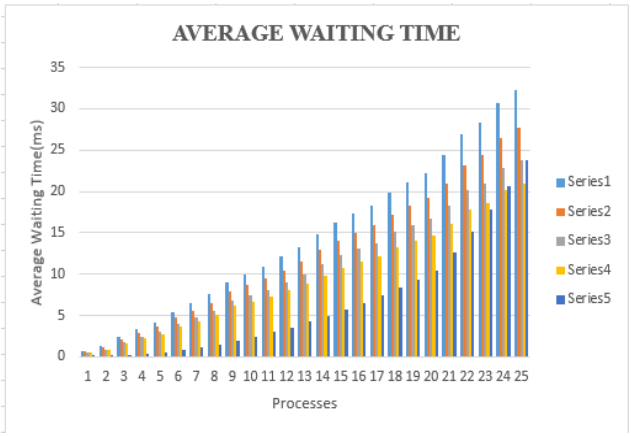
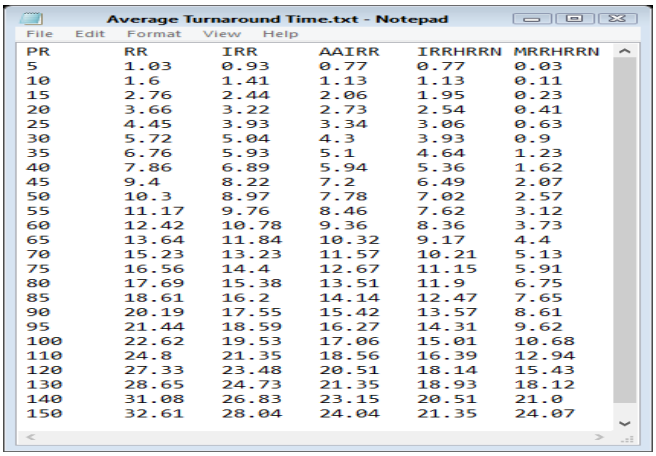


Figure 9B: Graph

Figure 9. Output Results and Graph of AWT for Zero Arrival Time Processes



PR	RR	IRR	AAIRR	IRRHRRN	MRRHRRN
5	1.03	0.93	0.77	0.77	0.03
10	1.6	1.41	1.13	1.13	0.11
15	2.76	2.44	2.06	1.95	0.23
20	3.66	3.22	2.73	2.54	0.41
25	4.45	3.93	3.34	3.06	0.63
30	5.72	5.04	4.3	3.93	0.9
35	6.76	5.93	5.1	4.64	1.23
40	7.86	6.89	5.94	5.36	1.62
45	9.4	8.22	7.2	6.49	2.07
50	10.3	8.97	7.78	7.02	2.57
55	11.17	9.76	8.46	7.62	3.12
60	12.42	10.78	9.36	8.36	3.73
65	13.64	11.84	10.32	9.17	4.4
70	15.23	13.23	11.57	10.21	5.13
75	16.56	14.4	12.67	11.15	5.91
80	17.69	15.38	13.51	11.9	6.75
85	18.61	16.2	14.14	12.47	7.65
90	20.19	17.55	15.42	13.57	8.61
95	21.44	18.59	16.27	14.31	9.62
100	22.62	19.53	17.06	15.01	10.68
110	24.8	21.35	18.56	16.39	12.94
120	27.33	23.48	20.51	18.14	15.43
130	28.65	24.73	21.35	18.93	18.12
140	31.08	26.83	23.15	20.51	21.0
150	32.61	28.04	24.04	21.35	24.07

Figure 10A: Output Results

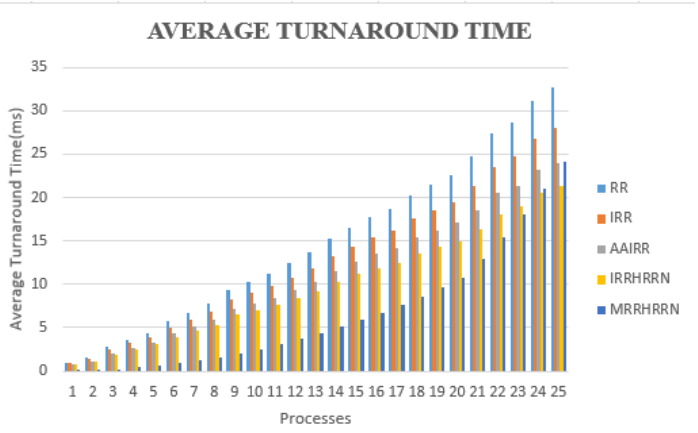
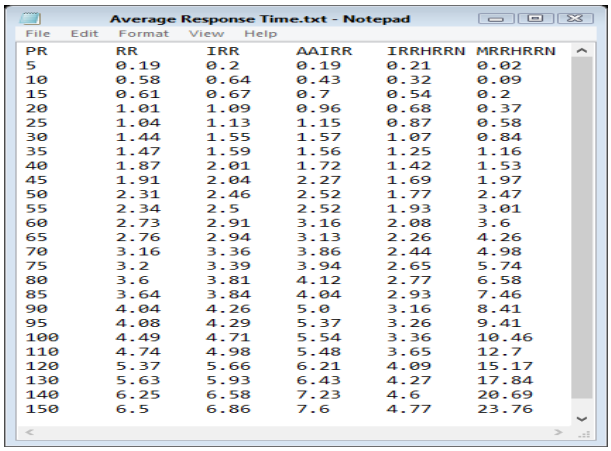


Figure 10B: Graph

Figure 10. Output Results and Graph of ATAT for Zero Arrival Time

Sule Lamido University Journal of Science & Technology Vol. 6 No. 1&2 [March, 2023], pp. 87-99
https://doi.org/10.56471/slujst.v6i.363
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng


Average Response Time.txt - Notepad

File Edit Format View Help

PR	RR	IRR	AAIRR	IRRHRRN	MRRHRRN
5	0.19	0.2	0.19	0.21	0.02
10	0.58	0.64	0.43	0.32	0.09
15	0.61	0.67	0.7	0.54	0.2
20	1.01	1.09	0.96	0.68	0.37
25	1.04	1.13	1.15	0.87	0.58
30	1.44	1.55	1.57	1.07	0.84
35	1.47	1.59	1.56	1.25	1.16
40	1.87	2.01	1.72	1.42	1.53
45	1.91	2.04	2.27	1.69	1.97
50	2.31	2.46	2.52	1.77	2.47
55	2.34	2.5	2.52	1.93	3.01
60	2.73	2.91	3.16	2.08	3.6
65	2.76	2.94	3.13	2.26	4.26
70	3.16	3.36	3.86	2.44	4.98
75	3.2	3.39	3.94	2.65	5.74
80	3.6	3.81	4.12	2.77	6.58
85	3.64	3.84	4.04	2.93	7.46
90	4.04	4.26	5.0	3.16	8.41
95	4.08	4.29	5.37	3.26	9.41
100	4.49	4.71	5.54	3.36	10.46
110	4.74	4.98	5.48	3.65	12.7
120	5.37	5.66	6.21	4.09	15.17
130	5.63	5.93	6.43	4.27	17.84
140	6.25	6.58	7.23	4.6	20.69
150	6.5	6.86	7.6	4.77	23.76

Figure 11A: Output Results

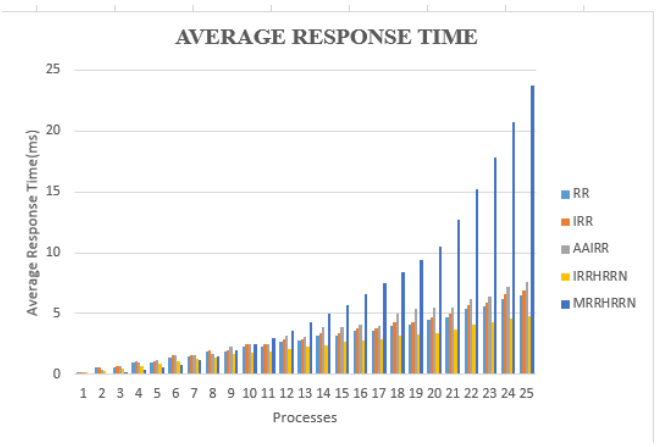
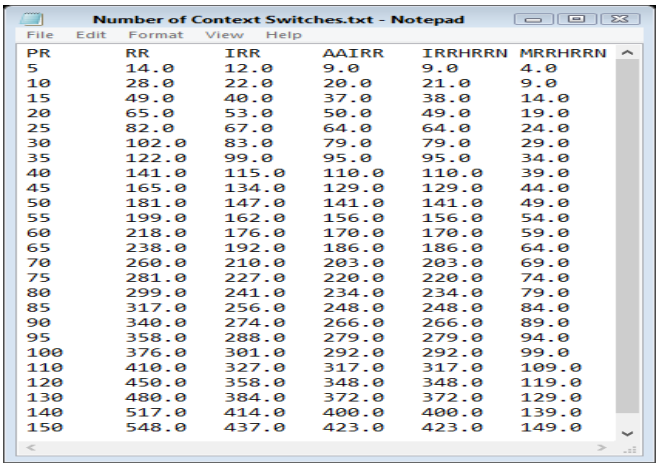


Figure 11B: Graph

Figure 11. Output Results and Graph of ART for Zero Arrival Time


Number of Context Switches.txt - Notepad

File Edit Format View Help

PR	RR	IRR	AAIRR	IRRHRRN	MRRHRRN
5	14.0	12.0	9.0	9.0	4.0
10	28.0	22.0	20.0	21.0	9.0
15	49.0	40.0	37.0	38.0	14.0
20	65.0	53.0	50.0	49.0	19.0
25	82.0	67.0	64.0	64.0	24.0
30	102.0	83.0	79.0	79.0	29.0
35	122.0	99.0	95.0	95.0	34.0
40	141.0	115.0	110.0	110.0	39.0
45	165.0	134.0	129.0	129.0	44.0
50	181.0	147.0	141.0	141.0	49.0
55	199.0	162.0	156.0	156.0	54.0
60	218.0	176.0	170.0	170.0	59.0
65	238.0	192.0	186.0	186.0	64.0
70	260.0	210.0	203.0	203.0	69.0
75	281.0	227.0	220.0	220.0	74.0
80	299.0	241.0	234.0	234.0	79.0
85	317.0	256.0	248.0	248.0	84.0
90	340.0	274.0	266.0	266.0	89.0
95	358.0	288.0	279.0	279.0	94.0
100	376.0	301.0	292.0	292.0	99.0
110	410.0	327.0	317.0	317.0	109.0
120	450.0	358.0	348.0	348.0	119.0
130	480.0	384.0	372.0	372.0	129.0
140	517.0	414.0	400.0	400.0	139.0
150	548.0	437.0	423.0	423.0	149.0

Figure 12A: Output Results

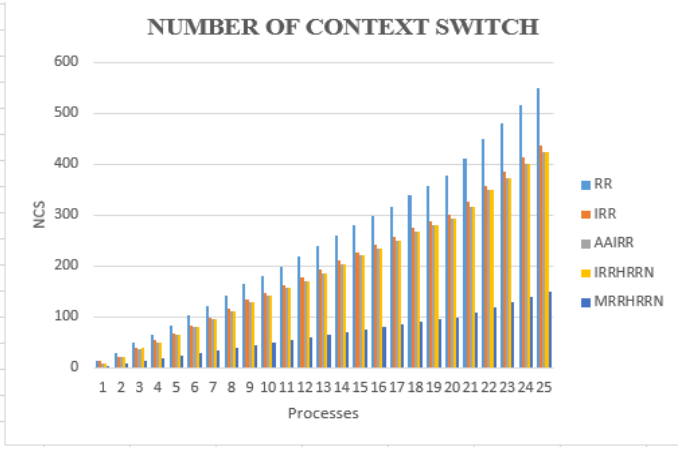


Figure 12A: Graph

Figure 12. Output Results and Graph of NCS for Zero Arrival Time Processes

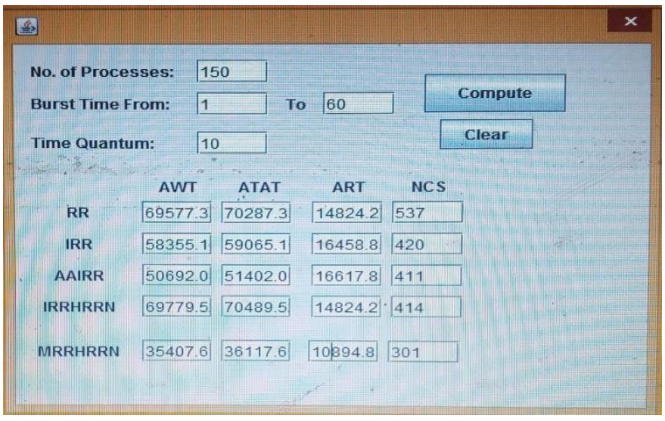


Figure 13. Overall Outputs for Non-Zero Arrival Time Processes

5. Summary of Findings

In first aspect of the implementation results, it shows that our proposed scheduling algorithm (MRRHRRN) performed much better in some instances of processes ID in term of reducing the average waiting time, average

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

turnaround time, and average response time as well as the context switches value as shown in Figures 5, 6, 7, and 8 respectively. This is to say that there is a reduction in the problem of starvation experience by longer burst time processes as encountered in RR, IRR, AIRR, IRRHRRN CPU scheduling algorithms.

In the second part of simulation, it indicates that the MRRHRRN algorithm performed better than RR, IRR, AIRR, and IRRHRRN CPU scheduling techniques with regard to average waiting time, average turnaround time, and average response time and the number of context switches as in Figures 9, 10, 11, and 12 above. This also show high reduction in the starvation problem experienced by higher burst time processes and also fairness in the selection process when assigning processes to the CPU for execution.

More also, Figure 13 above shows the overall summary of the Non-Zero Arrival Time processes generated. It shows that our proposed algorithm show much improvement in term of Average Waiting Time of 35407.6 ms, Average Turnaround Time of 36117.6 ms, Average Response Time of 10894.8 ms and Number of Context Switch of 301 compare to other algorithm such as: RR algorithm with Average Waiting Time of 69577.3ms, Average Turnaround Time of 70287.3ms, Average Response Time of 14824.2ms and Number of Context Switch of 537; IRR algorithm with Average Waiting Time of 58355.1ms, Average Turnaround Time of 59065.1ms, Average Response Time of 16458.8ms and Number of Context Switch of 420; AARR algorithm with Average Waiting Time of 50692.0 ms, Average Turnaround Time of 51402.0 ms, Average Response Time of 16617.8 ms and Number of Context Switch of 411;and IRRHRRN algorithm with Average Waiting Time of 69779.5ms, Average Turnaround Time of 70489.5ms, Average Response Time as 14824.2ms and Number of Context Switch of 414.

6. Conclusion

The CPU is one of a computer's most important components. CPU scheduling is the intelligent study of waiting processes to decide how best to respond to requests. There have been numerous CPU scheduling methods proposed, each having pros and cons. Based on the drawbacks of the existing algorithms, this study uses a dynamic time slice to reduce starvation problem among processes.

This research work is a modification of IRRHRRN and its goal is to increase the performance and reduce starvation problem experience in the Round Robin CPU scheduling technique by belittling the average response time, average waiting time, number of context switches, and average turnaround time. This algorithm (MRRHRRN) together with the simple RR, IRR, AIRR and IRRHRRN CPU scheduling algorithms were simulated using Java Programming Language and the outcomes were evaluated based on AWT, ATAT, ART and NCS for different categories of processes whose arrival time and burst time were generated randomly (i.e., Poison distribution for arrival time and uniform distribution for burst time).

It has also shown in Figure 13 above that MRRHRRN algorithm show much improvement in term of Average Waiting Time of 35407.6 ms, Average Turnaround Time of 36117.6 ms, Average Response Time of 10894.8 ms and Number of Context Switch of 301 for non-zero arrival times processes when compare to RR algorithm with Average Waiting Time of 69577.3ms, Average Turnaround Time of 70287.3ms, Average Response Time of 14824.2ms and Number of Context Switch of 537; IRR algorithm with Average Waiting Time of 58355.1ms, Average Turnaround Time of 59065.1ms, Average Response Time of 16458.8ms and Number of Context Switch of 420; AARR algorithm with Average Waiting Time of 50692.0 ms, Average Turnaround Time of 51402.0 ms, Average Response Time of 16617.8 ms and Number of Context Switch of 411;and IRRHRRN algorithm with Average Waiting Time of 69779.5ms, Average Turnaround Time of 70489.5ms, Average Response Time of 14824.2ms and Number of Context Switch of 414 respectively. In future, in order to increase the system's performance, this proposed technique should be implemented in systems that support Round Robin CPU scheduling algorithms.

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

References

Abdulrahim, A., Aliyu, S., Mustapha, A. M. and Abdullahi, S. E. (2014). An Additional Improvement in Round Robin (AAIRR) CPU Scheduling Algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering Volume 4, Issue 2, February 2014 ISSN: 2277 128X*

Baba'Abu, M. A. (2016). Improved Round Robin with Highest Response Ratio Next (IRRHRRN) CPU Scheduling Algorithm. Master Degree Thesis in Mathematic Department, Faculty of Physical Science, Ahmadu Bello University, Zaria, Kaduna State.

Galvin, P. B., Abraham, S., and Gagne, G. (2004). Operating System Concepts (7th Edition). (B. Zobrist, K. Santor, & M. Lesure, Eds.) NJ, USA: John Wiley and Sons Inc.

Haldar, S. (2009): Operating Systems. Pearson Education, India.

Jerry, B, John, S.C, Barry, L.N and David K. (2005). Discrete-Event System Simulation (Fourth Ed.). Pearson Education International.

MODY, S. & MIRKAR, S. 2019. Smart Round Robin CPU Scheduling Algorithm for Operating Systems. *4th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT).*

Saeidi, S. and Hakimeh, A. B. (2012). Determining the Optimum Time Quantum Value in Round Robin Process Scheduling Method. *International Journal of Information Technology and Computer Science, Vol. **4**(10). DOI: IO.5815/ijitcs.2012.10.08*

Silberschatz, A., Galvin, P. B., and Gagne, G. (2009). Operating System Concepts, Eighth Edition. Published by: John Wiley & Sons (July 29, 2008).

Simon, A, Dams, G. L., and Danjuma, S (2022). An Improved Half Life Variable Quantum Time with Mean Time Slice Round Robin CPU Scheduling (IMHLVQTRR). *Science World Journal Vol. 17(No 2) 2022. ISSN: 1597-6343 (Online), ISSN: 2756-391X (Print). Published by Faculty of Science, Kaduna State University. www.scienceworldjournal.org*

Singh, A., Goyal, P., and Batra, S. (2010). An Optimized Round Robin Scheduling Algorithm for CPU Scheduling. *International Journal on Computer Science and Engineering (IJCSE) Vol. 02, No. 07, 2010, 2383-23.*

SOHRAWORDI, M., ALI, U. A. M. E., UDDIN, M. P. & HOSSAIN, M. M. 2019. A Modified Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum. *International Journal of Advanced Research,* 7**,** pp. 422-429.

Srinivasu, N., Balakrishna, A. S. V., and Durgalakshmi, R. (2015). An Augmented Dynamic Round Robin CPU Scheduling Algorithm. *Journal of Theoretical and Applied Information Technology, Vol. 76, No 1, pp. 119-124.*