MPRI-JPAS | INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE
VOL. 10 NO.9 DEC-2019 ISSN: 1660-5332

A NEW IMPROVED ROUND ROBIN CPU SCHEDULING ALGORITHM

UDDIN O. OSEMENGBE[1] AND JOHN TEMITOPE OGBITI[2]

[1]*Department of Mathematics and Computer Science, Edo University Iyamho, Nigeria.*

[2]*Department of Mathematics and Computer Science, Edo University Iyamho, Nigeria.*

Abstract

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. Scheduling is the method by which threads, processes or data flows are given access to system resources. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously). The Round Robin 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 thesis proposes a new algorithm that further improved on the Improved Round Robin (IRR) CPU scheduling algorithm. It was observed that the proposed algorithm (i.e. A New Improved Round Robin (NIRR)) compared with the other Round Robin scheduling algorithms, produces best Average Waiting Time (AWT), Average Turnaround Time (ATAT) and Number of Context Switches (NCS) in all categories of the statistical distributions used. Based on these results, the proposed algorithm outperformed other scheduling algorithms for systems that adopt RR CPU scheduling.

KEYWORDS: *CPU Scheduling, Multiprogramming, Processors, Simulation*

INTERNATIONAL JOURNAL – PAS
VOL. 4 NO.1 MAR-2018 ISSN: 1660-5332

Introduction

CPU Scheduling is the task of selecting a waiting process from the ready queue and allocating the CPU to it. The CPU is allocated to the selected process by a dispatcher, giving the process autonomous use of the CPU and access to all the resources during the allotted time based on the algorithm used.

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. Scheduling is the method by which threads, processes or data flows are given access to system resources. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously) (Ishwari and Deepa, 2012). The basic idea is to keep the CPU busy as much as possible by executing a user process until it must wait for an event, and then switch to another process. In multiprogramming systems, when there are more than one runnable process (i.e., ready), the operating system must decide which one to activate. The decision is made by the part of the operating system called the scheduler, using a scheduling algorithm (Suri and Sumit, 2012).

The round-robin (RR) scheduling algorithm is designed especially for timesharing systems. It is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes. A small unit of time, called a time quantum or time slice, is defined. A time quantum is generally from10 to 100 milliseconds in length. The ready queue is treated as a circular queue.

The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

To implement RR scheduling, we again treat the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.

One of two things will then happen. The process may have a CPU burst of less than 1 time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then proceed to the next process in the ready queue. If the CPU burst of the currently running process is longer than 1 time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and

INTERNATIONAL JOURNAL – PAS
VOL. 4 NO.1 MAR-2018 ISSN: 1660-5332

the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue.

Process State

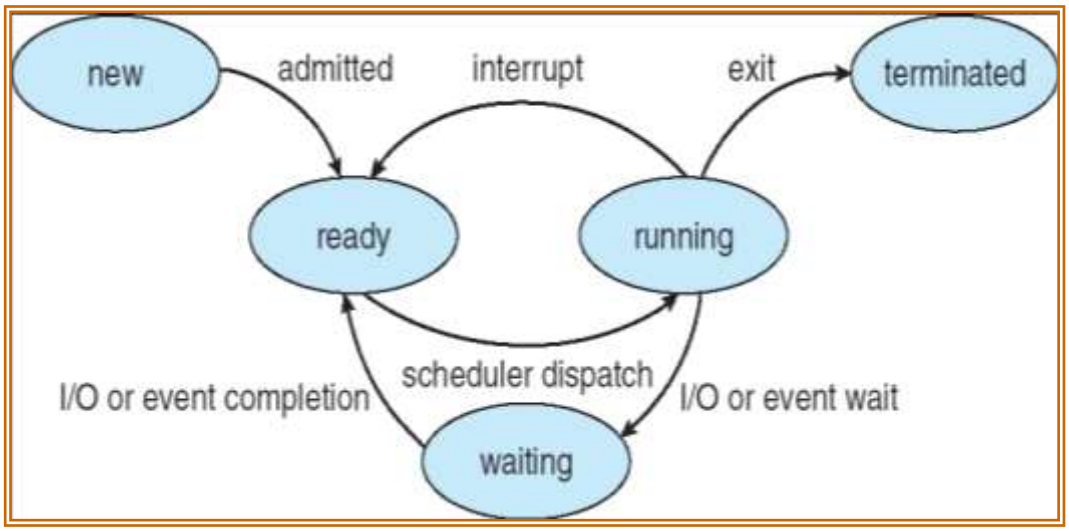
As a process executes, it changes **state**. The state of a process is defined in part by the current activity of that process. A process may be in one of the following states:

- **New**. The process is being created.
- **Running**. Instructions are being executed.
- **Waiting**. The process is waiting for some event to occur (such as an I/O completion or reception of a signal).
- **Ready**. The process is waiting to be assigned to a processor.
- **Terminated**. The process has finished execution

These names are arbitrary, and they vary across operating systems. The states that they represent are found on all systems, however. Certain operating systems also more finely delineate process states. It is important to realize that only one process can be *running* on any processor at any instant. Many processes may be *ready* and *waiting*, however. The state diagram corresponding to these states is presented in Figure below. (Silberschatz *et.al.* 2013)

A multiprogramming operating system allows more than one process to be loaded into the executable memory at a time and for the loaded process to share the CPU using time-multiplexing.

- Switches from running to ready state
- Switches from waiting to ready



A Diagram Illustrating Process State (Silberschatz *et.al.* 2013)

INTERNATIONAL JOURNAL – PAS
VOL. 4 NO.1 MAR-2018 ISSN: 1660-5332

Literature review

There are some selected literature reviews

Behera *et al* (2010) developed an algorithm and proved the experimental results of its performance over simple Round Robin scheduling algorithm. This algorithm reduces the number of context switching, average waiting time and average turnaround time. It does this by arranging the processes in ascending order of their burst times present in the ready queue. Then, the time quantum is calculated. For finding an optimal time quantum, median method is followed. Then, the time quantum is assigned to the processes. This time quantum is recalculated taking the remaining burst time in account after each cycle. In the next step, the algorithm have to rearrange the sorted processes, i.e. among n processes, the process which needs minimum CPU burst time will be replaced as the first process and then the process with highest CPU burst time from the queue, will be replaced as the second process and so on. This approach is similar to the approach that will be adopted in this thesis work in the sense that it calculates the time quantum dynamically and applies it to the processes in the ready queue which are arranged in some order.

Dibyendu and Iti (2013) developed an algorithm (Dynamic Time Quantum in Shortest Job First Scheduling Algorithm (DTQSJF)) that predicts burst times of processes as in the case of Shortest Job First scheduling and then use the predicted burst times to determine the time quantum to be used in Round Robin scheduling of processes. The algorithm gets the time quantum by taking the average of the predicted burst times and applying the time quantum to the processes in the ready queue in Round Robin fashion. If processes remain in ready queue after completion of their predicted burst time, a new quantum will be generated dynamically depending on another prediction of burst times of remaining processes, and each non completed process will get that amount of time quantum to execute in CPU, if any process does not complete its execution after application of that amount of time (Time Quantum), it will go to tail of the ready queue and wait for its turn. This algorithm was not demonstrated with examples or implemented, but the author suggested that implementing it will decrease the possibilities of starvation.

INTERNATIONAL JOURNAL – PAS
VOL. 4 NO.1 MAR-2018 ISSN: 1660-5332

Lalit *et al* (2011) developed, analyzed the operation and performance of Optimized Scheduling Algorithm over the simple Round Robin scheduling algorithm. The algorithm arranges the processes in ascending order of the burst time, and then it calculates the time quantum for Round Robin by taking the average of the burst times. This algorithm assumes that all processes arrive at the time t=0. This approach is similar to the approach that will be adopted in this thesis work in the sense that the time quantum will be calculated in similar manner

The Proposed New Improved Round Robin (NIRR) Scheduling Algorithm

The proposed NIRR CPU scheduling algorithm is the modification of the Improved Round Robin (IRR) CPU scheduling algorithm (i.e. the algorithm by Manish and AbdulKadir (2012)). It introduced 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.) that enters the ARRIVE queue, and allocates the CPU to it for the period of its burst time (i.e.*bt*). Processes that arrive while the CPU is executing this process will be added to the ARRIVE queue according to their arrival time. After execution of the first 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

$$time\ quantum = \left\lceil \frac{\sum_{i=1}^{n} burst\ time[i]}{n} \right\rceil$$

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.

The Random Number Generator

Poisson distribution is used in this simulation to generate randomly the necessary data required for the algorithms. Recall that if events, say arrivals to a queue, have inter-event times which are independent exponentially distributed random variables with rate ƛ, then the number of events N(t) up to time t has a Poisson distribution with parameter ƛ t:

$$P(N(t) = k) = \frac{e^{-\lambda}(\lambda t)^k}{K!}, \quad k = 0,1,2,\ldots$$

So, if we wish to generate random variates having a Poisson distribution, we can do the following:

1 Count = 0
2 Sum = 0
3 while 1
4 Count = Count + 1
5 Sum = Sum + expon(alpha)
6 if Sum > 1 return Count-1

The arrival of jobs to the CPU for processing can be described as a Poisson distribution. The Poisson distribution is a discrete distribution that uses integers as random variables in which the occurrence of events given an interval is independent.

In this simulation, we considered three algorithms

- First Come First Serve (FCFS)
- Shortest Job First
- Round Robin

INTERNATIONAL JOURNAL – PAS
VOL. 4 NO.1 MAR-2018 ISSN: 1660-5332

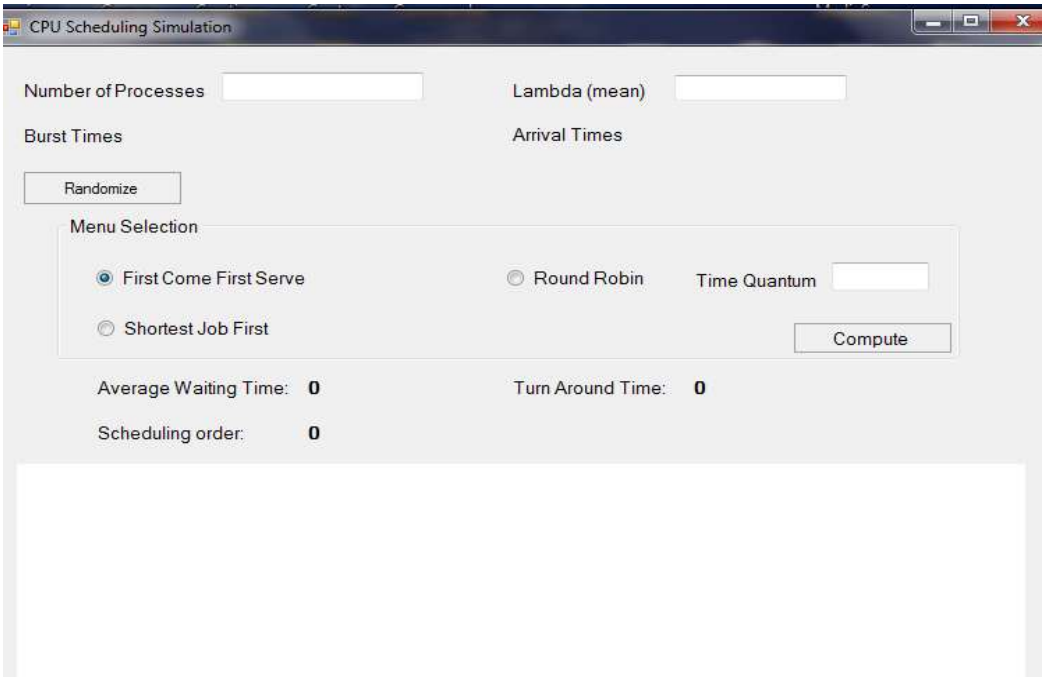
The CPU burst time, arrival time are random variables from Poisson distribution. To generate these variables, Poisson distribution requires the mean approximation of the distribution called lambda, and the number of events, in this case processes.

Installation

The software can be installed on any system with basic resources necessary to perform basic user task. In addition it is recommended that you have visual studio 2010 installed on your system before installing CPU scheduling simulation software. If you don't have visual studio 2010 or your system does not support visual studio 2010, you will not be able to run the application because the functionalities and platform for the software is the visual studio2010.

Generating the Burst Time and Arrival Time

The module that handles the Poisson random number is an open source library called math.net. a reference was made to the class module. An install of the Poisson distribution class was made in the btnrandomize() method which generates the required burst time and arrival time. Locate the executable file from start menu "CPU Scheduling", a simulation screen shown below will appear.



- Enter an integer value as the number of process. E.g 5

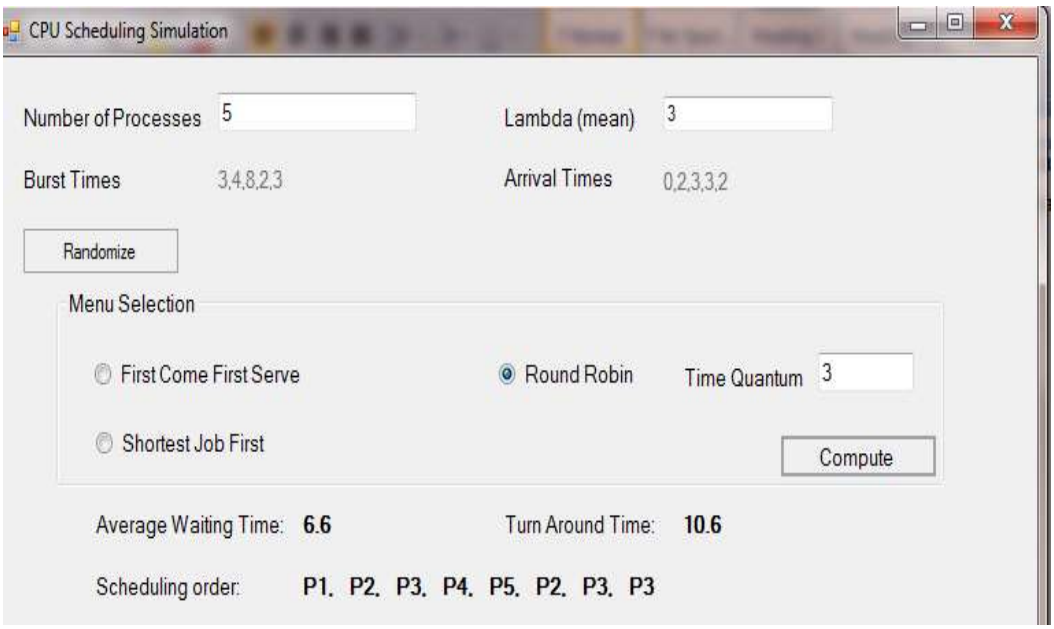
INTERNATIONAL JOURNAL – PAS
VOL. 4 NO.1 MAR-2018 ISSN: 1660-5332

- Estimate the mean of distribution say 3
- Click on the randomize button.

The randomize button generate random values as the CPU burst time, ensure that there is no zero value in the CPU burst time. If zero appears, click on the button again until a better distribution is obtained.
Note that the arrival time for the processes is also generated. The initial arrival time for the first process is assumed to be zero. That is, no jobs were present when the first job arrived.

Simulating the Algorithm
To simulate any of the three algorithms provided, click on the algorithm and then click the compute button.

Round Robin
The round robin algorithm is handled by the RR() module on the code behind. The round robin algorithm is based on time slice called the quantum number. Enter an integer as the quantum number in the space provided, and using the random numbers generated click on the compute button. The average turnaround time and waiting time for the process will be computed. Note also that the scheduling order. See figure bellow.



INTERNATIONAL JOURNAL – PAS
VOL. 4 NO.1 MAR-2018 ISSN: 1660-5332

Examples

The following example will be considered, in which each process has its burst and arrival time as shown in Table 4.1. The time quantum to be used by RR and IRR is 10*ms*. All processes are considered to arrive at the same time. Considers all processes to arrive at the same time (i.e. t=0).

Table 4.1: Process table

PR_ID	Bust time	Arrival Time
P1	9	0
P2	27	0
P3	14	0
P4	30	0
P5	22	0
P6	19	0
P7	45	0
P8	39	0
P9	7	0
P10	10	0

Improved Round Robin (IRR)

Figure 4.4 shows the Gantt chart of Improved Round Robin CPU scheduling algorithm

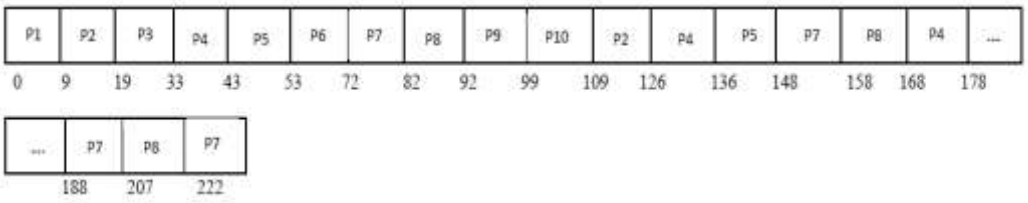


Figure 4.1 Gantt chart for IRR

Waiting Time

P1: (0-0) = 0, P2: ((9+90)-0) =99, P3: (19-0) = 19, P4: ((33+83+32)-0) = 148, P5: ((43+83)-0) = 126, P6: (53)-0) = 53, P7: ((72+66+20+19)-0) = 177, P8: ((82+66+20)-0) = 168, P9: ((92)-0) = 92, P10: (99-0) = 99.

Average Waiting Time

$$AWT = \frac{(0 + 99 + 19 + 148 + 126 + 53 + 177 + 168 + 92 + 99)}{10} = \frac{981}{10} = 98.1$$

Turnaround Time

P1: (9-0) = 9, P2: (126-0) =126, P3: (33-0) = 33, P4: (178-0) =178, P5: (148-0) = 148, P6: (72-0) = 72, P7: (222-0) = 222, P8: (207-0) = 207, P9: (99-0) = 99, P10: (109-0) = 109.

Average Turnaround Time

$$ATAT = \frac{9 + 126 + 33 + 178 + 148 + 72 + 222 + 207 + 99 + 109}{10} = \frac{1203}{10} = 120.3$$

Response Time

P1: (0-0) = 0, P2: (9-0) = 9, P3: (19-0) = 19, P4: (33-0) = 33, P5: (43-0) = 43, P6: (53-0) = 53, P7: (72-0) = 72, P8: (82-0) = 82, P9: (92-0) = 92, and P10: (99-0) = 99.

Average Response Time

$$ART = \frac{(0 + 9 + 19 + 33 + 43 + 53 + 72 + 82 + 92 + 99)}{10} = \frac{502}{10} = 50.2$$

Number of Context Switches = 18

Conclusion

Round Robin algorithm performs better with a turnaround Time of and Average Waiting. Its Presents an algorithm that is based on the modification of improved round robin scheduling algorithm. Its goal is to increase the performance of the Round Robin CPU scheduling algorithm by reducing the average waiting time, number of context switch, average response time and average turnaround time. 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 AWT, ATAT, ART and NCS.

References:

Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne (2013) *Operating System Concepts*. John Wiley and Sons, inc. ninth Edition Norm Matloff, (February 21, 2006) *Random Number Generation*

Lawrence Leemis and Steve Park (December, 1994) *Descrete-Event Simulation: A First Course* www.matlab.net

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) , 5* (5), 10-15.

Dibyendu, B and Iti, S. (2013). Dynamic Time Quantum in Shortest Job First Scheduling Algorithm (DTQSJF) for Unpredictable Burst Time of Processes. *International Journal of Computer Science & Engineering Technology , 4* (3), 208-212.

Lalit, K, Rajendra, S and Praveen, S. (2011). Optimized Scheduling Algorithm. *International Journal of Computer Applications* , 106-109.

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) , 49* (17), 4-9.

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 , 1* (3), 1-11