CSC3002F DPRMAR021 OS1 Assignment

Mark Du Preez

May 2025

Investigation and evaluation of CPU process scheduling algorithms using a simulation model.

1 Introduction

The purpose of this assignment is to investigate, compare and evaluate different CPU process scheduling algorithms using a simulation model. The simulation is based on the context of a barman and the patrons entering the bar and placing a drink order. Using the standard performance metrics and criteria for scheduling algorithms, a recommendation will be made on the best scheduling algorithm for the simulation model (Sarah the Barman).

2 Background

CPU process scheduling is concerned with the decision-making on which process in the ready queue gets to be allocated to the CPU core at any time. We can categorize these algorithms into 2 categories: preemptive (**Round-Robin**) and non-preemptive (**First-Come First-Serve** and **Shortest-Job First**). This investigation will assume a uni-processor system. In order to investigate the scheduling criteria of these algorithms, a simulation is used involving one barman and many patrons placing different drink orders. In the simulation, a single barman acts as both a scheduler and the CPU core, whilst a patron acts as a process. Each patron will place five different drink orders (kept constant for all patrons). The preparation of a drink by the barman is considered a CPU burst, and the imbibing of that drink by a patron is considered an IO burst. A drink is considered to be on the ready queue when it is waiting to be prepared, and said drink is taken off the ready queue when it is being actively prepared by the barman. A context switch for the barman was also implemented to simulate the context switch of a process. The time quantum which is the fixed-time a process gets to spend on the CPU for the round-robin algorithm was also implemented.

3 Methodology

3.1 Implementation Details

This simulation was created in a Java multi-threaded environment, where system timers were inserted throughout the Patron and Barman class to measure the times of different drink orders being placed, prepared, completed and imbibed as well as patron arrival and completion times. The SchedulerSimulation class then prints these times to standard output which is later redirected to a file. A python file called dataprocessing.py takes the raw output data from the experiment, calculates the different performance metrics for each algorithm and then graphs and outputs the distribution, mean and median of these metrics for further analysis and evaluation.

An experiment to determine an optimal time quantum for the round-robin algorithm was first conducted where for each time quantum (5-150 in intervals of 5) the simulation was ran ten times (with a constant number of patrons, fixed random seed and the same context switch length) to generate reliable sample means of the response and turnaround times of a patron for each time quantum. The time quantum that has the lowest average response time as well as turnaround time for patrons was chosen as the optimal time quantum.

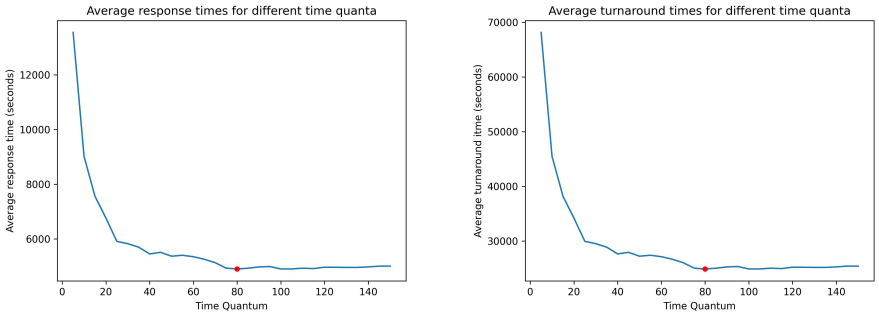
The second (main) experiment was then conducted, where the simulation was ran 30 times for each different scheduling algorithm with the same random seed, constant number of patrons and fixed context switch and time quantum (determined in previous experiment) to generate reliable sample means of the turnaround times, response times, waiting times of each patron as well as the throughput and CPU utilization of the barman for each scheduling algorithm, respectively, for further evaluation.

For both experiments, a sample size of 100 patrons was used and kept constant throughout the experiment to ensure the sample was random and representative enough to extract reliable data. Throughout both experiments for all simulation runs, a context switch of 5 seconds (5 milliseconds in simulation) was used as a reasonable estimate for the context switch of a barman which would be the time to move from preparing one drink to preparing another drink.

The scheduling algorithm metrics investigated are turnaround, waiting and response times of patrons/processes as well as the CPU utilization and throughput of the barman. The distribution, mean and median for each of these metrics were determined for each scheduling algorithm. The predictability, fairness and possibility of starvation for each scheduling algorithm was also evaluated.

4 Results, Analysis and Discussion

4.1 Time Quantum Experiment



(a) Average Response Time vs Time Quantum

(b) Average Turnaround Time vs Time Quantum

Figure 1: Effect of Time Quantum on Turnaround & Response Times

The data aligns well with our expectations regarding the behaviour of a varying time quantum. Notably, average turnaround time does not consistently decrease as the time quantum increases. For very small time quanta, both response and turnaround times tend to be high. This is because drink preparation progresses in very short time slices, leading to frequent context switches and delays in completion. As the time quantum increases, both response time and turnaround time decrease, since drinks are allowed to progress further during each time quantum. This reduces the total overhead incurred by many context switches.

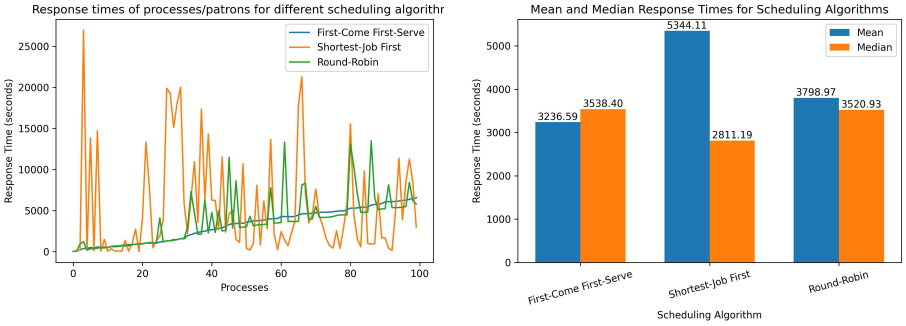
The time quantum that yields the lowest average response and turnaround times is 80 seconds, which was selected as the optimal time quantum and used in the main experiment. This value also aligns with the "80% Rule of Thumb," which recommends that the time quantum be greater than 80% of CPU burst durations. In our case, the fourth quintile (80th percentile) of drink preparation times (CPU bursts) is 75 seconds, making 80 seconds a suitable and theoretically sound choice according to this heuristic.

Note: Average waiting times for different time quanta were not plotted, as waiting time is simply the turnaround time minus the drink preparation times for all drinks (All CPU burst):

$$\text{Waiting Time} = \text{Turnaround Time} - \text{Total Drink Preparation Time} \quad (1)$$

4.2 Main Experiment

4.2.1 Response Time



(a) Average Response Time vs patron/process

(b) Mean and median response time for each scheduling algorithm

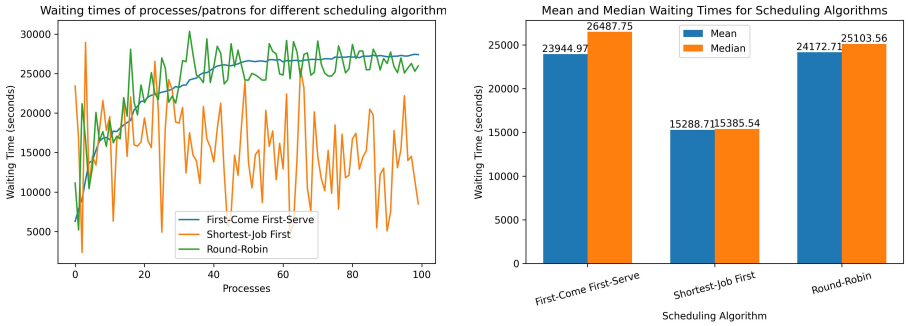
Figure 2: Distribution, mean and median of average response time per patron

The **First-Come-First-Serve** (FCFS) algorithm yields the lowest average response time among the three scheduling algorithms. It also exhibits the highest predictability, as indicated by the lowest variance in response times per patron. The distribution shows mean $<$ median, suggesting that earlier-arriving processes experience slightly better response times than those arriving, most likely due to a drink with a long preperation time temporarily blocking the ready queue.

In comparison, **Shortest-Job First** (SJF) produces the highest average response time and the lowest predictability, demonstrated by a high variance in response times. The distribution shows mean $>$ median indicating a positively-skewed distribution, where later-arriving patrons experience substantially longer response times. This skewness reflects poor fairness in process scheduling, as response times should ideally remain relatively consistent across all patrons.

Round-Robin (RR) achieves an average response time higher than FCFS but lower than SJF. Its average response time distribution is more predictable than SJF but less so than FCFS (indicated by the variance). The distribution exhibits symmetry similar to FCFS, indicating more balanced response times across different patrons which is desirable.

4.2.2 Waiting Time



(a) Average Waiting Time vs patron/process

(b) Mean and median waiting time for each scheduling algorithm

Figure 3: Distribution, mean and median of average waiting time per patron

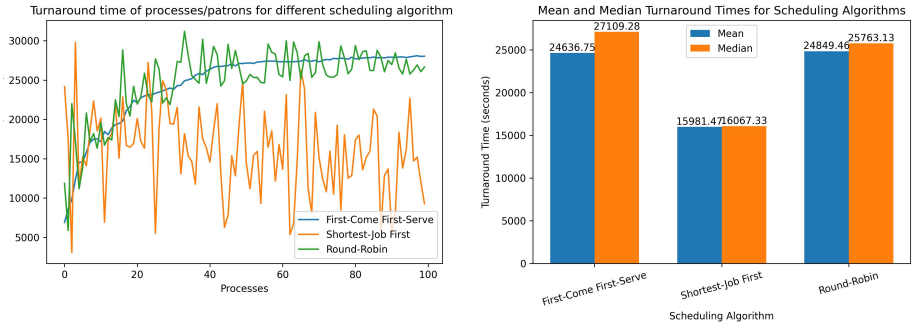
For **First-Come-First-Serve**, the average waiting time is relatively high, as expected. The **convoy effect** is clearly evident in the waiting time distribution; once a long-duration job (drink order) enters the queue, particularly early or mid-sequence, it delays all subsequent processes, driving up their waiting times. This introduces a risk of starvation, as some processes may wait significantly longer than others to complete. The distribution appears negatively-skewed (mean $<$ median), suggesting that earlier-arriving patrons experience shorter waiting times. The lack of symmetry in the distribution indicates unfairness, as waiting times are not evenly distributed among patrons. However, FCFS also exhibits the lowest variance in waiting times, implying that its performance is highly predictable, even if with higher waiting times which is often desirable.

Shortest-Job First, by design, ensures the minimum average waiting time for a given set of processes, which is reflected in our results as it yields the lowest mean and median waiting times across all algorithms. The mean and median are nearly identical, indicating a symmetrical distribution, which Figure 4(a) supports. This symmetry suggests that SJF is the most fair in terms of waiting times, regardless of when patrons arrive, they tend to experience similar waiting times. However, SJF is less predictable than FCFS, as indicated by its higher variance. Importantly, SJF shows little to no evidence of starvation, with no outliers or unusually high waiting times, which is very desirable for a good scheduling algorithm.

Round-Robin is known to produce relatively high average waiting times, and this notion is confirmed by our data, which show RR having the highest average waiting time among the three algorithms. Despite this, the waiting time distribution for RR is more symmetrical than that of FCFS, suggesting a greater degree of fairness. RR combines characteristics of both FCFS and SJF: it shares the predictability and potential for starvation seen in FCFS, yet

exhibits distributional fairness similar to SJF. Overall, RR represents a trade-off between the extremes of FCFS and SJF, offering more balanced waiting times than FCFS but at the cost of higher average waiting time than SJF.

4.2.3 Turnaround Time



(a) Average Turnaround Time vs patron/process

(b) Mean and median turnaround time for each scheduling algorithm

Figure 4: Distribution, mean and median of average turnaround time per patron

$$\text{Turnaround Time} = \text{Waiting Time} + \text{Total Drink Preparation Time} \quad (2)$$

The turnaround time plot closely mirrors the waiting time plot, differing primarily by a slight vertical shift upward due to the addition of drink preparation time as showed in equation (2). Therefore, the patterns and interpretations for waiting time are largely preserved for turnaround times.

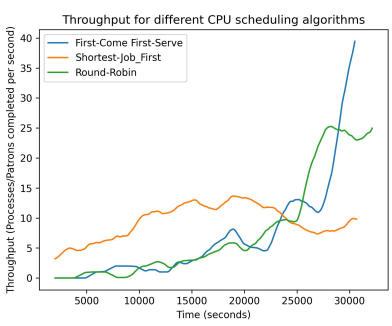
For **First-Come-First-Serve**, the average turnaround time is relatively high, which is expected given its high waiting time. The *convoy effect* is also evident here, when a drink order with a large preparation time enters early or mid-sequence, it increases not just the waiting time but also the turnaround time of all subsequent processes. This can cause a form of starvation, where short processes have uncharacteristically long turnaround times. The distribution appears negatively-skewed (mean $<$ median), indicating that earlier processes tend to finish sooner. The asymmetry of the distribution again suggests unfairness, as turnaround times vary disproportionately across processes. However, FCFS still shows low variance, indicating predictable but generally longer turnaround times.

Shortest-Job First achieves the lowest average turnaround time among all algorithms, as it minimizes waiting time by prioritizing short jobs. Since turnaround time includes burst time, which is shortest for these jobs, the benefit compounds and persists over the entire distribution. The distribution is symmetrical, with the mean roughly being equal to the median, indicating an equitable distribution of turnaround times, suggesting fairness and consistency in turnaround times across jobs. Although SJF shows slightly higher variance

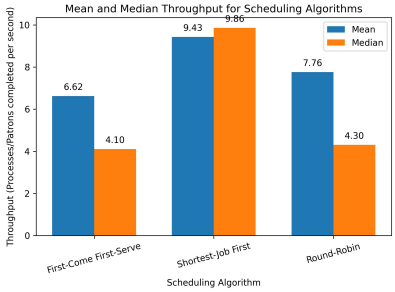
than FCFS, making it slightly less predictable, it remains efficient and equitable, and no outliers suggest starvation is not a concern.

Round-Robin has the highest average turnaround time due to frequent context switching and time-slicing overhead, particularly for processes with short jobs (drink orders will small preparation time) that might finish earlier under SJF. Still, its nearly symmetrical distribution suggests improved fairness compared to FCFS. RR exhibits a moderate variance, striking a balance between FCFS's predictability and SJF's performance. Overall, RR represents a middle ground: it avoids extreme unfairness and starvation, but at the cost of higher turnaround times, particularly for short-duration processes.

4.2.4 Throughput



(a) Average throughput vs time



(b) Mean and median throughput for each scheduling algorithm

Figure 5: Distribution, mean and median of average throughput over time

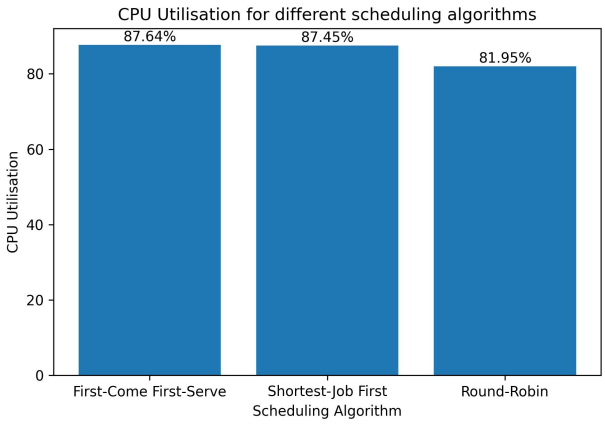
First-Come First-Serve exhibits the lowest mean and median throughput among the three scheduling algorithms. The throughput distribution is positively skewed, indicating that throughput tends to increase towards the end of the simulation period. This behaviour may be attributed to the *convoy effect*, wherein long-duration CPU bursts of a process (analogous to drink orders with large preparation times in the simulation context) occupy the processor (barman) for extended periods of time, causing shorter jobs to accumulate and only be serviced later in the simulation. As a result, the system appears more productive towards the end, but overall throughput remains low due to earlier inefficiencies. This delayed servicing leads to uneven job completion and contributes to the lower overall efficiency of FCFS in high-load environments.

Shortest-Job First demonstrates the highest mean and median throughput among the evaluated scheduling algorithms. The distribution of throughput is approximately symmetrical, with a slight negative skew, likely resulting from the nature of the algorithm: a large number of short jobs (drink orders) are completed rapidly at the beginning of the simulation, leading to high initial throughput, while longer jobs are completed more slowly toward the end,

resulting in a slightly reduced throughput. This pattern reflects the core behavior of SJF (prioritizing shorter jobs increases early system productivity), and although throughput slightly declines later, the overall performance remains consistently high. The relatively narrow spread and symmetry of the distribution also indicate a balanced and steady rate of process completion, suggesting that SJF maintains a stable level of productivity throughout the simulation. In contrast to algorithms such as FCFS, which may suffer from issues such as the **convoy effect**, SJF appears more efficient and better optimized for maximizing throughput under these conditions.

Round-Robin demonstrates a mean and median throughput that lies between those of FCFS and SJF. The throughput distribution is highly positively skewed, indicating that throughput increases significantly towards the latter stage of the simulation. This skewness can be attributed to the frequent context switching inherent to the RR algorithm, which delays the completion of longer jobs during the early stages of execution. Since each job receives only a limited time quantum before being preempted, many jobs remain partially complete for a substantial portion of the simulation. Over time, as more jobs accumulate partial progress (preparation before preemption), they begin completing in quick succession towards the end of the simulation, thus increasing throughput later on. This behavior highlights the trade-off in RR scheduling: while it improves fairness and responsiveness by ensuring that all processes receive CPU time relatively quickly, it also introduces overhead due to preemption and delays the completion of individual jobs, especially in high-load scenarios. Consequently, the system experiences bursts of productivity later in the simulation, rather than a consistent rate of process completion. Despite these inefficiencies, RR provides a more equitable distribution of service time compared to FCFS, albeit with a less balanced and equitable distribution than SJF.

4.2.5 CPU Utilization



(a) CPU Utilization

Figure 6: CPU Utilization of the different scheduling algorithms

CPU utilization appears to be relatively similar for both **First-Come First-Serve** and **Shortest-Job First**, but is noticeably lower for **Round-Robin**. The reduced CPU utilization observed in RR is likely due to the increased number of context switches caused by preemption. These frequent context switches introduce context switching overhead, decreasing the time the CPU spends actually processing (preparing drinks).

From the experiment, FCFS emerged as the most efficient algorithm in terms of CPU utilization, while RR proved to be the least efficient. In this context, higher CPU utilization is desirable, as it indicates more effective use of processing time. Therefore, FCFS and SJF are favored over RR when maximizing CPU efficiency is the primary goal.

5 Conclusion

This investigation explored three different CPU scheduling algorithms: ***First-Come-First-Serve***, ***Shortest-Job First***, and ***Round-Robin***, using a Java-based simulation of a barman preparing drinks for patrons. The experiment evaluated each algorithm based on response time, waiting time, turnaround time, CPU utilization, and throughput. Other scheduling criteria such as predictability, fairness and risk of starvation were also explored.

The time quantum experiment revealed that a quantum of 80 seconds yielded the best balance between response and turnaround times for Round-Robin scheduling, aligning with theoretical heuristics.

In the main experiment, FCFS demonstrated the most predictable response and waiting times, though at the cost of fairness and higher waiting time. SJF minimized average waiting and turnaround times, as expected, but suffered from unpredictability, poor fairness and potential starvation. RR offered a compromise between fairness and efficiency, balancing response and turnaround times more evenly across all patrons while maintaining a lower but still overall good CPU utilization.

Given these findings, Round-Robin with a properly chosen time quantum (80 seconds) emerges as the most balanced scheduling strategy in terms of fairness, predictability, and overall efficiency for the simulated bar environment. However, in systems where predictability is prioritized, FCFS may be preferred, while SJF is best suited for throughput-optimized environments where starvation is less of a concern.

6 References

References

Anderson, Thomas and Michael Dahlin (2014). *Operating Systems: Principles and Practice*. Recursive Books.

Silberschatz, Abraham, James L. Peterson, and Peter B. Galvin (1991). *Operating System Concepts*. 3rd ed. Addison-Wesley Longman Publishing Co., Inc.