# OPERATING SYSTEMS PROJECT

Group - 19

Muskan Matwani - 1741027
Yesha Shastri - 1741035

## CPU SCHEDULING ALGORITHMS
## (FCFS, RR, DRR)

Table of Contents

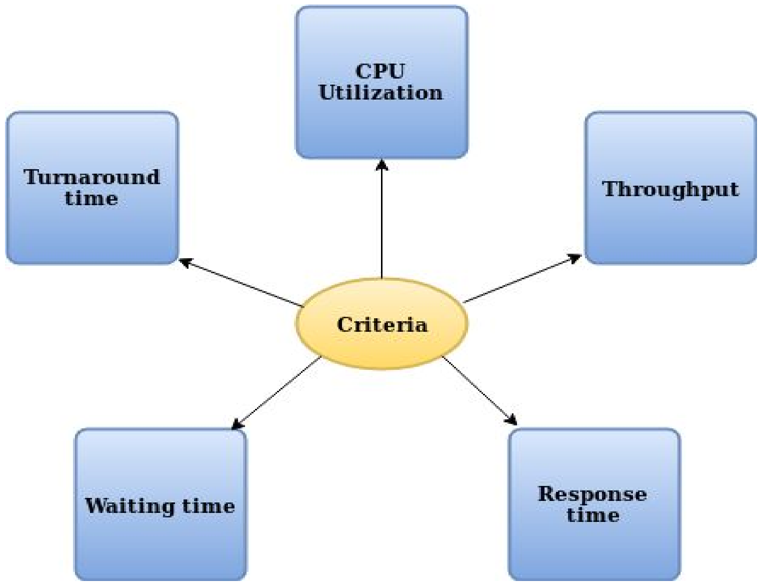
## Ⅰ. What is scheduling?

Operating system has to perform several processes therefore it is necessary for the operating system to allow the processes to execute in a way that avoids wastage of CPU time. This has made way for multi-programming which allows multiple processes to run together. Scheduling the processes will help improve throughput and system utilisation. Now, in order to choose from a large pool of processes, the scheduler takes the decision of which process will be executed next based on scheduling algorithms.

There are three kinds of scheduler jobs:

1. Long term scheduler - The decision of adding a process from new state (creation of processes) to ready state (process loaded into main memory) is taken by long term scheduler.
2. Short term scheduler - The decision of adding a process from ready state to running state is made by short term scheduler. After the process has been scheduled, dispatcher will load the process on the CPU for ready to running state. This scheduler is majorly responsible for enhancing CPU performance and improving execution rate of the processes.
3. Medium term scheduler - The decision of swapping a process from one state to another is made by medium term scheduler. The need for swapping may arise when let's say a process wants to move from blocked state to ready state or whenever it wants to switch from one state to another.

## Ⅱ. Criteria to determine the efficiency of a scheduling algorithm



An algorithm is chosen which is able to perform best in accordance with the below mentioned trends:

- ➢ Maximise throughput - It allows for efficient use of system resources like CPU and I/O devices. It also minimizes overhead which includes context switching.
- ➢ Minimize average response time - The CPU should respond to processes as soon as they arrive.
- ➢ Minimize waiting time - Amount of time a process will have to spend in order to finish its execution should be minimized.

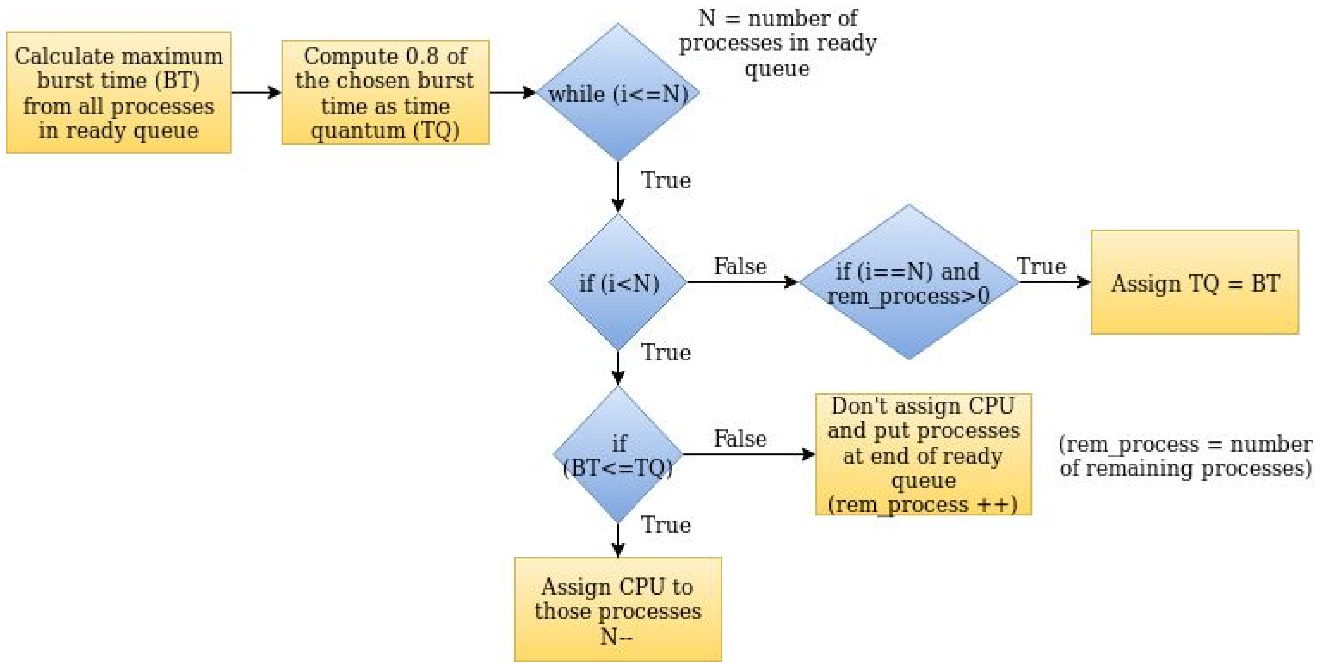
## Ⅲ. Scheduling algorithms implemented in project

- ➢ **Round Robin** - It is a preemptive scheduling algorithm which allows processes to get executed in a cyclic manner. The preemption is based on a fixed time slice which is called time quanta. Every process is allowed to execute for the given time quantum and is preempted when the time quantum gets over. The processes having their burst time lesser than time quantum can finish their execution in the allotted time whereas when the process fails to complete execution within a time quantum then it is sent back to the ready queue. This algorithm is utilized in time-sharing systems.
  **Fairness** to all process, **reduction in starvation** are the major advantages of using this algorithm. On the other hand, more number of **context switches** can render overhead and waste CPU time. Additionally, choosing an **appropriate time slice** is a critical task.

- ➢ **First Come First Served** - It is a non-preemptive algorithm in which jobs will be executed by the CPU in the order in which they arrive in the ready queue. This algorithm might lead to **starvation** since shorter jobs arriving later will have to wait for longer jobs to complete execution. Another disadvantage is that **CPU-bound processes** will suffer as they take relatively longer to execute as opposed to I/O bound processes. On the better part, it is **easy to implement** and fairly simple.

- ➢ **Dynamic Round Robin** - As choosing a constant time quantum is a difficult task, a more advanced approach is to choose a time quantum dynamically. The selected time quantum can change after the arrival of the next process in ready queue or could be just after a cycle. The time quantum in this algorithm is chosen in a way which reduces the number of context switches with lesser average waiting and turnaround times. The methodology devised to compute the time quantum is as shown below -

Time quantum calculation for Dynamic Round Robin:


Calculate maximum burst time (BT) from all processes in ready queue
Compute 0.8 of the chosen burst time as time quantum (TQ)
while (i<=N)
N = number of processes in ready queue
True
if (i<N)
False
if (i==N) and rem_process>0
True
Assign TQ = BT
True
if (BT<=TQ)
False
Don't assign CPU and put processes at end of ready queue (rem_process ++)
(rem_process = number of remaining processes)
True
Assign CPU to those processes N--

## Ⅳ. Use of OS System Concepts in our project

There are three queues in our implementation - New, Ready and Ready Suspend.

- ➢ Long Term Scheduler - We have implemented a thread which is generating random processes and they are getting adding dynamically to the new queue. The size of new queue is limited to 8. If the ready queue is filled to its maximum capacity, the processes are added to the ready suspend queue. The adding of processes from new queue to ready queue occurs simultaneously with addition of processes to new queue.
- ➢ Short Term Scheduler - The removal of processes from the ready queue to acquire the running state is done by the short term scheduler. Since the maximum capacity of the ready queue is constrained to 5, any new processes coming from the new queue will be added to the ready suspend queue when maximum capacity has been reached.
- ➢ Medium Term Scheduler - This scheduler will take care of swapping between ready queue and ready suspend queue. Whenever the ready queue is not full, new processes will be added from either the new queue or the ready suspend queue.

Concepts of concurrency are utilised with the help of synchronised blocks and wait and notify.

- ➢ Synchronisation - The addition and removal of processes from the queues takes place inside a synchronised block which helps to ensure concurrency since a single queue cannot be accessed together for both the operations - addition and removal.

- ➢ Wait and Notify - When the queues are full, they will have to wait till at least one of the processes is removed and then only a new process could enter the queue. Notify is used to send a signal to the consumer queue that the queue from which it wants to consume is not empty and it can start removing the processes.

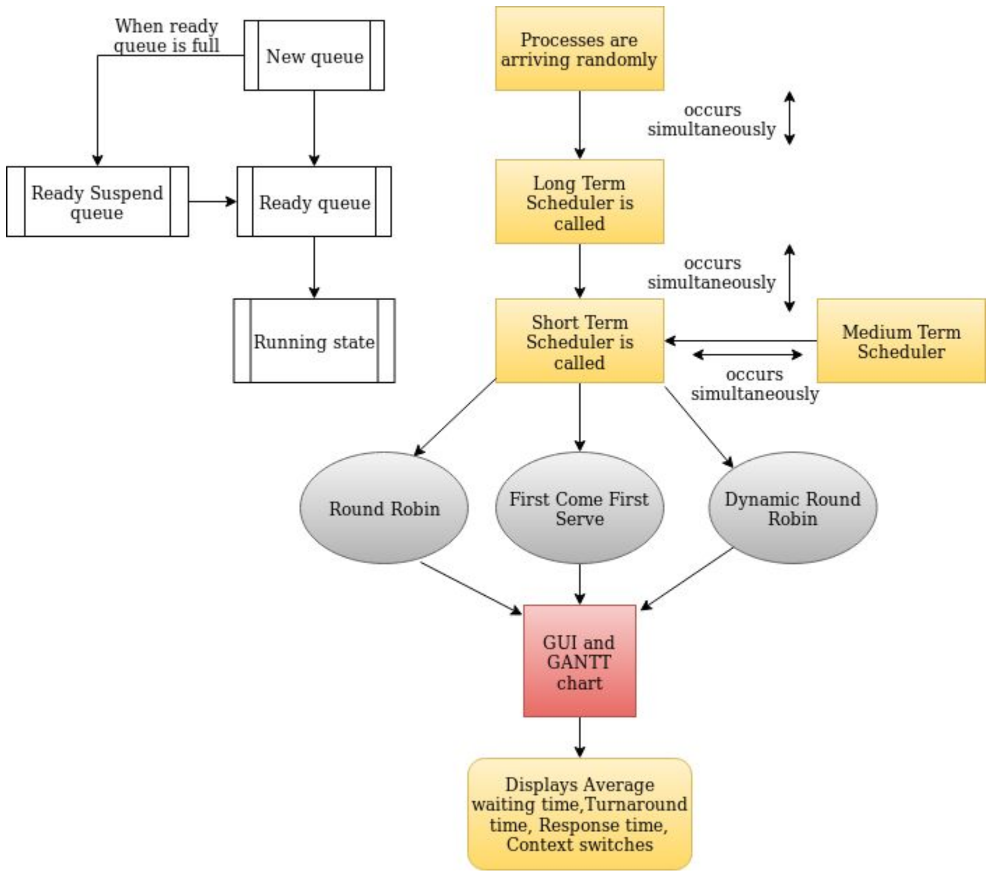
## V. Working Flow

Major java classes and their description:

1. **DRR** - This class is used to implement the logic of Dynamic Round Robin. It inherits the Scheduler class and contains two functions; one is 'process' - It implements the main logic of DRR and other 'checkForProcess' which is used to add processes that arrive till current time.
2. **MainThreads** - This class comprises of three methods, 'makeProcess' - Generates random arrival time and burst time, 'longTermScheduler' - implements long term scheduling logic and 'shorttermScheduler' - implements short term scheduling logic.
3. **FCFS** - This class implements the logic of First Come First Served and inherits Scheduler class. Contains the 'process' method to implement the logic.
4. **RR** - This class implements the logic of Round Robin and inherits the Scheduler class. Contains the 'process' method to implement the logic.
5. **Scheduler** - Contains getter and setter for computing the results like average waiting time, average turnaround time and so on.
6. **ProcessState** - This class is used to create a row for each process with appropriate comparison metrics.
7. **Event** - A class ideally made to store the details like name of process, start time and finish time.
8. **GUI** - A class to display graphical user interface for easy interaction with a user. It has a drop down list to choose an algorithm, after pressing display button the calculations of the chosen algorithm is displayed.

**Flow Description**

This project primarily creates instances of all the algorithms present - that are - First Come First Serve, Round Robin with different time quantums, and Dynamic Round Robin. Then, it creates 3 separate threads - for making processes, long term scheduler and short term scheduler. First thread makes process with random parameters assigned to it such as arrival time and burst time. Burst time is assigned randomly such that we get a mix of short burst and long burst processes in order to compare various algorithms. And then, makeProcess thread adds this process into new queue, whereas long term scheduler waits till there is at-least one process entry in the new queue. This can be accomplished by using 'synchronised' inbuilt command to facilitate only one thread to access the new queue. Then, long term scheduler extracts the processes residing inside new queue and adds them into ready queue one by one. If the ready queue size is full indicating in shortage of memory available, it adds that process into ready suspend queue. Also, each time a process is extracted from new queue, first thread is notified to fill in other processes into new queue. However, short term scheduler waits till there is at least one entry in the ready queue and then processes all the algorithms according to the processes present. Hence, now RR, FCFS, and DRR are called and starts processing. As soon as there is one empty space in the ready queue caused by removal by short terms scheduler, processes in the ready suspend queue are added in the ready queue by the medium term scheduler. This process executes till both the ready suspend and ready queues are empty indicating the end of the execution of all processes and then, the project displays Average waiting time, Average Turn Around time, Average response time and number of context switches of each algorithm. Additionally, it displays waiting time, turn around time and response time for each process in all algorithms. Furthermore, GUI is implemented to display the details of respective algorithms along with chart that shows the order of execution of processes for each algorithm.



## Ⅵ. Assumptions

- ➢ Maximum number of processes in new queue at a time is 8.
- ➢ The fixed size of the ready queue is taken as 5.
- ➢ It takes 3 different time quantums with values - 5, 20, 90. Moreover, we can extend and update to any value.
- ➢ The process creation includes processes with both short burst time and large burst times.

## Ⅶ. Limitations and Efforts put to resolve them

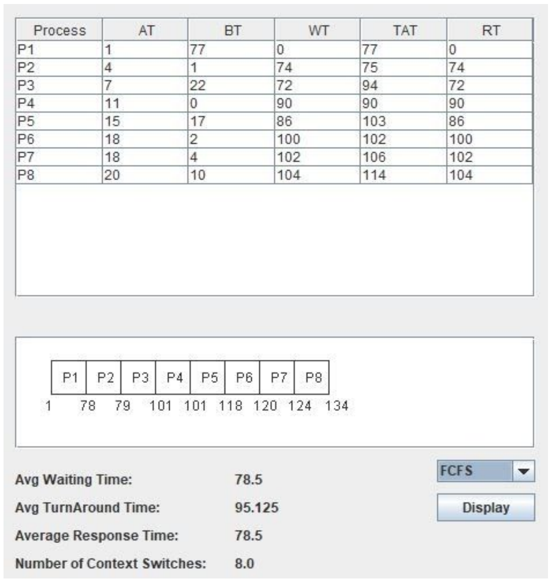
- ➢ When the ready queue is not full, new processes are added from either new queue or ready suspend queue randomly. We do not have control over which process will come first from either of them.
- ➢ Short term takes the processes from ready queue to execute when the ready queue is at its maximum size. Problem faced: Whenever executing ready process, we don't know the next process arrival time, and after processing with short term, current time exceeds till the ready process finishes, and thus, next process waiting time shoots up. Problem is we can't determine priorily about what next process's arrival time would be.

## Ⅷ. Comparison between FCFS, RR and DRR

**FCFS vs RR:**

1. Average Waiting Time and Average Turnaround Time of FCFS is more than RR. In our case, there is a mix of short burst time and long burst time processes.This poses a major drawback for FCFS for short burst processes as they have to wait longer to execute for small time and those processes then lead starvation and significant increase in the average waiting and turn around time.RR overcomes this problem and execute each process with the specified time quantum which lessens the waiting and turnaround time.

**FCFS**

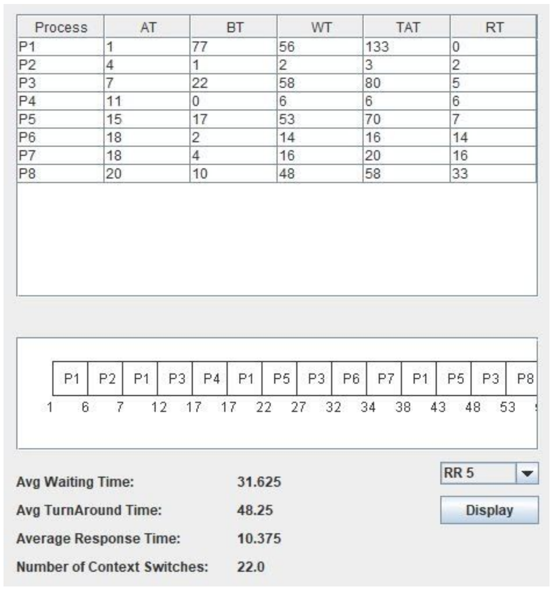


| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 1 | 77 | 0 | 77 | 0 |
| P2 | 4 | 1 | 74 | 75 | 74 |
| P3 | 7 | 22 | 72 | 94 | 72 |
| P4 | 11 | 0 | 90 | 90 | 90 |
| P5 | 15 | 17 | 86 | 103 | 86 |
| P6 | 18 | 2 | 100 | 102 | 100 |
| P7 | 18 | 4 | 102 | 106 | 102 |
| P8 | 20 | 10 | 104 | 114 | 104 |

```
  | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 |
  1   78   79   101  101  118  120  124  134
```

Avg Waiting Time: 78.5
Avg TurnAround Time: 95.125
Average Response Time: 78.5
Number of Context Switches: 8.0

**RR** with time quantum = 5



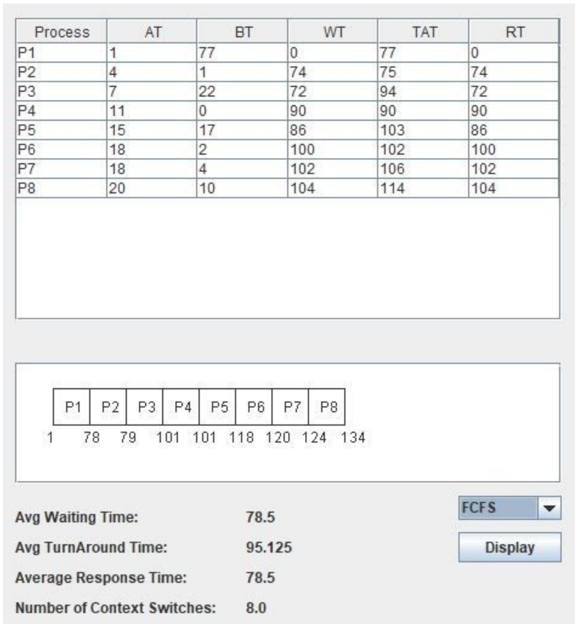
| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 1 | 77 | 56 | 133 | 0 |
| P2 | 4 | 1 | 2 | 3 | 2 |
| P3 | 7 | 22 | 58 | 80 | 5 |
| P4 | 11 | 0 | 6 | 6 | 6 |
| P5 | 15 | 17 | 53 | 70 | 7 |
| P6 | 18 | 2 | 14 | 16 | 14 |
| P7 | 18 | 4 | 16 | 20 | 16 |
| P8 | 20 | 10 | 48 | 58 | 33 |

```
  | P1 | P2 | P1 | P3 | P4 | P1 | P5 | P3 | P6 | P7 | P1 | P5 | P3 | P8
  1    6    7    12   17   17   22   27   32   34   38   43   48   53
```

Avg Waiting Time: 31.625
Avg TurnAround Time: 48.25
Average Response Time: 10.375
Number of Context Switches: 22.0

2. When time quantum of RR is bigger than the burst time of all the processes, then it behaves like FCFS, whereas when it is small it faces lot of overhead and number of context switches increases.

**FCFS**

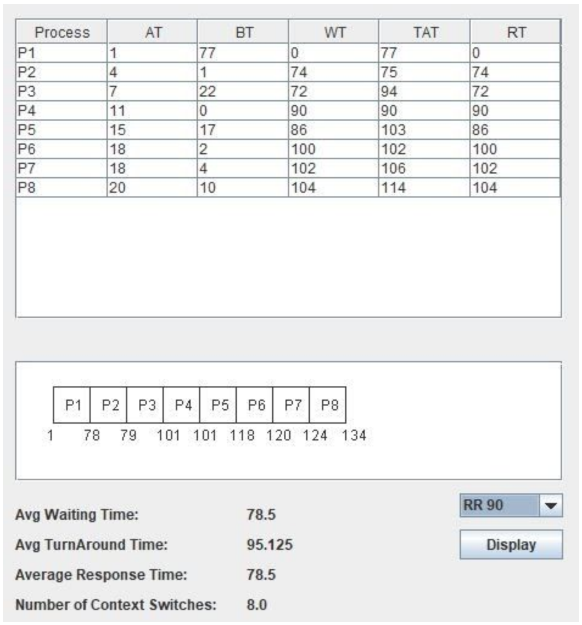


| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 1 | 77 | 0 | 77 | 0 |
| P2 | 4 | 1 | 74 | 75 | 74 |
| P3 | 7 | 22 | 72 | 94 | 72 |
| P4 | 11 | 0 | 90 | 90 | 90 |
| P5 | 15 | 17 | 86 | 103 | 86 |
| P6 | 18 | 2 | 100 | 102 | 100 |
| P7 | 18 | 4 | 102 | 106 | 102 |
| P8 | 20 | 10 | 104 | 114 | 104 |

```
  | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 |
  1   78   79  101  101  118  120  124  134
```

Avg Waiting Time: 78.5
Avg TurnAround Time: 95.125
Average Response Time: 78.5
Number of Context Switches: 8.0

**RR** with Time quantum = 90



| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 1 | 77 | 0 | 77 | 0 |
| P2 | 4 | 1 | 74 | 75 | 74 |
| P3 | 7 | 22 | 72 | 94 | 72 |
| P4 | 11 | 0 | 90 | 90 | 90 |
| P5 | 15 | 17 | 86 | 103 | 86 |
| P6 | 18 | 2 | 100 | 102 | 100 |
| P7 | 18 | 4 | 102 | 106 | 102 |
| P8 | 20 | 10 | 104 | 114 | 104 |

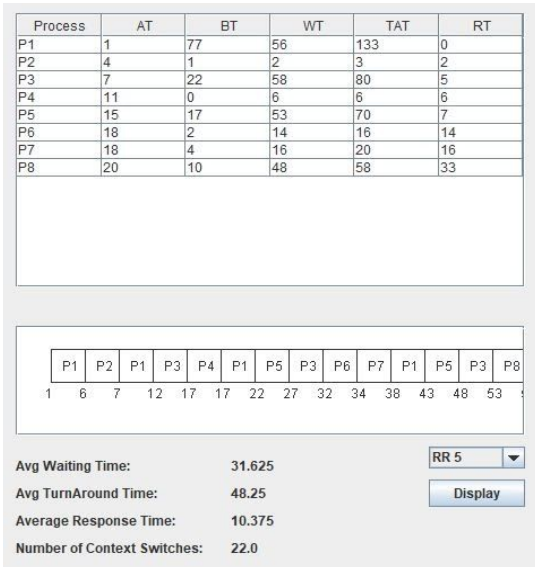
```
  | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 |
  1   78   79  101  101  118  120  124  134
```

Avg Waiting Time: 78.5
Avg TurnAround Time: 95.125
Average Response Time: 78.5
Number of Context Switches: 8.0

## RR vs DRR

1. Average waiting time, Average turnaround time and number of context switches of DRR is less than RR. DRR dynamically assigns time quantum according to the burst time of the processes present in ready queue and changes on addition of new processes, whereas, RR's performance varies with the fixed time quantum value assigned but is lower than DRR.

**RR with time quantum 5**

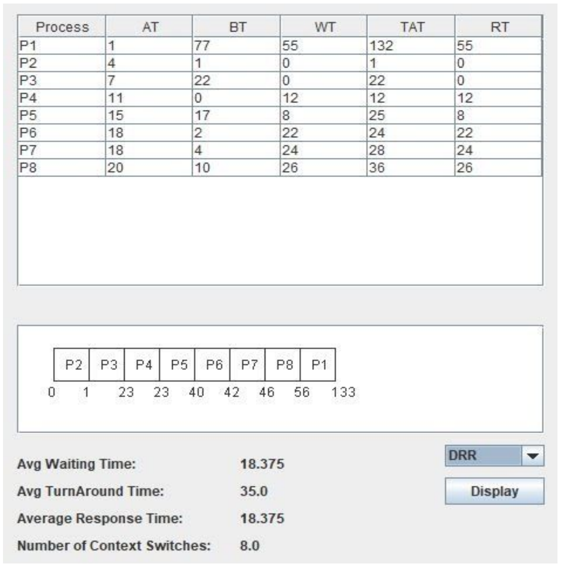


| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 1 | 77 | 56 | 133 | 0 |
| P2 | 4 | 1 | 2 | 3 | 2 |
| P3 | 7 | 22 | 58 | 80 | 5 |
| P4 | 11 | 0 | 6 | 6 | 6 |
| P5 | 15 | 17 | 53 | 70 | 7 |
| P6 | 18 | 2 | 14 | 16 | 14 |
| P7 | 18 | 4 | 16 | 20 | 16 |
| P8 | 20 | 10 | 48 | 58 | 33 |

```
  | P1 | P2 | P1 | P3 | P4 | P1 | P5 | P3 | P6 | P7 | P1 | P5 | P3 | P8
  1    6    7   12   17   17   22   27   32   34   38   43   48   53
```

Avg Waiting Time: 31.625
Avg TurnAround Time: 48.25
Average Response Time: 10.375
Number of Context Switches: 22.0

**DRR**



| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 1 | 77 | 55 | 132 | 55 |
| P2 | 4 | 1 | 0 | 1 | 0 |
| P3 | 7 | 22 | 0 | 22 | 0 |
| P4 | 11 | 0 | 12 | 12 | 12 |
| P5 | 15 | 17 | 8 | 25 | 8 |
| P6 | 18 | 2 | 22 | 24 | 22 |
| P7 | 18 | 4 | 24 | 28 | 24 |
| P8 | 20 | 10 | 26 | 36 | 26 |

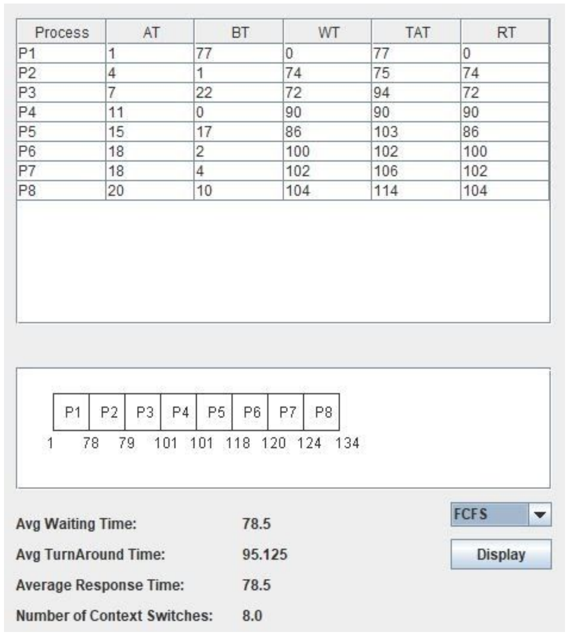
```
  | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P1 |
  0    1   23   23   40   42   46   56  133
```

Avg Waiting Time: 18.375
Avg TurnAround Time: 35.0
Average Response Time: 18.375
Number of Context Switches: 8.0

## FCFS vs DRR

1. Average waiting time, Average turnaround time and the number of context switches of FCFS is more than DRR. Since DRR overcomes the problem of starvation by introducing the time slice concept with the dynamic nature, it significantly improves performance and thereby, reduce the Average waiting and turnaround times. Note: The number of context switches in FCFS and DRR are almost similar.

**FCFS**

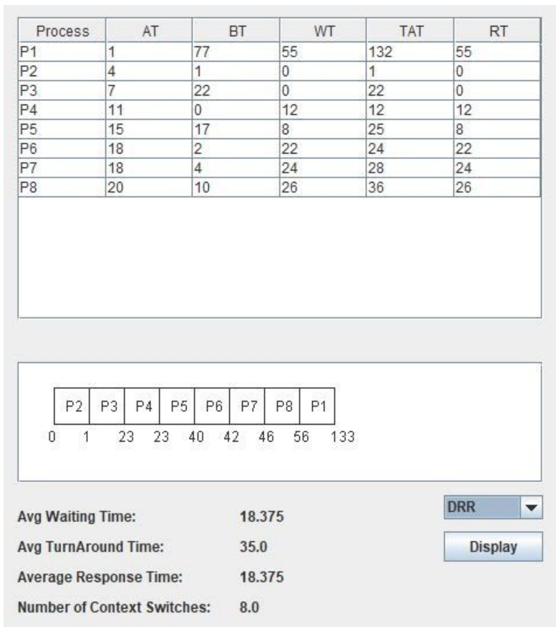


| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 1 | 77 | 0 | 77 | 0 |
| P2 | 4 | 1 | 74 | 75 | 74 |
| P3 | 7 | 22 | 72 | 94 | 72 |
| P4 | 11 | 0 | 90 | 90 | 90 |
| P5 | 15 | 17 | 86 | 103 | 86 |
| P6 | 18 | 2 | 100 | 102 | 100 |
| P7 | 18 | 4 | 102 | 106 | 102 |
| P8 | 20 | 10 | 104 | 114 | 104 |

```
 | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 |
 1   78   79  101  101  118  120  124  134
```

| | | |
|---|---|---|
| Avg Waiting Time: | 78.5 | FCFS |
| Avg TurnAround Time: | 95.125 | Display |
| Average Response Time: | 78.5 | |
| Number of Context Switches: | 8.0 | |

**DRR**



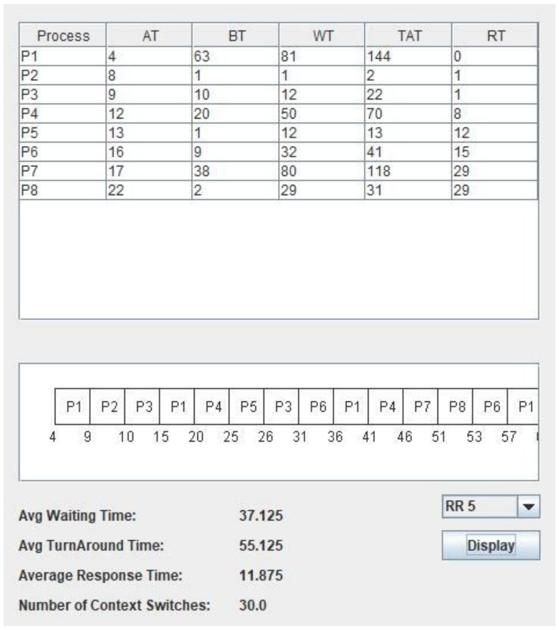
| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 1 | 77 | 55 | 132 | 55 |
| P2 | 4 | 1 | 0 | 1 | 0 |
| P3 | 7 | 22 | 0 | 22 | 0 |
| P4 | 11 | 0 | 12 | 12 | 12 |
| P5 | 15 | 17 | 8 | 25 | 8 |
| P6 | 18 | 2 | 22 | 24 | 22 |
| P7 | 18 | 4 | 24 | 28 | 24 |
| P8 | 20 | 10 | 26 | 36 | 26 |

```
 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P1 |
 0    1   23   23   40   42   46   56   133
```

| | | |
|---|---|---|
| Avg Waiting Time: | 18.375 | DRR |
| Avg TurnAround Time: | 35.0 | Display |
| Average Response Time: | 18.375 | |
| Number of Context Switches: | 8.0 | |

**RR with different Time Quantums -** 5, 10, 20

RR - 5

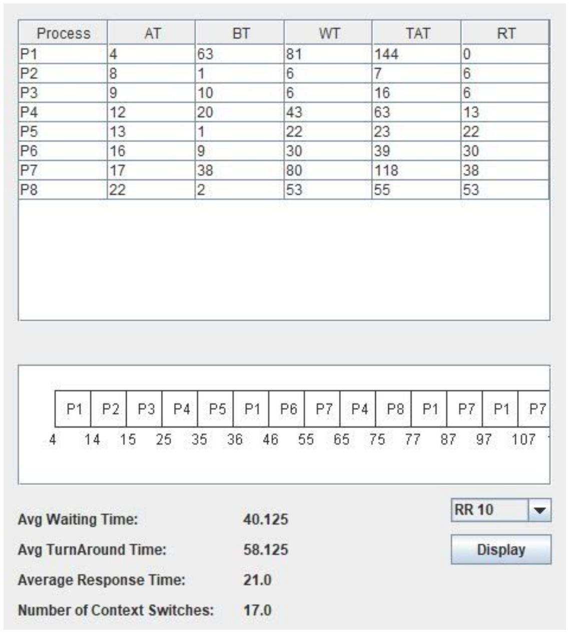


| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 4 | 63 | 81 | 144 | 0 |
| P2 | 8 | 1 | 1 | 2 | 1 |
| P3 | 9 | 10 | 12 | 22 | 1 |
| P4 | 12 | 20 | 50 | 70 | 8 |
| P5 | 13 | 1 | 12 | 13 | 12 |
| P6 | 16 | 9 | 32 | 41 | 15 |
| P7 | 17 | 38 | 80 | 118 | 29 |
| P8 | 22 | 2 | 29 | 31 | 29 |

```
 | P1 | P2 | P3 | P1 | P4 | P5 | P3 | P6 | P1 | P4 | P7 | P8 | P6 | P1
 4    9   10   15   20   25   26   31   36   41   46   51   53   57
```

| | | |
|---|---|---|
| Avg Waiting Time: | 37.125 | RR 5 |
| Avg TurnAround Time: | 55.125 | Display |
| Average Response Time: | 11.875 | |
| Number of Context Switches: | 30.0 | |

RR - 10

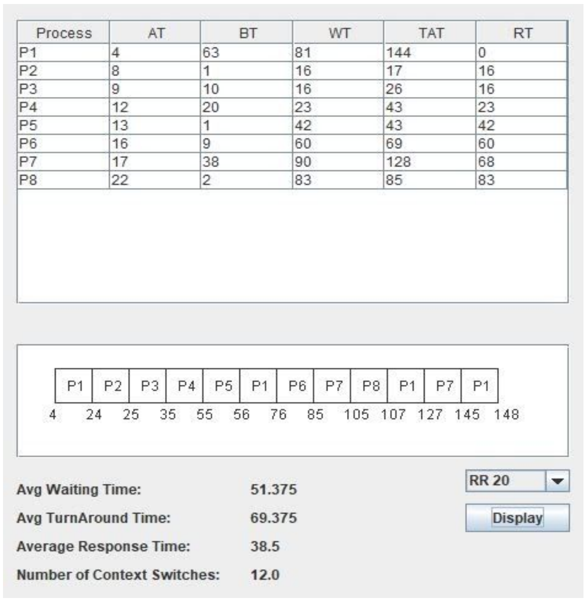


| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 4 | 63 | 81 | 144 | 0 |
| P2 | 8 | 1 | 6 | 7 | 6 |
| P3 | 9 | 10 | 6 | 16 | 6 |
| P4 | 12 | 20 | 43 | 63 | 13 |
| P5 | 13 | 1 | 22 | 23 | 22 |
| P6 | 16 | 9 | 30 | 39 | 30 |
| P7 | 17 | 38 | 80 | 118 | 38 |
| P8 | 22 | 2 | 53 | 55 | 53 |

```
 | P1 | P2 | P3 | P4 | P5 | P1 | P6 | P7 | P4 | P8 | P1 | P7 | P1 | P7
 4   14   15   25   35   36   46   55   65   75   77   87   97   107
```

| | | |
|---|---|---|
| Avg Waiting Time: | 40.125 | RR 10 |
| Avg TurnAround Time: | 58.125 | Display |
| Average Response Time: | 21.0 | |
| Number of Context Switches: | 17.0 | |

**RR - 20**



| Process | AT | BT | WT | TAT | RT |
|---|---|---|---|---|---|
| P1 | 4 | 63 | 81 | 144 | 0 |
| P2 | 8 | 1 | 16 | 17 | 16 |
| P3 | 9 | 10 | 16 | 26 | 16 |
| P4 | 12 | 20 | 23 | 43 | 23 |
| P5 | 13 | 1 | 42 | 43 | 42 |
| P6 | 16 | 9 | 60 | 69 | 60 |
| P7 | 17 | 38 | 90 | 128 | 68 |
| P8 | 22 | 2 | 83 | 85 | 83 |

```
| P1 | P2 | P3 | P4 | P5 | P1 | P6 | P7 | P8 | P1 | P7 | P1 |
4    24   25   35   55   56   76   85   105  107  127  145  148
```

Avg Waiting Time: 51.375

Avg TurnAround Time: 69.375

Average Response Time: 38.5

Number of Context Switches: 12.0

RR 20

We observed that small time quantums results in large context switches and also, produces less waiting time and turn-around time than the quantums of larger ones. Hence, it can be considered as a tradeoff between different time quantums.

## Ⅸ. CONCLUSION

Overall, the performance of DRR is most optimal in comparison with Round Robin and First Come First Served Scheduling algorithm, as it reduces the number of context switches as well as prevent short burst time processes from starvation problem.

## Ⅹ. Table of Contributions

| | | | |
|---|---|---|---|
| Muskan Matwani | RR, DRR, GUI and main thread files | Integration and Execution | Report |
| Yesha Shastri | FCFS, DRR and main thread files | Integration and Execution | Report |