# A New Improved Round Robin (NIRR) CPU Scheduling Algorithm

Abdulrazaq Abdulrahim
Department of Mathematics, Ahmadu Bello University, Zaria, Nigeria

Saleh E Abdullahi
Department of Mathematics, Ahmadu Bello University, Zaria, Nigeria

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

## ABSTRACT

The Round Robin (RR) CPU scheduling algorithm is a fair scheduling algorithm that gives equal time quantum to all processes. The choice of the time quantum is critical as it affects the algorithm's performance. This paper proposes a new algorithm that further improved on the Improved Round Robin CPU (IRR) scheduling algorithm by Manish and AbdulKadir. The proposed algorithm was implemented and benchmarked against five other algorithms available in the literature. The proposed algorithm compared with the other algorithms, produces minimal average waiting time (AWT), average turnaround time (ATAT), and number of context switches (NCS). Based on these results, the proposed algorithm should be preferred over other scheduling algorithms for systems that adopt RR CPU scheduling.

## Keywords

Operating system, Scheduling algorithms, Round Robin, Time quantum, Time sharing systems, Real time systems

## 1 INTRODUCTION

Multiprogramming is one of the most important aspects of operating systems. It requires several processes to be kept simultaneously in memory, the aim of which is maximum CPU utilization. If these several processes in the memory are ready to run at the same time, the operating system must choose which one among them to run first. Making this decision is CPU scheduling. CPU scheduling is the basis of multiprogramming systems. It refers to a set of policies and mechanisms to control the order of work to be performed by a computer system. It is made by the part of the operating system called the scheduler, using a CPU scheduling algorithm [9].

## 1.1 CPU scheduling algorithms

The basic CPU scheduling algorithms are First Come First Serve (FCFS), Shortest Job First (SJF), Priority Scheduling (PS) and Round Robin (RR). The FCFS is the simplest form of CPU scheduling algorithms, which allocates CPU to the processes on the basis of their arrival to the ready queue. Arriving processes are inserted into the tail (rear) of the ready queue and the process to be executed next is removed from the head (front) of the ready queue. A long CPU-bound process may dominate the CPU and may force shorter CPU-bound processes to wait prolonged periods. In the SJF, the scheduler arranges processes according to shortest burst times in the ready queue, so that the process with least burst time is scheduled first. If two processes have equal burst times, then FCFS procedure is followed. Long running processes may wait for prolonged periods, because the CPU has a steady supply of short processes. It has been proven to be the fastest scheduling algorithm, but it suffers from one important problem: How does the scheduler know how long the next CPU burst is going to be? [7]. The PS associates each process with a priority number. The CPU is allocated to the process with the highest priority. If there are multiple processes with same priority, then FCFS will be used to allocate the CPU. Lower priority processes may starve, because the CPU may have a steady supply of higher priority processes. Round Robin (RR) is specially designed for time-sharing systems; each process gets a small unit of CPU time (time quantum). This algorithm will allow the first process in the ready queue to run until its time quantum expires, and then run the next process in the ready queue. In a situation where the process needs more time, the process runs for the full length of the time quantum and then it is preempted and then added to the tail of the queue.

## 1.2 Scheduling Criteria

The various CPU scheduling algorithms have different properties as mentioned above. The choice of a particular algorithm may favor one class of processes over another. For selection of an algorithm for a particular situation, the properties of various algorithms must be considered [4]. Many criteria have been suggested for comparing CPU scheduling algorithms. Those characteristics are used for comparison and to make a substantial difference in which algorithm is judged to be the best. The criteria include the following:

1. Context Switch: This is the process of storing and restoring context (state) of a preempted process, so that execution can be resumed from same point at a later time.
2. Throughput: This is the number of processes completed per unit time.
3. CPU Utilization: This is a measure of how much busy the CPU is.
4. Turnaround Time: This refers to the total time it takes the CPU to execute a process.
5. Waiting Time: This is the total time a process has been waiting in ready queue.
6. Response Time: It is approximately the time of submission of a process until its first access to the CPU.

So, a good scheduling algorithm should possess the following characteristics [2]:

- Minimum context switches.
- Maximum CPU utilization.
- Maximum throughput.
- Minimum turnaround time.
- Minimum waiting time.
- Minimum response time.

Due to a number of disadvantages the various CPU scheduling algorithms have, they are rarely used in timesharing and real time operating systems except for RR scheduling which is considered the most widely used CPU scheduling algorithms [2][4].

The performance of RR scheduling is sensitive to time quantum selection, because if time quantum is very large then RR will be the same as the FCFS scheduling. If the time quantum is extremely too small then RR will be the same as Processor Sharing algorithm and number of context switches will be very high. Each value of time quantum will lead to a specific performance and will affect the algorithm's efficiency by affecting the processes waiting time, turnaround time, response time and number of context switches.

In this paper, an algorithm is proposed that determines the time quantum dynamically, by taking the average of the available burst time of processes in the system. This algorithm together with FCFS, SJF, RR, IRR and LJF+CBT (Longest Job First with Combinational Burst Time) are implemented and their results were compared based on average waiting time, average turnaround time, average response time and number of context switches. Results of the analyses show that the proposed algorithm is promising as it outperforms other algorithms with respect to the average waiting time, average turnaround time and number of context switches scheduling criteria.

## 2 LITERATURE REVIEW

Various modifications to Round Robin CPU scheduling algorithm have been proposed by several authors. These modifications can be classified as follows:

### 2.1 Statically allocated time quantum

Ajit *et al* [2] proposed an algorithm that allocates the CPU to every process in RR fashion for an initial time quantum (say k units). After completing first cycle, it doubles the initial time quantum (2k units) and allocates the CPU to the processes in SJF format. It alternates the doubling and halving of the time quantum if processes remain in the ready queue after completing any execution cycle.

Ishwari and Deepa [4] proposed an algorithm that allocates the CPU to every process in RR fashion for only one time quantum. The CPU is then allocated to the remaining processes in the ready queue after completion of the execution in SJF fashion.

Manish and AbdulKadir [6] proposed an algorithm that allocates the CPU to processes in RR fashion. After executing each process for one time quantum, it checks if the remaining burst time of the currently running process is less than the time quantum. If so, it allocates the CPU to the process for the remaining burst time, else it moves the process to the tail of the ready queue.

### 2.2 Dynamically determined time quantum

Behera *et al* [3] developed an algorithm that arranges the processes in the ready queue in ascending order of burst time. Then, the time quantum is calculated. For finding an optimal time quantum, it takes the median of the processes in the ready queue. The time quantum is recalculated taking the remaining burst time into account after each execution cycle.

Lalit *et al* [5] developed an algorithm that arranges the processes in ascending order of burst time, and then calculate the time quantum for RR by taking the average of the burst times. This algorithm assumes that all processes arrive at the time t=0.

Soraj and Roy [8] presented a new algorithm that arranges the processes in ascending order of burst time, and then it chooses the smart time slice (STS), which is mainly dependant on the number of processes. It is equal to the burst time of the mid process when number of processes is odd and average of the processes burst times when the number of processes is even. This algorithm assumes that all processes arrive at the time t=0.

Abdullahi and Junaidu [1] made an improvement to the Longest Job First (LJF) CPU scheduling algorithm. It works by sorting the processes in descending order of their burst times and then it determines a threshold known as Combined Weighted Average (Cwa) which is the average of the processes. This threshold is used to categorize the processes into long and short processes. A Long process is a process with burst time greater than Cwa while a short process is one with burst time less than or equal to Cwa. New burst times are created from this categorization by merging two consecutive shorter processes until no shorter process has one to merge with or no shorter process exist in the categorization. After the merging, new queue is created by sorting the categorized and merged processes in descending order of burst times. The CPU is then allocated to the processes based on Longest Job First.

This paper presents a modification in RR CPU scheduling algorithm by modifying [6] and also determining the time quantum dynamically. Based on results of a simulation, application of this proposed algorithm in time sharing and real time systems will increase the performance of the systems by reducing average waiting time, average turnaround time and number of context switches.

## 3 THE PROPOSED ALGORITHM

The proposed CPU scheduling algorithm is a modification of the algorithm presented in [6]. It assumes another queue called the ARRIVE queue which holds processes according to their arrival times while there are other processes in the ready queue (say REQUEST) waiting for CPU allocation.

The algorithm takes to the REQUEST queue, the first process (i.e.$pr[1]$) that enters the ARRIVE queue, and allocates the CPU to it for the period of its burst time (i.e.$bt[1]$). Processes that arrive while the CPU is executing this process will be added to the ARRIVE queue according to arrival time. After execution of the process, all the processes in the ARRIVE queue will be moved to the REQUEST queue and arranged in ascending order of burst times. The algorithm takes the ceiling of the average of burst times of the processes in the REQUEST queue as the time quantum and allocates the CPU to first process in REQUEST queue for the period of the determined time quantum. When the time quantum for the process expires, the algorithm checks the remaining CPU burst time of the currently running process. If the remaining CPU burst time is less than or equal to half of the time quantum, the CPU will again be allocated to the currently running process for the remaining CPU burst time. In this case, this process will finish its execution and will be removed from the REQUEST queue. Otherwise, if the remaining CPU burst time of the currently running process is longer than half of the time quantum, the process will be moved to the ARRIVE queue. The CPU scheduler will then proceed to the next process in the REQUEST queue. During the execution of the processes in the REQUEST queue, any process that arrives the system will be placed in the ARRIVE queue. These activities continue until no process is available in the REQUEST queue.

After execution of the processes in the REQUEST queue, the transferred processes from the REQUEST queue to the ARRIVE queue in the previous execution cycle and the newly arrived processes in the ARRIVE queue will be moved to the

REQUEST queue in ascending order of burst times and a new time quantum will be calculated (i.e. the ceiling of the average of burst times of the processes). The CPU will be allocated to the processes in the REQUEST queue as usual using the newly determined time quantum. These activities continue until no process is available in the REQUEST and ARRIVE queues.

## 3.1 Pseudo-code of the proposed algorithm

```
Step 1: Start
Step 2: Create a queue, ARRIVE, where processes will be placed when they arrive the system before they are moved to the ready queue
Step 3: Create a ready queue, REQUEST
Step 4: Do
Step 5: If ($process\_Index = 1$) {
            $time\_quantum = burst\_time[1]$
        Move the first process ($pr[1]$) to REQUEST queue
            }
        Else {
        Move all processes in ARRIVE queue to REQUEST queue in ascending burst time order
            $time\_quantum = \left\lceil \frac{\sum_{i=1}^{n} burst\_time[i]}{n} \right\rceil$
        }
Step 6: Do
Step7: Allocate the CPU to the first process in REQUEST queue for a period of 1 time quantum.
Step 8: If the remaining CPU burst time of the currently running process is less than or equal to half time quantum then allocate the CPU again to the currently running process for remaining CPU burst time. After completion of execution, remove the process from the ready queue and go to step 7.
Step 9: If the remaining CPU burst time of the currently running process is longer than half time quantum, remove the process from the REQUEST queue and put it in the ARRIVE queue and go to step 7.
Step 10: If a new process arrives the system, it is placed in the ARRIVE queue.
Step 11: WHILE queue REQUEST is not empty.
Step 12: WHILE queue ARRIVE is not empty.
Step13: Calculate AWT, ATAT, ART and NCS.
Step 14: END
```

## 3.2 The Flow Chart

Figure 1 shows the flow chart of the proposed Round Robin algorithm.

## 3.3 Illustrative Example

The processes shown in Table 1 were used to demonstrate the proposed algorithm. All processes are assumed to arrive at the same time, as required by one of the benchmark algorithms [1]. The time quantum used in RR and IRR is 50*ms*.

**Table 1: Processes with their burst times**

| PROCESS ID | BURST TIME (ms) | ARRIVAL TIME (ms) |
|---|---|---|
| P1 | 23 | 0 |
| P2 | 75 | 0 |
| P3 | 93 | 0 |
| P4 | 48 | 0 |
| P5 | 2 | 0 |

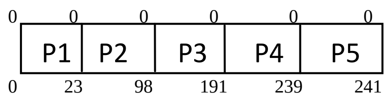

**Figure 2: The Gantt chart representation of FCFS scheduling**

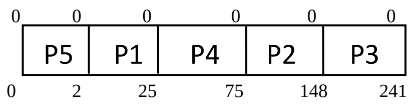

**Figure 3: The Gantt chart representation of SJF scheduling**

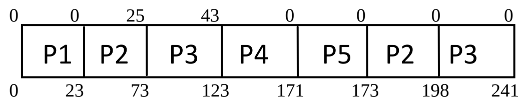

**Figure 4: The Gantt chart representation of RR scheduling with tq=50ms**

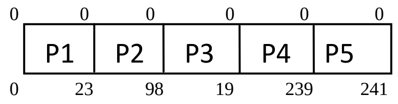

**Figure 5: The Gantt chart representation of IRR scheduling with tq=50ms**

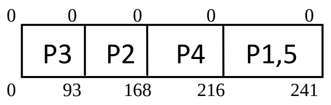

**Figure 6: The Gantt chart representation of LJF+CBT scheduling**

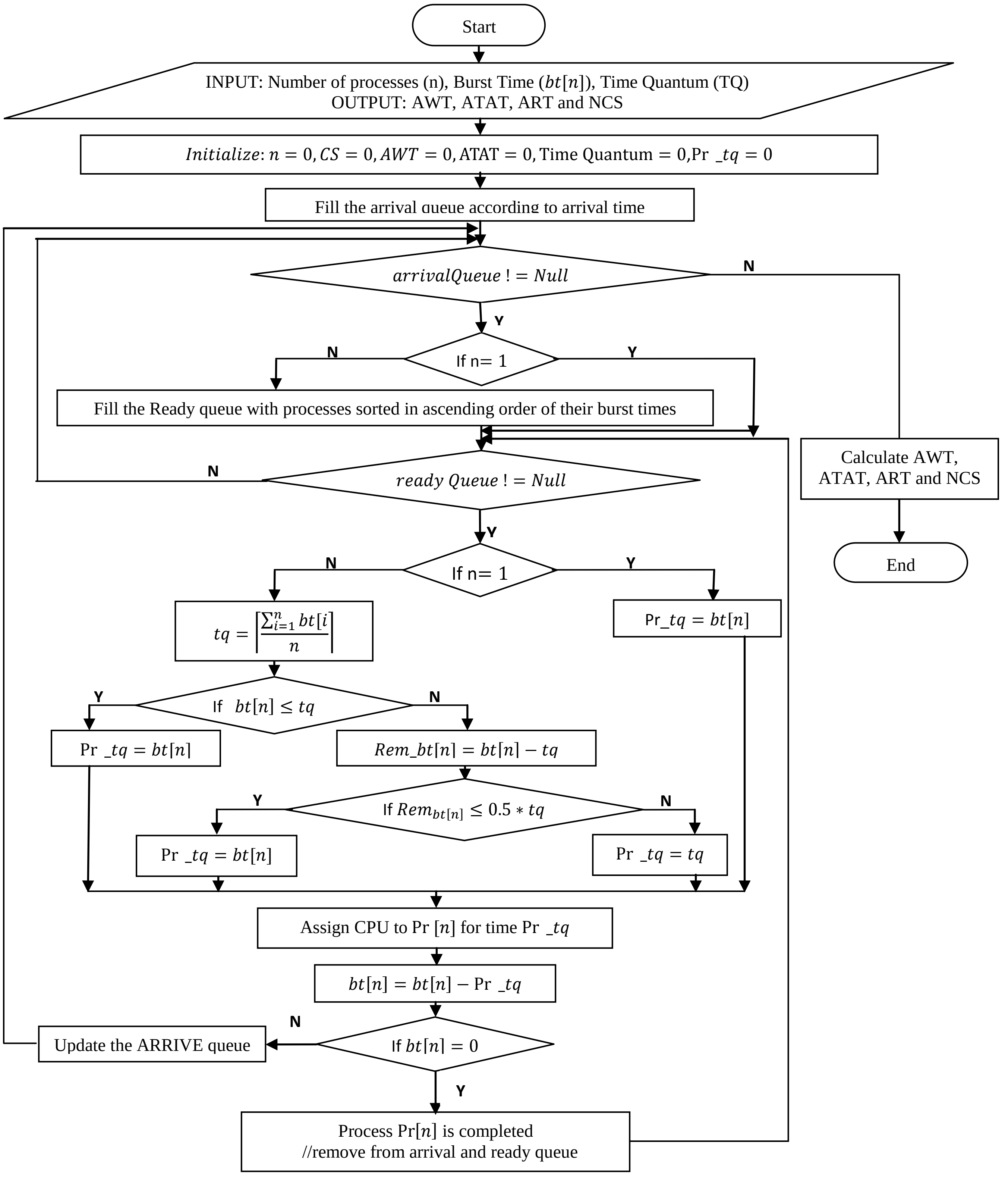
Start
INPUT: Number of processes (n), Burst Time ($bt[n]$), Time Quantum (TQ)
OUTPUT: AWT, ATAT, ART and NCS
$Initialize: n = 0, CS = 0, AWT = 0, \text{ATAT} = 0, \text{Time Quantum} = 0, \text{Pr}\ \_tq = 0$
Fill the arrival queue according to arrival time
$arrivalQueue\ != Null$
N
Y
If n= 1
N
Y
Fill the Ready queue with processes sorted in ascending order of their burst times
Calculate AWT, ATAT, ART and NCS
End
N
$ready\ Queue\ != Null$
Y
If n= 1
N
Y
$tq = \left\lceil \frac{\sum_{i=1}^{n} bt[i]}{n} \right\rceil$
$\text{Pr}\_tq = bt[n]$
Y
If $bt[n] \le tq$
N
$\text{Pr}\ \_tq = bt[n]$
$Rem\_bt[n] = bt[n] - tq$
Y
If $Rem_{bt[n]} \le 0.5 * tq$
N
$\text{Pr}\ \_tq = bt[n]$
$\text{Pr}\ \_tq = tq$
Assign CPU to Pr $[n]$ for time Pr $\_tq$
$bt[n] = bt[n] - \text{Pr}\ \_tq$
N
Update the ARRIVE queue
If $bt[n] = 0$
Y
Process Pr$[n]$ is completed
//remove from arrival and ready queue

**Figure 1: The Flow Chart of the proposed Round Robin algorithm**

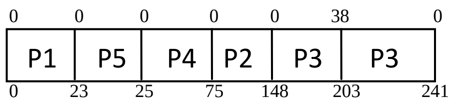

**Figure 7: The Gantt chart representation of NIRR scheduling**

**Table 2: Comparative table**

| Algorithms | AWT | ATAT | ART | NCS |
|---|---|---|---|---|
| **FCFS** | 110.2 | 158.4 | 110.2 | 4 |
| **SJF** | 49.6 | 97.8 | 49.6 | 4 |
| **RR** | 113 | 161.2 | 78.4 | 6 |
| **IRR** | 110.2 | 158.4 | 110.2 | 4 |
| **LJF+CBT** | 95.4 | 143.6 | 95.4 | 3 |
| **NIRR** | 53.8 | 102 | 53.2 | 4 |

**Table 2** shows the comparative results of the algorithms under study. SJF has the minimal AWT and ATAT while LJF+CBT and RR have the minimal NCS and ART respectively. In the RR category, the proposed algorithm has the minimal AWT, ATAT and NCS.

## 3.4 Simulation

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

A process generator routine was built to generate the process sets. Each process in the process set is a tuple: <process_*id*, *CPU_time*)>.

The Burst time (i.e. the *CPU_time*) was generated using uniform distribution. A process burst time generator was developed to take care of the random burst time of different processes in the system.

### *3.4.1 Experimental Setup*

Hardware

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

Software

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

The following figures show results of the algorithms for processes varying from 5 to 1000 taking the time quantum of 10*ms* (that will be used for RR and IRR) and burst time ranges between 1 and 50*ms*.

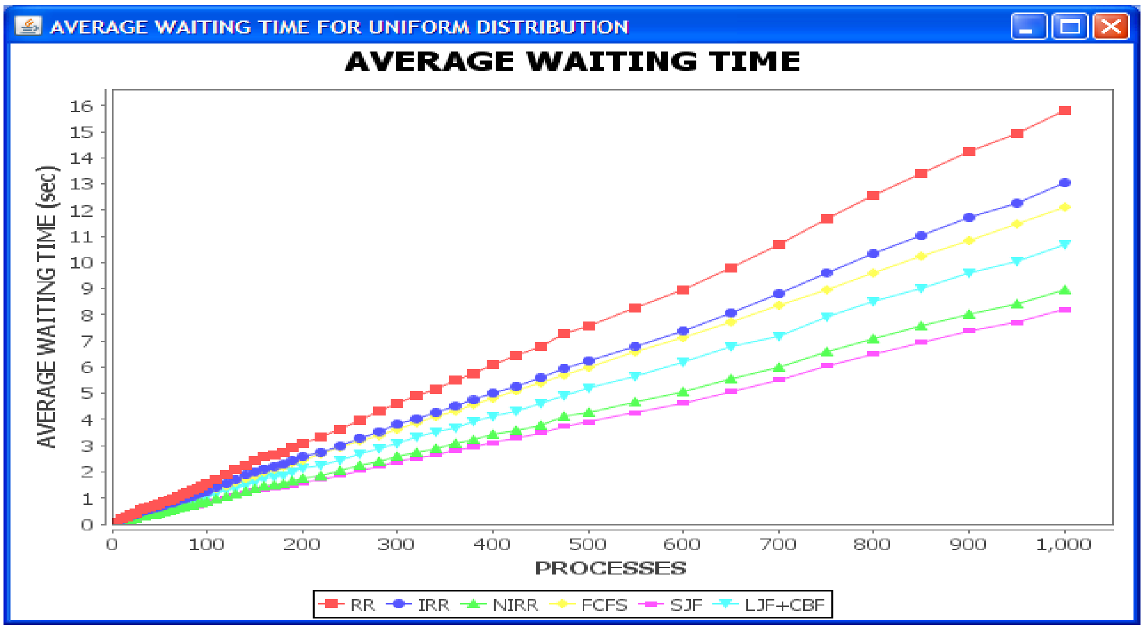

**Figure 8: Graph of Average Waiting Time**

Figure 8 above shows the graphical representation of the result of AWT. SJF produces the minimal result followed by the proposed algorithm (NIRR). This is followed by LJF+CBT, FCFS, IRR and RR respectively. And Figure 9 below shows the graphical representation of the result of ATAT. SJF produces the minimal result followed by the proposed algorithm (NIRR). This is followed by LJF+CBT, FCFS, IRR and RR respectively.

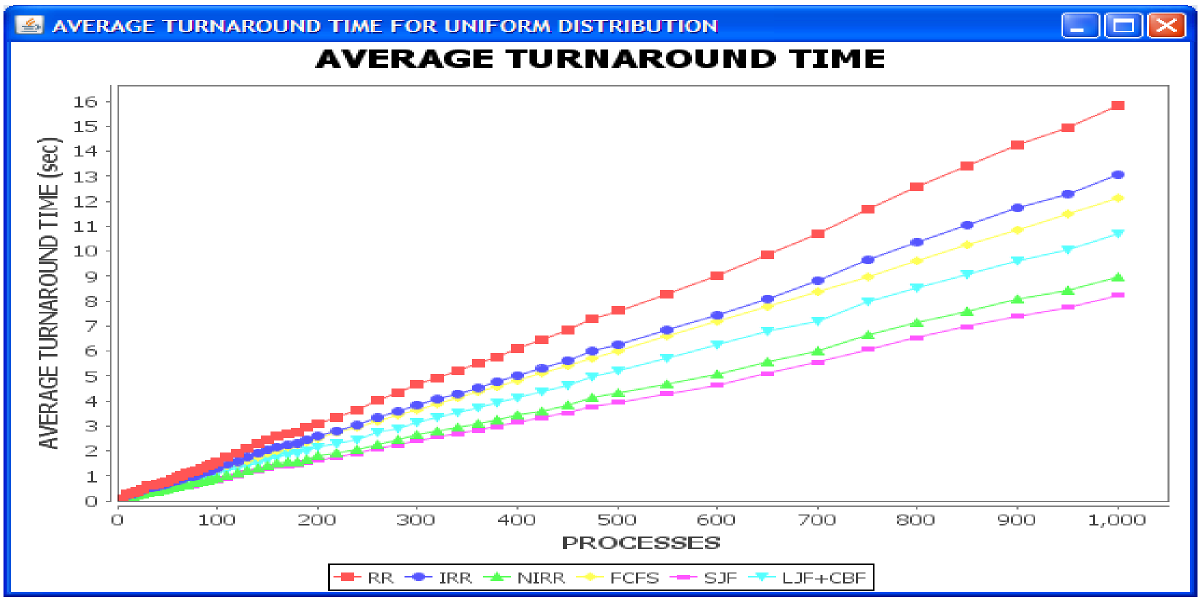

**Figure 9: Graph of Average Turnaround Time**

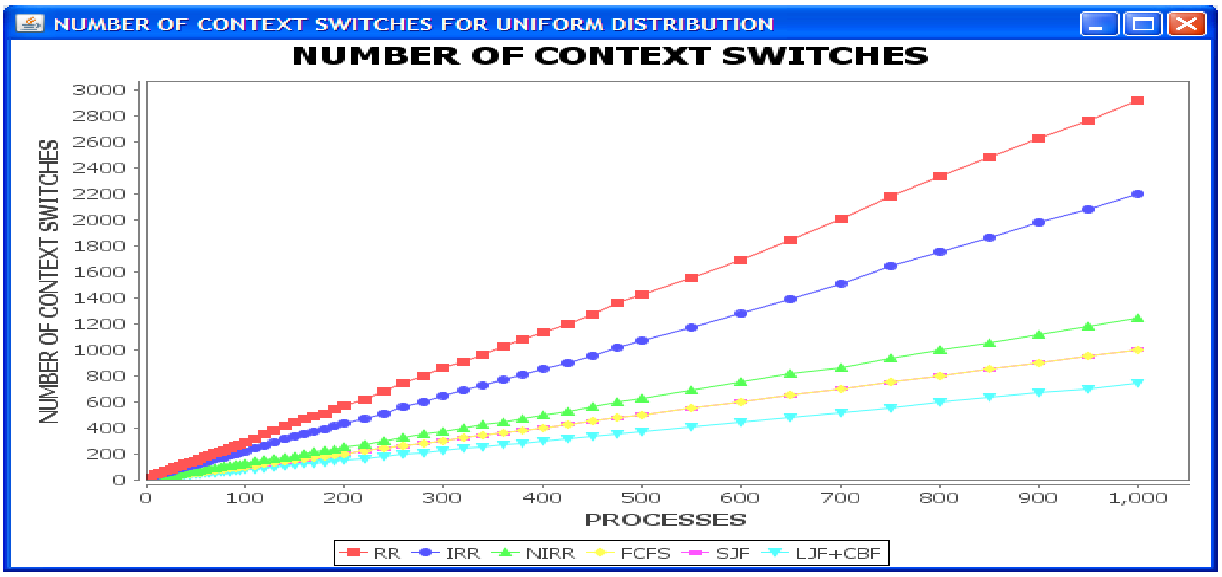

**Figure 10: Graph of Number of Context Switches**

Figure 10 above shows the graphical representation of the results of number of context switches. LJF+CBT produces the minimal result followed by SJF and FCFS producing the same results. This is followed by the proposed algorithm (NIRR), IRR and RR respectively. And Figure 11 below shows the graphical representation of the result of average response time. RR produces the minimal result followed by IRR, then the proposed algorithm (NIRR). This is followed by SJF, LJF+CBT and FCFS respectively.

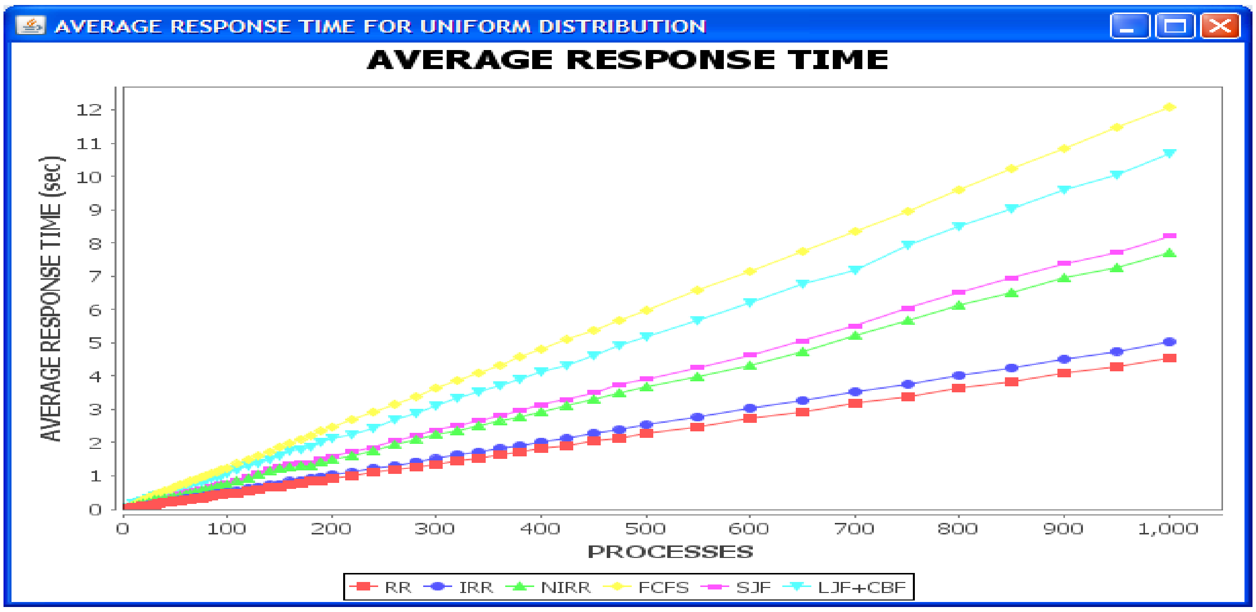

**Figure 11: Graph of Average Response Time**

## 4 CONCLUSION

A new algorithm based on improvement on the IRR known as A New Improved Round Robin (NIRR) CPU Scheduling Algorithm was proposed. This proposed algorithm (NIRR) together with FCFS, SJF, RR, IRR and LJF+CBT CPU scheduling algorithms were implemented in Java and their results were compared based on four scheduling criteria namely, AWT, ATAT, ART and NCS.

The simulation results show that SJF is the optimal scheduling algorithm in terms of minimizing AWT and ATAT. LJF+CBT and RR are the optimal algorithms in terms of minimizing NCS and ART respectively.

Based on the results obtained, the proposed algorithm (NIRR) is preferred for systems that adopt the RR Scheduling because it produces minimal AWT, ATAT and NCS compared to RR and IRR. In the future work, more tests should be done based on the burst time of processes that follow different patterns of statistical distributions.

## 5 REFERENCES

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

[2] Ajit, S, Priyanka, G and Sahil, B (2010): An Optimized Round Robin Scheduling Algorithm for CPU Scheduling, International Journal on Computer Science and Engineering (IJCSE), Vol. 02, No. 07, 2383-2385, pp 2382-2385.

[3] Behera, H.S, Mohanty, R and Debashree, N (2010): A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis, International Journal of Computer Applications (0975 – 8887) Volume 5, No.5, pp 10-15.

[4] Ishwari, S. R and Deepa, G (2012): A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems, International Journal of Innovations in Engineering and Technology (IJIET), Vol. 1 Issue 3, pp 1-11.

[5] Lalit, K, Rajendra, S and Praveen, S (2011): Optimized Scheduling Algorithm, International Journal of Computer Applications, pp 106-109.

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

[7] Operating Systems_ CPU Scheduling, http://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/5_CPU_Scheduling.html ,accessed 8th October 2013.

[8] Soraj, H and Roy, K.C: Adaptive Round Robin scheduling using shortest burst approach, based on smart time slice", International Journal of Data Engineering (IJDE), Volume 2, Issue 3, www.cscjournals.org/csc/manuscript/Journals/IJDE/.../IJDE-57.pdf ,accessed 10th December 2012.

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