An advanced approach to traditional round robin CPU scheduling algorithm to prioritize processes with residual burst time nearest to the specified time quantum

IOP Conf. Series: Materials Science and Engineering **263** (2017) 042001 doi:10.1088/1757-899X/263/4/042001

An advanced approach to traditional round robin CPU scheduling algorithm to prioritize processes with residual burst time nearest to the specified time quantum

Mythili.N Swaraj Pati, Pranav Korde, and Pallav Dey

SITE School, VIT University, Vellore-632014, Tamil Nadu, India
E mail: nmythili@vit.ac.in

Abstract. The purpose of this paper is to introduce an optimised variant to the round robin scheduling algorithm. Every algorithm works in its own way and has its own merits and demerits. The proposed algorithm overcomes the shortfalls of the existing scheduling algorithms in terms of waiting time, turnaround time, throughput and number of context switches. The algorithm is pre-emptive and works based on the priority of the associated processes. The priority is decided on the basis of the remaining burst time of a particular process, that is; lower the burst time, higher the priority and higher the burst time, lower the priority. To complete the execution, a time quantum is initially specified. In case if the burst time of a particular process is less than 2X of the specified time quantum but more than 1X of the specified time quantum; the process is given high priority and is allowed to execute until it completes entirely and finishes. Such processes do not have to wait for their next burst cycle.

1. Introduction

Operating system manages the work load by allocating tasks appropriately depending upon their arrival time. If resources are available, it allocates resources to the processes as and when it arrives. This can be done in multiple ways. There are numerous scheduling algorithms which have been introduced and work well for different situation based on the requirement.

Round Robin scheduling algorithm is one of the most widely used CPU scheduling algorithms which are implemented in various software systems. It is used in various operating systems, such as Windows, Linux, etc.

At a time, only one process can run in a single processor system. Any other process must wait for the resources until the CPU is idle again. The purpose of multi programming is to have multiple processes running continuously at all times. This will maximize CPU utilization and increase efficiency.

All the resources like the CPU are scheduled before allocation in order to avoid in consistencies like deadlocks. CPU scheduling is done by allocating resources to a specific process by slicing the total time. This is called as time slicing. There are various conditions based on which the CPU distributes resources to the processes.

They are:

CPU Utilization: CPU must be kept in use as much as possible. Involving maximum usage ensures that the CPU is not idle, thus minimizing resources being stagnant.

Throughput: The number of processes that are completed in one unit of time should always be high in order to improve the system's performance. The efficiency is automatically increased if there is high throughput.

Turnaround Time: The time required for a process to complete the task after it is submitted is turnaround time. This time associated with a process must be always minimal.

Waiting Time: The time that any process additionally waits for its execution has to be minimal. Less waiting time ensures faster completion of tasks.

Response Time: The time taken to produce the first response after a process has been submitted is response time. Response time has to be minimal. The response time can be optimised as per the requirement.

The proposed system works in an efficient manner. The algorithm is non – pre-emptive and works based on the priority of the associated processes. To complete the execution, a time quantum is initially specified. In case if the burst time of a particular process is less than 2X of the specified time quantum but more than 1X of the specified time quantum; the process is given a higher priority and is allowed to execute until it completes entirely. Such processes do not have to wait for their next burst cycle.

2. Problem Statement

One of the most crucial problems in operating systems concepts is Scheduling the CPU to different processes and to design a particular system that will attain accurate results in scheduling. In case of priority based Round Robin scheduling algorithm when similar priority jobs arrive, the processes are executed based on FCFS. The processes have a specific burst time associated.

A time quantum is set and the processes utilize the resources available for that time quantum only. Once the time quantum is crosses, the control is passed to the next process. In case if the burst time of a particular process is less than 2X of the specified time quantum but more than 1X of the specified time quantum; the process is given a higher priority and is allowed to execute until it completes entirely. Such processes do not have to wait for their next burst cycle. In case of long burst time processes that execute before other processes execute, remain in waiting queue for a long time. This concept of combination of different processes that wait in the queue, for the resources result in high turnaround time and the average waiting time increases.

This paper focuses on reducing the waiting time and turnaround time of the traditional algorithm. It deals with the modification of the exit criteria during execution. If the process has remaining burst time within the specified limits (i.e. the process should continue if the remaining burst time is between N and 2N), the process is executed without making it wait for any further.

3. Literature Survey

In past years, several different modifications to the different scheduling algorithms have been proposed by different authors. Some of the authors have proposed that by increasing the quantum size in the beginning in Round Robin fashion. After completing the first cycle, the time quantum is increase by 2X. Then in the further stages the time quantum is decreased and increased in alternative fashion. It does that if the processes remaining the ready queue after completing any execution cycle.

Some have also proposed that first in RR fashion the time quantum can be allocated but then all the processes will be given to the CPU on SJF algorithm. Also few checked that after allocating in RR fashion, after executing each process for one time quantum, it will see for the remaining burst time of the presently running process is less than the time quantum. If it is, then it will allocate the process the CPU for remaining burst time. Also the processes can be arranged in the ready queue increasing order of burst time and then the quantum is computed. Then, to find an optimal time quantum it will take the median of the processes in the ready queue. The time quantum is then recomputed considering the remaining burst time for each execution cycle.

3.1. Survey

Sahalu, Abdullahi and Abdulrahim proposed a system called NIRR (New improved Round Robin) which suggested some new things. It also gave us a working model implementation in java. It also told that Longest Job First so also CBT and Round Robin are the optimal solutions for reducing mean response time and rate of context switches respectively. Their working principal was that, they have two queues, when the process has arrived, we have to put it to the arrive queue and when the process is ready, it can be put in the ready queue. When the execution is going on, if a process comes, it is sent to the arrive queue. If the residual burst time is lower than or equal to half of the considered time-quantum, the CPU will be re-allocated to the process that is running at present in order to complete its residual burst time. With the results they got, the new system NIRR produces minimum NCS, ATAT

and AWT compared to IRR and RR, therefore it is used by the systems that adopt RR scheduling. They also said that more work has to be done with the residual burst time of the processes that follow different decoration of statistical distribution. This may be its drawback. The burst time should be properly managed and optimal solution should be got [1].

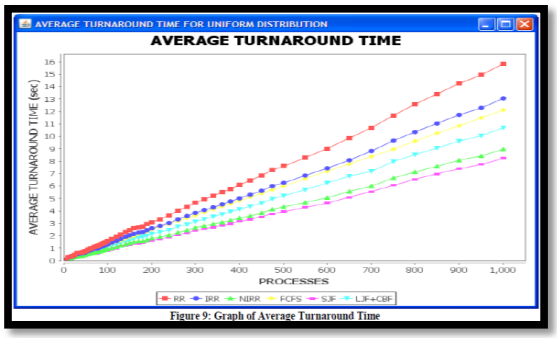


Figure 9: Graph of Average Turnaround Time

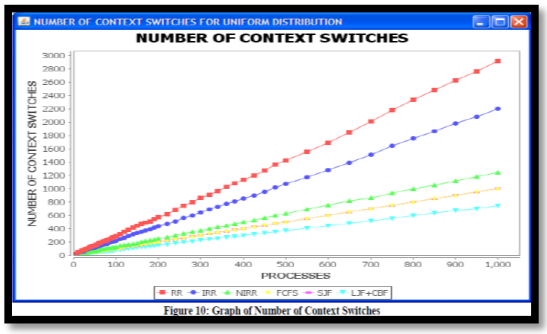


Figure 10: Graph of Number of Context Switches

Pushpraj Singh, Vinod Singh and Anjani Pandey studied several reviews of various scheduling algorithms which can be performed. The objective of the paper is to find out which algorithm is apt for different type of tasks. To find out the best algorithm for each circumstance, the WT and the TAT for each is computed and the results are compared. The SJF scheduling algorithm provides good results with all types of jobs with optimum scheduling criteria. This algorithm gives the minimum average waiting time. But also that long processes have to wait for a long time to get served and can cause starvation. This could be one of its drawbacks. Also the paper suggests that coded and implemented algorithms can give you a better understanding on how a particular algorithm can behave and fare [2].

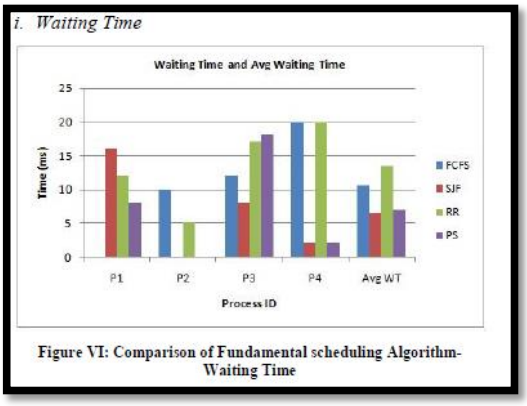


Figure VI: Comparison of Fundamental scheduling Algorithm-Waiting Time

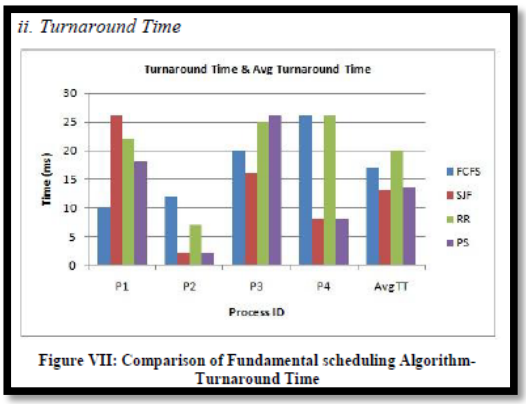


Figure VII: Comparison of Fundamental scheduling Algorithm-Turnaround Time

Rahul Joshi and Shashi Bhushan Tyagi jointly published a paper. Their new CPU scheduling algorithm is expected to perform better than simple Round Robin Algorithm, Improved Round Robin (IRR) Algorithm and An Additional Improvement Round Robin (AAIRR) Algorithm. The proposed calculation named As Smart Upgraded Round Robin (SORR) goes for enhancing AAIRR calculation. The calculation lessens the number of setting switch, holding up time and turnaround time massively when contrasted with the IRR scheduling and basic Round Robin scheduling. The proposed SRR CPU Scheduling is a little change to AAIRR CPU scheduling. It executes the briefest occupation having least blasted time first rather than FCFS amid the AAIRR, IRR and basic Round robin calculation and it likewise employments Smart time quantum rather than static time quantum. The calculation computes the Smart time quantum itself as per the burst time of all procedures as opposed to giving static quantum. This calculation wipes out the inefficiencies of actualizing straightforward round robin design. In this paper, a Smart and Advanced Round Robin CPU planning calculation has been exhibited which is a change to the An Advanced Improved Round Robin Scheduling calculation and Enhanced Round Robin CPU planning calculation [3].

Table 1: Comparative results of RR, IRR,AAIRR and SORR

Algorithm	Average Waiting Time(ms)	Average Turnaround Time (ms)	Number of Context Switch
RR	38.4	57.8	10
IRR	30.4	49.8	7
AAIRR	26.4	45.8	6
SORR	24.4	43.7	4

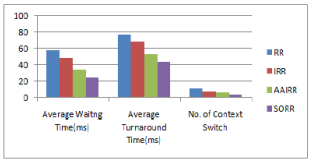


Figure 6: Bar chart of Case 1 (Increasing order)

Ishwari Singh Rajput and Deepa Gupta published a paper with a fundamental goal. The fundamental goal of this paper is to come up with another approach for RRCPU planning algorithm that improves the running of CPU progressively working system. The combination of round-robin and need planning calculation determines the said Priority based Round-Robin CPU Scheduling. It possesses the advantage of round robin in decreasing starvation and further incorporates the advantage of need booking. The said calculation likewise realises the concept of maturing by doling out new demands to procedures. Simple RR CPU booking calculation can't be actualized continuously working system because of the high switch rates, stable holding up time, expansive RT, huge TAT and, fewer throughputs. The proposed calculation enhances every one of the downsides of round robin CPU boo king calculation. The paper additionally exhibits the near examination of put forth calculation with current round robin planning calculation on premise of fluctuating time-quantum, normal holding up time, normal TAT and amount of context switches. They have effectively thought about both of the calculation which is basic RR and the said one that the proposed set up is more productive in light of the fact that it possess small normal WT, normal TAT and number of setting switches when contrasted with basic round robin, thus diminishing the working framework overhead and henceforth dispatch inertness. It decreases the issue of keeping the process waiting because the procedures with, less outstanding residual burst-time are allocated with the greater needs and are run first in the 2nd iteration of calculation. Execution of time-sharing frameworks could be enhanced with this said calculation and likewise be changed, to improve execution of on-going framework [4].

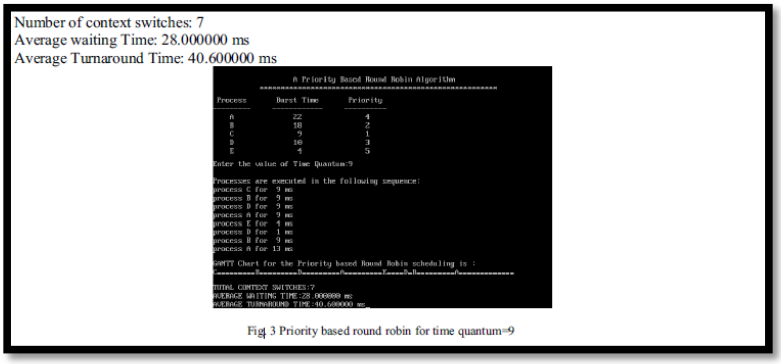
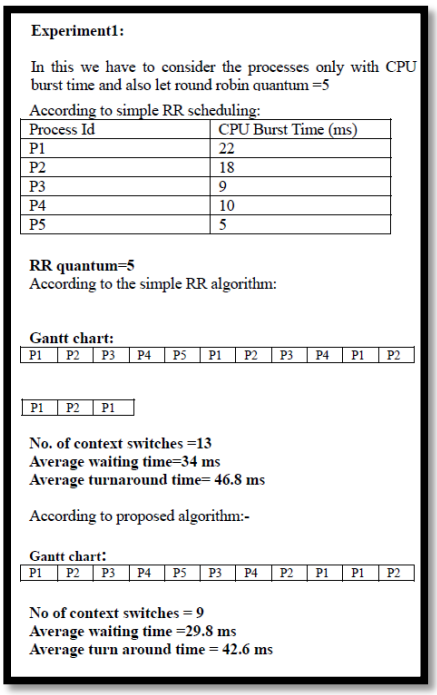


Fig 3 Priority based round robin for time quantum=9

Singh, A., Goyal, P., & Batra S. proposed and introduced an innovative model which is an adaptation to the traditional round robin algorithm. This approach aims to enhance the efficiency of the CPU in various operating system environments. There are several CPU scheduling algorithms available, but not all can prove to be good in a given scenario. Hugh context switch rates, high WT and TAT, high response time are few issues that needs to be considered. The introduction of the proposed algorithm aims to surpass all the drawbacks of the traditional round robin algorithm. A comparative analysis of the traditional Round Robin and the modified Round Robin algorithm has been performed. The authors state that the introduced algorithm overcomes all the drawbacks of the

traditional Round Robin and decreases the response time thus increasing the throughput of the system. Based on the experiment performed on a single data set, the mean WT and the mean TAT are less when the introduced algorithm is used. The traditional algorithm failed to provide efficiency. We can ensure higher CPU deployment and lower response time and proves to be better when compared to traditional Round Robin algorithm. The conclusion is that the introduced algorithm is better as it has lesser W T and response time, thereby decreasing the overhead and saving of memory and time. Future enhancements for the proposed algorithm can be implementation of the system for any heavy ongoing framework where the systems having a particular deadline require fractional outcomes to anticipate negative occasions [5].



Experiment1:

In this we have to consider the processes only with CPU burst time and also let round robin quantum =5

According to simple RR scheduling:

Process Id	CPU Burst Time (ms)
P1	22
P2	18
P3	9
P4	10
P5	5

RR quantum=5
According to the simple RR algorithm:

Gantt chart:

P1	P2	P3	P4	P5	P1	P2	P3	P4	P1	P2

P1	P2	P1

No. of context switches =13
Average waiting time=34 ms
Average turnaround time= 46.8 ms

According to proposed algorithm:-

Gantt chart:

P1	P2	P3	P4	P5	P3	P4	P2	P1	P1	P2

No of context switches = 9
Average waiting time =29.8 ms
Average turn around time = 42.6 ms

3.2. Summary Table

Sr. No.	Author	Method Name	Advantage	Disadvantage
1.	Abdulrazaq Abdulrahim, Saleh E Abdullahi and Junaidu B. Sahalu	New Improved Round Robin (NIRR) CPU Scheduling Algorithm	Minimum AWT, ATAT and NCS compared to RR and IRR	More work is needed to be done with Burst time of the processes that follow different patterns of statistical distribution
2.	Pushpraj Singh, Vinod Singh and Anjani Pandey	Analysis And Comparison Of CPU Scheduling Algorithms	The turnaround time, waiting time and response time of the processes prove to be the best for SJF compared to all other scheduling algorithms.	FCFS is suitable for batch systems alone where waiting time is large. The priority scheduling algorithm can create a problem of starvation.
3.	Rahul Joshi and Shashi Bhushan Tyagi	Smart Optimized Round Robin (SORR) CPU Scheduling Algorithm	In SORR, the time quantum is calculated dynamically using the burst time of all	A more accurate dynamic time quantum or specific time quantum needs to be determined for each process

			processes and it will find out a smart time quantum for all processes which gives good performance as compared to RR, IRR and AAIRR.	so as to improve the performance.
4.	Rajput, I. S., & Gupta, D.	A Priority Based Round Robin CPU Scheduling Algorithm For Real Time Systems	Reduces the operating system overhead and hence dispatch latency; it has less average waiting time, average turnaround time and number of context switches	It is complex to implement. Have to be tested in real time systems.
5.	Ajit Singh, Priyanka Goyal, Sahil Batra	An Optimized Round Robin Scheduling Algorithm For CPU Scheduling	Less waiting response time, usually less pre-emption and context switching thereby reducing the overhead and saving of memory space	Future work needed; implementation difficult on hard real time systems.

4. Existing System

The traditional Round-Robin is a scheduling algorithm where a time quantum is decided and time-slices are assigned to each process in equal proportions and in a circular approach. It is a simple algorithm and the development is effortless, hence is used in many applications. The above said algorithm is a concept of operating system which proves to be starvation-free and can be used to perform several scheduling tasks, such as engaging data packets in any networking scenario.

The mean WT in round robin scheduling algorithm proves to be longer than the others many a times. Each process is given a fixed time period to execute which is known time slice. Once the process has executed for the decided time quantum, it is pre-empted and the control is given to the next process for computation. This happens in a circular fashion.

5. Proposed System

Round Robin scheduling algorithm has been modified to propose and introduce a new methodology which aims in taking the efficiency of the CPU to a higher level by reducing the mean WT and the mean TAT. The CPU is allotted to the process for a specific time-quantum to perform the authorized functions.

An optimized variant to the round robin scheduling algorithm is introduced. The algorithm is pre-emptive and works based on the priority of the associated processes. The priority is decided on the basis of the residual burst time of a particular process, that is; lower the burst time, higher the priority and higher the burst time, lower the priority. To complete the computation, a time quantum is initially set.

In case if the residual burst time of a particular process is lower than 2X of the specified time quantum but more than 1X of the specified time quantum; the process is given high priority and is allowed to execute until it completes entirely and finishes. Such processes do not have to wait for their next burst cycle. This will reduce the mean WT and mean TAT of the process.

6. Architecture Of The Proposed System

In the first block, the waiting queue holds the processes that are yet to be deployed to the scheduler. Once the processes are ready to get executed, they are sent from the Waiting queue to the Ready queue for further processing. The processes move in a sequential manner until here. From the Ready queue the processes are scheduled based on the proposed algorithm conditions. The processes are sent

to the execution box which will execute the processes one by one according to the specified time quantum. This is a modified round robin scheduler which will follow the algorithm which we have proposed. If any process does not complete execution in that particular time slice, it is sent to the Ready queue. If a process competes, it is declared as Completed and removed from the ready queue.

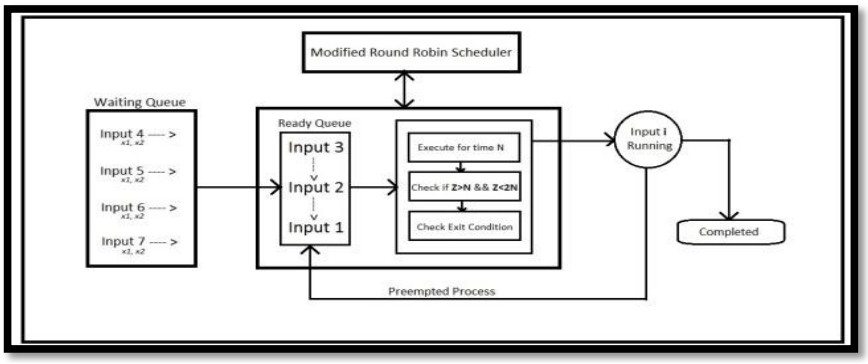


Figure 1: Architecture of Proposed System

7. Flowchart

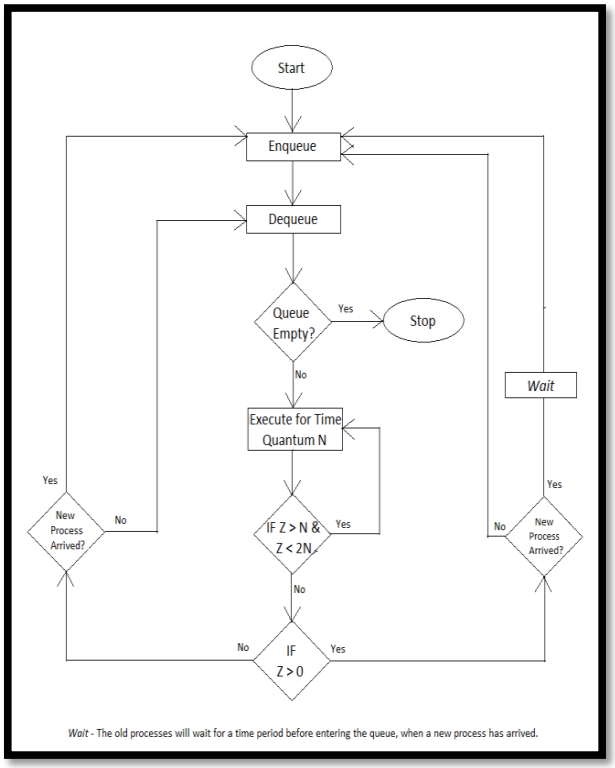


Figure 2: Flowchart of the Proposed System

Steps Explained:
Start: The process starts.
Enqueue: A process is inserted to the ready queue.
Dequeue: A process is withdrawn from ready queue and sent for execution.
Condition: If the Queue is empty, the flow terminates via Stop. If the queue is not empty, execution takes place.
Execute for Time Quantum N: The process executes for a specified time quantum that has been agreed upon initially. Condition: After Execution, there will be a conditional checking, if Z > N and Z < 2N; where Z is the Remaining Burst Time and N is the Time Quantum, the process will be re-

executed. Else it will be forwarded to the next step. Condition: If there is no remaining time of a particular process, it checks for a new process arrival. If there is one, it adds the new process to the queue, else gets itself out of the program.

Condition: On the other hand, if Z > 0, first the system checks if there is a new process arriving. If a new process arrives, it waits for the new process to enter the queue and then enters the queue (This is why W AIT is shown in the diagram). Else, if there is no new process, it adds itself to the queue.

P.S.:

- Wait denotes a time period that an old process waits for, when a new process has arrived and is yet to be put in the queue.
- Z denotes the Remaining Burst Time and N denote the Time Quantum.

8. Block Diagram

This block diagram shows that the processes are placed in a queue and removed for execution from the ready queue as and when the CPU is idle. Process B here is an active process and Process C and Process A are being executed in a circular Round Robin fashion.

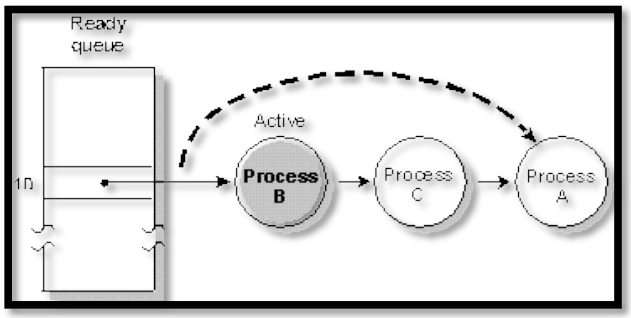


Figure 3: Block Diagram of the Proposed System

9. Result Analysis

The traditional Round Robin algorithm proves to be inefficient when compared to the modified Round Robin algorithm. A total of **100 processes** with respective Arrival Time and Burst Time have been taken for the experiment as the data set. The same data set has been supplied to both the algorithms, Traditional Round Robin as well as Modified Round Robin.

The traditional round robin gives **1053.30 ms** as the average Waiting Time and **1076.79 ms** as the average TAT. Whereas, the modified round robin gives **903.23 ms** as the average waiting time and **926.72 ms** as the average TAT. The execution time also decreases in case of the modified Round Robin. The traditional Round Robin, takes **35.19 secon**ds to complete the execution whereas the modified Round Robin takes **23.94 seconds** to complete the execution.

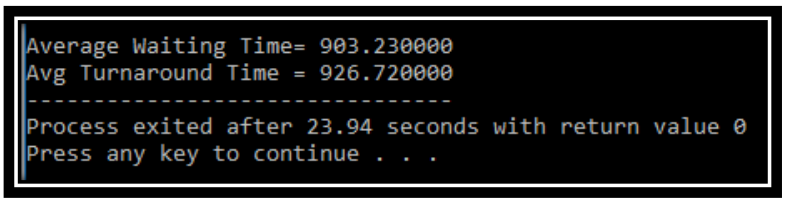
This depicts that the traditional Round Robin is not a good alternative when compared to the modified Round Robin.

Result:

Traditional Round Robin:

```
Average Waiting Time= 1053.300000
Avg Turnaround Time = 1076.790000
--------------------------------
Process exited after 35.19 seconds with return value 0
Press any key to continue . . .
```

Modified Round Robin:



10. Conclusion

The traditional Round Robin gives higher mean WT and higher the mean TAT as correlated to he modified Round Robin. The modified Round Robin proves to be better and efficient when it compares the performance aspect. The execution time also decreases in case of the modified Round Robin. This will in turn increase the efficiency of the system as the waiting time and TAT is reduced. This will be very much helpful, as the overall performance is increased through increase in throughput and decrease in response time.

11. Future Enhancements

In the proposed algorithm, the CPU utilization, throughput and response time is not considered. This algorithm is a basic enhancement to the traditional Round Robin.

In the future, the time and space complexity will also be calculated and reduced if required. The working will be enhanced by incorporating Priority based scenario along with Shortest Job First.

12. References

[1] Abdulrahim A., Abdullahi S. E. and Sahalu J. B. (2014) A new improved round robin (NIRR) CPU scheduling algorithm *International Journal of Computer Applications* **90(4)**.

[2] Singh P., Singh V. and Pandey A. (2014) Analysis and comparison of CPU scheduling algorithms *International Journal of Emerging Technology and Advanced Engineering* **4(1)** 91-95.

[3] Joshi R. and Tyagi S. B. (2015) Smart optimized round robin (SORR) CPU scheduling algorithm *International Journal of Advanced Research in Computer Science and Software Engineering* **5(7)** 568-574

[4] Rajput I. S., and Gupta D. (2012) A priority based round robin CPU scheduling algorithm for real time systems *International Journal of Innovations in Engineering and Technology* **1(3)** 1-11.

[5] Singh A., Goyal P. and Batra S. (2010) An optimized round robin scheduling algorithm for CPU scheduling *International Journal on Computer Science and Engineering* **2(07)** 2383-2385.