Research Proposal on

An Optimized Round Robin Scheduling Algorithm

Team Members

Md. Al-Amin Bhuiyan
ID: 1804084
S.A.M Jubyar
ID: 1804085
Ohcitya Bhattacharjee
ID: 1804086
Arnab Chakraborty
ID: 1804087
Ahammed Zayed Uddin Rahat
ID: 1804088
Fahimul Alam Araf
ID: 1804089

Course Information

Course Name :	Operating Systems(Sessional)
Course Code :	CSE-336

Course Teacher

MD. SHAFIUL ALAM FORHAD
Assistant Professor
Department of Computer Science & Engineering,
Chittagong University Engineering & Technology, Bangladesh

HASAN MURAD
Lecturer
Department of Computer Science Engineering,
Chittagong University Engineering Technology, Bangladesh.

August, 2022

An Optimized Round Robin Scheduling Algorithm

ABSTRACT

The operating system performs a wide range of tasks, including managing processes, memory, files, input/output, networking, protection systems, and command interpreter systems. Since the operating system is a system program and processes interact with hardware at runtime, process management is the most crucial function in these operations. As a result, we may infer that managing all processes is necessary for increasing a CPU's efficiency. We utilize a number of scheduling algorithms to manage the process. There are multiple scheduling algorithms for CPUs. But every algorithm has its own drawbacks and constraints. In this research, we proposed a new method for the round robin scheduling algorithm, which enhances CPU effectiveness.

Keywords: CPU Scheduling, Round Robin Scheduling Algorithm, Turnaround Time, Waiting Time, Gantt Chart

1. INTRODUCTION

A process is defined as an entity which represents the basic unit of work to be implemented in the system. In other words, when a program is loaded in memory, it is called a process. A process control block (PCB) is a data structure maintained by the operating system for every process. A PCB keeps all the information such as process state, process id, process privileges, CPU scheduling information etc. that are needed to keep track of the process and is identified by the process id. A CPU scheduling algorithm is necessary because a number of processes can be kept in memory simultaneously but only one process can occupy the CPU at a time. A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are some criteria for designing a CPU scheduling algorithm. An algorithm should be designed in such a way that the CPU remains as busy as possible and the less amount of time a process needs to wait on average. Any scheduling algorithm is either preemptive or non-preemptive. Preemption means if a process enters in running state, it can be interrupted and can be resumed in later stages. Whereas in case of non-preemptive scheduling, interruption can not be done. Instead, a process entered in the running state, will be executed uninterrupted and will leave the queue. There are some common non-preemptive process scheduling algorithms namely, First Come First Serve (FCFS), Shortest Job First (SJF), Priority based Scheduling. Some common preemptive scheduling algorithms are Round Robin Scheduling algorithm, Shortest Remaining Time First algorithm, preemptive version of Priority based Scheduling algorithm. As CPU is a primary resource, a good scheduling algorithm can increase the performance of a system by reducing the average waiting time of the process than traditional algorithms. Round Robin scheduling algorithm is best known for its nature of providing good response time in process scheduling. Here, the key factor and the tricky part is the time quantum. Depending on the time quantum, the performance of the algorithm varies. In this research work, we tried to improvise the traditional Round Robin scheduling algorithm and compared it with other traditional scheduling algorithms.

2.RELATED WORK

First Come First Serve(FCFS) is an operating system scheduling algorithm that automatically executes queued requests and processes in order of their arrival.

Shortest Job First (SJF) is an algorithm in which the process having the smallest execution time is chosen for the next execution. This scheduling method can be preemptive or non-preemptive.In non-preemptive scheduling, once the CPU cycle is allocated to process, the process holds it till it reaches a waiting state or terminates.

In Preemptive SJF Scheduling, jobs are put into the ready queue as they come. A process with the shortest burst time begins execution. If a process with even a shorter burst time arrives, the current process is removed or preempted from execution, and the shorter job is allocated CPU cycle.

Priority Scheduling is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the tasks to work as per the priority.In non-preemptive priority scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy, will release the CPU either by switching context or terminating.

In Preemptive Priority Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running.

Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way. It is basically the preemptive version of First come First Serve CPU Scheduling algorithm.

Seltzer, M P. Chen and J Outerhout 1990[1] has developed scheduling algorithms suited for certain goals sometimes with provable properties.

Silberschatz, Galvin, Gagne [2] proposed a multi-programmed operating system which allows the process as many as possible running at all times in order to maximize the CPU utilization.

Sabrina, F.C.D, Nguyen, S.Jha, D.Platt and F. Safaei[3] proposed a fundamental operating system function which has a big effect on resource utilization and overall performance of the system.

Rakesh Kumar, Yadav Abhishek K Mishra , Navin Prakash and Himanshu Sharma[4] proposed an improved Round Robin CPU Scheduling Algorithm which maximizes CPU utilization and minimizes response time by reducing total waiting time and turn around time.

3. OBJECTIVES OF OUR WORK

Objectives of our research can be summarized as:

1. To optimize Round Robin Scheduling Algorithm for the efficiency of CPU.
2. To optimize average waiting time from traditional Round Robin Scheduling.
3. To optimize average turnaround time from traditional Round Robin Scheduling.

4. PROPOSED METHODOLOGY

1. Allocate all processes to the CPU only one time as like the present Round Robin scheduling algorithm.

2. After the first time we select the job which is closest to the average remaining time from the waiting queue and assign it to the CPU.

3. After that we calculate the average of all remaining jobs from the waiting queue then select the nearest value from the waiting queue and do step 2.

4. Till the complete execution of all processes we repeat steps 2 and 3 that means while all the processes have not been finished(executed).

5. Time complexity of our proposed algorithm is O(total_execution_time*search_time)

Brief Comprehension
In our proposed algorithm,

p[]	:	a structure containing all processes
q	:	ready queue
n	:	the total number of process
avg_var	:	average variation computed between array's element
comArr	:	arrival time comparison

t	:	time quantum

5. PROPOSED METHODOLOGY

Our proposed algorithm differs at the second step from the traditional round robin scheduling algorithm.

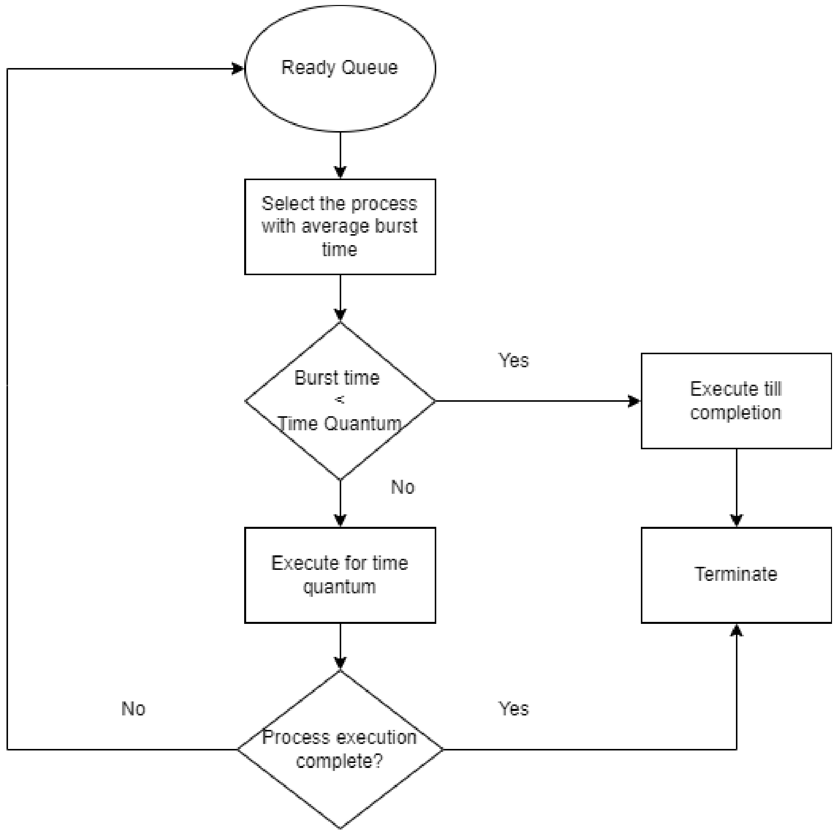

Ready Queue
Select the process with average burst time
Burst time < Time Quantum
Yes
Execute till completion
No
Execute for time quantum
Terminate
No
Yes
Process execution complete?

Figure 1: Flow chart of the proposed algorithm

6. Experiment Discussion

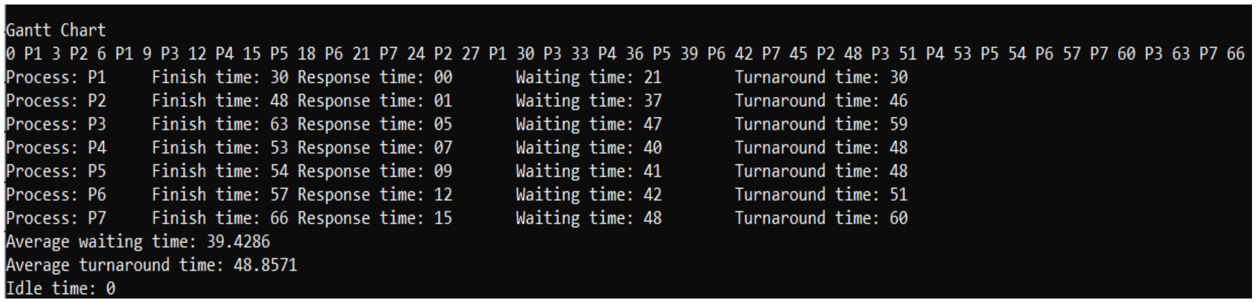
In our proposed round robin algorithm we have included the idea of searching the process with average remaining burst time.
Suppose state of processes is according to given below table and also time quantum=3

Process	Arrival Time	Burst Time
P1	0	9
P2	2	9

P3	4	12
P4	5	8
P5	6	7
P6	6	9
P7	6	12

Table 1: The arrival time and the burst time of each process

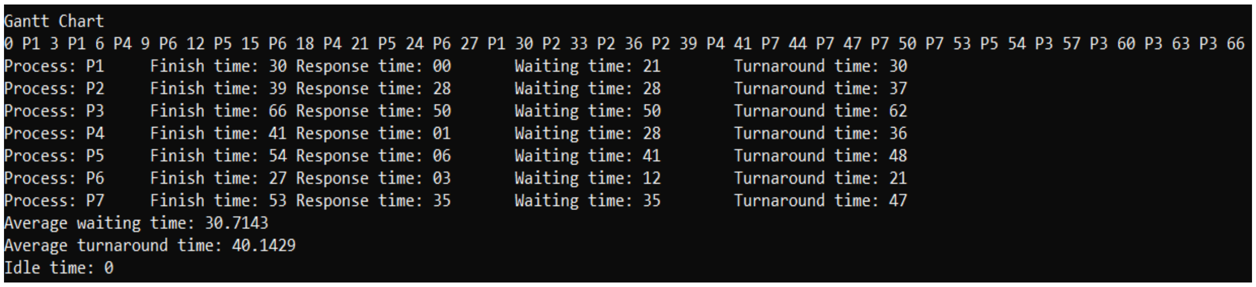
We will see the outputs according to Table 1.
The round robin algorithm creates the output given below:



```
Gantt Chart
0 P1 3 P2 6 P1 9 P3 12 P4 15 P5 18 P6 21 P7 24 P2 27 P1 30 P3 33 P4 36 P5 39 P6 42 P7 45 P2 48 P3 51 P4 53 P5 54 P6 57 P7 60 P3 63 P7 66
Process: P1     Finish time: 30 Response time: 00       Waiting time: 21        Turnaround time: 30
Process: P2     Finish time: 48 Response time: 01       Waiting time: 37        Turnaround time: 46
Process: P3     Finish time: 63 Response time: 05       Waiting time: 47        Turnaround time: 59
Process: P4     Finish time: 53 Response time: 07       Waiting time: 40        Turnaround time: 48
Process: P5     Finish time: 54 Response time: 09       Waiting time: 41        Turnaround time: 48
Process: P6     Finish time: 57 Response time: 12       Waiting time: 42        Turnaround time: 51
Process: P7     Finish time: 66 Response time: 15       Waiting time: 48        Turnaround time: 60
Average waiting time: 39.4286
Average turnaround time: 48.8571
Idle time: 0
```

Figure 2: Gantt chart, Finish time, Waiting time, Turnaround time for each process and Average waiting time, Average turnaround time, Idle time for Round Robin Algorithm

Our proposed algorithm gives the output for the corresponding input is given below:



```
Gantt Chart
0 P1 3 P1 6 P4 9 P6 12 P5 15 P6 18 P4 21 P5 24 P6 27 P1 30 P2 33 P2 36 P2 39 P4 41 P7 44 P7 47 P7 50 P7 53 P5 54 P3 57 P3 60 P3 63 P3 66
Process: P1     Finish time: 30 Response time: 00       Waiting time: 21        Turnaround time: 30
Process: P2     Finish time: 39 Response time: 28       Waiting time: 28        Turnaround time: 37
Process: P3     Finish time: 66 Response time: 50       Waiting time: 50        Turnaround time: 62
Process: P4     Finish time: 41 Response time: 01       Waiting time: 28        Turnaround time: 36
Process: P5     Finish time: 54 Response time: 06       Waiting time: 41        Turnaround time: 48
Process: P6     Finish time: 27 Response time: 03       Waiting time: 12        Turnaround time: 21
Process: P7     Finish time: 53 Response time: 35       Waiting time: 35        Turnaround time: 47
Average waiting time: 30.7143
Average turnaround time: 40.1429
Idle time: 0
```

Figure 3: Gantt chart, Finish time, Waiting time, Turnaround time for each process and Average waiting time, Average turnaround time, Idle time for Our Proposed Algorithm

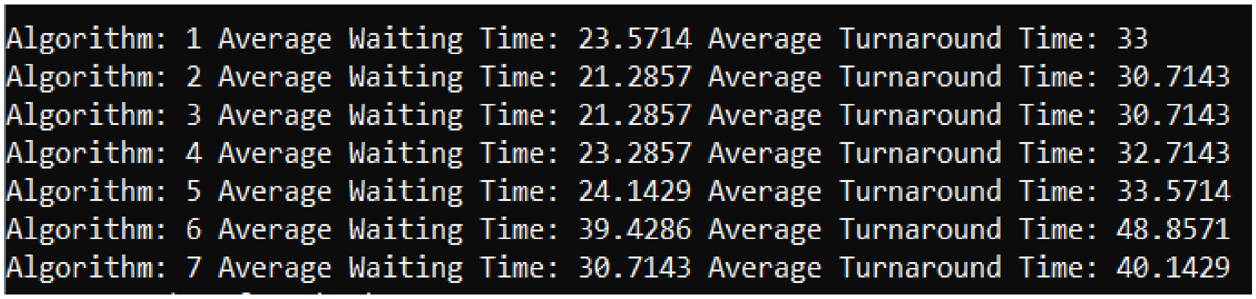
From the aforementioned experiment, it is clear that our suggested approach reduces both the average waiting time and the average turnaround time.

Maximum CPU use is indicated by the reduction of total waiting time and turnaround time. We may say that the new approach is significantly more effective than the current algorithm.

7. Theoretical Comparison with related works

According to the previous example we have compared the seven CPU Scheduling algorithms including our proposed algorithm. Here we have used the **output format** as per our instructions. For better understanding we are going to mention the name and the serial number.

Serial No	**Algorithm Name**
Algorithm 1	First Come First Service(FCFS)
Algorithm 2	Non-preemptive SJF
Algorithm 3	Preemptive SJF
Algorithm 4	Non-preemptive Priority
Algorithm 5	Preemptive Priority
Algorithm 6	Round Robin
Algorithm 7	Our Proposed Algorithm



```
Algorithm: 1 Average Waiting Time: 23.5714 Average Turnaround Time: 33
Algorithm: 2 Average Waiting Time: 21.2857 Average Turnaround Time: 30.7143
Algorithm: 3 Average Waiting Time: 21.2857 Average Turnaround Time: 30.7143
Algorithm: 4 Average Waiting Time: 23.2857 Average Turnaround Time: 32.7143
Algorithm: 5 Average Waiting Time: 24.1429 Average Turnaround Time: 33.5714
Algorithm: 6 Average Waiting Time: 39.4286 Average Turnaround Time: 48.8571
Algorithm: 7 Average Waiting Time: 30.7143 Average Turnaround Time: 40.1429
```

Figure 4: Compared output of Average Waiting Time and Average Turnaround Time among seven algorithms.

From the above output we can see the exact comparison among seven algorithms. Our proposed algorithm decreases the average waiting time and average turnaround time more than round robin. We can see the difference from their graphical comparison.

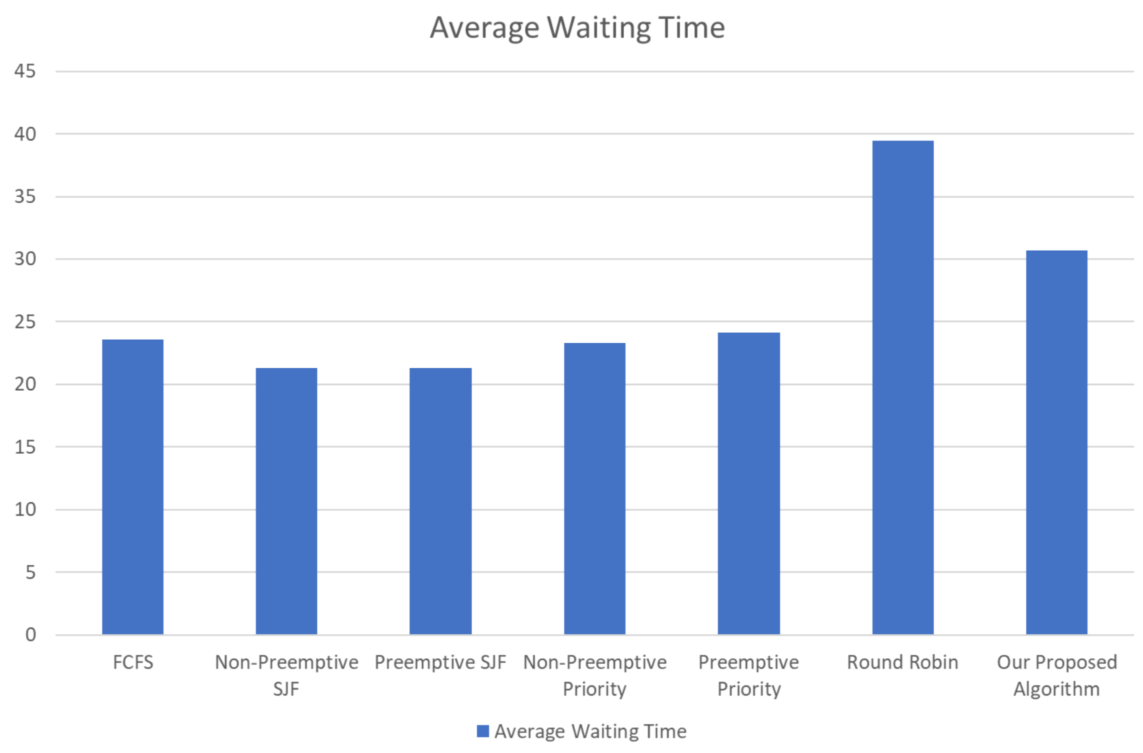


Figure 5: Compared bar chart of average waiting time of seven algorithm

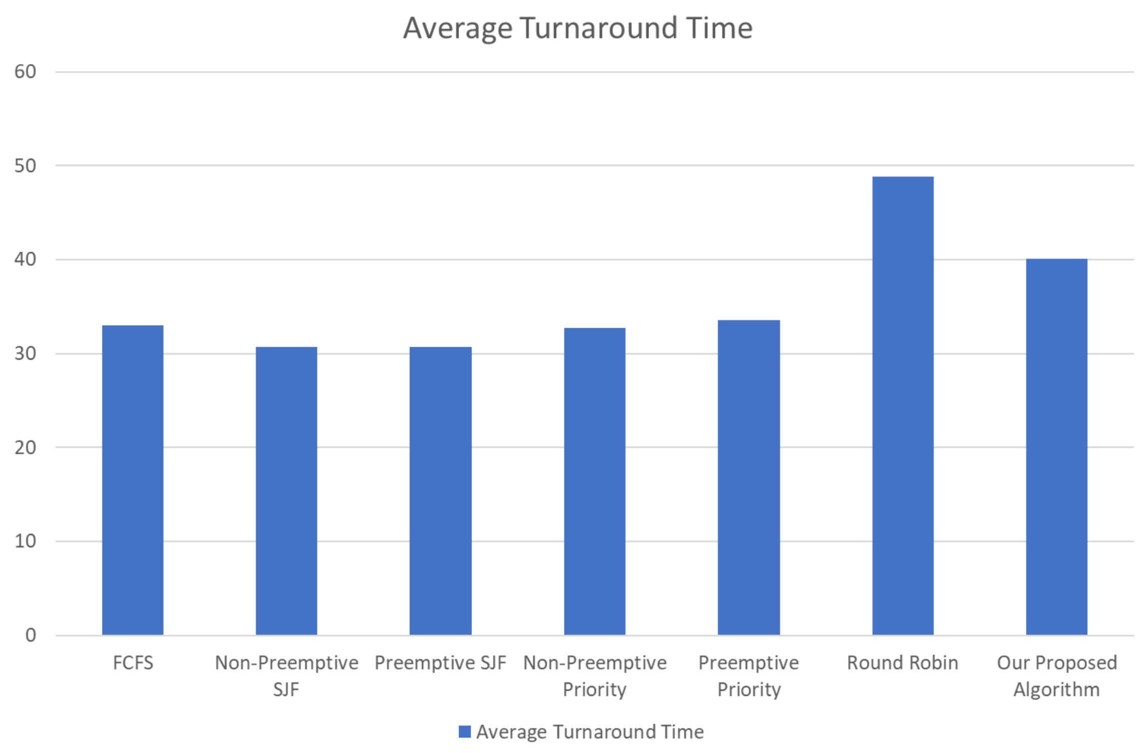


Figure 6: Compared bar chart of average turnaround time of seven algorithms

8. Conclusion

It is clear that our suggested algorithm is better than the current RR algorithm. Knowing that the RR scheduling algorithm was created specifically for a system for sharing time. Consequently, we can get better soon by employing this method in a time-sharing system.

REFERENCES

1. Sabrina, F.C.D, Nguyen, S.Jha, D. Platt and F. Safaei, 2005. Processing resources scheduling in programmable networks. Computer commun, 28:676-687.
2. Silberchatz, Galvin and Gagne ,2003 .operating systems concepts,(6th edn, John Wiley and Sons)
3. Seltzer, M P. Chen and J outerhout, 1990.Disk scheduling revisited in USENIX. Winter technical conference.
4. Shamim H M 1998. Operating system, DCSA-2302. School of sciences and Technology. Bangladesh open university Gazipur-1705
5. D. M. Dhamdhere Operating Systems A Concept Based Approach, Second edition, Tata McGraw-Hill, 2006
6. Operating Systems Sibsankar Haldar 2009 , Pearson Education, India
7. http://en.wikipedia.org/wiki/Scheduling_(computing)