INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Process Scheduling Simulation
Team Deadlock

1. Introduction:

Process Scheduling is important and yet an integral part of the Operating System. Basically, process scheduling algorithms decide which one of the processes from the ready queue will get a chance to execute. Different types of Computing Systems require different types of Scheduling algorithms. Some of the Computer systems have the requirement that the processes get executed according to priority. For Example, the College Server. In such systems, priority scheduling algorithms are used. We can define the priorities and the processes will get executed based on priorities. Some computing systems support user interaction. Therefore, such systems have as less response time as possible. This class of systems uses Round Robin Scheduling Algorithm. All these algorithms are measured with respect to an Ideal Algorithm, namely Shortest Job First(SJF) (Non-Preemptive) or Shortest Remaining Time First (SRTF) (Preemptive). SJF and SRTF are measuring sticks. These algorithms can't be implemented practically but can be used to measure the performance of other algorithms in relation to it. Of course one can argue that we can implement them by estimating their bursts time but in most situations, it is not used. It is just a theoretical algorithm. Most modern operating systems use a hybrid of these scheduling algorithms to get the best of all worlds. Let us take a case study of the Ubuntu Operating System. It uses the following scheduling algorithms:

a) Earliest Deadline First(EDF): Execute the task which has indicated that it should execute first.
b) First In First Out(FIFO): It schedules the algorithm based on arrival time. It is non preemptive in nature.
c) Round Robin(RR): It is similar to FIFO. It maintains a preemptive and schedules the algorithm in cycles with fixed cycle length. Each process executes for a fixed time quantum and is pushed again to the ready queue if its execution is not over.

d) Magic: High-priority tasks should be run for longer periods of time than lower-priority jobs. The operating system has the ability to adjust the priority of a task based on its behavior. This is the default scheduler in a typical Linux system, and it is used for practically everything.

Let us now look at the working of these algorithms and interpret the statistical data extracted from these algorithms. All the codes are implemented in the C++ programming language and the plots are made in Python.

2. Problem Description:

For this process scheduling simulator, the following algorithms are needed to implement:

- First Come First Serve
- Shortest Job First Preemptive
- Shortest Job First Non-Preemptive
- Priority Preemptive
- Priority Non-Preemptive
- Round Robin
- Multi-level queue
- Multi-level feedback queue

These above algorithms were to be stimulated under different parameters. Processes are of different types . Most commonly they are either CPU bound or IO bound . CPU bound processes require more CPU time to execute, while IO bound processes require IO devices to execute. CPU scheduling is extremely necessary, as it makes a multi-tasking environment that keeps the CPU and I/O devices busy at all times which results in increased CPU utilization. So choosing a perfect optimized scheduling algorithm plays an important role in CPU's performance. So we have stimulated different types of processes under different algorithms ,and compared each other to get an idea which scheduling algorithm to choose.

3. Solution Approach

First Come First Serve Scheduling: This scheduling algorithm schedules the process which comes earlier. To implement this algorithm, processes are sorted according to the arrival times. And then, by iterating all the processes, completion time, Turnaround time, and waiting times can be found.
Input requirement : PID, Pname,AT(Arrival Time), BT(Burst Time), PT(Process Type).
Output : RT(Response Time), CT(Completion Time), TAT(Turn around Time), WT(Waiting Time).

Round Robin Scheduling: This scheduling algorithm schedules the process for a particular time quantum. As the time quantum of that process completes, it schedules the next process in a the queue. To implement this algorithm, the Queue of processes is to be maintained, and each process in the queue is given the same time quantum to execute.
Input requirement : PID, Pname,AT(Arrival Time), BT(Burst Time), PT(Process Type),Time Quantum.
Output : RT(Response Time), CT(Completion Time), TAT(Turn around Time), WT(Waiting Time).

Shortest Job First (SJF): SJF schedules the processes according to burst time. The job whose burst time is less gets higher priority. If two processes have the same burst time, then the deciding factor is arrival time. If two processes have the same burst time as well as arrival time, then the process with less ProcessID gets higher priority. This scheduling algorithm is non preemptive in nature.
Input requirement: PID, Pname, AT(Arrival Time), BT(Burst TIme), PT(Process type).
Output: RT(Response Time), CT(Completion Time), TAT(Turn around Time), WT(Waiting Time).

Shortest Remaining Time First(SRTF): SRTF is a preemptive version of the SJF scheduling algorithm. SRTF schedules the processes according to burst time. The job whose burst time is less gets higher priority. If two processes have the same burst time, then the deciding factor is arrival time. If two processes have the same burst time as well as arrival time, then the process with less ProcessID gets higher priority. If at any time a process with less burst time arrives in the ready queue, then it gets scheduled.
Input requirement: PID, Pname, AT(Arrival Time), BT(Burst TIme), PT(Process type).
Output: RT(Response Time), CT(Completion Time), TAT(Turn around Time), WT(Waiting Time).

Multilevel Queue Scheduling: In this Scheduling method there are multiple levels available in the form of queues in which we can use different scheduling algorithms and also use another scheduling method to maintain all queues.
In this, we are personally using the Shortest Job First Algorithm(SJF) at each level because we know that the waiting time of SJF is less. And also preferring SJF among each level.
Input requirement: PID, AT(Arrival Time), BT(Burst Time), PT(Process Type).
Output: RT(Response Time), CT(Completion Time), TAT(Turn around Time), WT(Waiting Time).

Multilevel Feedback Queue Scheduling: This is the same as multilevel Queue Scheduling only difference is they are virtually connected with each other, so we are manipulating processes among different levels, and for this, we are personally using Round robin algorithm with different Time Quantum at each level. At the last level we are using the FCFS fashion algorithm for processes. Aging is a technique that is used to allocate/move processes into higher level Queues so that they do not get trapped into starvation.
Input Requirements: PID, AT, BT, PT, TQ(Time Quantum)
Output :RT(Response Time), CT(Completion Time), TAT(Turn around Time), WT(Waiting Time).

At the end of computations of Scheduling methods, we are comparing these processes with the help of graphs in Python.
For that we might be using matplotlib or some other library of python.

Priority Scheduling Algorithm: In this scheduling algorithm each process is given a priority. The process with the highest priority is executed first and so on.
Processes with the same priority are executed on the first come first serve basis.
This version of priority scheduling algorithm is non preemptive as the process that is assigned the cpu will not leave the cpu till it is completed. A major drawback of this scheduling is infinite blocking or starvation. Aging is the method used to tackle this problem by gradually decreasing priority of the process.

Input Requirements : PID, AT(Arrival Time), BT(Burst time), PT(Process Type), Priority
Output :RT(Response Time),CT(Completion Time), TAT(TurnAround Time), WT(Waiting Time)

Preemptive Priority Scheduling Algorithm : Preemptive priority scheduling algorithm is a version of priority scheduling algorithm in which a currently scheduled process can be preempted on the arrival of a higher priority process. The waiting time for the higher priority process is always zero. It is used in applications where higher priority processes can not be kept waiting.

Input Requirements : PID, AT, BT, PT, Priority
Output :RT(Response Time), CT(Completion Time), TAT(Turn around Time), WT(Waiting Time).

4. Work Distribution

Everyone has contributed in plotting the different graphs.

Roll Number	**Name**	**Contribution**
2021201006	Darshan Tripathi	FCFS,RR
2021201011	Alok Jain	Preemptive Priority,Non-preemptive priority
2021201077	Hussain Kagdi	Shortest Job First, Shortest Remaining Time First
2021202009	Marmik Patel	Multilevel Queue Scheduling And Multilevel Feedback Queue Scheduling

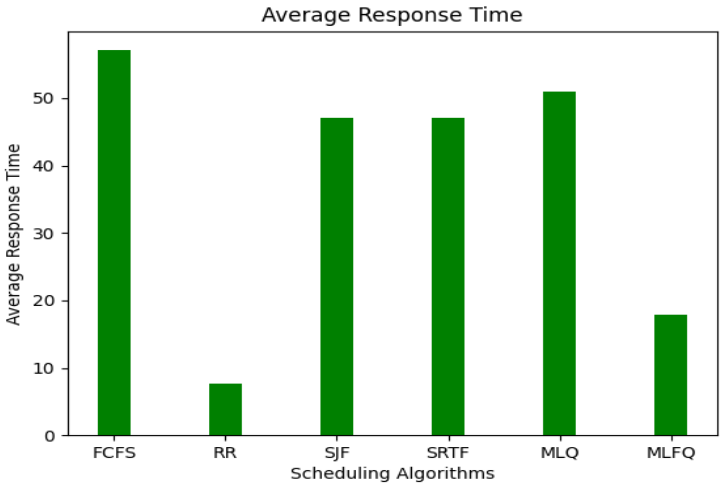
5. Statistical Visualization

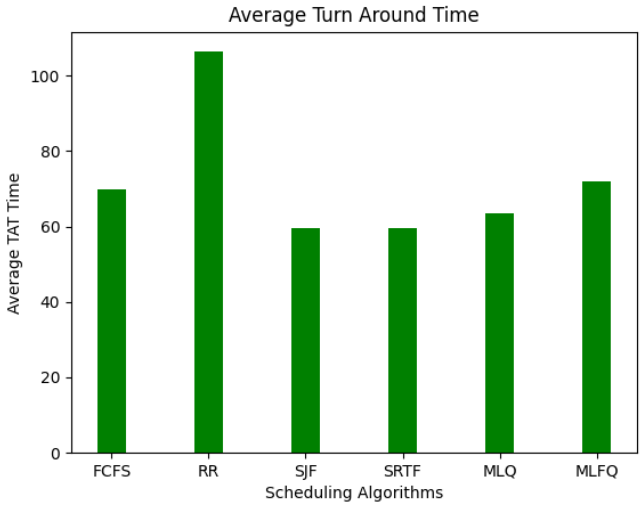
We have generated graphs for visualization and comparison of different scheduling algorithms.

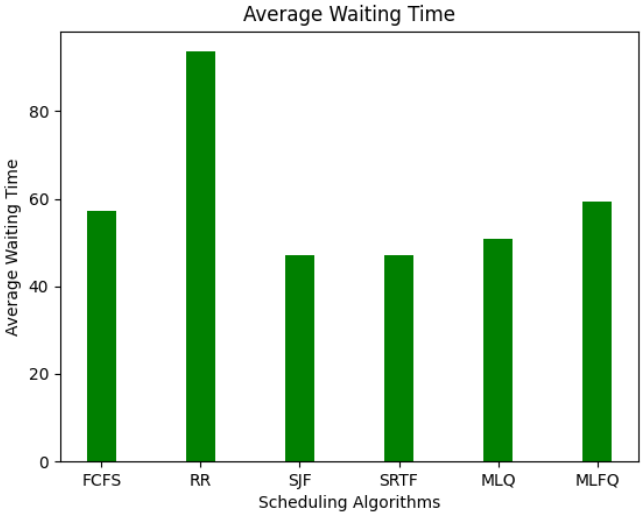
Sample Input:

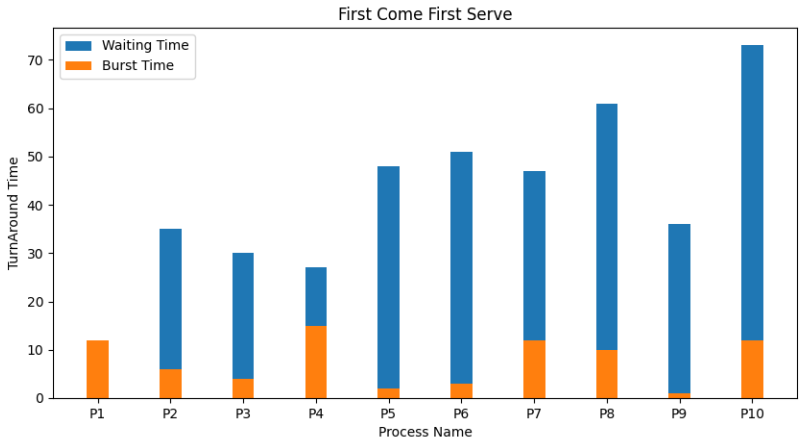
PID	PNAME	AT	BT	BOUND
0	P1	0	12	CPU
1	P2	2	6	CPU
2	P3	1	4	CPU
3	P4	0	15	CPU
4	P5	4	2	CPU
5	P6	4	3	CPU
6	P7	3	12	CPU
7	P8	4	10	CPU
8	P9	2	1	CPU
9	P10	4	4	CPU

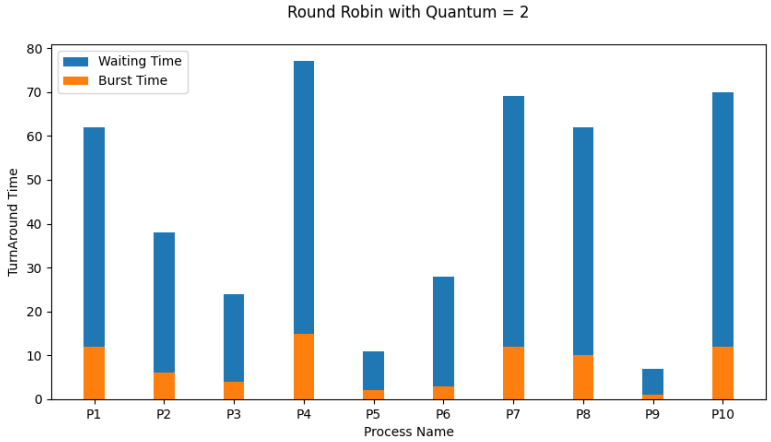
Average RT, Average TAT and Average WT of all Scheduling Algorithm

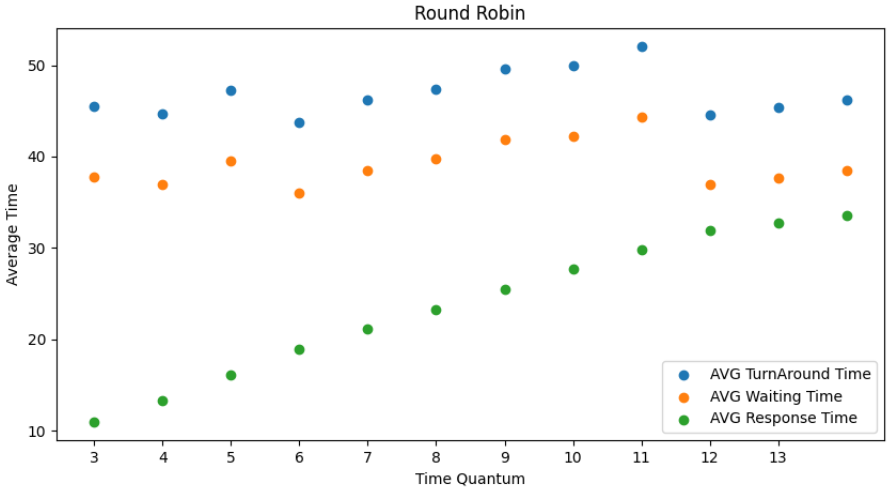


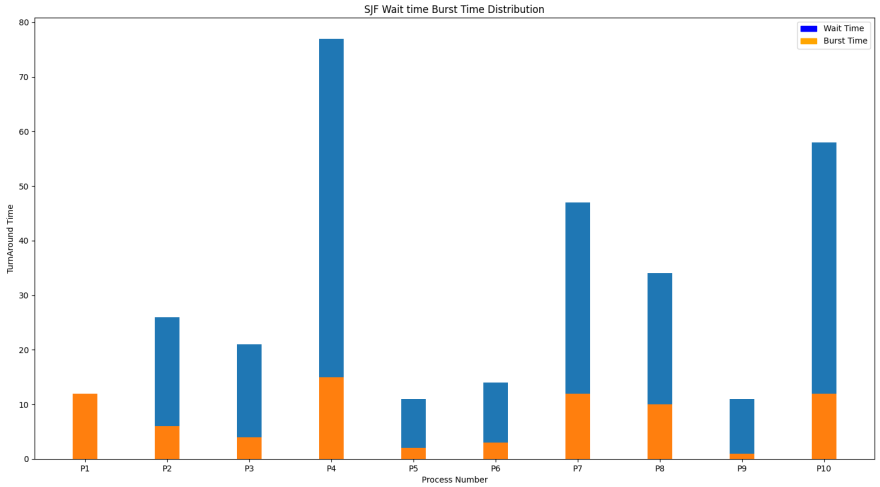


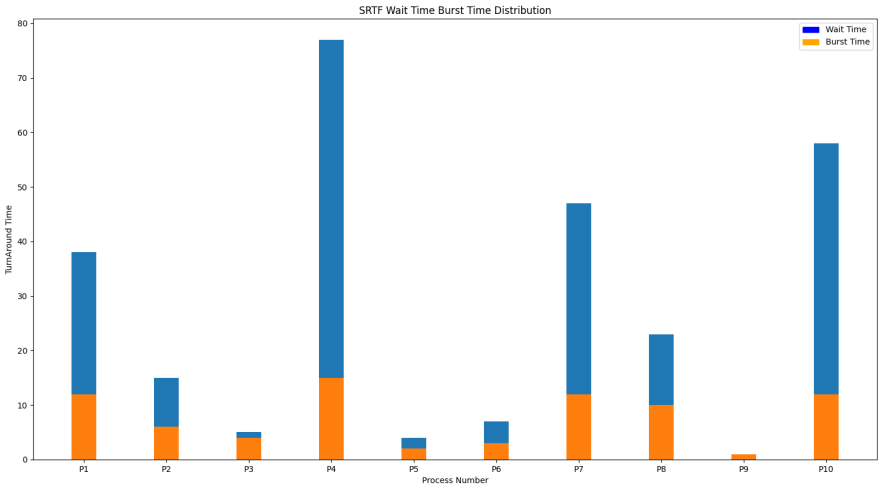


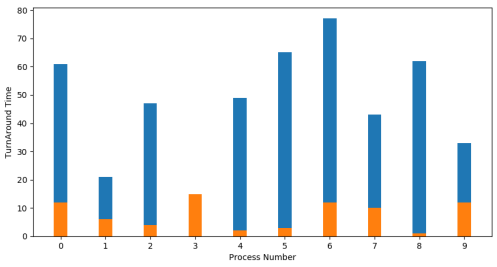

First Come First Serve
Waiting Time
Burst Time
TurnAround Time
Process Name
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10

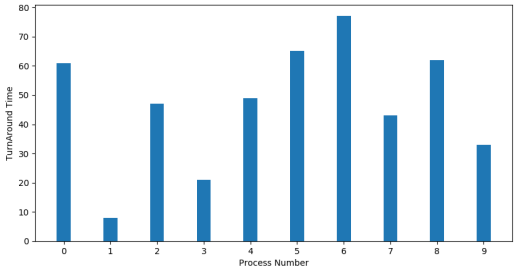

Round Robin with Quantum = 2
Waiting Time
Burst Time
TurnAround Time
Process Name
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10

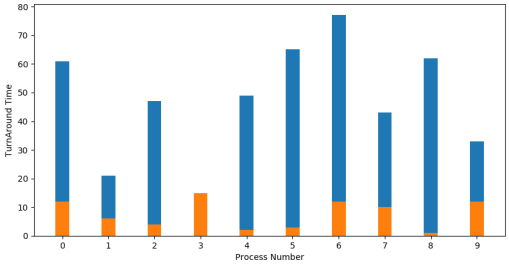

Round Robin
Average Time
Time Quantum
AVG TurnAround Time
AVG Waiting Time
AVG Response Time

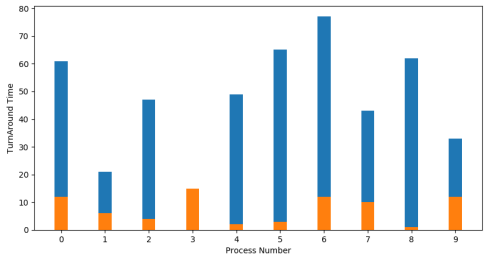

SJF Wait time Burst Time Distribution
Wait Time
Burst Time
TurnAround Time
Process Number
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10

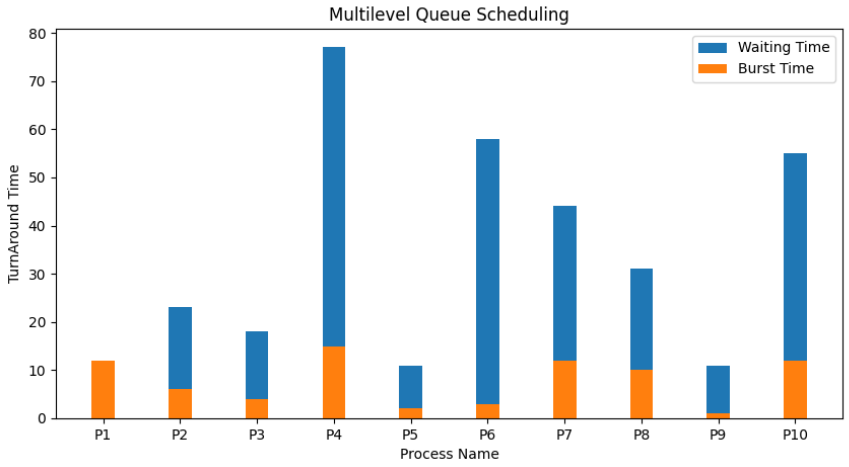

SRTF Wait Time Burst Time Distribution
Wait Time
Burst Time
TurnAround Time
Process Number
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10

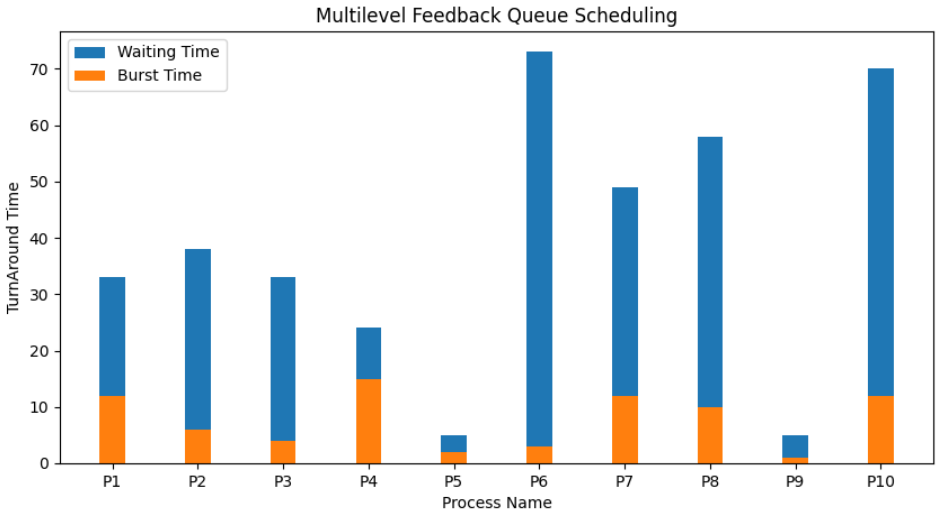

TurnAround Time
Process Number

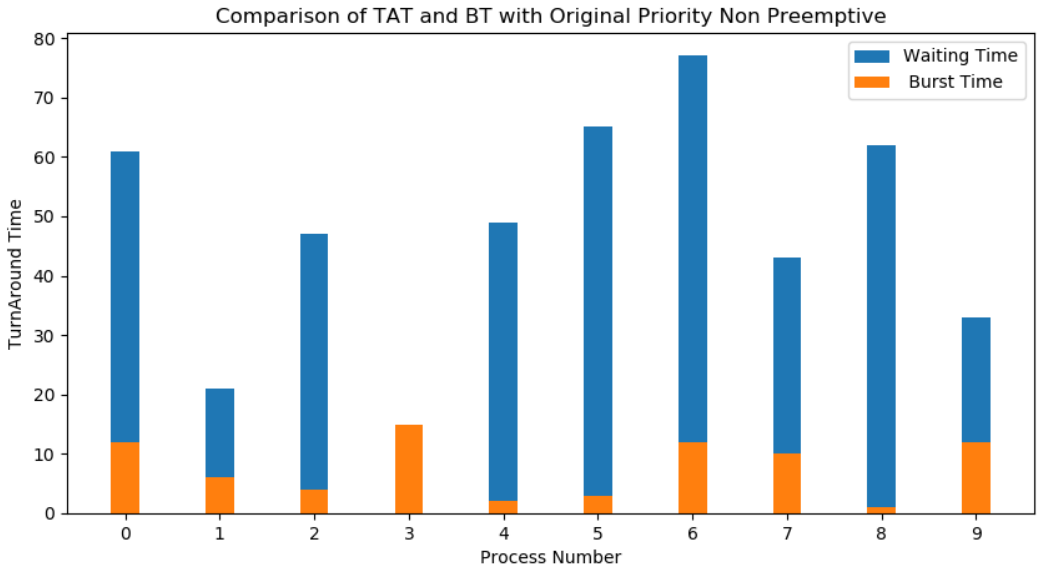

TurnAround Time
Process Number

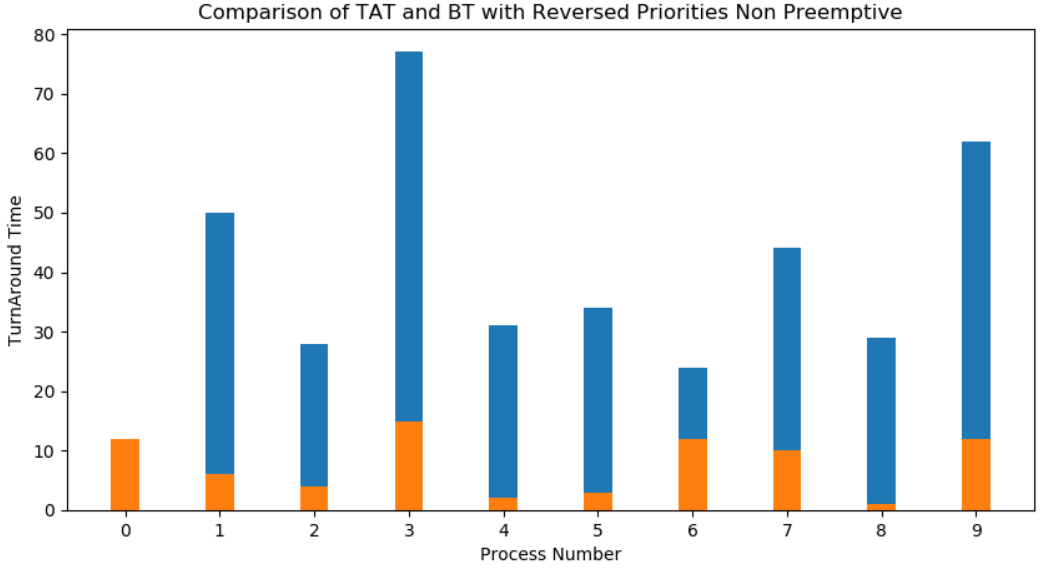

TurnAround Time
Process Number

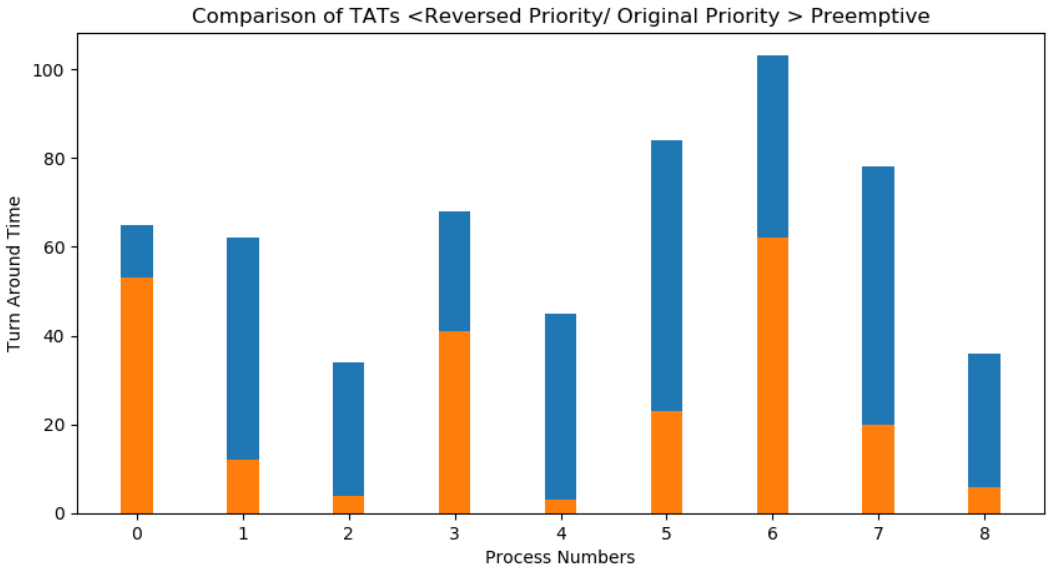

TurnAround Time
Process Number

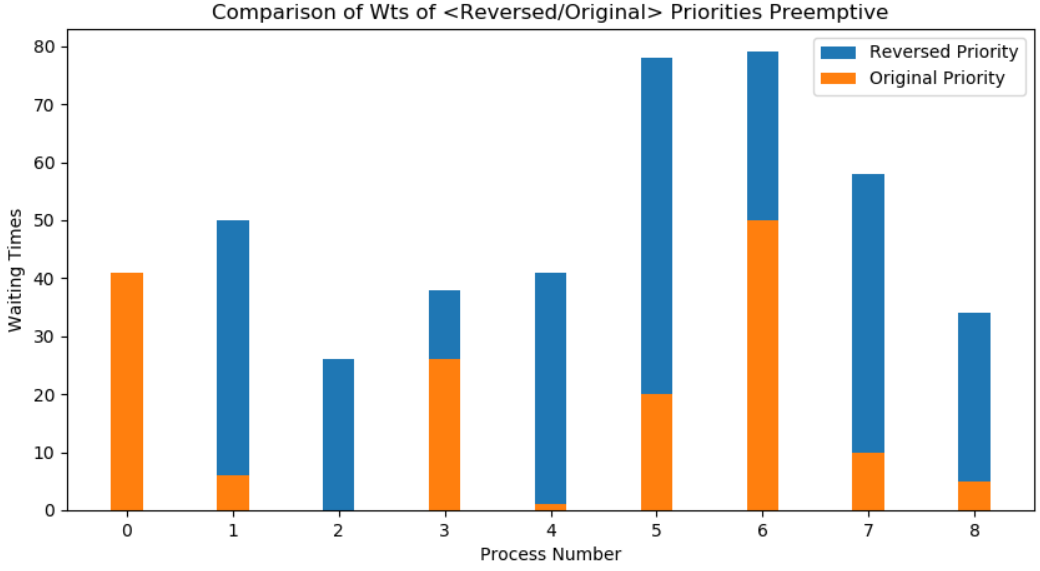

Multilevel Queue Scheduling
Waiting Time
Burst Time
TurnAround Time
Process Name
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
0
10
20
30
40
50
60
70
80


Multilevel Feedback Queue Scheduling
Waiting Time
Burst Time
TurnAround Time
Process Name
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
0
10
20
30
40
50
60
70

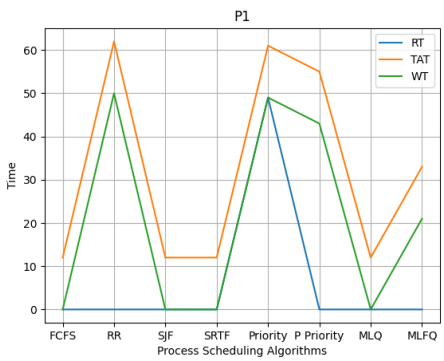

Comparison of TAT and BT with Original Priority Non Preemptive
Waiting Time
Burst Time
TurnAround Time
Process Number
0
10
20
30
40
50
60
70
80
0
1
2
3
4
5
6
7
8
9

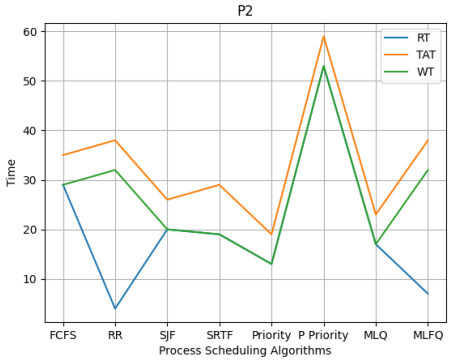

Comparison of TAT and BT with Reversed Priorities Non Preemptive
TurnAround Time
Process Number
0
10
20
30
40
50
60
70
80
0
1
2
3
4
5
6
7
8
9

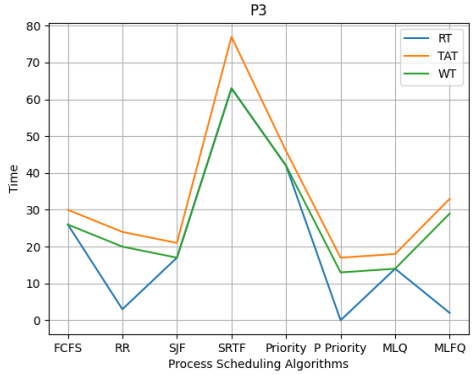

Comparison of TATs <Reversed Priority/ Original Priority > Preemptive
Turn Around Time
Process Numbers
0
20
40
60
80
100
0
1
2
3
4
5
6
7
8

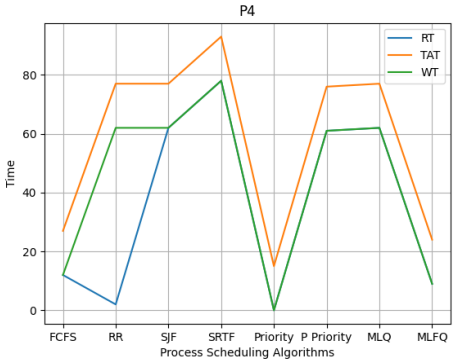

Comparison of Wts of <Reversed/Original> Priorities Preemptive
Waiting Times
Process Number
Reversed Priority
Original Priority
0
10
20
30
40
50
60
70
80
0
1
2
3
4
5
6
7
8

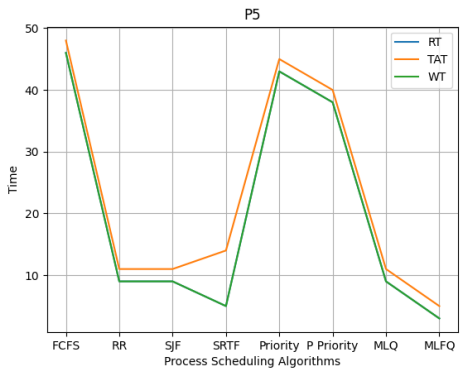
Process wise Scheduling Algorithms

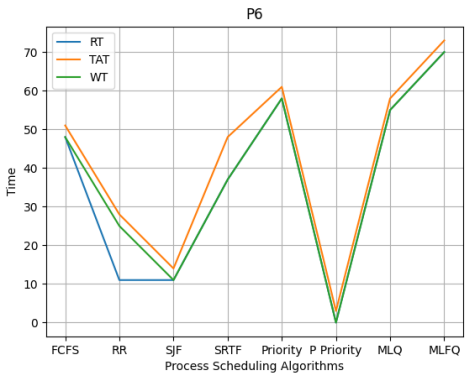


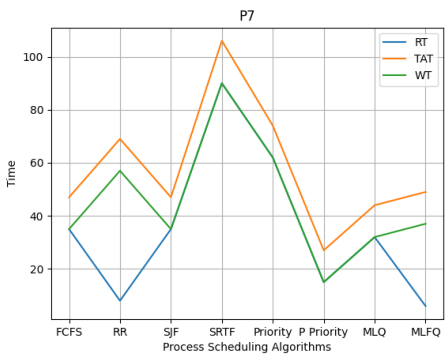


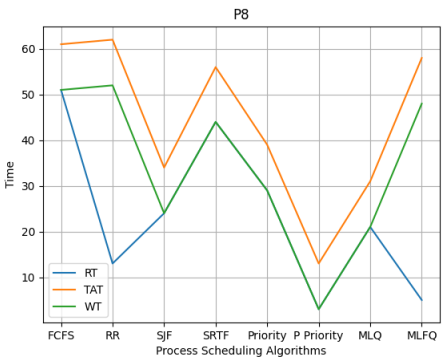


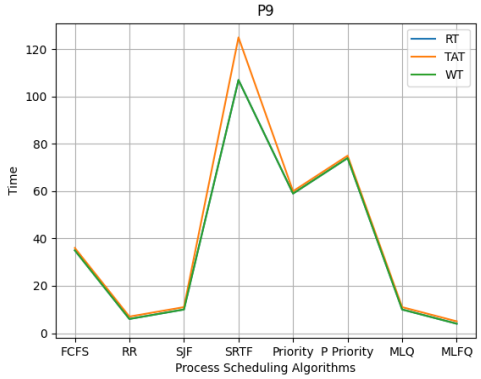


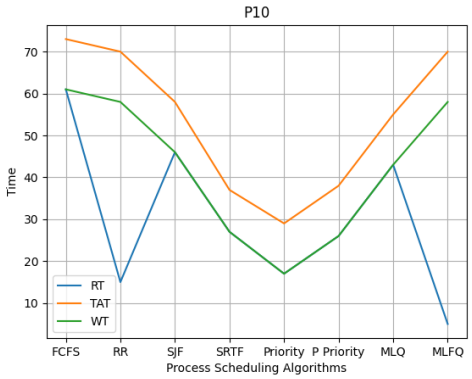












6. Conclusion

To evaluate all 8 algorithms, we had taken a set of 10 processes as a sample ,on which all the algorithms were executed.In turnaround times, they all behaved as expected. The average TAT of SJF is the lowest as compared to all others. Average TAT for Round Robin aws the highest among all. The average response time for Round Robin was lowest in all cases in our simulation.Long burst time processes will wait in the ready queue under SJF scheduling. Under FIFO, longer average response times could also be observed depending on the order in which jobs are received. Average wait time was found to be shortest for SJF and longest for RR. In Multilevel Queue Scheduling, there are different queues which is having different priority like all the processes in queue 1 will be executed first and then other which makes the processes in lower levels into starvation so to overcome/reduce this problem we have introduced multilevel feedback queue scheduling which solves up to some extent.