Experiment 1- Evaluating Scheduler Efficiency in High Utilization Scenarios

This study aims to identify and evaluate different CPU scheduling methods, specifically FCFS, Round Robin, Feedback Round Robin, SJF, and Ideal SJF, under consistent load (50 user processes) and under a system load condition (high CPU utilisation). The objective is to assess scheduling efficiency by examining feedback on key performance indicators, including CPU utilisation, turnaround time, waiting time, and response time. The responses of each scheduler to load will be examined. In the end, this study will inform the most effective scheduling algorithm for scheduling when the system is under load.

Methodology

To simulate a high CPU load environment, I configured the input generator with the following parameters:

- numberOfProcesses=50
- staticPriority=0
- meanInterArrival=20.0
- meanCpuBurst=30.0
- meanIOBurst=10.0
- meanNumberBursts=6.0

The 5 different seeds used for generating the input files are:

- 270826029269605
- 983472389012348
- 174839201837465
- 657382910283746
- 890123456789012

The simulator configuration that I kept consistent across all the runs:

- timeLimit=10000
- periodic=false
- interruptTime=0
- timeQuantum=5
- initialBurstEstimate=5
- alphaBurstEstimate=0.9

Input Files (With different seeds)

- Input1_highload.in (270826029269605)
- Input2_highload.in (983472389012348)
- input3_highload.in (174839201837465)
- input4_highload.in (657382910283746)
- input5_highload.in (890123456789012)

Output Files (For each scheduler and input file)

- **FCFS Scheduler**
  - output_fcfs_highload1.out
  - output_fcfs_highload2.out
  - output_fcfs_highload3.out
  - output_fcfs_highload4.out
  - output_fcfs_highload5.out
- **FeedbackRR Scheduler**
  - output_feedbackrr_highload1.out
  - output_feedbackrr_highload2.out
  - output_feedbackrr_highload3.out
  - output_feedbackrr_highload4.out
  - output_feedbackrr_highload5.out
- **IdealSJF Scheduler**
  - output_idealsjf_highload1.out
  - output_idealsjf_highload2.out
  - output_idealsjf_highload3.out
  - output_idealsjf_highload4.out
  - output_idealsjf_highload5.out
- **RR Scheduler**
  - output_rr_highload1.out
  - output_rr_highload2.out
  - output_rr_highload3.out
  - output_rr_highload4.out
  - output_rr_highload5.out
- **SJF Scheduler**
  - output_sjf_highload1.out
  - output_sjf_highload2.out
  - output_sjf_highload3.out
  - output_sjf_highload4.out
  - output_sjf_highload5.out

File used for Input parameters

- input_parameters_highload.prp

File used for Simulator parameters

- simulator_Fcfs_highload.prp
- simulator_FeedbackRR_highload.prp
- simulator_IdealSJF_highload.prp
- simulator_RR_highload.prp
- simulator_SJF_highload.prp

Metrics Calculated

1. **CPU Utilisation:** The percentage of time the CPU was actively working and not idle during the simulation.
   Formula: CPU Utilisation = 1 - (Idle Time / Total Time)
2. **Turnaround Time:** The total time taken by a process from the moment it is created to the moment it is terminated.
   Formula: Terminated Time - Created Time
3. **Waiting Time:** The total time a process spends waiting in the ready queue not being executed.
   Formula: Turnaround Time - CPU Time - Blocked Time
4. **Response Time:** The time from when a process is created to when it is first scheduled on the CPU.
   Formula: Started Time - Created Time

Validation of results

The results shown in this experiment are valid within the constraints of the simulation environment, including the parameters chosen.

1. **Multiple Seeds for Input Diversity**

   To mitigate the bias of any one input scenario, five separate input files were produced with different random seeds. This enables some variation in process inter-arrival times, burst times, and the number of bursts, averaging out some outliers or struck algorithmic patterns.

2. **Same Simulator Configuration**

   All scheduling algorithms were tested using the same simulation parameters, Same input files for each scheduler, Same simulation time limit and alpha/burst estimates, Same number of processes (which was 50), and Same time quantum where it was applicable. This stops any differences between the algorithms from being related to the environment (in which they run).

3. **Controlled high-load condition**

We designed the system to run under a high CPU load so that the CPU is intentionally almost always busy. Given that we want to test the algorithms under challenging conditions, as relevant to the real world, we wanted all the metrics, like idle time to be low to confirm high utilisation.

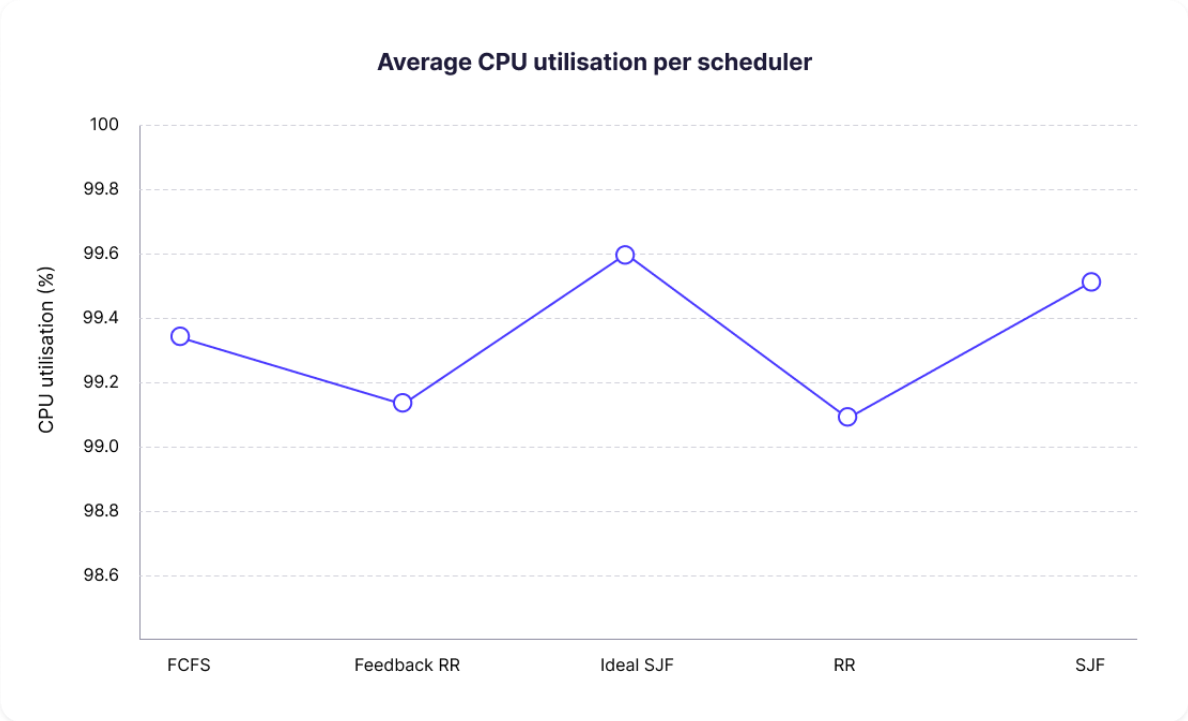
4. **Averages taken from multiple runs**

All performance measures (turnaround time, waiting time, response time, and CPU utilisation) were averaged across the 5 runs. This takes away, on average, the impact of any one anomalous point and provides a more robust evaluation.

Results

The graphs below compare the average CPU utilisation, Turnaround time, waiting time, and Response time of five different scheduler algorithms under high system load across five different seeded simulations.

CPU Utilisation



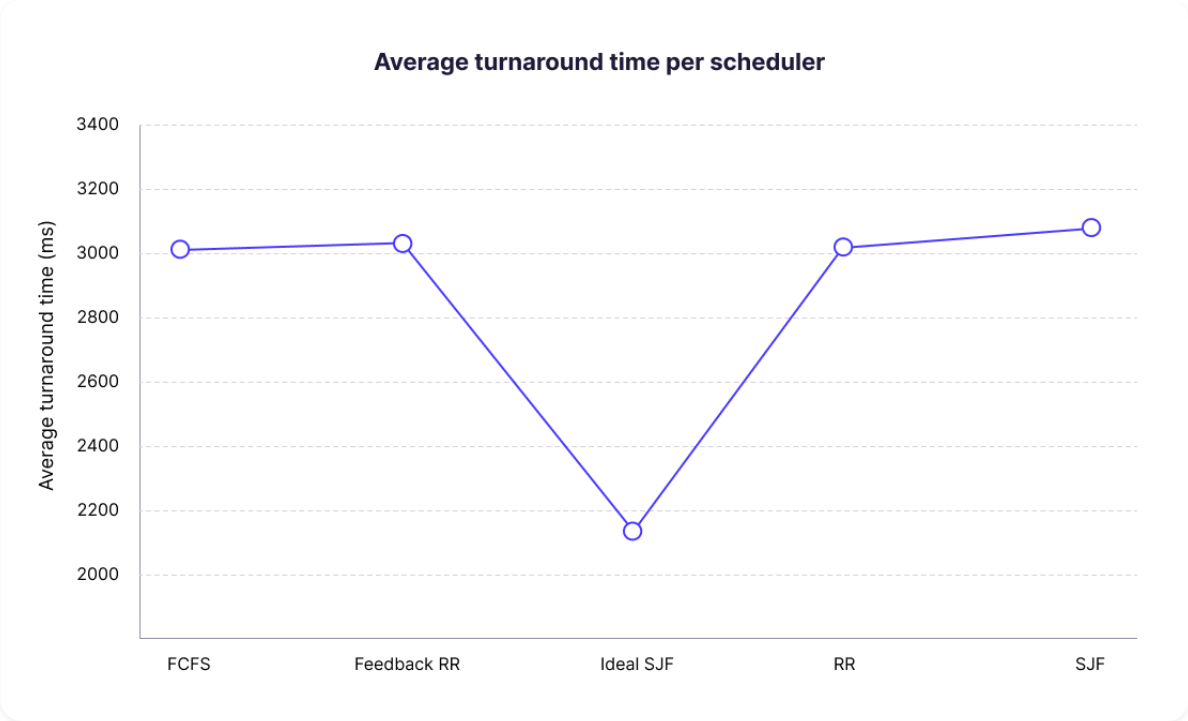
All schedulers performed well on utilisation, being greater than 99%. This is predictable in a high-load situation, where the CPU is nearly always engaged.

The ideal SJF (99.59%) and SJF (99.52%) are slightly ahead. This performance indicates their effectiveness in reducing idle time by efficiently selecting shorter bursts, thereby limiting waiting for longer jobs under high load while keeping the CPU busy with available processes.

FCFS (99.37%) is average as it does not have any optimisation - it strictly schedules processes in order of arrival. It benefits solely from the fact that there is always a process in the queue in high-load situations that it can distribute amongst the processes.

Feedback RR (99.16%) and simply RR (99.11%) are slightly lower as the time-slicing and the switching of the queue introduce a small amount of overhead that creates slightly more context switching, then subsequent very brief time periods of conflict in the queues where the CPU might be very briefly idle.

Turnaround Time

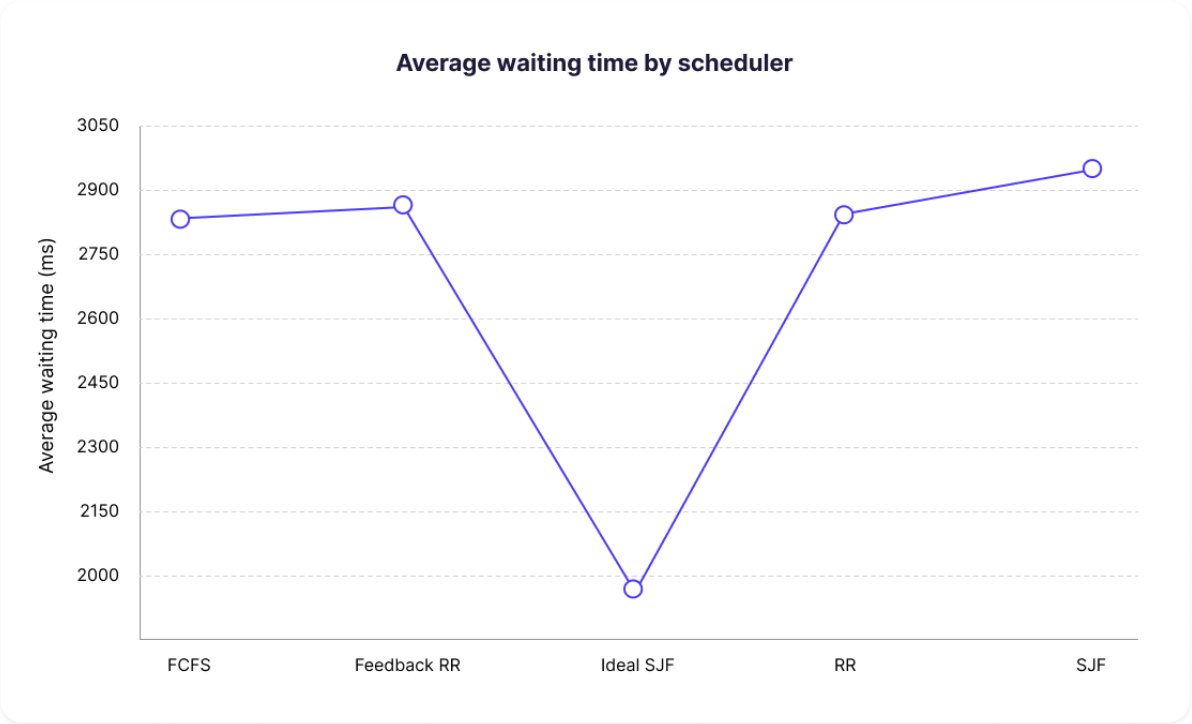


The ideal SJF demonstrates the most effective (2133.05). It is always aware of the next job that occupies the shortest run time while minimising waiting and improving flow.

Regular SJF demonstrates the most ineffective performance (3070.55), which may be surprising. We can assume the prediction of bursts is incorrect, resulting in suboptimal jobs getting scheduled and jobs of longer run times being postponed.

The RR, Feedback RR, and FCFS are fairly close in the ~3010–3020 range. Because they are less sensitive to the length of burst times and follow a fair, i.e., an arrival-based policy, they tend to b

Waiting Time

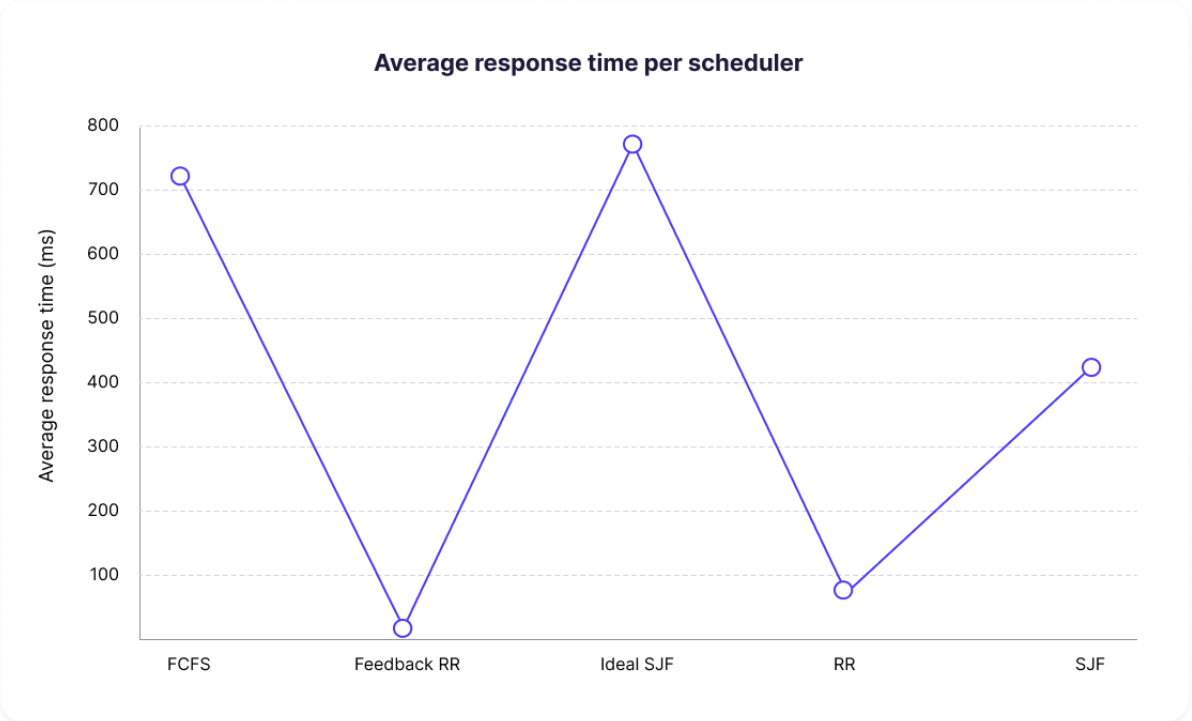


Ideal SJF has the best performance for the average wait time with a value of 1976.65, which is quite low compared to the other schedulers. This is expected from the Ideal SJF algorithm, which always selects the process with the shortest actual CPU burst time and minimises time spent in the ready queue.

SJF has the highest average wait time of 2914.15. This indicates that an incorrect estimate for a predicted burst time will better scheduling, but in a high-load situation, even a small estimation error can lead to longer waiting times before the process gets the CPU.

FCFS, RR and Feedback RR have relatively the same average wait times (ranging from 2849.14 to 2865.51). FCFS, RR and Feedback RR all stick to simple queues and fair time-slicing approaches as opposed to prioritising a process based on the time-burst duration, leading to a less favourable wait time compared to Ideal SJF.

Response Time



Feedback RR has the best mean response time of 0.40 ms, which is effectively negligible. This is expected from Feedback RR schedulers, which try to dispatch short processes and will grant a new process the CPU quickly after it arrives. For the workload in this test, there is very little time from arrival to start-up for each process.

RR was also strong for mean response time at 96.64 ms. Since RR uses time-slicing to allow processes to move to execution fairly quickly upon arrival (even if they are not going to finish in that time slice), it is good for workloads with interactive or time-sensitive processes.

SJF and Ideal SJF were very efficient in terms of mean turnaround and waiting time while processes were in a queue, but they had higher mean response times of 417.84 ms and 786.76 ms, clearly due to scheduling processes in favour of shorter jobs. This meant that longer processes often experienced the longest delays, in some cases very long delays, from the time of first execution to their first execution.

Surprisingly, FCFS has a better mean response time (710.74 ms) than Ideal SJF, likely as a consequence of the fact that FCFS schedules processes strictly in the order of arrival, allowing every process the CPU in-favour of shorter burst times.

Experiment 2- Real-Time Sensitivity in Scheduling Algorithms

This experiment is aimed at knowing how five different scheduling algorithms (FCFS, Feedback RR, Ideal SJF, RR, and SJF) behave in a real-time situation where timing and responsiveness matter. The experiment seeks to discern how well each scheduling algorithm performs when scheduling processes that arrive at different times and have different lengths of execution. Through a real-time sensitive perspective, the goal of the experiment is to assess each scheduler's ability to respond and minimize delay to their tasks, as time is critical for most real-time sensitive components in the system.

Methodology

To simulate real-time conditions, I configured the input generator with the following parameters:

- numberOfProcesses=20
- staticPriority=0
- meanInterArrival=80.0
- meanCpuBurst=5.0
- meanIOBurst=10.0
- meanNumberBursts=3.0

The 5 different seeds used for generating the input files are:

- 270826029269605
- 983472389012348
- 174839201837465
- 657382910283746
- 890123456789012

The simulator configuration that I kept consistent across all the runs:

- timeLimit=10000
- periodic=false
- interruptTime=0
- timeQuantum=5
- initialBurstEstimate=5
- alphaBurstEstimate=0.9

Input Files (With different seeds)

- input1_realtime.in (270826029269605)
- Input2_realtime.in (983472389012348)
- input3_realtime.in (174839201837465)
- input4_realtime.in (657382910283746)
- input5_realtime.in (890123456789012)

Output Files (For each scheduler and input file)

- **FCFS Scheduler**
  - output_fcfs_realtime1.out
  - output_fcfs_realtime2.out
  - output_fcfs_realtime3.out
  - output_fcfs_realtime4.out
  - output_fcfs_realtime5.out
- **FeedbackRR Scheduler**
  - output_feedbackrr_realtime1.out
  - output_feedbackrr_realtime2.out
  - output_feedbackrr_realtime3.out
  - output_feedbackrr_realtime4.out
  - output_feedbackrr_realtime5.out
- **IdealSJF Scheduler**
  - output_idealsjf_realtime1.out
  - output_idealsjf_realtime2.out
  - output_idealsjf_realtime3.out
  - output_idealsjf_realtime4.out
  - output_idealsjf_realtime5.out
- **RR Scheduler**
  - output_rr_realtime1.out
  - output_rr_realtime2.out
  - output_rr_realtime3.out
  - output_rr_realtime4.out
  - output_rr_realtime5.out
- **SJF Scheduler**
  - output_sjf_realtime1.out
  - output_sjf_realtime2.out
  - output_sjf_realtime3.out
  - output_sjf_realtime4.out
  - output_sjf_realtime5.out

File used for Input parameters

- input_parameters_realtime.prp

File used for Simulator parameters

- simulator_Fcfs_realtime.prp
- simulator_FeedbackRR_realtime.prp
- simulator_IdealSJF_realtime.prp
- simulator_RR_realtime.prp
- simulator_SJF_realtime.prp

Metrics Calculated

1. **CPU Utilisation:** The percentage of time the CPU was actively working and not idle during the simulation.
   Formula: CPU Utilisation = 1 - (Idle Time / Total Time)
2. **Turnaround Time:** The total time taken by a process from the moment it is created to the moment it is terminated.
   Formula: Terminated Time - Created Time
3. **Waiting Time:** The total time a process spends waiting in the ready queue not being executed.
   Formula: Turnaround Time - CPU Time - Blocked Time
4. **Response Time:** The time from when a process is created to when it is first scheduled on the CPU.
   Formula: Started Time - Created Time

Validation of results

The results shown in this experiment are valid within the constraints of the simulation environment, including the parameters chosen.

1. **Multiple Seeds for Input Diversity**

   To mitigate the bias of any one input scenario, five separate input files were produced with different random seeds. This enables some variation in process inter-arrival times, burst times, and the number of bursts, averaging out some outliers or struck algorithmic patterns.

2. **Validation of Real-world simulations**

   In this study the aim was to simulate the real-time behaviour of systems in which processes arrive at different times and require service immediately. A real-time system requires a scheduler that can provide a predictable and efficient response time. The metrics chosen (turnaround time, waiting time, response time, and CPU utilization) measure the scheduling algorithms' effectiveness in scheduling these timed jobs

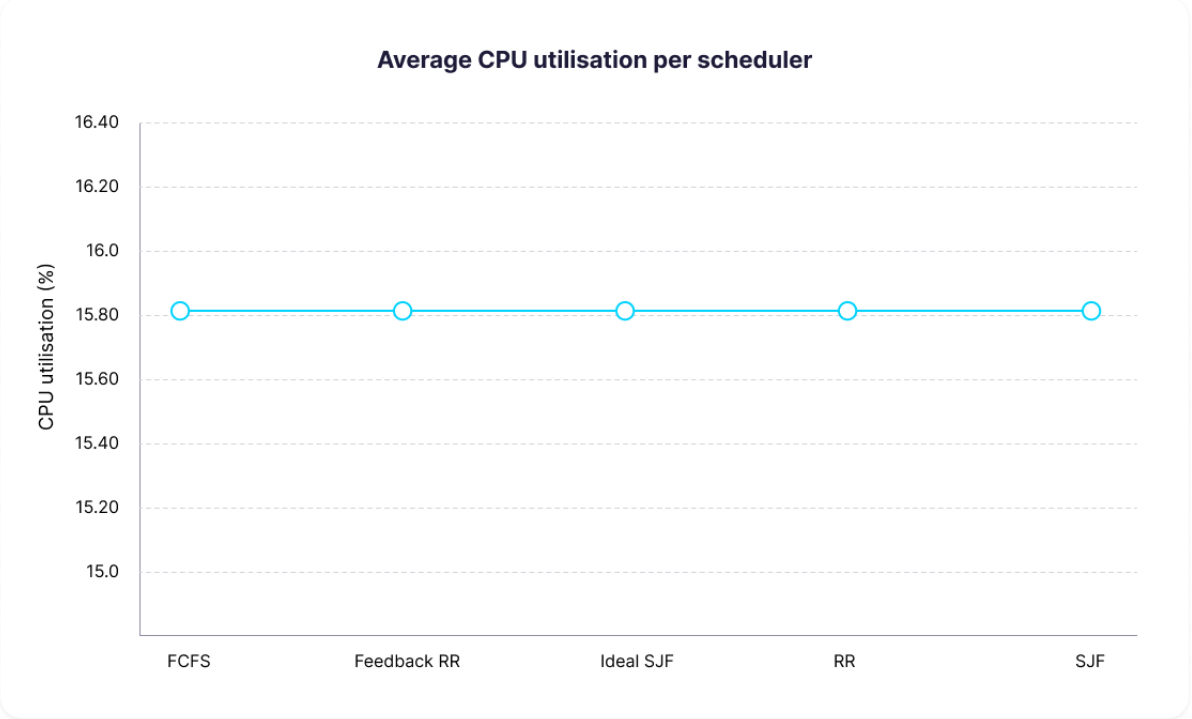
3. **Averages taken from multiple runs**

   All performance measures (turnaround time, waiting time, response time, and CPU utilisation) were averaged across the 5 runs. This takes away, on average, the impact of any one anomalous point and provides a more robust evaluation.

Results

The graphs below compare the average CPU utilisation, Turnaround time, waiting time, and Response time of five different scheduler algorithms under real-time load across five different seeded simulations.

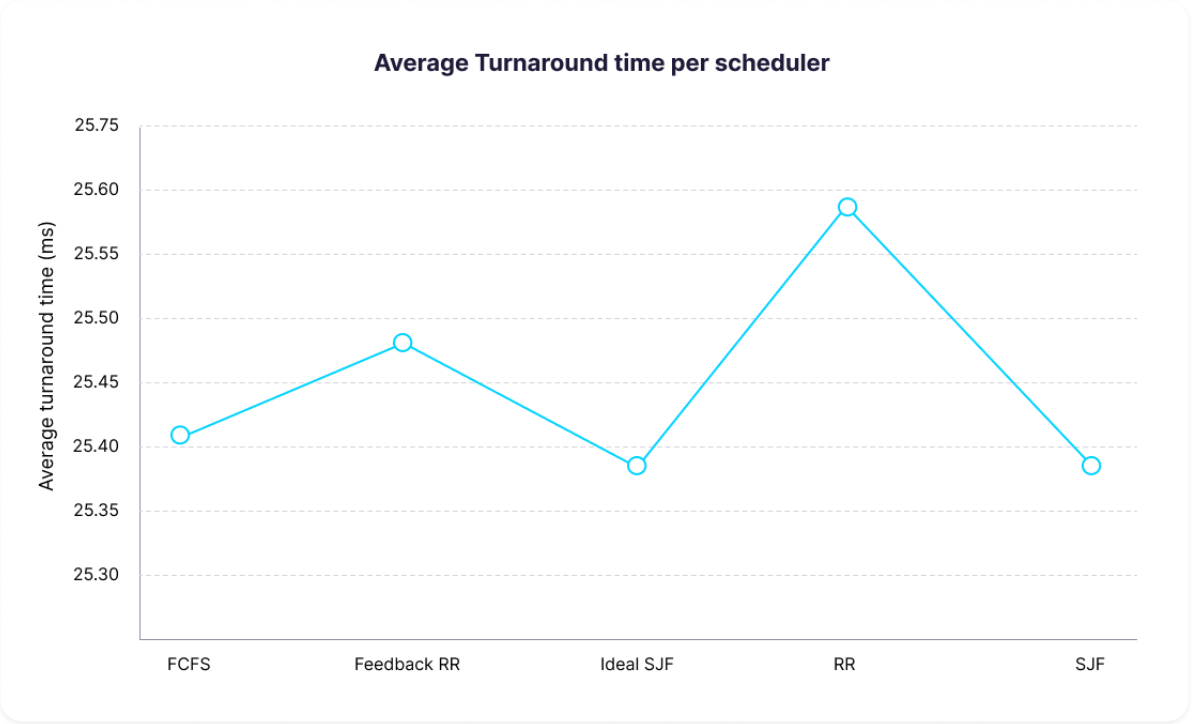
CPU Utilisation



On average, CPU utilisation was the same (15.81%) across all five schedulers (FCFS, Feedback RR, Ideal SJF, RR, and SJF). This consistency in utilisation suggests a pattern of underutilisation in the CPU for all scheduling approaches, which is likely due to a light load on the workers and idle time that results from workers arriving at different times.

This result illustrates that one of the aspects of scheduling behaviour under low to mid-load system conditions is that the CPU will be underutilized regardless of the scheduling approach or scheduler until conditions are sufficient enough or the load is consistent enough that a scheduler will have more impact on CPU utilisation.

Turnaround Time



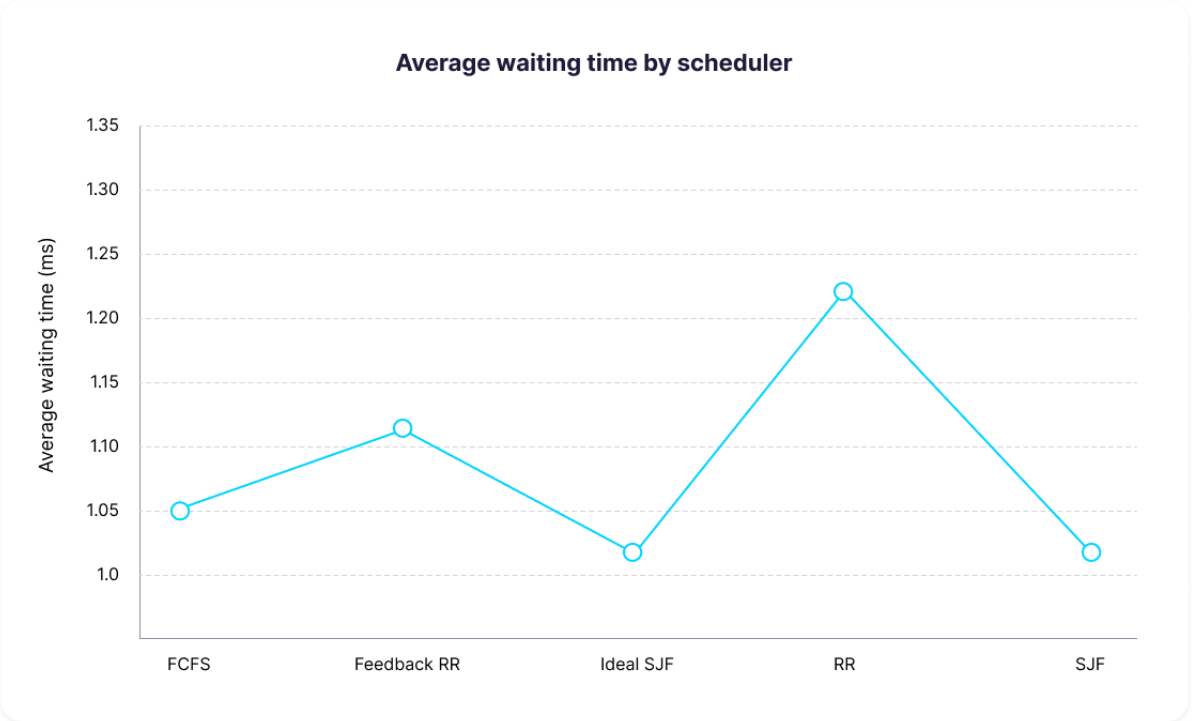
The turnaround time for all of the schedulers was relatively close, with turnaround times in a range of 25.38 ms to 25.58 ms.

The Ideal SJF and SJF schedulers had the lowest of turnaround times of 25.38 ms, indicating their potential for minimizing total completion time by scheduling the shorter jobs.

The RR scheduler had the highest turnaround time at 25.58 ms, which likely correlates with the time-slicing associated with the RR strategy, especially when each time slice causes a delay while completing longer processes.

Feedback RR and FCFS schedulers had turnaround times of 25.48 ms and 25.41 ms, respectively. Overall, the small variability in turnaround times suggests that when running at a lower system load with fewer processes, the differences in different scheduling strategies may be less noticeable for this particular metric.

Waiting Time



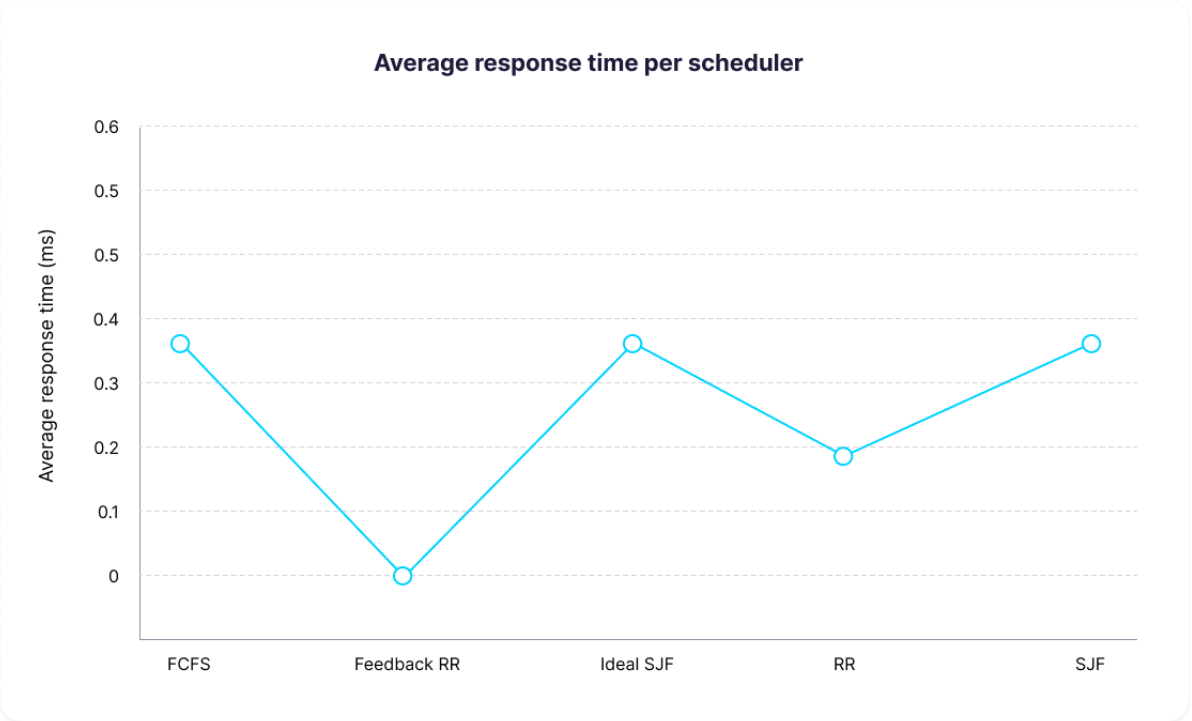
Both the Ideal SJF and SJF schedulers demonstrated the lowest average waiting time, at 1.02 ms, which shows their capability to minimize waiting times by selecting shorter jobs.

The FCFS scheduler was close behind at 1.05 ms, which is fairly good evidence that FCFS can be effective despite its simple nature and non-preemptive condition.

The Feedback RR and RR schedulers had higher average waiting times of 1.12 ms and 1.22 ms, respectively, than both SJF-based methods.

In general, the experiment highlighted the advantages of scheduling awareness of burst time when applying the scheduling methods in low latency scenarios.

Response Time



Feedback RR (0.00 ms) had instantaneous response times, which suggests that all processes were granted access to the CPU 'immediately'. This feature is due to the way feedback round robin has been designed, which has new or preempted processes 'jump to the head of the queue' to obtain CPU time.

RR (0.19 ms) performed "very well" due to time-slicing allowing processes, once placed in a ready state, to be cycled through very quickly.

FCFS, Ideal SJF and SJF (average response time of 0.36 ms) all performed similarly to each other with quick response in 'non-contention' situations.

Under a moderate load, all of the schedulers provided excellent responsiveness, although to a higher level with Feedback RR and RR being more suited for systems requiring an immediate option for process access to the CPU. In terms of fairness and quick switching (over which processes preceded it on the scheduler), all had very little time to delay in terms of execution and arrival at the CPU.

Experiment 3- Impact of Scheduling Algorithms on System Performance with Varying Process Loads

The purpose of this experiment is to examine how different scheduling algorithms perform under different conditions of the system loads. This experiment will examine how each algorithm - FCFS, FeedbackRR, Ideal SJF, RR, and SJF - performs while handling multiple tasks with different specifications. Specifically, we will examine turnaround time, waiting time, response time, and CPU utilisation. The purpose of Experiment 3 is to investigate how these algorithms perform under heavy loads and get an idea of each algorithm's effectiveness with processes and CPU utilisation.

Methodology

To simulate mixed workload conditions, I configured the input generator with the following parameters:

- numberOfProcesses=50
- staticPriority=0
- meanInterArrival=120.0
- meanCpuBurst=5.0
- meanIOBurst=30.0
- meanNumberBursts=10.0

The 5 different seeds used for generating the input files are:

- 270826029269605
- 983472389012348
- 174839201837465
- 657382910283746
- 890123456789012

The simulator configuration that I kept consistent across all the runs:

- timeLimit=10000
- periodic=false
- interruptTime=0
- timeQuantum=5
- initialBurstEstimate=5
- alphaBurstEstimate=0.9

Input Files (With different seeds)

- input1_mixedworkload.in (270826029269605)
- Input2_mixedworkload.in (983472389012348)
- input3_mixedworkload.in (174839201837465)
- input4_mixedworkload.in (657382910283746)
- input5_mixedworkload.in (890123456789012)

Output Files (For each scheduler and input file)

- **FCFS Scheduler**
  - output_fcfs_mixedworkload1.out
  - output_fcfs_mixedworkload2.out
  - output_fcfs_mixedworkload3.out
  - output_fcfs_mixedworkload4.out
  - output_fcfs_mixedworkload5.out
- **FeedbackRR Scheduler**
  - output_feedbackrr_mixedworkload1.out
  - output_feedbackrr_mixedworkload2.out
  - output_feedbackrr_mixedworkload3.out
  - output_feedbackrr_mixedworkload4.out
  - output_feedbackrr_mixedworkload5.out
- **IdealSJF Scheduler**
  - output_idealsjf_mixedworkload1.out
  - output_idealsjf_mixedworkload2.out
  - output_idealsjf_mixedworkload3.out
  - output_idealsjf_mixedworkload4.out
  - output_idealsjf_mixedworkload5.out
- **RR Scheduler**
  - output_rr_mixedworkload1.out
  - output_rr_mixedworkload2.out
  - output_rr_mixedworkload3.out
  - output_rr_mixedworkload4.out
  - output_rr_mixedworkload5.out
- **SJF Scheduler**
  - output_sjf_mixedworkload1.out
  - output_sjf_mixedworkload2.out
  - output_sjf_mixedworkload3.out
  - output_sjf_mixedworkload4.out
  - output_sjf_mixedworkload5.out

File used for Input parameters

- input_parameters_mixedworkload.prp

File used for Simulator parameters

- simulator_Fcfs_mixedworkload.prp
- simulator_FeedbackRR_mixedworkload.prp
- simulator_IdealSJF_mixedworkload.prp
- simulator_RR_mixedworkload.prp
- simulator_SJF_mixedworkload.prp

Metrics Calculated

5. **CPU Utilisation:** The percentage of time the CPU was actively working and not idle during the simulation.
   Formula: CPU Utilisation = 1 - (Idle Time / Total Time)
6. **Turnaround Time:** The total time taken by a process from the moment it is created to the moment it is terminated.
   Formula: Terminated Time - Created Time
7. **Waiting Time:** The total time a process spends waiting in the ready queue not being executed.
   Formula: Turnaround Time - CPU Time - Blocked Time
8. **Response Time:** The time from when a process is created to when it is first scheduled on the CPU.
   Formula: Started Time - Created Time

Validation of results

The results shown in this experiment are valid within the constraints of the simulation environment, including the parameters chosen.

1. **Multiple Seeds for Input Diversity**

To mitigate the bias of any one input scenario, five separate input files were produced with different random seeds. This enables some variation in process inter-arrival times, burst times, and the number of bursts, averaging out some outliers or struck algorithmic patterns.

2. **Averages taken from multiple runs**

All performance measures (turnaround time, waiting time, response time, and CPU utilisation) were averaged across the 5 runs. This takes away, on average, the impact of any one anomalous point and provides a more robust evaluation.

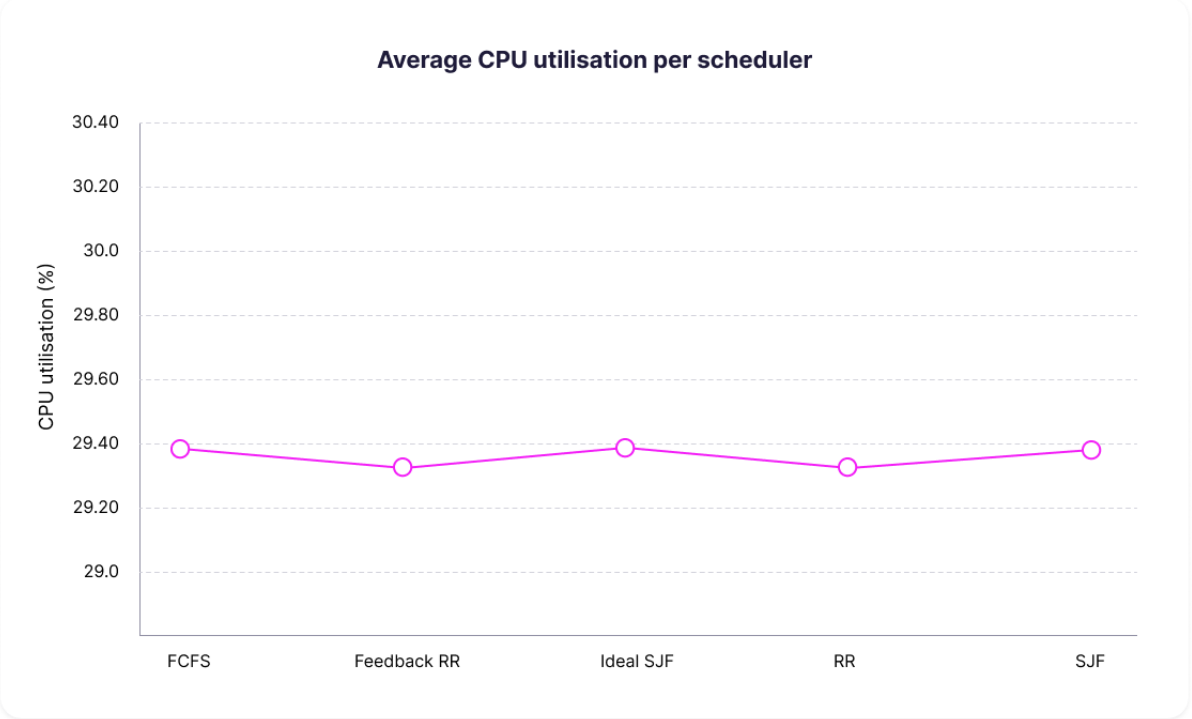
3. **Cross comparison between schedulers**

The across-application comparison for resiliency because it stimulated the various algorithms with the same input. This study will allow the reader to appreciate the particular strengths and weaknesses of the algorithms, and this conclusion will not be with any one algorithm. It specifically sheds light on the performance characteristics of the system between the algorithms in ideal conditions.

Results

The graphs below compare the average CPU utilisation, Turnaround time, waiting time, and Response time of five different scheduler algorithms under mixed workload across five different seeded simulations.

CPU Utilisation

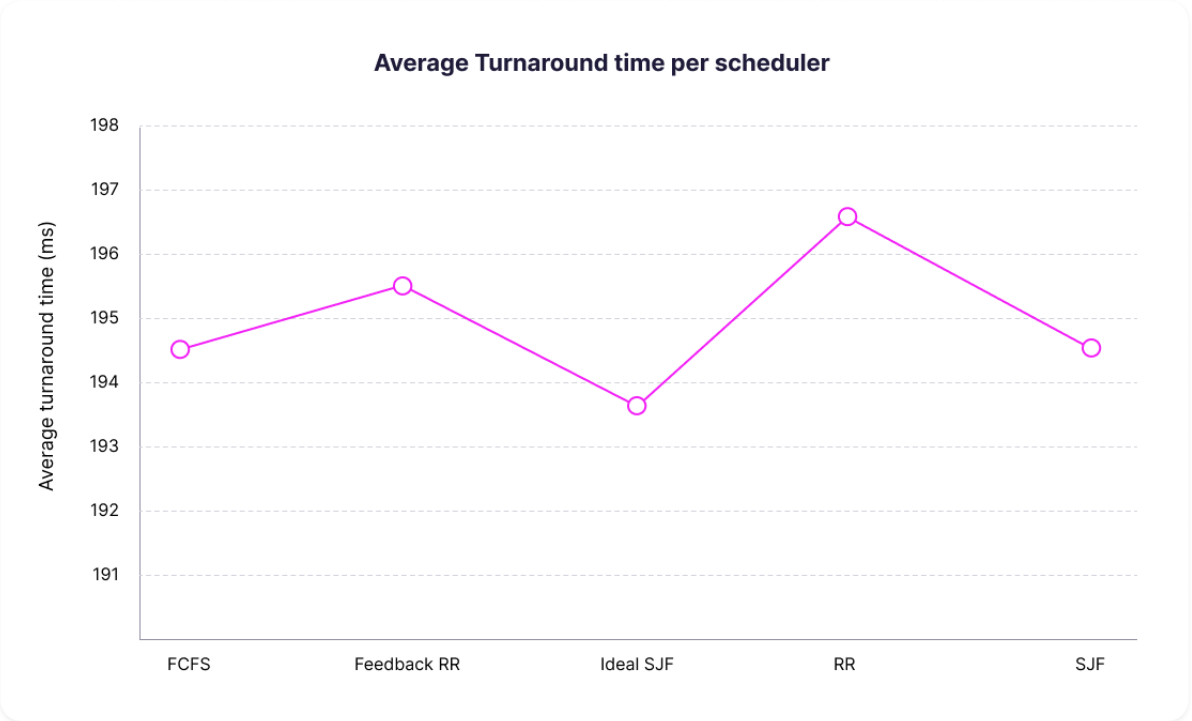


The CPU utilisation is nearly identical for all scheduling algorithms, remaining around 29%. This means that in Experiment 3, the system has similar levels of utilisation regardless of the scheduling algorithm.

This may be due to similar workloads in the experiment, indicating that the scheduling algorithms help balance the workloads and do not significantly impact utilisation.

The ideal SJF and SJF had slightly better utilisation of 29.39% than the FeedbackRR and RR at 29.35%. However, this is negligible. This shows that CPU utilisation was not significantly impacted by the scheduling algorithm run in this experiment.

Turnaround Time



FCFS has an average turnaround of 194.48, which is moderate. It works well when tasks fall in a similar range of burst time, but long jobs will cause longer delays.

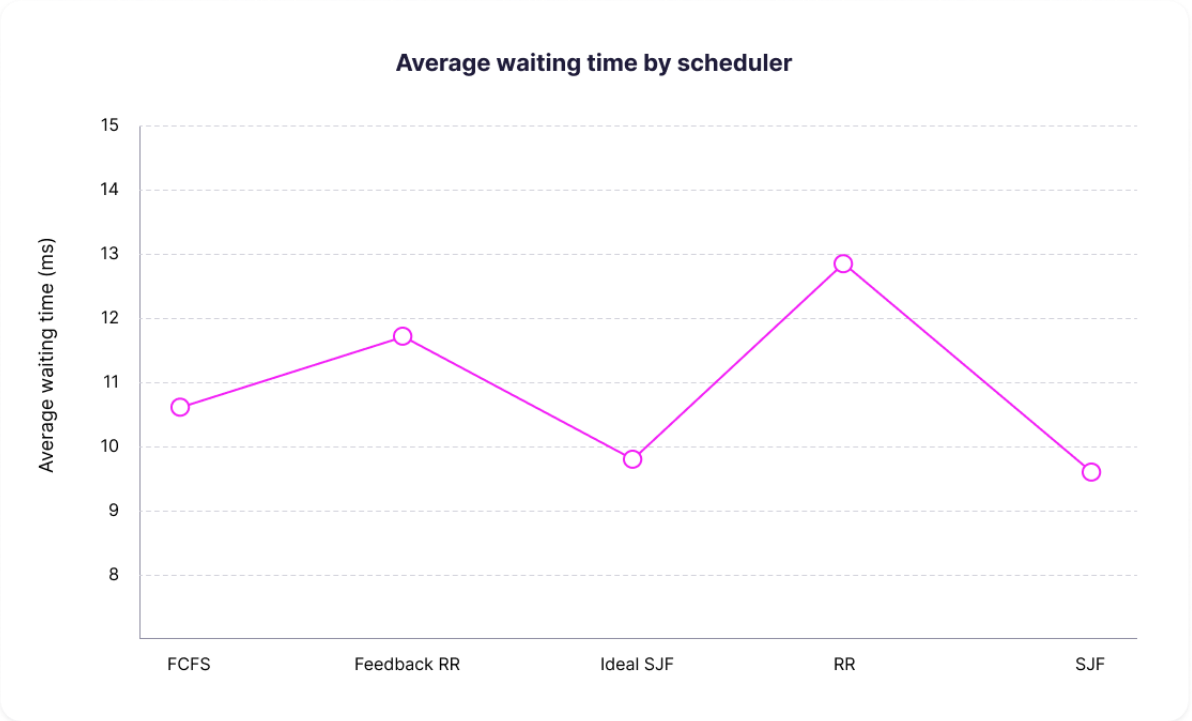
For FeedbackRR the average turnaround time is 195.50, which is a little worse than FCFS, because FeedbackRR can adapt the time slices. FeedbackRR struggled with long jobs due to the round-robin approach.

At 193.71, Ideal SJF had the lowest average turnaround time. Ideal SJF assigns priority to a shorter job first and performs better when the job type is shorter. However, it requires knowledge of burst time first.

RR had the average turnaround time of 196.65, the worst of all scenarios tested. The fixed time quantum of RR can cause additional time wasted, especially if tasks can vary in burst time.

SJF had a turnaround time of 194.49, similar to FCFS, and attempted to be the least wasteful in the minimum turnaround time by assigning the shortest job first, but like the Ideal SJF, it also faced a problem when the burst time was not known in advance.

Waiting time



The FCFS (10.68) scheduler has a fairly average waiting time in comparison to the waiting times of the rest of the schedulers which is indicative that a simple scheme relies on an average that is neither particularly high nor low.

The FeedbackRR (11.7) waiting time seems to be a little more than that of FCFS which indicates that the more complex feedback aspect of the FeedbackRR scheduler introduces a degree of waiting time for processes that are not completed as well and require the smaller quantum.

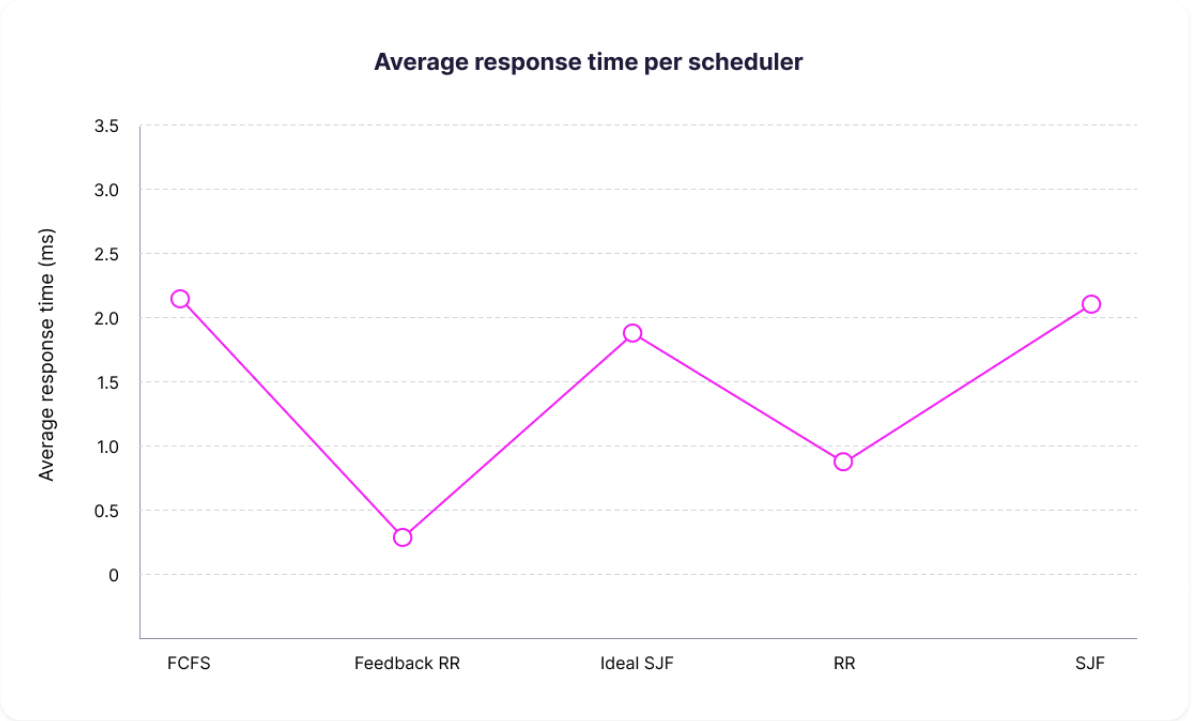
The ideal SJF scheduler reflects the shortest waiting time of all the schedulers (9.91) as expected due to the methodology of SJF, which executes shorter jobs first right after the job completion confirmation. As shorter jobs and tasks take less waiting time, shorter jobs finish faster than longer jobs which increases the overall average as far as the completion of smaller jobs as they are executed first.

The RR (12.85) is the highest amongst the waiting times which can arguably be associated with the static time quantum. Processes that require more CPU time likely tend to not be completed within one quantum of time are returned to the end of the queue as the fixed time span gets used in the one quantum of time as smaller processes are executed first and longer or larger processes have to wait for next availability to CPU time and thus create a higher waiting time average.

The average waiting time for SJF (10.69) is very close to FCFS. It can be interpreted that, while SJF is the optimal algorithm to minimise turnaround time, the results of SJF waiting

times do not differ significantly from FCFS waiting time when the processes are evenly distributed.

Response Time



FeedbackRR has the lowest mean response time (0.03), indicating it optimally and expediently responds to processes. FeedbackRR is intended to favour short jobs and quickly respond to new processes, resulting in negligible delay.

RR follows with a mean response time of 0.99, resulting in mean response slightly longer than FeedbackRR, but still quite low, indicating that Round Robin is still efficient in handling processes and quickly moving between processes.

The ideal SJF has a mean response time of 1.94, which is slightly larger than RR and FeedbackRR but demonstrates adequate execution time management. SJF processes the shortest job first, which minimizes waiting times and handles shorter jobs efficiently, resulting in a mean response time that is reasonable.

FCFS and SJF both result in larger response times (14 and 2.06, respectively), indicating that higher response times are probable. This is potentially a limitation to these algorithms if processes have a longer burst time to complete first, which will hinder computational efficiency. FCFS even more so suffers from longer response times due to the non-preemptive state and inability to process preemptively later arriving jobs.