Computer Science and Engineering 2015, 5(3): 59-65
DOI: 10.5923/j.computer.20150503.02

Weighted Pre-Emptive Modified Highest Response Ratio Next

G. A. Shidali*, S. B. Junaidu, S. E. Abdullahi

Department of Mathematics, Ahmadu Bello University, Zaria, Nigeria

Abstract Preemptive Modified Highest Response Ratio Next (PMHRRN) scheduling algorithm is a preemptive discipline in which the hybrid priority of each process determines which job is executed next. The hybrid priority of a job is obtained by giving equal weight to both its external and internal priority. The internal priority of a job is dependent on its response ratio and the amount of time it has spent waiting while the external priority is assigned by the system using any given algorithm. PMHRRN also prevents indefinite postponements but uses equal weights for both priorities. In this research, PMHRRN has been modified in such a way that the weights of the internal and external priority of jobs are staggered to favour either of the priorities. Hence, a weighted preemptive modified HRRN (WPMHRRN) algorithm has been developed.

Keywords Starvation, Priority, Preemption, CPU scheduling

1. Introduction

Process scheduling is a fundamental function of an operating system [1]. The main concept is to share computer resources among a number of processes. Almost each computer resource is scheduled before use [2]. The Central Processing Unit (CPU) is a primary computer resource. Process scheduling is important because it plays an important role in effective resource utilization and the overall performance of the system. CPU scheduling decisions may take place when a process:

1. Switches from running state to waiting state (for example as the result of an I/O request or an invocation of wait for the termination of one of the child processes).
2. Switches from running to ready state (for example when an interrupt occurs).
3. Switches from waiting state to ready state (for example at completion of an I/O).
4. Terminates [1]

Figure 1 shows the different process states. Scheduling under 1 and 4 is non-preemptive. All other scheduling is preemptive [3]. In other words, a scheduling discipline is preemptive if, once a process has been given the CPU; the CPU can be taken away. The strategy of allowing processes that are logically run-able to be temporarily suspended is called preemptive scheduling and it is contrast to the "run to completion" method. On the other hand, a scheduling discipline is non-preemptive if, once a process has been given the CPU; the CPU cannot be taken away from that process. In general preemptive scheduling algorithms are preferred due to their abilities to switch the CPU to another process even when the current running process is not completed [4].

The aim of process scheduling is to assign processes to be executed by the processor in order to meet system objectives such as response time, throughput and processor efficiency [5].

1.1. CPU Utilization

This is the percentage of time that the processor is busy [5]. The CPU should be as busy as possible [1].

1.2. Waiting Time

This is considered to be the sum of the periods a process spends waiting in the ready queue [1]. The goal is to minimize the waiting time [6].

1.3. Response Time

This is the time it takes for a process to start responding [1]. In an interactive system, the response time should be minimized and users maximized [5].

1.4. Throughput

This is the number of processes that are completed per unit time [6]. The number of jobs processed per unit time should be maximized [7].

1.5. Turnaround Time

This is the interval between the time of submission of a

* Corresponding author:
gloriashidali@gmail.com (G. A. Shidali)
Published online at http://journal.sapub.org/computer
Copyright © 2015 Scientific & Academic Publishing. All Rights Reserved

process and the time of completion of that process [5]. The turnaround time should be minimized.

In this research, a framework to evaluate the effect of staggering the weights of the internal and external priorities on the performance of a hybrid scheduling algorithm, Preemptive Modified Highest Response Ratio Next (PMHRRN) in terms of waiting time, turnaround time and response time of processes is proposed.

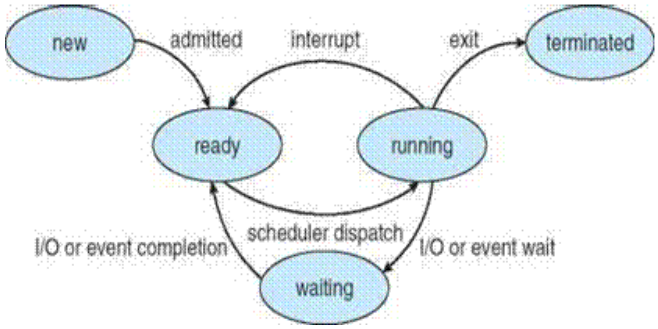

Figure 1. Process state diagram [1]

2. Related Work

2.1. First-Come-First-Served (FCFS)

FCFS is the simplest scheduling policy. With this policy, the process at the head of the ready queue, which has waited there for the longest time, is selected for execution [8].

2.2. Priority Scheduling

Each process is assigned a priority, and the process with the highest priority is allowed to run first [9]. Priority scheduling can be either preemptive or non-preemptive [1].

There are three possible ways of assigning priorities to processes. They are as follows;

1. Statically or externally: Priority is assigned by some external system manager before process is scheduled.
2. Dynamically or internally: Priority is assigned according to an algorithm.
3. Hybrid: Priority is assigned by some combination of external and internal schemes [4].

2.3. Highest Response Ratio Next

The response ratio of each process in the process pool, R is computed as:

$$R = (w + s)/s \quad (1)$$

Where w is the waiting time of a process and s is the expected service time.

The process with the greatest response ratio is executed next [8].

2.4. Highest Response Round Ratio Next (HRRRN)

Jobs with the highest response ratios which have not been executed completely are executed next. They are executed in RR manner. The quantum is determined by dividing the average burst time of processes by 1.5 [2].

2.5. Round Robin Highest Response Ratio Next (RRHRRN)

Processes with the highest response ratios are executed in RR manner. The time quantum is calculated by taking the mean of remaining burst time, RBT, of processes in the ready queue. After each round the process is repeated until the ready queue is empty [10].

2.6. Modified Highest Response Ratio Next (MHRRN)

The response ratio, R is considered as the internal priority of a process while the length of the service time of that process is considered as the external priority, E. The hybrid priority of each process is obtained by giving equal weight to both its external and internal priority. The hybrid priority, H_p, of each process is computed as follows:

$$H_p = 0.5 * R + 0.5 * E \quad (2)$$

Where R is the internal priority of each process and E is the external priority.

Processes with highest H_p are executed first [4].

2.7. Preemptive Modified Highest Response Ratio Next (PMHRRN)

Response ratio, R and hybrid priority, H_p are computed using equations (1) and (2) respectively. Processes with highest hybrid priority, H_p, are executed next. After the execution of a burst time, a running process may be preempted if there is another process with a higher hybrid priority [11].

3. Research Method

3.1. Design of Weighted Preemptive Modified Highest Response Ratio Next (PMHRRN)

For a given process with service time and arrival time, the length of service time of a process is considered as its external priority while its internal priority shall be determined by the response ratio of each process in the ready queue. The hybrid priority, H_p is computed using equation (3).

$$H_p = xR + yE \quad (3)$$

Where x and y can take on any value less than 1provided that the following condition takes place;

$$x + y = 1 \quad (4)$$

The process with highest hybrid priority, H_p, is executed next. After the execution of a burst time, a running process may be preempted if there is another process with a higher hybrid priority. If multiple processes have the highest hybrid priority, the process with the earliest arrival time among those processes is executed next which then preempts the current running process. However, if the current running process is the process with the highest hybrid priority, it

continues to run.

3.2. WPMHRRN Algorithm

1. Start
2. Processes arrive at the ready queue, RQ.
3. Processes in RQ are sorted and assigned internal and external priorities
4. Compute, H_p = xR + yE: x + y = 1
5. P_i = process with the highest value of H_p
6. P_i executes a burst time
7. While (RBT [P_i] = 0), process P_i leaves RQ. Calculate the WT, RT, and TAT of P_i
8. Is RQ = null? If yes, calculate AWT, ART, ATAT of all executed processes; go to step 9 else go to step 3.
9. Stop

3.3. Results and Discussion

Assumptions of the model are;

1. All experiments are performed in a uniprocessor environment.
2. All processes are independent of each other.
3. Attributes such as burst times and arrival times are known prior to submission of process.
4. All processes are CPU bound.

Table 1. Data set of five processes

Process	Arrival Time	Burst Time	Priority (E)
P1	0	9	1
P2	2	6	3
P3	4	5	4
P4	5	3	5
P5	8	7	2

Using the data in Table 1, the Average Turnaround Time (ATAT), Average Waiting Time (AWT) and Average Response Time (ART) are computed for a set of five processes. The range of burst time used is between 0-10.

3.3.1. PMHRRN

Using the information in Table 1, the PMHRRN algorithm is demonstrated.

At time t = 0, only process P1 is available and runs for 2 time units. Note that process P1 runs for 2 time units because it is the only process available for that time period.

At time, t = 2, processes P1 and P2 are available. Their priorities are determined as shown in Table 2 and their hybrid priority, H_p, is computed. Note that the longer the burst time of the process, the lower the value of the priority of the process. The burst time of each process is also updated (as shown in Table 2) to indicate the remaining burst time of the process which is used in computing the response ratio (internal priority) of the process.

Table 2. Process state at t=2

Process	Burst Time	Priority (E)
P1	7	1
P2	6	2

P1: R = (0+7) /7 = 1; H_p = 0.5(1) + 0.5(1) = 1
P2: R = (0+6)/6 = 1; H_p = 0.5(1) + 0.5(2) = 1.5

Process P2 has the highest hybrid priority, H_p. P1 is therefore pre-empted and P2 runs for 2 time units. Note that process P2 runs for 2 time units because after the execution of a burst time, when the hybrid priority is recomputed, it turns out to be the process with the highest value.

At time t = 4, processes P1, P2 and P3 are available. Their priorities are determined as shown in Table 3 and their hybrid priority, H_p, is computed. The burst time of each process is also updated to indicate the remaining burst time of the process which is used in computing the response ratio (internal priority) of the process.

Table 3. Process state at t=4

Process	Burst Time	Priority (E)
P1	7	1
P2	4	3
P3	5	2

P1: R= (2+7)/7 = 1.285; H_p= 0.5(1.285) + 0.5(1) = 1.143
P2: R= (0+4)/4 = 1; H_p= 0.5(1) + 0.5(3) = 2
P3: R= (0+5)/5 = 1; H_p= 0.5(1) + 0.5(2) = 1.5

Process P2 has the highest hybrid priority, H_p and so runs for 1 time unit.

At time, t = 5, processes P1, P2, P3 and P4 are available. Their priorities are determined as shown in Table 4 and their hybrid priority, H_p, is computed.

P1: R = (3+7)/7 = 1.43; H_p= 0.5(1.43) + 0.5(1) = 1.2
P2: R = (0+3)/3 = 1; H_p= 0.5(1) + 0.5(3) = 2
P3: R = (1+5)/5 = 1.2; H_p= 0.5(1.2) + 0.5(2) = 1.6
P4: R = (0+3)/3 = 1; H_p= 0.5(1) + 0.5(3) = 2

Table 4. Process state at t=5

Process	Burst Time	Priority (E)
P1	7	1
P2	3	3
P3	5	2
P4	3	3

Processes P2 and P4 have the highest hybrid priority, H_p, but process P2 runs for 3 time units because it arrived earlier than P4. It then leaves the queue. Note that process P2 runs for 3 more time units because after the execution of a burst time, when the hybrid priority is recomputed, it turns out to be the process with the highest value.

At time, t = 8, processes P1, P3, P4 and P5 are available.

Their priorities are determined as shown in Table 5 and their hybrid priority, H_p, is computed.

Table 5. Process state at t=8

Process	Burst Time	Priority (E)
P1	7	1
P3	5	2
P4	3	3
P5	7	1

P1: R = (6+7)/7 = 1.86; H_p= 0.5(1.86) + 0.5(1) = 1.423
P3: R = (4+5)/5 = 1.8; H_p= 0.5(1.8) + 0.5(2) = 1.9
P4: R = (1+3)/3 = 1.333; H_p= 0.5(1.333) + 0.5(3) = 2.167
P5: R = (0+7)/7 = 1; H_p= 0.5(1) + 0.5(1) = 1

Process P4 has the highest hybrid priority, H_p, and runs for 3 time units and leaves the queue. Note that process P4 runs for 3 time units because after the execution of a burst time, when the hybrid priority is recomputed, it turns out to be the process with the highest value.

At time, t = 11, processes P1, P3 and P5 are available. Their priorities are determined as shown in Table 6 and their hybrid priority, H_p, is computed.

P1: R = (9+7)/7 = 2.29; H_p = 0.5(2.29) + 0.5(1) = 1.643
P3: R = (7+5)/5 = 2.4; H_p = 0.5(2.4) + 0.5(2) = 2.2
P5: R = (3+7)/7 = 1.429; H_p = 0.5(1.429) + 0.5(1) = 1.214

Table 6. Process state at t=11

Process	Burst Time	Priority (E)
P1	7	1
P3	5	2
P5	7	1

Process P3 has the highest hybrid priority, H_p, runs for 5 time units and leaves the queue.

At time, t = 16, processes P1 and P5 are available. Their priorities are determined as shown in Table 7 and their hybrid priority, H_p, is computed.

Table 7. Process state at t=16

Process	Burst Time	Priority (E)
P1	7	1
P5	7	1

P1: R = (14+7)/7 = 3; H_p= 0.5(3) + 0.5(1) = 2
P5: R = (8+7)/7 = 2.143; H_p= 0.5(2.143) + 0.5(1) = 1.57

Process P1 has the highest hybrid priority, H_p, runs for 7 time units and leaves the queue.

At time t=23, only process P5 is available. P5 runs for 7 time units and leaves the queue. Table 8 summarizes the result.

3.3.2. Weighted Pmhrrn

The proposed algorithm, WPMHRRN, is demonstrated below.

The same data in Table 1 is employed. For the purpose of this demonstration, the hybrid priority, H_p is computed with the weight of the internal priority, x and external priority, y given as 0.9 and 0.1 respectively. Note that x + y = 1.

At time t=0, only process P1is available. P1 runs for 1 time unit. At time t=1. Only process P1 is available still so process P1 runs for another 1 time unit.

At time t=2, processes P1 and P2 are available.

P1: R = (0+7)/7 = 1; H_p = 0.9(1) + 0.1(1) = 1
P2: R = (0+6)/6 = 1; H_p = 0.9(1) + 0.1(2) = 1.1

Process P2 has the highest hybrid priority, H_p. P1 is therefore pre-empted and P2 runs for 1 time unit.

At time t=3, P1 and P2 are available.

P1: R = (1+7)/7 = 1.143; H_p = 0.9(1.143) + 0.1(1) = 1.129
P2: R = (0+5)/5 = 1; H_p = 0.9(1) + 0.1(2) = 1.1

Process P1 has the highest hybrid priority, H_p. P2 is therefore pre-empted and P1 runs for 1 time unit.

At time t=4, P1, P2 and P3 are available.

P1: R = (1+6)/6 = 1.167; H_p = 0.9(1.167) + 0.1(1) = 1.15
P2: R = (1+5)/5 = 1.2; H_p = 0.9(1.2) + 0.1(2) = 1.28
P3: R = (0+5)/5 = 1; H_p = 0.9(1) + 0.1(2) = 1.1

Process P2 has the highest hybrid priority, H_p. P1 is therefore pre-empted and P2 runs for 1 time unit.

At time t=5, P1, P2, P3 and P4 are available.

P1: R = (2+6)/6 = 1.33; H_p = 0.9(1.33) + 0.1(1) = 1.3
P2: R = (1+4)/4 = 1.25; H_p = 0.9(1.25) + 0.1(3) = 1.425
P3: R = (1+5)/5 = 1.2; H_p = 0.9(1.2) + 0.1(2) = 1.28
P4: R = (0+3)/3 = 1; H_p = 0.9(1) + 0.1(4) = 1.3

Process P2 has the highest hybrid priority, H_p. continues to runs for another 1 time unit.

Table 8. Summary of PMHRRN result

Process	Arrival Time	Burst Time	Priority (E)	Start Time	Finish Time	TAT	WT	RT
P1	0	9	1	0, 16	2, 23	23	14	2.56
P2	2	6	3	2	8	6	0	1
P3	4	5	4	11	16	12	7	2.4
P4	5	3	5	8	11	6	3	2
P5	8	7	2	23	30	22	15	3.14
Average						13.8	7.8	2.2

At time t=6, P1, P2, P3 and P4 are available.

P1: R = (3+6)/6 = 1.5; H_p= 0.9(1.5) + 0.1(1) = 1.45
P2: R = (1+3)/3 = 1.33; H_p= 0.9(1.33) + 0.1(3) = 1.5
P3: R = (2+5)/5 = 1.4; H_p= 0.9(1.4) + 0.1(2) = 1.46
P4: R = (1+3)/3 = 1.33; H_p= 0.9(1.33) + 0.1(3) = 1.5

Processes P2 and P4 have the highest hybrid priority, H_p, but process P2 is selected to run because it arrived earlier than P4. P2 therefore runs for 1 time unit.

At time t=7, P1, P2, P3 and P4 are available.

P1: R = (4+6)/6 = 1.67; H_p= 0.9(1.67) + 0.1(1) = 1.6
P2: R = (1+2)/2 = 1.5; H_p= 0.9(1.5) + 0.1(4) = 1.75
P3: R = (3+5)/5 = 1.6; H_p= 0.9(1.6) + 0.1(2) = 1.64
P4: R = (2+3)/3 = 1.67; H_p= 0.9(1.67) + 0.1(3) = 1.8

Process P4 has the highest hybrid priority, H_p. P2 is therefore pre-empted and P4 runs for 1 time unit.

At t=8, P1, P2, P3, P4 and P5 are available.

P1: R = (5+6)/6 = 1.83; H_p= 0.9(1.83) + 0.1(2) = 1.85
P2: R = (2+2)/2 = 2; H_p= 0.9(2) + 0.1(4) = 2.2
P3: R = (4+5)/5 = 1.8; H_p= 0.9(1.8) + 0.1(3) = 1.92
P4: R = (2+2)/2 = 2; H_p= 0.9(2) + 0.1(4) = 2.2
P5: R = (0+7)/7 = 1; H_p= 0.9(1) + 0.1(1) = 1

Processes P2 and P4 have the highest hybrid priority, H_p, but process P2 is selected to run because it arrived earlier than P4. P2 therefore runs for 1 time unit.

At time t=9, P1, P2, P3, P4 and P5 are available.

P1: R = (6+6)/6 = 2; H_p= 0.9(2) + 0.1(2) = 2
P2: R = (2+1)/1 = 3; H_p= 0.9(3) + 0.1(5) = 3.5
P3: R = (5+5)/5 = 2; H_p= 0.9(2) + 0.1(3) = 2.1
P4: R = (3+2)/2 = 2.5; H_p= 0.9(2.5) + 0.1(4) = 2.65
P5: R = (1+7)/7 = 1.143; H_p= 0.9(1.143) + 0.1(1) = 1.13

Process P2 has the highest hybrid priority, H_p. continues to runs for another 1 time unit and then leaves the queue.

At time t=10, P1, P3, P4 and P5 are available.

P1: R = (7+6)/6 = 2.167; H_p= 0.9(2.167) + 0.1(2) = 2.15
P3: R = (6+5)/5 = 2.2; H_p= 0.9(2.2) + 0.1(3) = 2.28
P4: R = (4+2)/2 = 3; H_p= 0.9(3) + 0.1(4) = 3.1
P5: R = (2+7)/7 = 1.286; H_p= 0.9(1.286) + 0.1(1) = 1.257

Process P4 has the highest hybrid priority, H_p. P4 runs for 1 time unit.

At time t=11, P1, P3, P4 and P5 are available.

P1: R = (8+6)/6 = 2.33; H_p= 0.9(2.33) + 0.1(2) = 2.3
P3: R = (7+5)/5 = 2.4; H_p= 0.9(2.4) + 0.1(3) = 2.46
P4: R = (4+1)/1 = 5; H_p= 0.9(5) + 0.1(4) = 4.9
P5: R = (3+7)/7 = 1.429; H_p= 0.9(1.429) + 0.1(1) = 1.39

Process P4 has the highest hybrid priority, H_p. P4 runs for another 1 time unit and then leaves the queue.

At t=12, P1, P3 and P5 are available.

P1: R = (9+6)/6 = 2.5; H_p= 0.9(2.5) + 0.1(2) = 2.45
P3: R = (8+5)/5 = 2.6; H_p= 0.9(2.6) + 0.1(3) = 2.64
P5: R = (4+7)/7 = 1.57; H_p= 0.9(1.57) + 0.1(1) = 1.514

Process P3 has the highest hybrid priority, H_p. P3 runs for 5 time units and leaves the queue. Note that for subsequent computations after process P3 runs its first time unit, it remains the process with the highest hybrid priority. Hence it runs for 5 time units.

At time t=17, processes P1 and P5 are available.

P1: R = (14+6)/6 = 3.33; H_p= 0.9(3.33) + 0.1(2) = 3.2
P5: R = (9+7)/7 = 2.286; H_p= 0.9(2.286) + 0.1(1) = 2.157

Process P1 has the highest hybrid priority, H_p. P1 runs for 6 time units and leaves the queue. Note that for subsequent computations after process P6 runs its first time unit after process P3 leaves the queue, it remains the process with the highest hybrid priority. Hence it runs for 6 time units.

At time t=23, only process P5 is available. P5 runs for 7 time units and leaves the queue.

Figure 2 shows the comparison of parameters studied for PMHRRN and WPMHRRN (90/10) algorithms based on Table 1.

3.4. Comparing Existing Scheduling Algorithms with PMHRRN

To study the performance of WPMHRRN, a simulator was developed which produces a simulation of the scheduling algorithms discussed namely PMHRRN and WPMHRRN for a single CPU. Burst times and arrival times of processes were generated by the simulator using Poisson distribution. The simulator computes the scheduling parameters discussed. The user is also provided with the facility to check whether his or her answer is correct or not. The simulator also provides detailed results of each process as well as a summarised result of all processes run.

Figure 3 shows the summary of results obtained for a one-time run of 100 processes with burst range of 1-17. Table 10 shows a summary of results obtained for ATAT, AWT and ART for 1000 processes with a burst range of 1-1200 run 10 different times for WPMHRRN with varied weights assigned as internal and external priorities.

Table 9. Summary of WPMHRRN (90/10) result

Process	Arrival Time	Burst Time	Priority (E)	Start Time	Finish Time	TAT	WT	RT
P1	0	9	1	0, 3, 17	2, 4, 23	23	14	2.56
P2	2	6	3	2, 4, 8	3, 7, 10	8	2	1.3
P3	4	5	4	12	17	13	8	2.6
P4	5	3	5	7, 10	8, 12	7	4	2.3
P5	8	7	2	23	30	22	15	3.14
Average						14.6	8.6	2.38

Table 10. A comparison of PMHRRN and variations of PMHRRN

	Weighted PMHRRN				PMHRRN
Variations	**10/90**	**30/70**	**70/30**	**90/10**	**50/50**
ATAT	195383.8	195383.8	195384.35	195389.93	195383.85
AWT	194816.28	194816.28	194816.84	194822.41	194816.34
ART	274.475	274.475	274.482	274.504	274.476

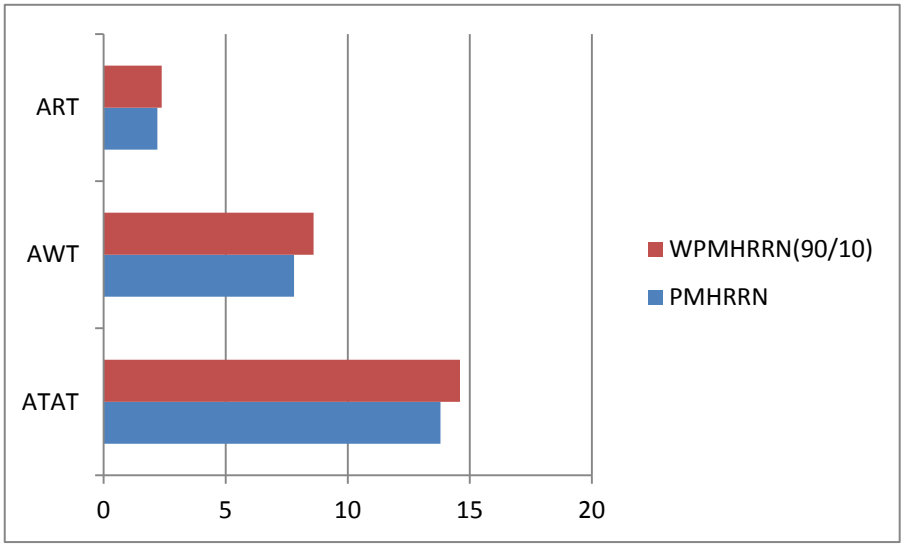

Figure 2. Comparing parameters for PMHRRN and WPMHRRN (90/10)

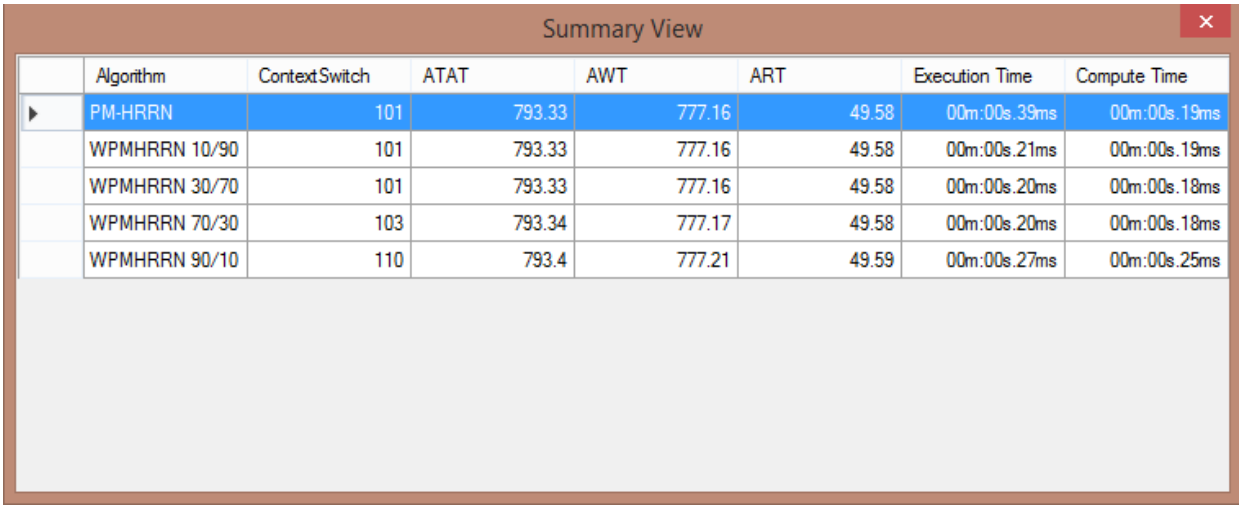

Figure 3. Screen shot of a one-time run of 100 processes with burst range of 1-17

Based on the research, the following is found in terms of performance metrics studied; an increase in internal priority causes an increase in average turnaround time, average waiting time and average response time. This implies that if the external priority is favoured over the internal priority the system will perform better than if otherwise.

4. Conclusions

The aim of the research was to study the performance of PMHRRN (a preemptive algorithm that incorporated fixed internal and external priorities to determine which process gets the CPU) when its priorities are staggered or altered.

Upon successful completion of the study, it was found that response times, waiting times and turnaround times of processes in a uniprocessor are found to be maximized with an increase in internal priority of the system. Table 9 shows a summary of results obtained from the system showing the behaviour of the processes with respect to performance criteria. Users and designers of operating systems will find that placing more weight on the external priority which in this case is the shortest job first is helpful in addressing the issue of starvation and response times especially in interactive systems.

In future, the proposed scheduling algorithm shall be applied on tasks in a multiprocessor environment and that have dependencies among one another.

REFERENCES

[1] A. Silberchatz, P. B. Galvin and G. Gagne. Operating System Concepts. 7th ed., John Wiley & Sons. Hoboken, USA. 2005.

[2] J. Patel and A. K. Solanki. "Performance evaluation of CPU scheduling by using hybrid approach," International Journal of Engineering Research & Technology (IJERT) ., 1 (4), 1-8, June 2012.

[3] J. Breecher. (2013). *web.cs.wpi.edu/~cs3013/c07/lectures.* Retrieved March 3, 2013.

[4] S. Varma. "Design of modified HRRN scheduling algorithm for priority systems using hybrid priority scheme," Journal of Telematics and Informatics (JTI), 1 (1), 14-19, 2013.

[5] W. Stallings. Operating Systems: Internals and Design Principles.7th ed., Prentice Hall, USA. 2012.

[6] E. O. Oyetunji and E. Oluleye. "Performance assessment of some CPU scheduling algorithms," Research Journal of Information Technology , 1 (1), 22-26, 2009.

[7] M. Abur, A. Mohammed, S. Danjuma, S. Abdullahi S. "A critical simulation of CPU scheduling algorithm using exponential distribution," International Journal of Computer Science Issues , 8 (6), 201-206, 2011.

[8] J. Niu. http://www.sci.brooklyn.cuny.edu/~jniu/teaching/csc 33200/files/1201-UniprocessorScheduling.pdf. Retrieved August 26, 2012.

[9] S. S. Monemi. http://www.csupomona.edu/~carich/classes/cs 499/201001/notes/os_fundamentals.pdf. Retrieved April 10, 2013.

[10] H. S. Behera, B. K. Swain, A. K. Parida, G. Sahu. (2012). "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.

[11] G. A. Shidali, S. B. Junaidu, S. E. Abdullahi. "A new hybrid process scheduling algorithm (Pre-Emptive Modified Highest Response Ratio Next)," Computer Science and Engineering, Vol. 5 No. 1, 2015, pp. 1-7.