Operating Systems – Assignment 1

Candidate Number: 184514 - Bayley Cowen-Seagrove

University of Sussex 2020

Abstract

The purpose of this report is to examine the performance of different processing scheduling algorithms with exponential average. The scheduling algorithms being tested consist of: First Come First Serve, Round Robin, Shortest Job First, Ideal Shortest Job First and Multilevel Feedback Queue with Round Robin. The report will discuss three different experiments which will each investigate the performance of one or more schedulers, given specific inputs. Each experiment's motivations, methodologies, hypotheses and results will be discussed.

Contents

Experiment 1

The question investigated in this experiment is as follows;

Does the Mean CPU Burst affect the overall performance for different scheduling algorithm, and which if the scheduling algorithms experiences the biggest difference in performance from this alteration?

Introduction

The motivation behind this question is to examine how the three chosen scheduling algorithms deal with processes strikingly better than others. This experiment is relevant as the Mean CPU Burst for a collection of processes could influence the implementation of a scheduling algorithm. The mean CPU Burst was chosen as the independent variable for this experiment, as it is a good measure, since it determines which algorithm is best at starting processes upon arrival. The rate of waiting times, turnaround times and the maximum response time will all be observed as metrics to determine the scheduling algorithm with the best performance.

The three scheduling algorithms that have been chosen for this experiment are: First Come First Served, Round Robin and Ideal Shortest Job scheduling. These three schedulers have been chosen to participate in this experiment as I believe the three schedulers should convey all different results to the performance tests. The hypothesis for this experiment was that the Ideal Shortest Job First and Round Robin to perform the best for all Mean CPU Bursts, when compared to the First Come First Served scheduler. This is since these algorithms ensure smaller processes are given higher priorities in the ready queue, reducing the Turnaround time, which is the time difference between the processes creation and termination. These high performance schedulers should have low waiting & turnaround average times, and a lower maximum response time. Furthermore a more generic hypothesis, is that as the mean CPU bursts increase, the performance of the schedulers should decrease, most drastically present in the First Come First Served scheduler, so the Average waiting / turnaround time will increase, along with the maximum response time. For maximum response time, I theorize that the Round Robin scheduler will have the lowest response time, since it gives each process an amount *fairly,* and the Ideal SJF Scheduler will be the least fair.

Methodology

In order to obtain useful metrics for performance, seven different Mean CPU burst values were selected, ranging from 10, to test the lower boundary, and 70, to test the upper boundary. These values will be able to estimate a value for where the schedulers' performance decreases, including an approximation where around the performance of the schedulers exponentially increases. For each Mean CPU burst value, the three scheduling algorithms were run on three sets of 110 processes, obtained using three different seeds. To obtain the input data, the input

parameters described in figure 1.1, stored as parameter files in the /experiment1 folder, with input parameters for each seed.

The three seeds in figure 1.1 are the different seeds being used for the schedulers described in figure 1.2, which describes the simulator parameters for the experiment. For each seed, there is 3 different simulator_parameters.prp, one for each of the scheduling algorithms. These schedulers were chosen as I believe they will all follow a similar trend and have varied averages, in order to be able to accurately describe which of the scheduling algorithms experiences the biggest hit in performance, from the increase in mean CPU bursts. In addition there are 7 different input_parameters.prp, one for each meanICPUBurst value. I have chosen to keep all the other default values as I believe they will produce the best results. In addition, the only value I have changed is the number of processes, as I believe 5 processes would not be enough to give an accurate representation of the performance decrease in each of the schedulers.

The results of these simulations are stored with the respective seed in the /experiment1 folder. The output files are named XoutputY.out, where the variables X and Y denote the type of scheduler. X = {FCFS | IDEAL | RR}, Y = {1|2|3|4|5|6|7}. The data from the output files was imported to Excel, so each output can have its average waiting and turnaround times calculated. For each seed, there will be a calculated average of waiting time, turnaround time and a maximum response time, for each scheduler. This is done by taking the sum of all waiting/turnaround times in an output file and dividing that sum by the number processes in that iteration. The maximum value was fund using the MAX() command in Excel after highlighting the column that describes the process' response time. The reason I did not use an the average response time, as many processes would possibly having a very low if not null response time, reducing the reliability of that performance metric. The maximum describes the worst case results that has been found.

```
seed=11111111 | 22222222 | 33333333
meanInterArrival=150
meanCpuBurst=10 | 20 | 30 | 40 | 50 | 60 | 70
meanIOBurst=15.0
meanNumberBursts=2.0
numberOfProcesses=110
staticPriority=0
```

Figure 1.1: Input parameters used to set up generation of input files in experiment 1, showing vareity of seeds and mean CPU bursts.

```
scheduler= FcfsScheduler | IdealSJFScheduler | RRScheduler*
timeLimit=10000
periodic=false
interruptTime=0
timeQuantum=20
initialBurstEstimate=10
alphaBurstEstimate=0.5
```

Figure 1. 2: Simulator parameters used to set up the simulator for experiment 1, with each scheduler having its own simulator parameter file.

Results

First Come First Served Scheduler Turnaround times

Mean CPU Bursts	Seed 1	Seed 2	Seed 3	Average Turnaround Time (ms)
10	29.680	38.227	38.862	35.590
20	54.973	61.409	70.123	62.168
30	88.616	94.333	119.354	100.768
40	139.932	147.485	216.046	167.821
50	207.699	227.197	341.662	258.852
60	290.134	360.894	502.241	384.423
70	396.091	546.938	571.667	504.899

Table 1. 1: FCFS - Average Turnaround time for each CPU Bursts value tested, rounded to 3.d.p

Ideal Shortest Job First Scheduler Turnaround times

Mean CPU Bursts	Seed 1	Seed 2	Seed 3	Average Turnaround Time (ms)
10	29.680	38.227	38.862	35.590
20	53.770	61.333	69.615	61.573
30	82.932	92.288	119.215	98.145
40	128.342	138.727	185.308	150.792
50	171.712	202.758	275.908	216.793
60	213.254	299.364	356.806	289.808
70	260.414	392.569	416.633	356.539

Table 1. 2: IdealSJF - Average Turnaround time for each CPU Bursts value tested, rounded to 3.d.p

Round Robin Scheduler Turnaround times

Mean CPU Bursts	Seed 1	Seed 2	Seed 3	Average Turnaround Time (ms)
10	29.743	39.167	39.785	36.232
20	56.014	61.894	69.877	62.595
30	82.863	96.955	125.708	101.842
40	135.836	153.061	231.769	173.555
50	201.500	235.985	382.815	273.433
60	260.418	344.773	535.589	380.260
70	313.222	511.954	633.943	486.373

Table 1. 3: RR- Average Turnaround time for each CPU Bursts value tested, rounded to 3.d.p

First Come First Served Scheduler Waiting times

Mean CPU Bursts	Seed 1	Seed 2	Seed 3	Average Waiting Time (ms)
10	20.613	36.561	31.738	29.637
20	28.096	41.742	42.231	37.356
30	44.178	56.652	70.677	57.169
40	77.712	91.697	146.615	105.342
50	127.808	153.364	251.477	177.550
60	194.164	269.045	387.655	283.622
70	282.591	435.600	437.852	385.348

Table 1. 4: FCFS - Average waiting time for each CPU Bursts value tested, rounded to 3.d.p

Ideal Shortest Job First Scheduler Waiting times

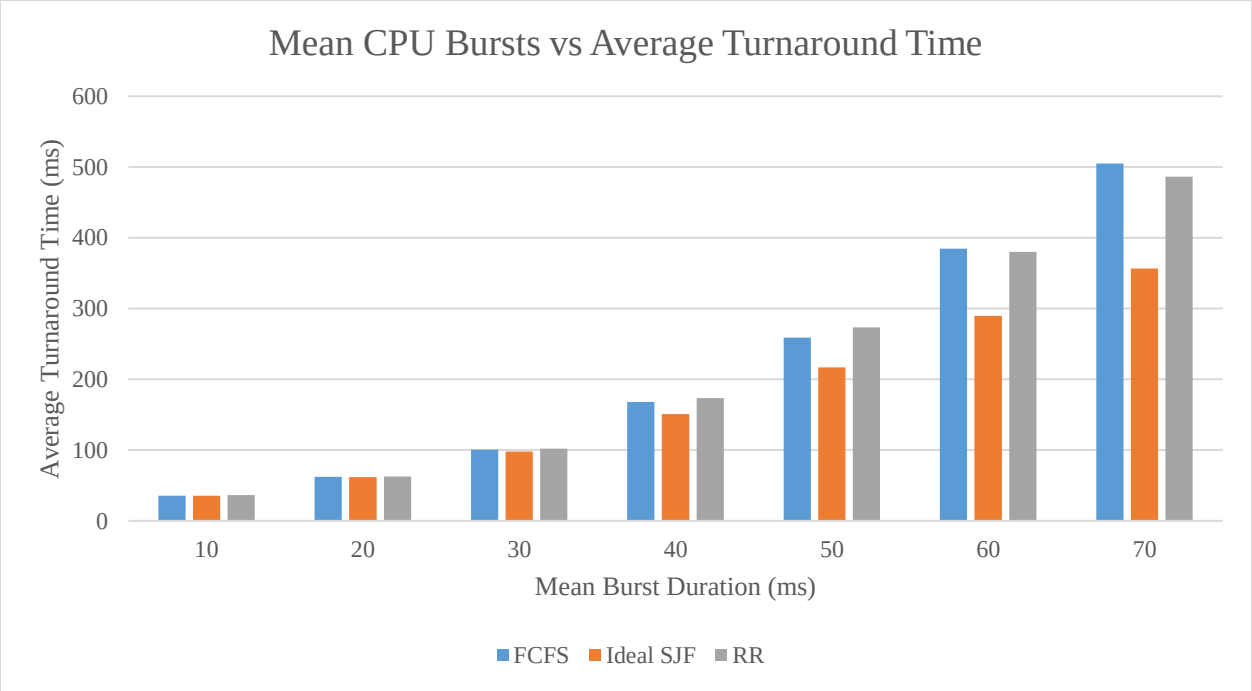
Mean CPU Bursts	Seed 1	Seed 2	Seed 3	Average Waiting Time (ms)
10	20.61333333	36.56060606	31.73846154	29.63746698
20	27.27027027	41.66666667	41.72307692	36.88667129
30	38.49315068	54.60606061	70.53846154	54.54589094
40	66.12328767	82.93939394	115.8769231	88.31320156
50	91.82191781	128.9242424	185.7230769	135.4897457
60	119.7323944	207.5151515	249.8225806	192.3567088
70	152.1142857	281.2307692	295.4166667	242.9205739

Table 1. 5: Ideal SJF- Average waiting time for each CPU Bursts value tested, rounded to 3.d.p

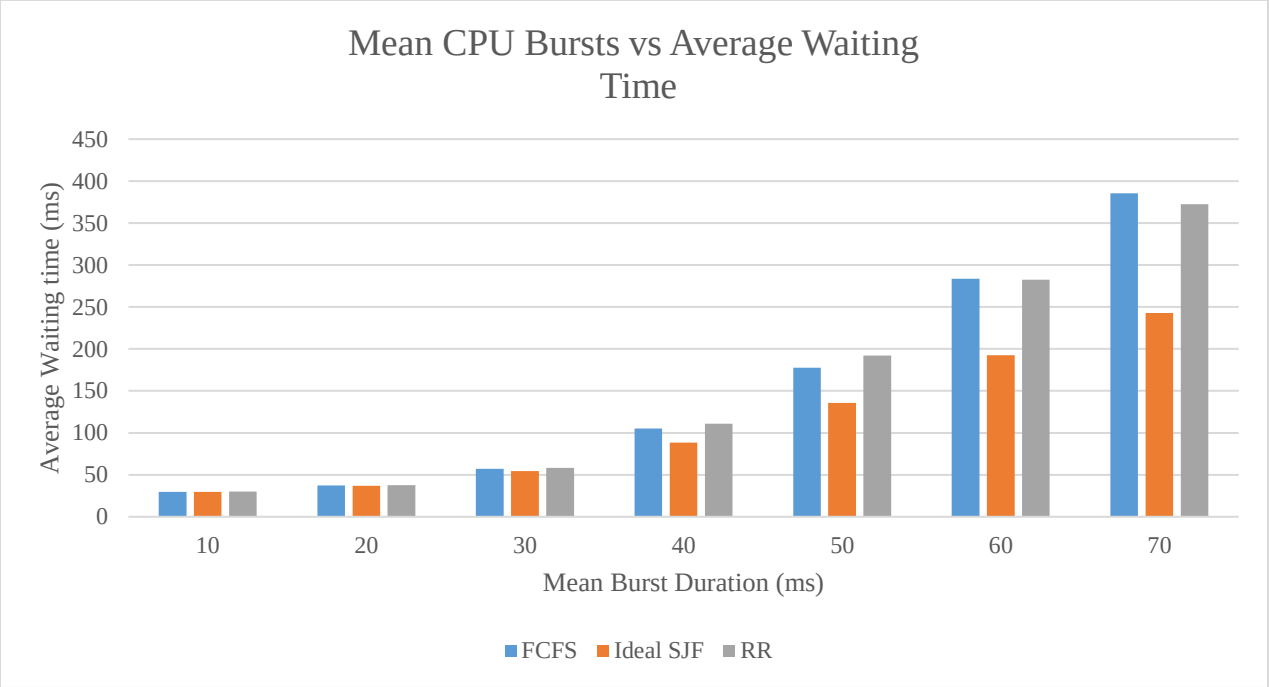
Round Robin Scheduler Waiting times

Mean CPU Bursts	Seed 1	Seed 2	Seed 3	Average Waiting Time (ms)
10	20.324	37.500	32.662	30.162
20	29.137	42.227	41.985	37.783
30	38.425	59.273	77.031	58.243
40	73.616	97.273	162.338	111.076
50	121.583	162.152	292.631	192.122
60	167.896	252.924	426.643	282.488
70	205.746	400.615	511.245	372.536

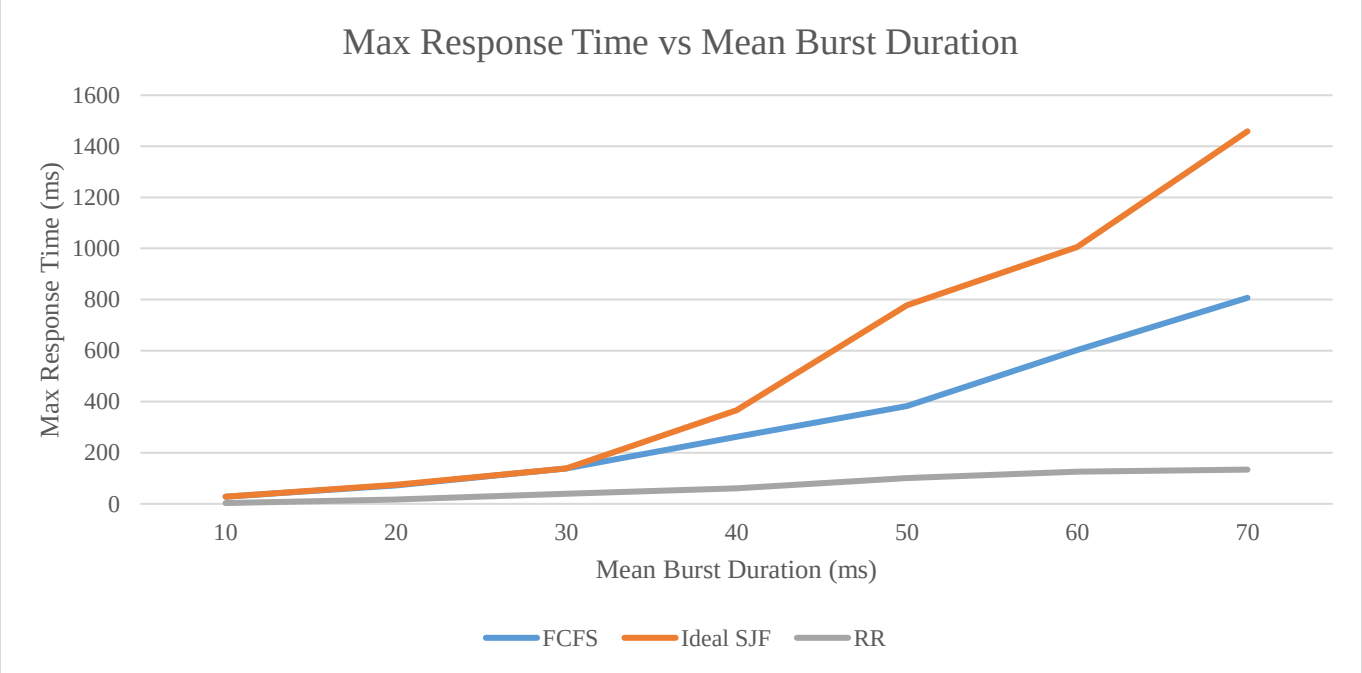
Table 1. 6: RR- Average waiting time for each CPU Bursts value tested, rounded to 3.d.p



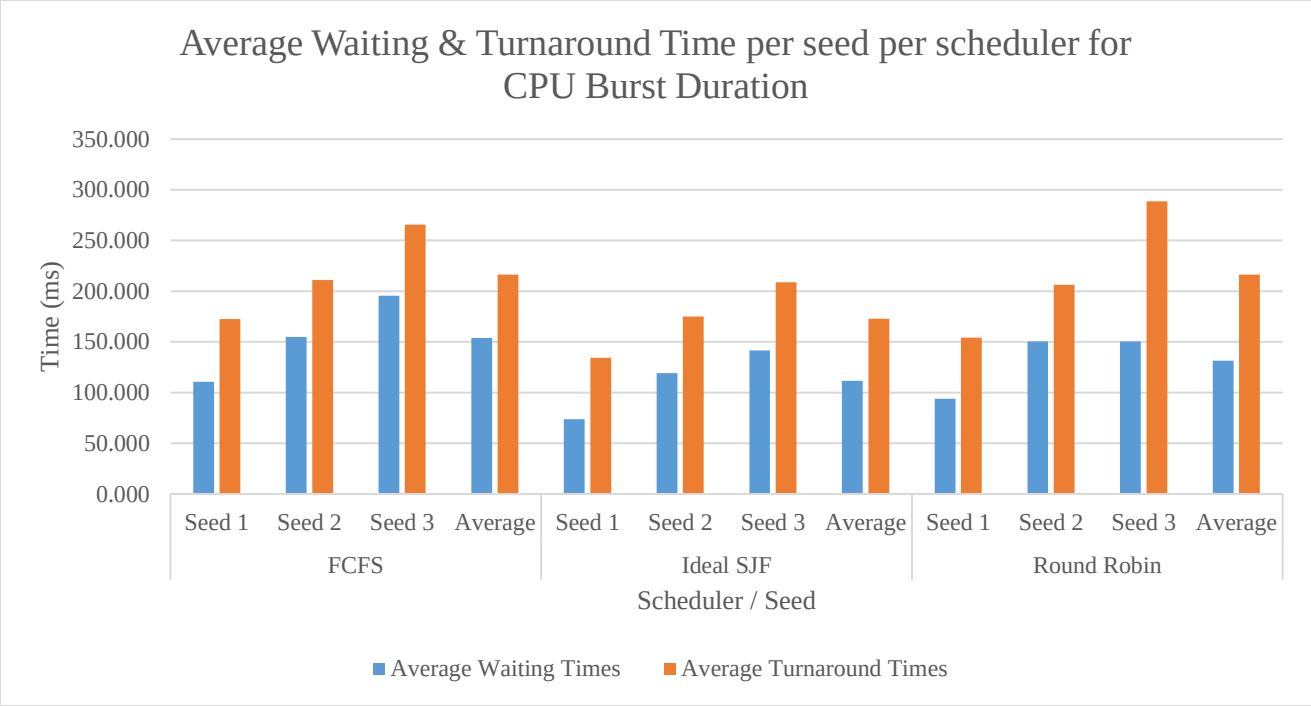
Graph 1. 1: Graph displaying the Average Turnaround time against Mean CPU Bursts duration. Note: FCFS: First Come First Served, Ideal SSJ: Ideal Shortest Job First, RR: Round Robin. Graphical display of the results from Table 1.1, 1.2 & 1.3



Graph 1. 2: Graph displaying the Average waiting time against Mean CPU Bursts duration. Note: FCFS: First Come First Served, Ideal SSJ: Ideal Shortest Job First, RR: Round Robin. Graphical display of the results from Table 1.4, 1.5 & 1.6.



Graph 1. 3: Graph displaying the Maximum response time against Mean CPU Bursts duration. Note: FCFS: First Come First Served, Ideal SSJ: Ideal Shortest Job First, RR: Round Robin. Graphical display of the results from table within .zip file named "Exp1MaxResponseTable"



Graph 1. 4: Graph displaying the Average Turnaround time & Waiting Time for each seed, per scheduler for CPU burst duration. Graphical display of the results from Table 1.1 - 1.6.

Discussion

The results show that the scheduling algorithm with the best performance in Waiting time and Turnaround time was the Ideal SJF Scheduler shown in Graph 1.1 and 1.2. By looking at Tables 1.1 – 1.3, we can confirm that for Turnaround time, the Ideal SJF scheduler was the best performing having the lowest score. In addition by looking at Tables 1.4-1.6, we can confirm that for also Waiting time, the Ideal SJF scheduler was the best performing since it had the lowest score. This algorithm had almost twice the performance compared to the other two, when referring to Turnaround/Waiting time, which was theorized in the hypothesis. The results further convey that the prediction of performance decreasing as the mean increases in all three performance metrics, evident in Graphs 1.1-1.3 with the positive correlation. The reason for the decrease in performance in waiting time and turnaround time is due to the increase of CPU burst durations, the terminated times of process is being extended, therefore a greater difference between time of creation and termination..

Another factor to note is that the Waiting time is always performing better than the turnaround time, shown evidently in Graph 1.4. The reason for this is because the CPU burst duration has increased, so has CPU time, which is a factor that reduces waiting time. In addition, the possibly may be an increase in blocked time, which also increases performance for waiting time. Overall, the adjustment on the Mean CPU Burst duration does make a very noticeable difference towards performance. As the CPU burst duration increases from 10ms to 70ms, the waiting time and turnaround time increase by on average ~400ms for the schedulers tested.

The hypothesis for this experiment, as made in the introduction, was correct to the results observed, as seen in Graph 1.1 – 1.3, as the CPU Burst duration increases, the performance of all three scheduler decrease. This is evident in Graph 1.1, where Ideal Shortest Job First was correctly predicted to be the least impacted by the performance dips. In Graph 1.2, the hypothesis correctly predicted that Ideal Shortest Job will retain the best performance. The reason for these is the scheduler has quicker times of termination (on average) as it prioritizes smaller processes. It is also the least impacted by the rise in Mean CPU burst duration, as the longer CPU bursts have low priority and left until the end, giving a good overall average for termination times, which is a performance factor in both Turnaround and Waiting times. The hypothesis further correctly predicted that Round Robin would be the best performer in Max Response Time, due to its fairness policy allowing all processes to be responded to in a quick and fair way, giving on average a much lower response time compared to the other two exponential like growths. These results are displayed clearly in Graph 1.3. Graphs 1.1 & 1.2 further display that FCFS scheduler is the worst performing in average waiting/turnaround times, due to its lack of priority when it comes to smaller and quicker processes. This algorithm however does introduce a some-what fairness in queuing (shown in Graph 1.3), with a reasonable Max Response Time relative to the response time of the Ideal SJF Scheduler.

The hypothesis failed to predict the exponential-like growth shown in Graph 1.3 for the Ideal SJF Scheduler. This makes sense, as the algorithm is very unfair, giving priority the shortest jobs, leaving some longer processes with very high response times. The growth to that scale however could be due to starvation, which can occur in Shortest Job First algorithms. Another aspect presented in the results that the hypothesis failed to predict correctly was the poor performance in Waiting/Turnaround time for the Round Robin scheduler, shown in Graphs 1.1 & 1.2. This makes sense, as by allowing all processes some time, and not prioritizing any certain jobs, on average, termination times will rapidly increase for all process, leaving the scheduling algorithm with poor performance in these aspects.

Evaluation

From the results presented in the tables, and graphs, I could confidently say that out of the scheduling algorithms tested, First Come First Served experiences the biggest difference in performance from this alteration of Mean CPU Burst duration. The combination of the worst performance in Waiting & Turnaround time (Graphs 1.1 & 1.2), along with a mediocre performance in max response time (Graph 1.3), leads me to believe it is the worst effected. As mentioned above, the number of iterations used may cause the validity of the experiment to be compromised. More data could have been used to provide a more realistic and reliable results from the simulations, but in order to preserve practicality in handling the output data, the number of processes and the number of tests was limited. In addition, shown in Graph 1.4, Seed1s performance was much better than Seed2 & 3, which can reduces the reliability of the results. Another issue that may have affected the results is the efficiency of the implementation of the scheduling algorithms used. This could affect the running time, or even the start times of the processes, which in turn effects the turnaround/waiting times of the processes, leading to worst

performance. To resolve this, multiple implementations of the same scheduling algorithms, or even more scheduling algorithms could be implemented & tested to validate the results and get a more tight average.

Experiment 2

The question investigated in this experiment is as follows;

Does the Mean Inter-Arrival Time affect the waiting and turnaround time for processes running a First Come First Served scheduling algorithm, more than a process running in a Shortest Job First scheduling algorithm which uses exponential averaging?

Introduction

The two scheduling algorithms chosen in this experiment have been chosen, as I would believe that by altering the Mean Inter-Arrival time for both of these scheduling algorithms will drastically improve the performance, as long as a viable Mean Inter-Arrival Time value is selected. The inter-arrival time is the time between processes arriving, so it's the difference between arrival times of a process and its predecessor. Therefore, the Mean Inter-Arrival time is the average of all the inter-arrival times. These times are represented by a distribution which is called Exponential Distribution. This displays the probability of an inter-arrival time being randomly selected from the distribution. The Mean Inter-Arrival time is where the majority of times are around the mean. Having inter-arrival times close to the mean, should increase performance, therefore having lower waiting & turnaround times.

The hypothesis for this experiment was that the First Come First Served algorithm will perform slightly worse to the Shortest Job First (with exponential averaging) implementation. Further, as the Mean Inter-arrival time is increased, performance will increase (lower waiting & turnaround times)

Methodology

In order to obtain useful averages for waiting time and turnaround time, five different Mean inter-arrival times values were selected; 50, 100, 150, 200, 250. These values where chosen to test the lower boundary (i.e. 50), as well as being able to estimate a value for where the schedulers performance increases, including an approximation where around the performance caps out / levels off. For each Mean Inter-Arrival value, the two scheduling algorithms were run on three sets of 100 processes, obtained using three different seeds. To obtain the input data, the input parameters described in figure 2.1, stored as parameter files in the /experiment2 folder,

```
seed=11111111 | 22222222 | 33333333
meanInterArrival=50 | 100 | 150.0 | 200 | 250
meanCpuBurst=15.0
meanIOBurst=15.0
meanNumberBursts=2.0
numberOfProcesses=100
staticPriority=0
```

Figure 2. 1: Input parameters used to set up generation of input files in experiment 2

with input parameters for each seed. This large number of processes is needed, else possible unrealistic waiting times of processes waiting in the ready queue due to the insufficient amount of processes, results in an inaccurate representation of the impact of increasing the meanInterArrival value against performance.

The three seeds in figure 2.1 are the different seeds being used for the schedulers described in figure 2.2, which describes the simulator parameters for the experiment. For each seed, there is 2 different simulator_parameters.prp, one for each of the two scheduling algorithms. These schedulers were chosen as I believe they will both follow a similar trend and have relatively close averages, despite being implemented differently. In addition there are 5 different input_parameters.prp, one for each meanInterArrival value.

The results of these simulations are stored with the respective seed in the /experiment2 folder. The output files are named outputXY.out, where the variables X and Y denote the type of scheduler. X = {F | S}, Y = {1|2|3|4|5}. The data from the output files was imported to Excel, so each output can have its average waiting and turnaround times calculated. For each seed, there will be a calculated average of waiting time & turnaround time, for each scheduler. This is done by taking the sum of all waiting/turnaround times in an output file and dividing that sum by the number processes in that iteration.

```
scheduler= FcfsScheduler | SJFScheduler
timeLimit=10000
periodic=false
interruptTime=0
timeQuantum=20
initialBurstEstimate=10
alphaBurstEstimate=0.5
```

Figure 2. 2: Simulator parameters used to set up the simulator for experiment 2

Results

First Come First Served Scheduler Turnaround times

Mean Inter-Arrival Time (ms)	Seed 1	Seed 2	Seed 3	Average Turnaround Time (ms)
50	63.100	97.240	91.110	83.817
100	47.160	61.410	55.190	54.587
150	41.452	48.773	52.754	47.660
200	35.500	52.796	54.894	47.730
250	36.233	49.282	54.531	46.682

Table 2. 1: FCFS - Average Turnaround time for each time Mean Inter-Arrival value, rounded to 3.d.p

Shortest Job First Scheduler Turnaround times

Mean Inter-Arrival Time (ms)	Seed 1	Seed 2	Seed 3	Average Turnaround Time (ms)
50	64.870	104.120	105.720	91.570
100	46.680	62.790	56.520	55.330
150	42.151	48.773	52.754	47.892
200	35.500	52.612	54.894	47.669
250	36.233	49.282	54.531	46.682

Table 2. 2: SJF - Average Turnaround time for each time Mean Inter-Arrival value, rounded to 3.d.p

First Come First Served Scheduler Waiting times

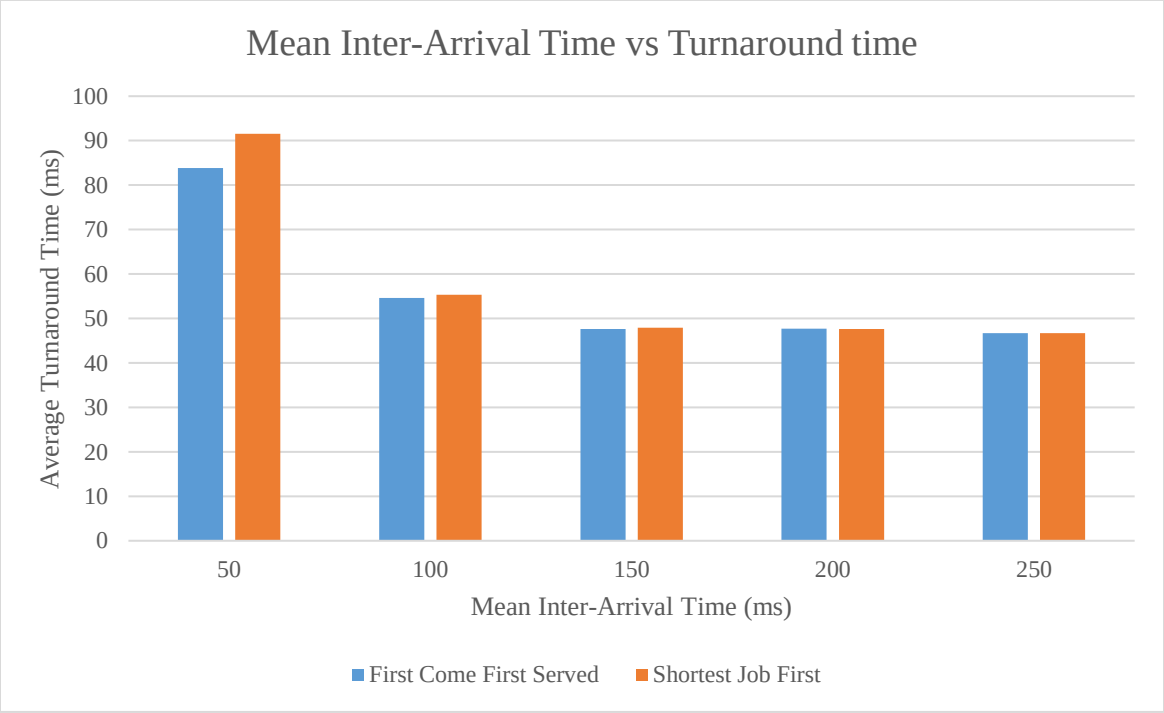
Mean Inter-Arrival Time (ms)	Seed 1	Seed 2	Seed 3	Average Waiting Time (ms)
50	47.310	84.790	74.220	68.773
100	31.370	48.960	38.300	39.543
150	23.438	38.167	35.262	32.289
200	15.769	42.571	38.362	32.234
250	15.442	37.128	37.563	30.044

Table 2. 3: FCFS - Average Waiting time for each time Mean Inter-Arrival value, rounded to 3.d.p

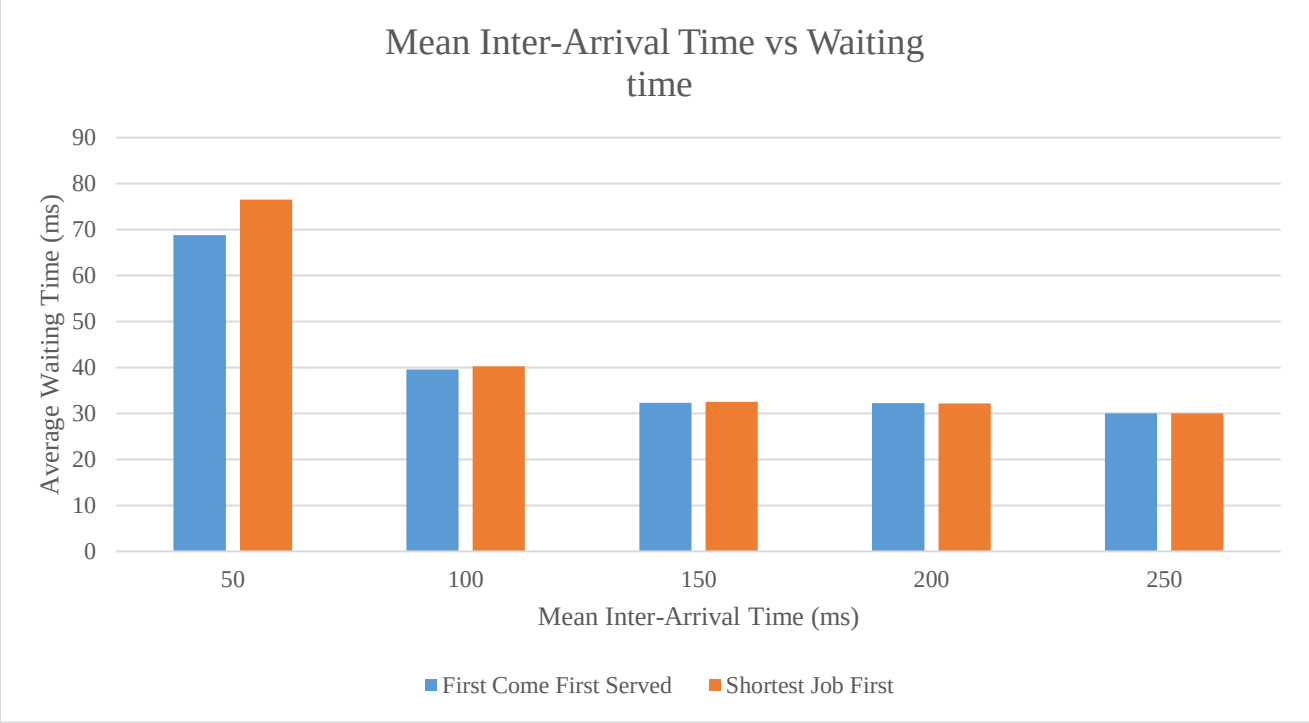
Shortest Job First Scheduler Waiting times

Mean Inter-Arrival Time (ms)	Seed 1	Seed 2	Seed 3	Average WaitingTime (ms)
50	49.080	91.670	88.830	76.527
100	30.890	50.340	39.630	40.287
150	24.137	38.167	35.262	32.522
200	15.769	42.388	38.362	32.173
250	15.442	37.128	37.563	30.044

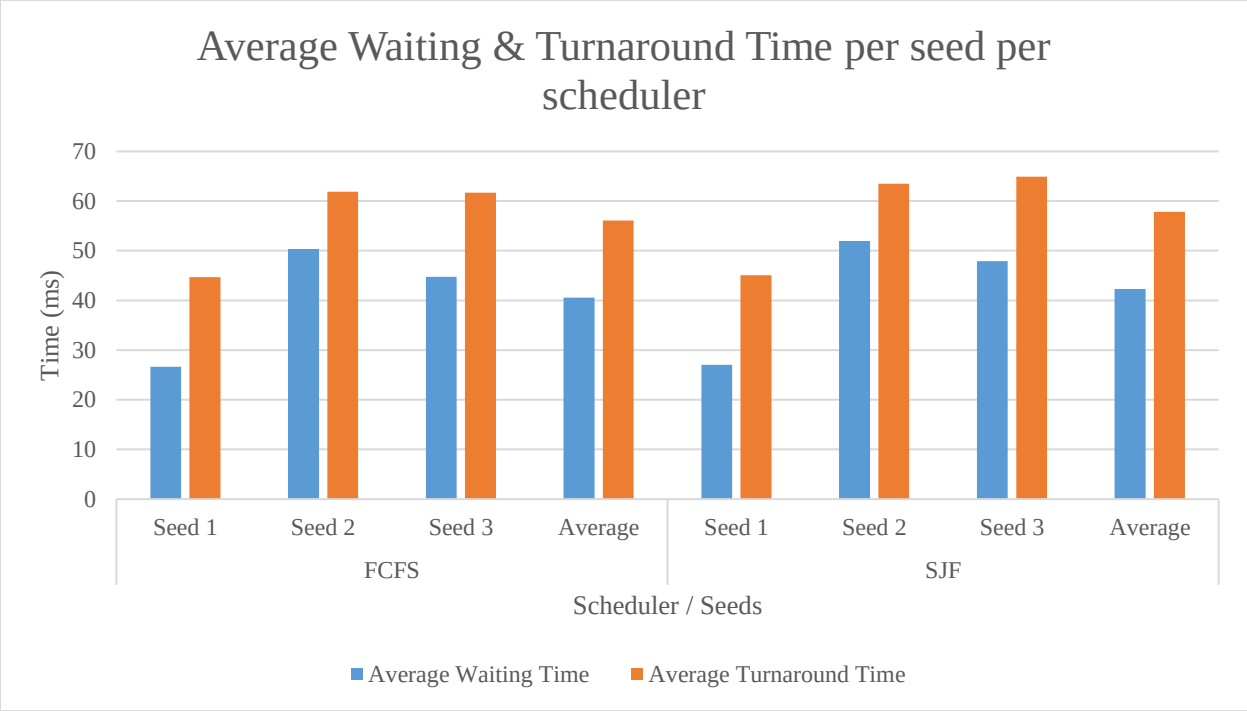
Table 2. 4: SJF - Average Waiting time for each time Mean Inter-Arrival value, rounded to 3.d.p



Graph 2. 1: Graph displaying the Average Turnaround time against Mean Inter-Arrival Time



Graph 2. 2: Graph displaying the Average Waiting time against Mean Inter-Arrival Time



Graph 2. 3: Graph displaying the Average Turnaround time & Waiting Time for each seed, per scheduler.

Discussion

The results show that both algorithms react similarly to the adjustments to the Mean Inter-Arrival. As the meanInterArrival value is increased, the average waiting/turnaround times decreases, leading to believe an improvement in performance, as more intervals arrive in the same time period, the number of processes that run in that time period would be greater meaning that the waiting/turnaround time for processes would be less.

Another factor to note is that the Waiting time is always performing better than the turnaround time, shown evidently in Graph 2.3. This would suggest either process do take longer to terminate, a factor that increases turnaround time, and/or the CPU or blocked time is reasonably enough to heavily reduce the waiting time when compared to the turnaround time. A way to check this would be to monitor the average CPU & blocked time to see which determining factor leads to a performance increase in waiting time over turnaround.

Overall, the adjustment on the Mean Inter-Arrival does make a very noticeable difference towards performance. For Turnaround time, an almost 50ms difference between the minimum and maximum value of Mean-Inter-Arrival tested, shown in Tables 2.1 & 2.2, as well as being displayed in Graph 2.1. In addition the waiting times, shown in Table 2.3 & 2.4 further support this idea of a performance increase with range of Average Turnaround times being approximately ~40ms, illustrated in Graph 3.2.

The hypothesis for this experiment, as made in the introduction, was correct to the results observed, as seen in Graph 2.1 & 2.2. The hypothesis did predict the drastic improvement in performance from using a low Mean Inter-Arrival (50ms) to much more realistic values. This value is seen as a poor mean value, due to the, in comparison, bad performance with waiting and turnaround times seen in especially Table 2.3 & 2.4.

The hypothesis failed to predict that the First Come First Served scheduling algorithm would out-perform the Shortest Job First scheduler for almost every result from every seed. By having a good mean value, the Inter-Arrival values chosen from the distribution at random, would result in a much more accurate probability for which values are chosen. Large Inter-arrival times where there are no processes could result in much worse performance increasing waiting times and turnaround times. From the results in Graph 2.1 & 2.2, the results convey that a mean value of 50ms is not a good value, due to poor performance in both waiting and turnaround time. In addition, at approximately 150ms the graphs begin to level off, suggesting a performance cap, suggested in the hypothesis. Therefore a Mean-Inter-Arrival time of around 200ms would be the ideal value suggested in Graph 2.1 & 2.2. The hypothesis further failed to predict the almost 20ms difference in performance between waiting and turnaround time, conveyed in Graph 2.3.

Evaluation

Whilst the simulation did produce data that appeared to match the predictions laid out in the hypothesis, there are areas that this experiment can be improved. As always, more data could have been used to provide a more realistic and reliable results from the simulations, but in order to preserve practicality in handling the output data, the number of processes and the number of tests was limited. From graph 2.3, the results show that for both the waiting times and turnaround times, Seed1 provides very low averages comparatively to Seed2 &3. Possibly in this case Seed1 was more optimized for the schedulers. A way to check this is to run more tests using more seeds to see whether Seed1 is an anomaly.

A threat to the validity of this experiment could be the efficiency of the implementation of the scheduling algorithms. This could in turn affect the performance of the scheduling algorithm, giving increased waiting & turnaround times. A way to resolve this is by conducting the same experiment with more than one implementation of the schedulers, allowing for more reliability towards the results.

In addition, to improve reliability and accuracy of results, more tests can be conducted such as; with more random seeds, an increased number mean inter-arrival values tested, allowing for an even narrower scope, or even a larger number of processes being tested. In this case, possibly more seeds may want to be tested due to the drastic difference in performance between Seed1 and the other two seeds, displayed in Graph 2.3.

Experiment 3

The question investigated in this experiment is as follows;

Does the Time Quantum parameter affect the waiting and turnaround time for processes running a Round Robin scheduling algorithm, more than a process running in a Multi-Level Feedback Queue with Round Robin scheduling?

Introduction

The two scheduling algorithms that have a dependence on the time quantum variable are the Round Robin and Multi-Level Feedback Queue with Round Robin. From this I found it of interest to observe how the performance of the algorithms are effected by altering the time quantum, if there is any. Waiting time and turnaround time are chosen as metrics for this experiment as these metrics given a good evaluation of the performance of a scheduling algorithm, in addition low waiting/turnaround is desirable for a Round Robin implementation. Waiting time is denoted as turnaround time subtract the CPU time blocked time. This is a time value for how long the process sat in the ready queue for. In practice, the lower the waiting time, the better performing the scheduler algorithm is. Moreover, the turnaround time is denoted as the terminated time subtract the created time. This can be seen as the time taken from entering the pipeline, until it leaves the pipeline. The lower the turnaround time, the better performing the scheduler is.

The hypothesis for this experiment was that the Multi-Level Feedback Queue with Round Robin algorithm will perform better than the Round Robin implementation. Further, as the time quantum increases, the performance of the Round Robin will improve, up to a point where if the quantum is too large, the scheduling will degenerate and work like a First Come First Served algorithm.

Methodology

In order to obtain useful averages for waiting time and turnaround time, six different time quantum values were selected; 2, 5, 10, 15, 20, 25. These values where chosen to test boundary limits (i.e. 2), as well as being able to estimate a value for where the scheduling degenerates, and

```
seed=11111111 | 22222222 | 33333333
meanInterArrival=150.0
meanCpuBurst=15.0
meanIOBurst=15.0
meanNumberBursts=2.0
numberOfProcesses=75
staticPriority=0
```

Figure 3. 1: Input parameters used to set up generation of input files in experiment 3

begins to mimic a First Come First Served implementation. For each time quantum value, the two scheduling algorithms were run on three sets of 75 processes, obtained using three different seeds. To obtain the input data, the input parameters described in figure 3.1, stored as parameter files in the /experiment3 folder, with input parameters for each seed.

The three seeds in figure 3.1 are the different seeds being used for the set of time quantum's, denoted in figure 3.2, which describes the simulator parameters for the experiment. For each seed, there is 12 different simulator_parameters.prp, since there is six different time quantum values for each of the two scheduling algorithms.

```
scheduler=RRScheduler | FeedbackRRScheduler*
timeLimit=10000
periodic=false
interruptTime=0
timeQuantum=2 | 5 | 10 | 15 | 20 | 25
initialBurstEstimate=10
alphaBurstEstimate=0.5
```

Figure 3. 2: Simulator parameters used to set up the simulator for experiment 3. (RR - Round Robin)

The results of these simulations are stored with the respective seed in the /experiment3 folder. The output files are named outputXY.out, where the variables X and Y denote the type of scheduler. X = {RR | Feed}, Y = {1|2|3|4|5|6}. The data from the output files was imported to Excel, so each output can have its average waiting and turnaround times calculated.

For each seed, there will be a calculated average of waiting time & turnaround time, for each scheduler. This is done by taking the sum of all waiting/turnaround times in an output file and dividing that sum by the number processes in that iteration.

Results

Round Robin Scheduler Turnaround times

Time Quantum	Seed 1	Seed 2	Seed 3	Average Turnaround Time (ms)
2	40.849	49.106	52.523	47.493
5	40.808	48.939	52.477	47.408
10	40.959	48.773	52.831	47.521
15	39.973	48.545	52.938	47.152
20	39.890	48.758	53.492	47.380
25	40.452	48.758	53.554	47.588

Table 3. 1: Round Robin - Average Turnaround time for each time quantum, rounded to 3.d.p

Multi-Level Feedback Queue with Round Robin Scheduler Turnaround times

Time Quantum	Seed 1	Seed 2	Seed 3	Average Turnaround Time (ms)
2	41.329	48.515	52.338	47.394
5	40.699	48.591	52.554	47.281
10	41.438	48.636	52.400	47.492
15	40.247	48.545	52.446	47.079
20	40.534	48.879	53.092	47.502
25	40.270	48.682	53.354	47.435

Table 3. 2: Multi-Level Feedback Queue with Round Robin - Average Turnaround time for each time quantum, rounded to 3.d.p

Round Robin Scheduler Waiting times

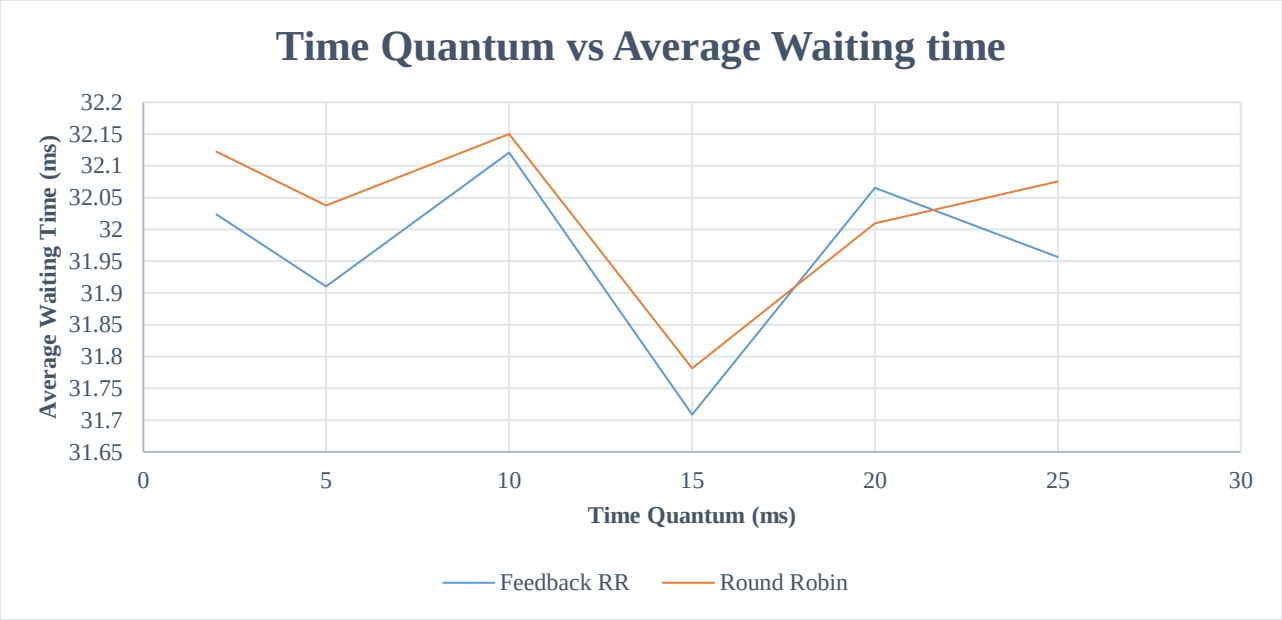
Time Quantum	Seed 1	Seed 2	Seed 3	Average Waiting Time (ms)
2	22.836	38.500	35.031	32.122
5	22.795	38.333	34.985	32.037
10	22.945	38.167	35.338	32.150
15	21.959	37.939	35.446	31.781
20	21.877	38.152	36.000	32.009
25	22.014	38.152	36.062	32.076

Table 3. 3: Round Robin - Average Waiting time for each time quantum, rounded to 3.d.p

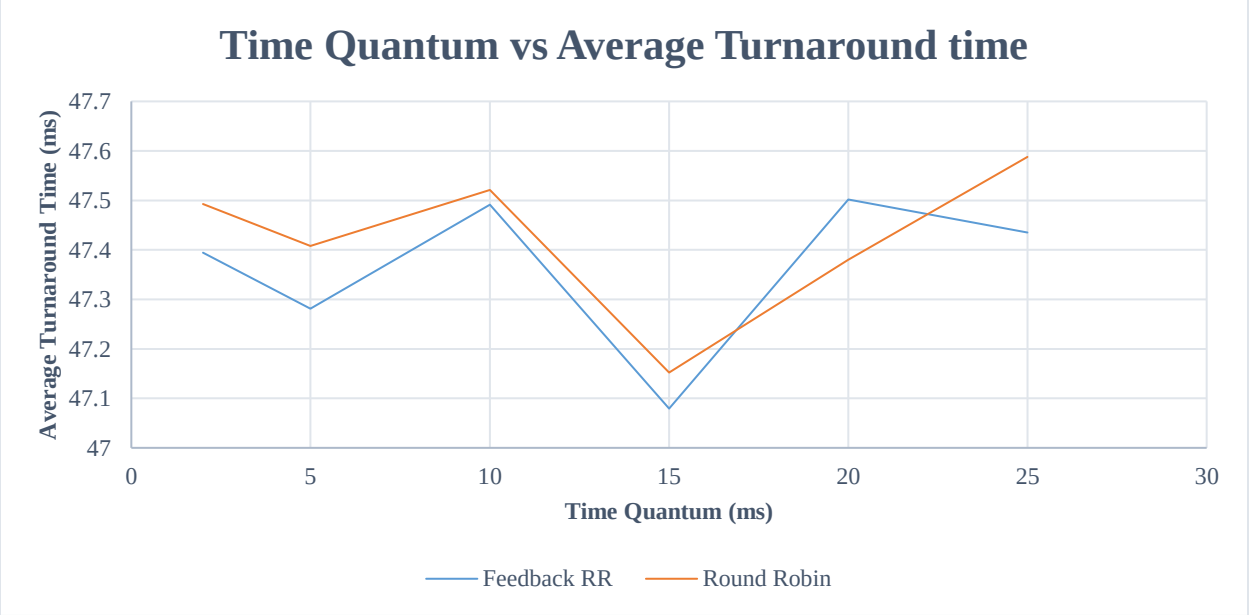
Multi-Level Feedback Queue with Round Robin Scheduler Waiting times

Time Quantum	Seed 1	Seed 2	Seed 3	Average Waiting Time (ms)
2	23.315	37.909	34.846	32.023
5	22.685	37.985	35.062	31.910
10	23.425	38.030	34.908	32.121
15	22.233	37.939	34.954	31.709
20	22.521	38.076	35.600	32.065
25	21.932	38.076	35.862	31.957

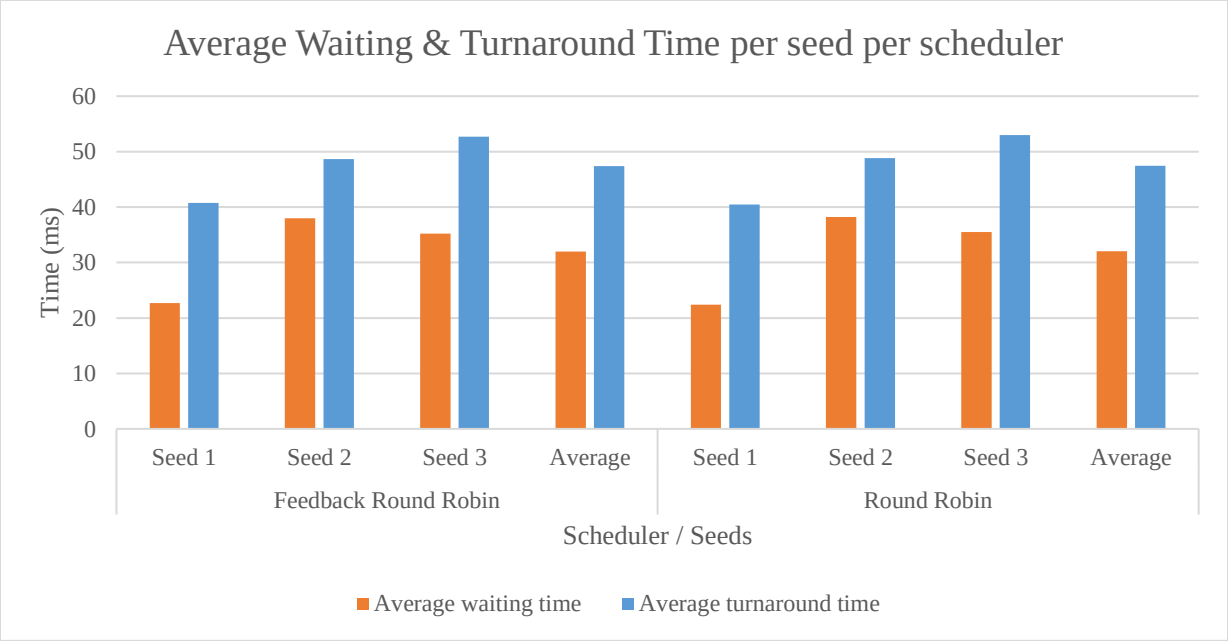
Table 3. 4: Multi-Level Feedback Queue with Round Robin - Average Waiting time for each time quantum, rounded to 3.d.p



Graph 3. 1: Graph displaying the Average Waiting time against the Time Quantum



Graph 3. 2: Graph displaying the Average Turnaround time against the Time Quantum



Graph 3. 3: Graph displaying the Average Turnaround time & Waiting Time for each seed, per scheduler.

Discussion

The results show that both algorithms react similarly to the adjustments to the time quantum. Overall, the adjustment on the time quantum does not make a noticeable difference towards performance since the maximum range of Average waiting times for both schedulers are within 0.5ms stated in Tables 3.3 & 3.4, as well as being displayed in Graph 3.1. In addition the turnaround times, shown in Table 3.1 & 3.2 support this with a maximum range of Average Turnaround times being ~0.4ms, further being illustrated in Graph 3.2. The reason for the limited difference in performance is the case, as by increasing the time quantum, increases the time a process would have to wait in the ready queue, along with how long the process runs for. This shows that a good time quantum value would be one which is as close to the average burst time, so most processes have enough time to be processed, along with not having to wait long in the ready queue. Graphs 3.1 & 3.2 suggest that a time quantum ~15ms for this data is the best performing, due to having to lowest wait time, for both schedulers, and the lowest turnaround time.

The hypothesis for this experiment, as made in the introduction, was correct to the results observed, as seen in Graph 3.1 & 3.2. The hypothesis did predict that the Multi-Level Feedback Queue with Round Robin would outperform the Round Robin scheduler, and in Graph 3.1, for almost all of experiment having lower turnaround & waiting times for all time quantum's except the time quantum of 20ms. This is because the Round Robin scheduler does not have access to priority, shorter processes are not prioritized to be finished earlier, therefore increasing the time other processes may need to wait until they are started and terminated, affecting the turnaround and waiting times. In addition, from the graphs, we can deduce that the time slice of

approximately ~17ms quantum is too large, thus the performance began to degenerate and work like a First Come First Served algorithm, since the increase in both Waiting & Turnaround times.

Evaluation

Whilst the simulation did produce data that appeared to match the predictions laid out in the hypothesis, there are areas that this experiment can be improved. More data could have been used to provide a more realistic and reliable results from the simulations, but in order to preserve practicality in handling the output data, no more data was used. From graph 3.3, the results show that for both the waiting times and turnaround times, there was a good distribution, since the average times are relatively around all the results for the seeds. Possibly by using more seeds, less of a distribution for waiting & turnaround times may occur, since more extensive testing, would suffice as more reliable results.

A threat to the validity of this experiment could be the efficiency of the implementation of the scheduling algorithms This could in turn affect the order in which the processes are added to the ready queue, as well as whether the run for more/less than the time slice value. A way to resolve this is by conducting the same experiment with more than one implementation of the schedulers, allowing for more reliability towards the results. In addition, to improve reliability and accuracy of results, more tests can be conducted such as; with more random seeds, an increased number of time quantum values tested, allowing for an even narrower scope, a larger number of processes being tested. In addition to improve on this experiment, more averages such as response time can be used to convey the performance of a scheduler. Having more ways to convey performance would ultimately give a more accurate and reliable representation of how performance is impacted when the time quantum value is increased.

Conclusion

In Conclusion, the experiments produced majority results that had been hypothesized, which does indicate the reliability of the experiments. In addition, by using graphs such as Graph 1.4, 2.3 & 3.3, that display the average results calculated for each seed, per scheduler, the results can be expressed visually as reliable as long as the averages have been calculated effectively, along with no anomaly results being used within averages. I don’t believe I have any large anomalies present in my testing, so all results collected were used towards averages. Through conducting these experiments, I believed it has given me an insight into how the different parameters tested in this report effect the schedulers along with impact parameters can have on efficiency.