Volume 3, Issue 3
June 2014

ITEE Journal

ISSN: - 2306-708X

Performance Analysis of Short Term Scheduling Algorithms

[1]Muhammad Usman, [2]Aamir Iqbal, [3]Ehsan Ahmed, [4]Shaukat Ghani, [5]Muhammad Adnan Khan

[1,2,3,4] Federal Urdu University of Arts, Science and Technology, Islamabad, Pakistan
[5,] School of Engineering and Applied Sciences (SEAS), ISRA University, Islamabad, Pakistan
Email: [1]musman@live.com, [2]aamircs@gmail.com, [3]ehsan225@gmail.com, [4]m.shoki@yahoo.com, [5]adnan_600@yahoo.com

ABSTRACT

Short term scheduling is the key function of a modern operating system. Since many jobs are entering into memory at a certain instant, this needs to be handled efficiently. The main objective behind short term scheduling is to keep the main resource, CPU, busy most of the time by executing more and more jobs. Many scheduling algorithms have been introduced like FCFS (Non-Preemptive), SJF (Preemptive & Non-Preemptive), Priority (Preemptive & Non-Preemptive), FCFS (SJF) etc. for a multiprogramming operating system. This paper presents a performance analysis between these existing scheduling algorithms. The algorithms have been analyzed using difference evaluation models like deterministic and queuing model to determine which algorithm has better performance.

Keywords: *Short term scheduling, CPU scheduling, performance analysis*

1. INTRODUCTION

1.1 Job and their types

A job is a program in execution. It is also called a process or task. Jobs are of two types; CPU bound and I/O bound. CPU bound jobs use their entire time quantum without performing any I/O operations. Whereas, I/O bound jobs use only a small amount of processor before performing I/O. These jobs do not use up their entire time quantum.

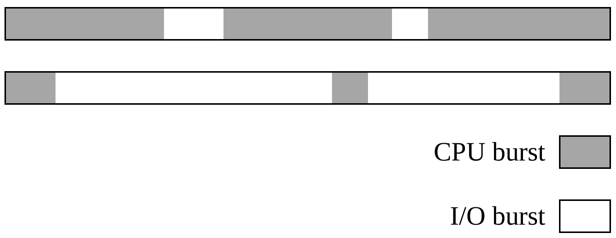

Figure 1: CPU bound vs I/O Bound jobs

1.2 Job states

A job occupies different states during execution. Figure-1 shows the different job states. [1]

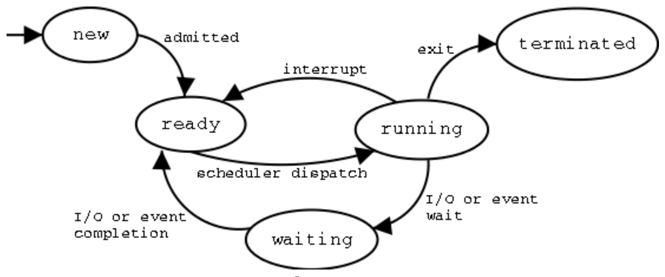

Figure 2: Job States Diagram

A job entered in job queue is assigned as new state. It admits the ready state afterwards. Here another queue is maintained. Scheduling techniques are applied here to assign CPU to the job. From running state a job can go back to ready state only when its quantum expires. But, if an I/O occurs in the running state, the running job has been assigned waiting state. Another queue is maintained in the waiting state. Here the jobs waiting for I/O are queued. The ready state has been assigned on completion of I/O. If the job completes the exaction, it has been terminated. Jobs can only be terminated from running state.

1.3 Reason behind short term scheduling

In early years, single user operating system can execute only one job at a certain time and other jobs keeps on waiting until the termination of first job and the main resource, CPU, remained idle most of the time. Afterwards, the multiprocessing scheme was introduced with the focus to maximize CPU utilization.[1][2] Hence, scheduling became a key factor in CPU performance. Short term scheduling is a core function of an operating system. Jobs are many in number and they keep on coming as the computer remains functional.[3] Therefore, managing the jobs before execution is a complexity. Several techniques are being introduced for scheduling purposes.

1.4 Types of schedulers

Scheduling is the major task of operating system. Its objective is to select the job from jobs queue and assign it to CPU for processing. Schedulers have three types:

i. Job scheduler
ii. Swapper
iii. Short term scheduler

Job scheduler is also named as long term scheduler. Its goal is to select a job from jobs queue and assign it to ready queue for CPU scheduling. The primary aim of the job scheduler is to provide a balance of mixed jobs, such as both I/O bound and processor bound. Job scheduler also controls the degree of multiprogramming. [5]

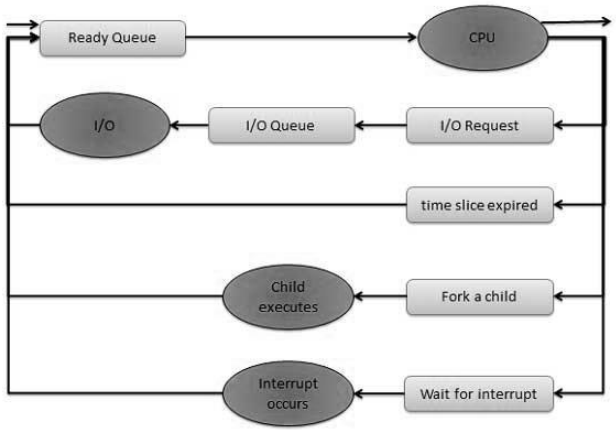

Figure 3: Job Queue Diagram

Swapper is also known as medium term scheduler. Its target is to remove the job from the memory. Swapper tries to reduce the degree of multi-programming. A job in running may become suspended and cannot make any further progress. It is the job of swapper to remove it from main memory and secondary memory to clear space for more jobs. [12]

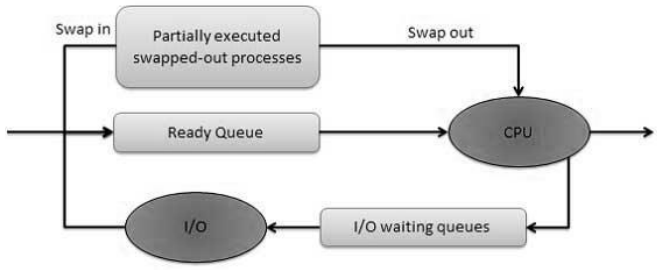

Figure 4: Addition of Swapper in Queue Diagram

Short term scheduler is also named as CPU scheduler. Its objective is to determine which job to move from ready to running state. Short term scheduler is called very frequently.

1.5 Context switching

Context switching is very important feature of modern operating systems. When CPU switches from one job to another, it must save the state of the already running job and load the new job. The context switching is the technique used to save and restore the job state in process control block (PCB).

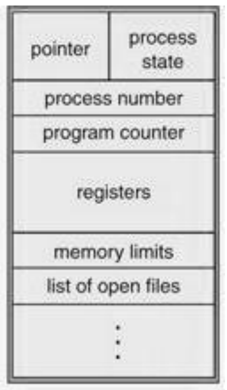

Figure 5: Process Control Block (PCB)

Context switch time is purely overhead. Context switching can affect the performance significantly as computers, these days, have a lot of general purpose and status registers to be saved. Context switching times are highly dependent on hardware support. [2]

When a process is running a process control block is created in the memory. All the necessary information of the running job is saved in the PCB. When the control switches from user to kernel mode, the context switcher saves the content of all registers in the memory.

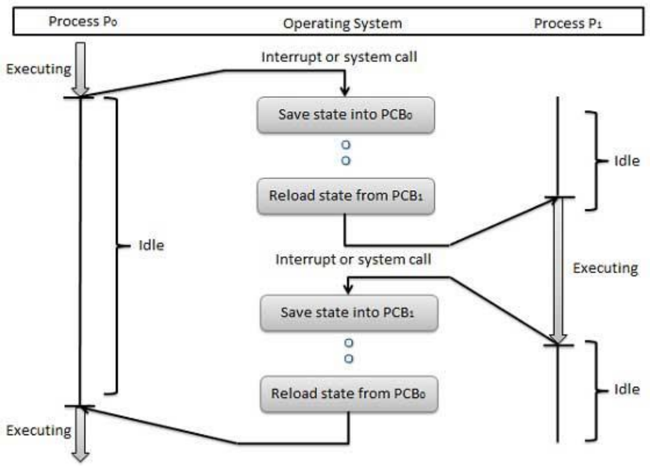

Figure 6: Context Switching Diagram

2. CRITERIA FOR CPU SCHEDULING

Various CPU scheduling techniques have been introduced with different properties and characteristics. It is a complex decision to apply a technique in a particular situation for scheduling a job from ready queue. However, there are some characteristics on which the comparison of different techniques is based. [8][9] The criteria for scheduling are based on:

2.1 Utilization/efficiency: This is an indicator for measuring how much CPU has been utilized.

2.2 Throughput: A certain amount of work done in an instant of time is called throughput. The greater the number of jobs completed, the more work is done by the system. [11] A better algorithm must have better throughput.

2.3 Turnaround time: The time taken by job from submission to termination state is considered as turnaround time. An algorithm with shorter turnaround time is considered better.

2.4 Waiting time: It is the total of times spent in ready queue.

2.5 Response time: The time consumed from new state of a job to assigning it to CPU for the first time is response time. Minimum response time makes an algorithm better.

Volume 3, Issue 3
June 2014

ITEE Journal

ISSN: - 2306-708X

2.6 Fairness: It states that each job is treated equally with fair policy in assigning CPU.

3. SHORT TERM SCHEDULING ALGORITHMS

This subsection explains some common short term scheduling algorithms, which are:

- First Come First Serve (Non-Preemptive)
- Shorter Job First (Non-Preemptive)
- Shorter Job First (Preemptive)
- Priority (Non-Preemptive)
- Priority (Preemptive)
- First Come First Serve (Shorter Job First)

3.1 First Come First Serve (Non-Preemptive)

First come first server is the simplest algorithm. It is also named as first in first out (FIFO) algorithm. The jobs are simply queued in order they reach and provided to CPU. When CPU is idle, a job at the head of queue is forwarded to CPU and it is removed from the ready queue. It is a non-preemptive algorithm. [7]

In FCFS algorithm, if multiple jobs are waiting for execution in the ready queue and a slow processing job with larger burst time is utilizing the CPU then due to the convoy all fast jobs with shorter burst time waiting for CPU waits for unnecessarily long time.[6] This is called convoy effect. Thus FCFS (Non-preemptive) is the troublesome for time sharing systems.

3.2 Shorter Job First (Non-Preemptive)

It is a non-preemptive algorithm. This scheduling algorithm attaches the length of the job's next CPU burst with each job. The job that has the smallest next CPU burst has been assigned to CPU for processing. If the next CPU burst of two jobs is same then FCFS technique is applied to select the next job. [14]

When a new job reaches in ready queue while another job is already executing, burst time of the newly reached job is compared with the remaining burst time of the job executing currently. If the new job has shorter burst time then SJF (non-preemptive) algorithm will continue to execute the current job till its burst time completes, as it is non-preemptive algorithm.

3.3 Shorter Job First (Preemptive)

It is preemptive algorithm. This algorithm associates with each job the length of the job's next CPU burst. The CPU is assigned to the job that has the smallest next CPU burst. If the next CPU bursts of two jobs are the same, FCFS algorithm is used to select the job. [14]

When a new job arrives in the ready queue while a previous job is already executing, burst time of the newly arrived job is compared with the burst time left of the currently executing job. If the new job has shorter burst time then SJF (preemptive) will preempt the currently executing job.

3.4 Priority (Non-Preemptive)

It is non-preemptive algorithm. A priority is defined with each job. The job with highest priority is assigned to the CPU. If two jobs have same priority, jobs are scheduled in FCFS order. Since it is a non-preemptive algorithm, the new arrived job is placed simply at the beginning of the ready queue.

A major problem with priority algorithm is that sometimes these can leave some low priority job waiting for CPU indefinitely. It is called starvation or blocking. A job ready to run but waiting for the CPU is considered as blocked. In heavily loaded systems, low priority jobs keeps on waiting for long time.

Aging is the solution to this blocking or starvation. It is a method of increasing the priority of jobs gradually that waits in the ready queue for a long time. [10] In this way their priority becomes higher.

3.5 Priority (Preemptive)

It is a preemptive algorithm. A priority is defined with each job. The job with highest priority is assigned to the CPU. If two jobs have same priority, jobs are scheduled in FCFS order.

When a new job arrives in the ready queue while a previous job is already executing, burst time of the newly arrived job is compared with the priority of currently executing job. A priority (preemptive) algorithm will swap out the currently executing job, if the priority of newly reached job is higher than the priority of the already executing job. Since, it is a priority scheduling algorithm therefore similar starvation/blocking happens in it too and hence aging is required to solve the issue.

3.6 First Come First Serve (Shorter Job First)

It is also named as round robin scheduling algorithm. The FCFS (SJF) scheduling algorithm is developed for timesharing systems. It is similar in function to FCFS algorithm but a preemption factor is added to switch between jobs. A time slice is introduced in FCFS (SJF). The ready queue is treated as a round queue. The short term scheduler goes around the ready queue, assigning the CPU to each job for a time interval of up to one unit time slice.

If the time slice of coming jobs is shorter than the frequently context switching is an overhead. [13] It is the major drawback of this scheduling algorithm.

4. MODELS OF ANALYTICAL EVALUATION

There are certain techniques for analysis of algorithm regarding short term scheduling:

Volume 3, Issue 3
June 2014

ITEE Journal

ISSN: - 2306-708X

4.1 Deterministic model

It is a mathematical model in which output is determined through already known values. Randomization element for values is not included in this model. [15] Due to this factor, a given input will always produce the same output.

The following example data will be applied using deterministic model on all the algorithms to find out the average waiting time and turnaround times:

Jobs	Burst Time	Arrival Time	Priority
J_1	6.3	2.7	3
J_2	1.8	1.8	1
J_3	5.9	0.9	1
J_4	1.1	2.6	0
J_5	4.8	0.1	2
J_6	0.2	0.1	2
J_7	0.5	0.2	3

(Time measured in μ seconds)

4.1.1 FCFS (Non-Preemptive)

Example:

Jobs	Burst Time	Arrival Time
J_1	6.3	2.7
J_2	1.8	1.8
J_3	5.9	0.9
J_4	1.1	2.6
J_5	4.8	0.1
J_6	0.2	0.1
J_7	0.5	0.2

Assumption: All jobs occur at time 0.

Gantt chart:

J_1	J_2	J_3	J_4	J_5	J_6	J_7

0 6.3 8.1 14.0 15.1 19.9 20.1 20.6

Waiting time (WT) of Ji = BT - Arrival time
WT of J1 = 0 - 0 = 0 μs
WT of J2 = 6.3 - 0 = 6.3 μs
WT of J3 = 8.1 - 0 = 8.1 μs
WT of J4 = 14.0 - 0 = 14.0 μs
WT of J5 = 15.1 - 0 = 15.1 μs
WT of J6 = 19.9 - 0 = 19.9 μs
WT of J7 = 20.1 - 0 = 20.1 μs
Average waiting time (AvgWT) = Sum of WTs of J1 to J7/7
AvgWT = 83.5/7 = 11.93 μs

Turnaround time (TAT) of Ji = BT of Ji + Waiting time of Ji
TAT of J1 = 6.3 + 0 = 6.3 μs
TAT of J2 = 1.8 + 6.3 = 8.1 μs
TAT of J3 = 5.9 + 8.1 = 14.0 μs
TAT of J4 = 1.1 + 14.0 = 15.1 μs
TAT of J5 = 4.8 + 15.1 = 19.9 μs
TAT of J6 = 0.2 + 19.9 = 20.1 μs
TAT of J7 = 0.5 + 20.1 = 20.6 μs
Average TAT (AvgTAT) = Sum of TATs of J1 to J7/7
AvgTAT = 104.1/7 = 14.87 μs

4.1.2 SJF (Non-Preemptive)

Example:

Jobs	Burst Time	Arrival Time
J_1	6.3	2.7
J_2	1.8	1.8
J_3	5.9	0.9
J_4	1.1	2.6
J_5	4.8	0.1
J_6	0.2	0.1
J_7	0.5	0.2

Assumption: All jobs reach at time 0.

Gantt chart:

J_6	J_7	J_5	J_4	J_2	J_3	J_1

0.1 0.3 0.8 5.6 6.7 8.5 14.4 20.7

Waiting time (WT) of Ji = BT - Arrival time
WT of J1 = 14.4 - 2.7 = 11.7 μs
WT of J2 = 6.7 - 1.8 = 4.9 μs
WT of J3 = 8.5 - 0.9 = 7.6 μs
WT of J4 = 5.6 - 2.6 = 3.0 μs
WT of J5 = 0.8 - 0.1 = 0.7 μs
WT of J6 = 0.1 - 0.1 = 0 μs
WT of J7 = 0.3 - 0.2 = 0.1 μs
Average waiting time (AvgWT) = Sum of WTs of J1 to J7/7
AvgWT = 28.9/7 = 4.13 μs

Turnaround time (TAT) of Ji = BT of Ji + Waiting time of Ji
TAT of J1 = 6.3 + 11.7 = 18.0 μs
TAT of J2 = 1.8 + 4.9 = 6.7 μs
TAT of J3 = 5.9 + 7.6 = 13.5 μs
TAT of J4 = 1.1 + 3.0 = 4.1 μs
TAT of J5 = 4.8 + 0.7 = 5.5 μs
TAT of J6 = 0.2 + 0 = 0.2 μs
TAT of J7 = 0.5 + 0.1 = 0.6 μs
Average TAT (AvgTAT) = Sum of TATs of J1 to J7/7
AvgTAT =48.6/7 = 6.94 μs

4.1.3 SJF (Preemptive)

Example:

Jobs	Burst Time	Arrival Time
J_1	6.3	2.7
J_2	1.8	1.8
J_3	5.9	0.9
J_4	1.1	2.6
J_5	4.8	0.1
J_6	0.2	0.1
J_7	0.5	0.2

Assumption: All jobs reach at time 0.

Gantt chart:

J_6	J_7	J_5	J_2	J_4	J_5	J_3	J_1

0.1 0.3 0.8 1.8 3.6 4.7 8.5 14.4 20.7

Waiting time (WT) of Ji = BT - Arrival time
WT of J1 = 14.4 - 2.7 = 11.7 μs
WT of J2 = 1.8 - 1.8 = 0 μs

Volume 3, Issue 3
June 2014

ITEE Journal

ISSN: - 2306-708X

©2012-14 International Journal of Information Technology and Electrical Engineering

WT of J3 = 8.5 - 0.9 = 7.6 µs
WT of J4 = 3.6 - 2.6 = 1.0 µs
WT of J5 = (0.8 - 0.1) + (4.7 - 1.8) = 3.6 µs
WT of J6 = 0.1 - 0.1 = 0 µs
WT of J7 = 0.3 - 0.2 = 0.1 µs
Average waiting time (AvgWT) = Sum of WTs of J1 to J7/7
AvgWT = 24.0/7 = 3.43 µs

Turnaround time (TAT) of Ji = BT of Ji + Waiting time of Ji
TAT of J1 = 6.3 + 11.7 = 18.0 µs
TAT of J2 = 1.8 + 0 = 1.8 µs
TAT of J3 = 5.9 + 7.6 = 13.5 µs
TAT of J4 = 1.1 + 1.0 = 4.1 µs
TAT of J5 = 4.8 + 3.6 = 8.4 µs
TAT of J6 = 0.2 + 0 = 0.2 µs
TAT of J7 = 0.5 + 0.1 = 0.6 µs
Average TAT (AvgTAT) = Sum of TATs of J1 to J7/7
AvgTAT =44.7/7 = 6.39 µs

4.1.4 Priority (Non-Preemptive)

Example:

Jobs	Burst Time	Arrival Time	Priority
J_1	6.3	2.7	3
J_2	1.8	1.8	1
J_3	5.9	0.9	1
J_4	1.1	2.6	0
J_5	4.8	0.1	2
J_6	0.2	0.1	2
J_7	0.5	0.2	3

Note: Lower value is treated as higher priority.

Gantt chart:

J_6	J_5	J_4	J_2	J_3	J_7	J_1
0.1 0.3	5.1	6.2	8.0	13.9	14.4	20.7

Waiting time (WT) of Ji = BT - Arrival time
WT of J1 = 14.4 - 2.7 = 11.7 µs
WT of J2 = 6.2 - 1.8 = 4.4 µs
WT of J3 = 8.0 - 0.9 = 7.1 µs
WT of J4 = 5.1 - 2.6 = 2.5 µs
WT of J5 = 0.3 - 0.1 = 0.2 µs
WT of J6 = 0.1 - 0.1 = 0 µs
WT of J7 = 13.9 – 0.2 = 13.7 µs
Average waiting time (AvgWT) = Sum of WTs of J1 to J7/7
AvgWT = 39.6/7 = 5.66 µs

Turnaround time (TAT) of Ji = BT of Ji + Waiting time of Ji
TAT of J1 = 6.3 + 11.7 = 18.0 µs
TAT of J2 = 1.8 + 4.4 = 6.2 µs
TAT of J3 = 5.9 + 7.1 = 13.0 µs
TAT of J4 = 1.1 + 2.5 = 3.6 µs
TAT of J5 = 4.8 + 0.2 = 5.0 µs
TAT of J6 = 0.2 + 0 = 0.2 µs
TAT of J7 = 0.5 + 13.7 = 14.2 µs
Average TAT (AvgTAT) = Sum of TATs of J1 to J7/7
AvgTAT =60.2/7 = 8.6 µs

4.1.5 Priority (Preemptive)

Example:

Jobs	Burst Time	Arrival Time	Priority
J_1	6.3	2.7	3
J_2	1.8	1.8	1
J_3	5.9	0.9	1
J_4	1.1	2.6	0
J_5	4.8	0.1	2
J_6	0.2	0.1	2
J_7	0.5	0.2	3

Note: Lower value is treated as higher priority.

Gantt chart:

J_6	J_5	J_3	J_2	J_4	J_2	J_3
0.1 0.3	0.9	1.8	2.6	3.7	4.7	9.7
J_5	J_7	J_1				
13.9	14.4	20.7				

Waiting time (WT) of Ji = BT - Arrival time
WT of J1 = 14.4 - 2.7 = 11.7 µs
WT of J2 = (1.8 - 1.8) + (3.7 - 2.6) = 1.1 µs
WT of J3 = (0.9 - 0.9) + (4.7 - 1.8) = 2.9 µs
WT of J4 = 2.6 - 2.6 = 0 µs
WT of J5 = (0.3 - 0.1) + (9.7 - 0.9) = 9.0 µs
WT of J6 = 0.1 - 0.1 = 0 µs
WT of J7 = 13.9 – 0.2 = 13.7 µs
Average waiting time (AvgWT) = Sum of WTs of J1 to J7/7
AvgWT = 38.4/7 = 5.49 µs

Turnaround time (TAT) of Ji = BT of Ji + Waiting time of Ji
TAT of J1 = 6.3 + 11.7 = 18.0 µs
TAT of J2 = 1.8 + 1.1 = 2.9 µs
TAT of J3 = 5.9 + 2.9 = 8.8 µs
TAT of J4 = 0 + 1.1 = 1.1 µs
TAT of J5 = 9.0 + 4.8 = 13.8 µs
TAT of J6 = 0.2 + 0 = 0.2 µs
TAT of J7 = 0.2 + 13.7 = 14.2 µs
Average TAT (AvgTAT) = Sum of TATs of J1 to J7/7
AvgTAT = 59.0/7 = 8.43 µs

4.1.6 FCFS (Shorter Job First)

Example:

Jobs	Burst Time	Arrival Time	Priority
J_1	6.3	2.7	3
J_2	1.8	1.8	1
J_3	5.9	0.9	1
J_4	1.1	2.6	0
J_5	4.8	0.1	2
J_6	0.2	0.1	2
J_7	0.5	0.2	3

Gantt chart:

J_6	J_7	J_5	J_2	J_4	J_2	J_4
0.1 0.3	0.8	1.8	2.8	3.8	4.6	4.7
J_5	J_3	J_5	J_3	J_5	J_3	J_5
5.7	6.7	7.7	8.7	9.7	10.7	11.5
J_3	J_1	J_3	J_1	J_3	J_1	
12.5	13.5	14.5	15.5	16.4	20.7	

Waiting time (WT) of Ji = BT - Arrival time
WT of J1 = (12.5 - 2.7) + (14.5 - 13.5) + (16.4 - 15.5) = 12.1 µs
WT of J2 = (1.8 - 1.8) + (3.8 - 2.8) = 1 µs
WT of J3 = (5.7 - 0.9) + (7.7 - 6.7) + (9.7 - 8.7) + (11.7 - 10.7) + (13.5 - 12.5) + (15.5 - 14.5) = 9.8 µs
WT of J4 = (2.8 - 2.6) + (4.6 - 3.8) = 1 µs
WT of J5 = (0.8 - 0.1) + (4.7 - 1.8) + (6.7-5.7) + (8.7 - 7.7) + (10.7 - 9.7) = 6.6 µs
WT of J6 = 0.1 - 0.1 = 0 µs
WT of J7 = 0.3 - 0.2 = 0.1 µs
Average waiting time (AvgWT) = Sum of WTs of J1 to J7/7
AvgWT = 30.6/7 = 4.37 µs

Turnaround time (TAT) of Ji = BT of Ji + Waiting time of Ji
TAT of J1 = 6.3 + 12.1 = 18.4 µs
TAT of J2 = 1.8 + 1 = 2.8 µs
TAT of J3 = 5.9 + 9.8 = 15.7 µs
TAT of J4 = 1.1 + 1 = 2.1 µs
TAT of J5 = 4.8 + 6.6 = 11.4 µs
TAT of J6 = 0.2 + 0 = 0.2 µs
TAT of J7 = 0.5 + 0.1 = 0.6 µs
Average TAT (AvgTAT) = Sum of TATs of J1 to J7/7
AvgTAT = 51.2/7 = 7.31 µs

4.2 Queuing model

In queuing model, waiting queues are analyzed. A model is built to predict the lengths and waiting times of a queue. [16] Little's formula is manipulated to predict the required queue length and waiting time:

$$n = \lambda w$$

In this formula: Average queue length is denoted by n
Average arrival time is denoted by λ
Average waiting time is denoted by w

In the instant case, apply above formula:

(Assuming λ = 0.5)

S #	Algorithm	Average waiting time (w)	Average queue length (n)
1	FCFS (NP)	11.93	5.97
2	SJF (NP)	4.14	2.07
3	SJF (P)	3.43	1.72
4	Priority (NP)	5.66	2.83
5	Priority (P)	5.49	2.75
6	FCFS (SJF)	4.37	2.19

4.3 Simulation model

It is the process of producing a prototype called standard functional model to find its functioning in the real world. Simulation modeling is, therefore, used to help understand the particular outcomes. However, this model has not been test in the paper.

4.4 Implementation model

The algorithm is tested in real mode in this model. Random variations are keys to this model. However, this model has not been implemented in this paper.

5. ANALYSIS CHART

S #	Algorithm	Avg WT	Avg TAT
1	FCFS (NP)	11.93	14.87
2	SJF (NP)	4.14	7.09
3	SJF (P)	3.43	6.39
4	Priority (NP)	5.66	8.60
5	Priority (P)	5.49	8.43
6	FCFS (SJF)	4.37	7.31

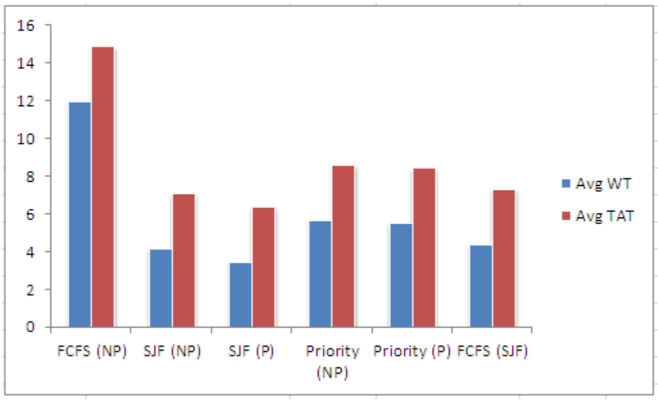

Figure 7: Comparison Graph of Average Waiting Time (µs) and Average Turnaround Time of Scheduling Algorithms

6. CONCLUSION

An analysis of some short terms scheduling algorithms has been presented in this paper. The algorithms have been analyzed using deterministic and queuing model. From our calculations it is evident that shorter job first (preemptive) has the minimum waiting time and turnaround time. Hence, it is the best algorithm among others when both waiting time and turnaround time is considered.

REFERENCES

[1]. Silberschatz A, Galvin PB, Gagne G. Operating System Concepts 2012, 8th edition, Wiley India.
[2]. Milan Milenkovic, "Operating Systems Concepts and Design", McGRAM-HILL, Computer Science Series, second edition.
[3]. Sabrian F, CD Nguyen, S Jha D Platt, F Safaei. Processing resource scheduling in programmable networks. Computer communication 2005. 28:676-87
[4]. http://www.tutorialspoint.com/operating_system/os_process_scheduling.htm
[5]. Geol N. A Comparative Study of CPU Scheduling Algorithms. International Journal of Graphics & Image Processing 2012. 245:251.
[6]. Saleem U, Javed MY. Simulation of CPU Scheduling Alogrithm. 0-7803-6355-8/00/$10.00@2000 IEEE.

[7]. Tanenbaum AS. Modern Operating System 2008. Prentice Hall.
[8]. Dhore A. Operating Systems. Technical Publications.
[9]. Bandarupalli SB, Nutulapati NP, Varma PS. A Novel CPU Scheduling Algorithm–Preemptive & Non-Preemptive. International Journal of Modern Engineering Research (IJMER) 2012. 6:4484-90.
[10]. http://www.geekinterview.com/Interview-Questions/Operating-System/Windows-Unix
[11]. Shrivastava M. Analysis and Review of CPU Scheduling Algorithms. IJSER 2014. 3:132-34
[12]. http://www.tutorialspoint.com/operating_system/os_process_scheduling.htm
[13]. Omar MA, Zwaid MJ. CPU Scheduling: A scientific research. University of Baghdad, Iraq.
[14]. Kumar A, Rohil H, Arya S. Analysis of CPU Scheduling Policies through Simulation. IJRCSSE 2013. 5:1158-62.
[15]. http://thelawdictionary.org/deterministic-model/
[16]. http://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/5_CPU_Scheduling.html

AUTHOR PROFILES

1. Muhammad Usman is doing MS (CS) degree from FUUAST, Islamabad, Pakistan. He is serving as Lecture in the department of Computer Science, Islamabad Model College for Boys, I-8/3, Islamabad, Pakistan, since 2011. His research interests include Operating System, DBMS, Software Development/ Engineering, Software Project Management and Data Mining.

2. Aamir Iqbal is doing MS (CS) degree from FUUAST, Islamabad, Pakistan. He is serving as Programmer, Department of MIS in Federal Urdu University or Arts Science and Technology, Islamabad, Pakistan, since 2013. His research interests include Visual Programming, Software Engineering, Operating System and Software Quality Management.

3. Ehsan Ahmed is doing MS (CS) degree from FUUAST, Islamabad, Pakistan. He is serving as SSC (CS) in Government High School, Thoha Khalsa, Kahuta, Pakistan, since 2010. His research interests include Network Programming, Network Security, Operating System and Analysis of Algorithm.

4. Shaukat Ghani is doing MS (CS) degree from FUUAST, Islamabad, Pakistan. His research interests include Wireless Network, Network Security, Operating System and Analysis of Algorithm.

5. Muhammad Adnan Khan received his MS (Electronic Engineering) degree from IIU, Islamabad, Pakistan, in 2010. He is PhD scholar at ISRA University, Islamabad. He has a number of publications in the field of receiver optimization, digital signal processing, space time coding, digital communication and operating system.