Operating Systems Coursework Report

Candidate Number: 279187

Experiment 1:

Introduction:

CPU scheduling is a crucial operation in the operating system, affecting the values of waiting time, response time, throughput, and turnaround time. The efficiencies/inefficiencies of different algorithms vary depending on the nature of the processes and the size of the workload. In this experiment, we aim to investigate how the two primary categories of processes—CPU-bound processes (characterized by long CPU bursts and minimal I/O activity) and I/O-bound processes (characterized by short CPU bursts and frequent I/O operations)—affect the performance and scalability of familiar CPU scheduling algorithms.

Specifically, I will compare the operation of five scheduling algorithms: First-Come, First-Served (FCFS), Round Robin (RR), Ideal Shortest Job First (Ideal SJF), Shortest Job First with Exponential Averaging (SJF), and Multi-Level Feedback Queue (MLFQ). By comparing each scheduler with a range of workload sizes (small, medium, large) for CPU-bound and I/O-Bound processes, respectively, I will be able to see which algorithms scale better and under what process conditions each algorithm performs well or poorly. This will help analyze scheduler behavior, enabling one to make decisions on selecting the best algorithm based on specific workload scenarios.

Methodology

Scope and Objective: Evaluate the impact of different process characteristics (CPU-bound vs. I/O-bound) and workload sizes on the performance and scalability of various CPU scheduling algorithms.

Data Values Experiment 1:

Aspect	Values	Rationale
Schedulers Tested	FCFS, RR, Ideal SJF, SJF, MLFQ (FeedbackRR)	Use of all schedulers to capture diverse scheduling behaviours and trade-offs clearly outlined by theory.
Process Types	CPU-bound (long CPU bursts), I/O-bound (frequent I/O bursts)	Real-world systems typically exhibit mixed workloads; explicitly including both CPU-intensive and I/O-intensive processes allows precise evaluation of scheduler efficiency, fairness, and responsiveness under realistic conditions.
Workload Sizes	Small (10 processes), Medium (50 processes), Large (100 processes)	Analyze how the scheduler performs and determine to what extent it is scalable for varying workload conditions, ranging from minimal to heavy loads.
Input Generation	**Static Priority:** 1 **Inter-Arrival (CPU-bound)**: 8, 7, **2** *(Small, Medium, Large)* **Inter-Arrival (I/O-bound):** 14, 12, 5 **CPU Burst (CPU-bound):** 140, 195, 280 ms **CPU Burst (I/O-bound):** 15, 25, 35 ms **I/O Burst (CPU-bound):** 4, 8, 15 ms **I/O Burst (I/O-bound):** 50, 65, 70 **Number of Bursts:** 2, 5, 8	The parameters model realistic variations in system conditions and intensity of workload. CPU bursts and I/O inter-arrivals change significantly between I/O-bound and CPU-bound cases, allowing various scheduler reactions to the type of workloads to be easily highlighted. Larger examples of longer CPU bursts and shorter inter-arrival times model heavy loads, while variations in I/O bursts model actual I/O operation times typically seen in actual systems.
Seeds	666666, 777777, 888888	Multiple distinct seeds mitigate the influence of random variability in workload generation, ensuring

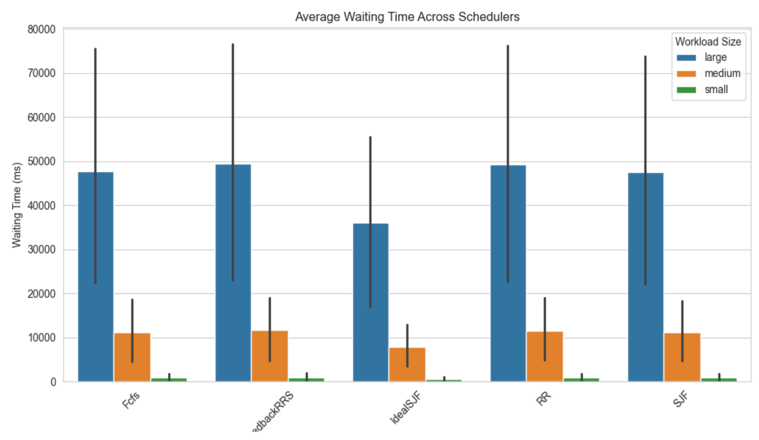
		reproducibility and statistical reliability of the performance results.
Simulator Parameters	**Time Limit:** 100000 ms (ensures workload completion) **Interrupt Time:** 1 **Time Quantum (RR, MLFQ):** 20 ms **Alpha Burst (SJF):** 0.5 **Initial Burst (SJF):** 5	The parameters reflect a realistic operating system scheduler as follows: the Time Quantum (RR, MLFQ) of 20 ms balances responsiveness and context switch overhead, the Alpha Burst (SJF) of 0.5 ms is the standard for predictive smoothing, and the Initial Burst of 5 ms serves as a neutral starting point.
Performance Metrics	Turnaround Time, Waiting Time, Throughput, CPU Utilization	Through these metrics, we provide a comprehensive and detailed evaluation of each scheduler, measuring efficiency through turnaround and waiting time, while response time is used to assess scheduler interactivity and fairness. Additionally, CPU utilization allows us to measure the effectiveness of resource management. Finally, with throughput we asses overall productivity and highlights algorithmic efficiency trade-offs respectively.
Data Collection	***.in*** Generated Input Files (18 total: 6 for each seed (small, medium, large for CPU-bound and I/O-bound)) ***.out*** Generated Output Files (90 total: 30 for each seed (6 input files * 5 schedulers) Aggregated results for all the seeds in a **.csv** file for analysis	This clear and systematic structure enables easy validation, reproduction, and comparison of data, while the aggregated file facilitates straightforward statistical analysis.

Validity:

- **Averaging through multiple distinct seeds** reduces the impact of random variance, ensuring statistical reliability and consistency.
- **Controlled Input Generation** with defined realistic workload parameters that give a systematic basis for consistency in performance comparisons and minimal variability.
- Cross-referenced metrics on final outputs (.out files) against expected results to ensure accuracy.
- **Consistent with theoretical expectations** as the outcomes match the established theoretical expectations (Fcfs convoy effect, SJF's efficiency for shorter processes, and RR/MLFQ fairness-efficiency trade-offs) * See Results

Results:

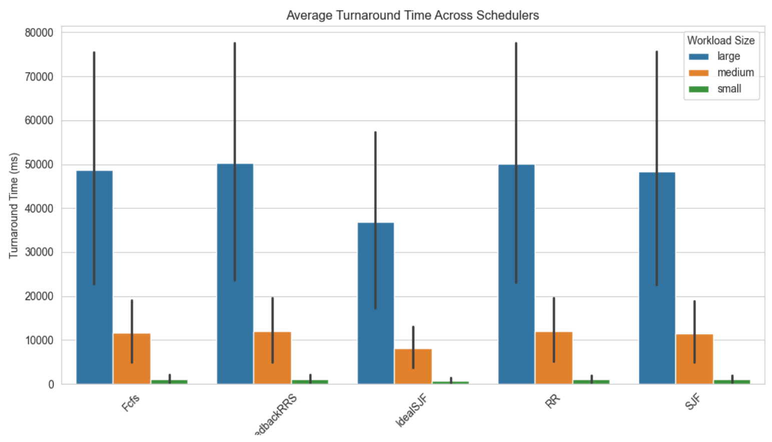
The experiment measures: **waiting time**, **turnaround time**, **response time**, **CPU utilization**, and **throughput** across five different schedulers—**FCFS**, **RR**, **Ideal SJF**, **SJF (Exponential Averaging)**, and **FeedbackRR(MLFQ)**—under varying workload sizes (small, medium, large) and process types (CPU-bound vs. I/O-bound).



1. Average Waiting Time Across Schedulers(ms)

Scheduler	Average Waiting Time (CPU-Bound)	Average Waiting Time (I/O-Bound)
Fcfs	32765.2	3942.0
FeedbackRR	33196.8	4034.0
IdealSJFS	22758.6	2993.7
RR	33190.8	3968.3
SJFS	31964.9	3892.1

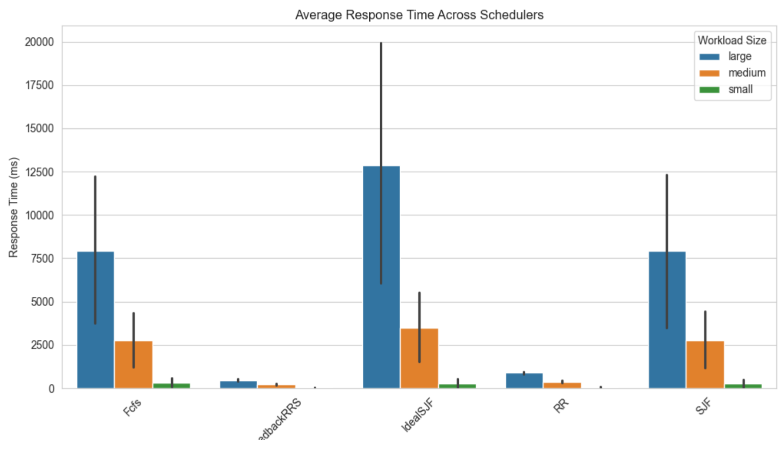
Table 1



2. Average Turnaround Time Across Schedulers (ms)

Scheduler	Average Turnaround Time (CPU-Bound)	Average Turnaround Time (I/O-Bound)
Fcfs	33522.1	4045.2
FeedbackRR	33950.4	4133.2
IdealSJFS	23515.4	3095.0
RR	33944.7	4069.7
SJFS	32721.8	3996.2

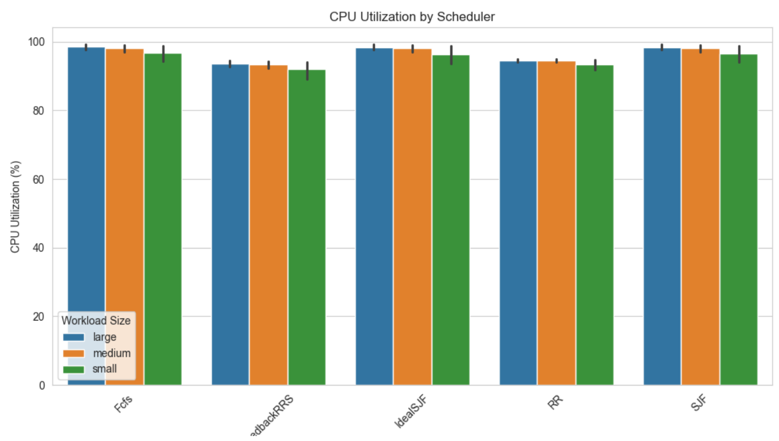
Table 2



3. Average Response Time Across Schedulers (ms)

Scheduler	Average Response Time ms (CPU-Bound)	Average Response Time ms (I/O-Bound)
Fcfs	6923.3	741.7
FeedbackRR	278.5	179.1
IdealSJFS	9827.8	1086.8
RR	511.5	368.3
SJFS	6666.5	737.1

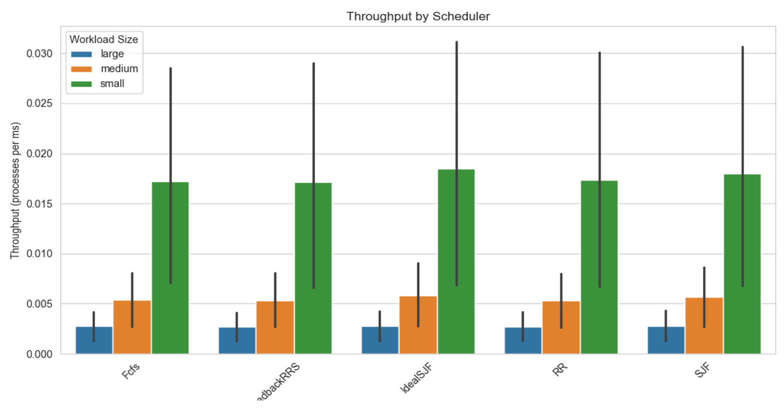
Table 3



4. CPU Utilization by Scheduler (ms)

Scheduler	CPU-Utilization% % (CPU-Bound)	CPU-Utilization % (I/O-Bound)
Fcfs	99.4	95.9
FeedbackRR	94.6	91.0
IdealSJFS	99.4	95.8
RR	94.9	93.4
SJFS	99.4	95.9

Table 4



5. Throughput by Scheduler (Processes per ms)

Scheduler	Throughput (CPU-Bound)	Throughput (I/O-Bound)
Fcfs	.0021	.0126
FeedbackRR	.0020	.0129
IdealSJFS	.0021	.0131
RR	.0020	.0130
SJFS	.0021	.0127

Table 5

Discussion:

The graphs and data tables provide clear illustrations of the distinctions in each scheduler's approach to balancing responsiveness, overall efficiency, and overhead. Firstly, in CPU-bound scenarios, we observe that **Ideal SJF** achieves the lowest waiting time (23,515 ms) and turnaround time (22,758 ms) (Figures 1 and 2, Tables 1 and 2), as it leverages its perfect knowledge of burst lengths. On the other hand, this leads to increased response times (figure 3), in contrast because of frequent pre-emptions **MLFQ** (FeedbackRR) and Round Robin showcase rapid response times (Table 3), however said premptions yield high waiting and turnaround times when we asses CPU-bound loads (Tables 1, 2). Meanwhile, in Fcfs, we observe the convoy effect, as long bursts inflate queue delays, leading to high waiting and turnaround times (Tables 1, 2) under CPU-bound processes. The differentiation between schedulers becomes more apparent through the I/O-bound scenarios. Specifically, because of the short time quantum used (20ms) and their preemptive designs, MLFQ and RR schedulers produce much faster response times (Figure 3, Table 3) because of the frequent bursts by rapidly cycling the tasks. Ideal SJF maintains low waiting and turnaround times (Figures 1,2), with relatively slower response times (Table 3). SJF maintains its consistency across the process types as it balances burst estimates to control waiting and turnaround times, also achieving a good throughput (0.0127 proc/ms) (Table 4). However, it is unable to compare favorably to MLFQ and RR in heavily interactive situations. (Figures 1, 2, 3). Summarizing, the results confirm that preemptive algorithms (RR, MLFQ) perform better in scenarios with short bursts and frequent I/O requests, as they incur some extra overhead in exchange for faster responses. In contrast, SJF-based models (particularly IdealSJFS) excel in scenarios with predictable or heavier CPU bursts, illustrating greater efficiency at the cost of interactivity.

Threats to Validity:

1. **Limited Seeds & Parameters:** Only used 3 seeds, and a narrow range of CPU/I/O burst values were used, which may not conform, as real-world workloads might differ significantly.

2. **Simulator Simplifications:** The context-switch overhead was set with a fixed interrupt time, which may not match the dynamic overhead in live systems.

3. **Fixed Scheduler Parameters:** The time quantum (20) and alpha values were fixed, making my current results for the schedulers that utilize said parameters specific to my chosen setting.

Conclusion:

As showcased by the results and discussion, the experiment clearly demonstrates scheduler performance trade-offs. In CPU-bound workloads, however, Ideal SJF performs best through optimal burst management, which comes at the cost of responsiveness. On the other hand, RR and MLFQ, due to their preemptive nature, have faster response time (better interactivity) for I/O-bound operations. Fcfs underperforms comparatively under heavy CPU-bound loads. The experiment concludes that the choice of scheduler depends strongly on workload characteristics and the desired level of responsiveness.

Experiment 2:

Introduction:

Context switching is defined as the process of switching the CPU from one process, task, or thread to another, a factor that directly affects system performance in CPU scheduling. In this experiment, we aim to investigate how scheduler efficiency varies across 5 different CPU scheduling algorithms (RR, SJF, IdealSJFS, MLFQ, Fcfs)[1] It is impacted by context switch overhead. In further detail, we will use waiting time, turnaround time, response time, and CPU utilization as performance metrics to investigate the effect of context switching frequency on the performance of the scheduling algorithms.

Methodology:

Scope and Objective: Evaluate how context switch overhead influences the performance of the different CPU scheduling algorithms through a balanced workload (50% CPU-bound processes + 50% I/O-bound processes), analyzing 5 scenarios (Balanced/Baseline, CPU-Intensive, I/O Intensive, Lower Load, Higher Load), aiming to determine the distinct scheduling strategies' balance between responsiveness and CPU overhead.

Data Values Experiment 2:

Aspect	Values	Rationale
Schedulers Tested	FCFS, RR, Ideal SJF, SJF, MLFQ (FeedbackRR)	Use of all schedulers to capture diverse scheduling behaviours and trade-offs clearly outlined by theory.
Process Types	CPU-bound (long CPU bursts, minimal I/O), I/O-bound (frequent I/O bursts, short CPU bursts)	Using a balanced workload of processes, which explicitly includes both CPU-intensive and I/O-intensive processes, allows for a precise evaluation of scheduler efficiency, fairness, and responsiveness under realistic conditions.
Workload Sizes	50 processes	
Input Generation	**Baseline CPU-Intensive I/O-Intensive Lower Load Higher Load** **Static Priority:** 0 (same across scenarios) **Inter-Arrival**: 5, 4, 6, 2, 8 **CPU Burst:** 15, 20, 10, 18, 25 **I/O Burst:** 10, 8, 15, 12, 10 **Number of Bursts:** -, 12, 8, 10, 12	Varying inter-arrival times, CPU Bursts, and I/O bursts establishes the 5 scenarios as the parameters highlight the distinct conditions (CPU-Intensive has increased CPU bursts and reduced I/O bursts to stress the CPU-bound aspect, I/O-intensive shortness CPU-bursts and prolongs I/O-burst favouring frequent I/O operations, Lower Load lengthens inter-arrival times, reduces burst counts lowering overall system pressure, In contrast Higher Load has shorter inter-arrivals, more bursts—to intensify system demand) and isolate how each scheduler handles different levels of intensity and mix of CPU vs I/O demands.

[1] See experiment 1/ Introduction

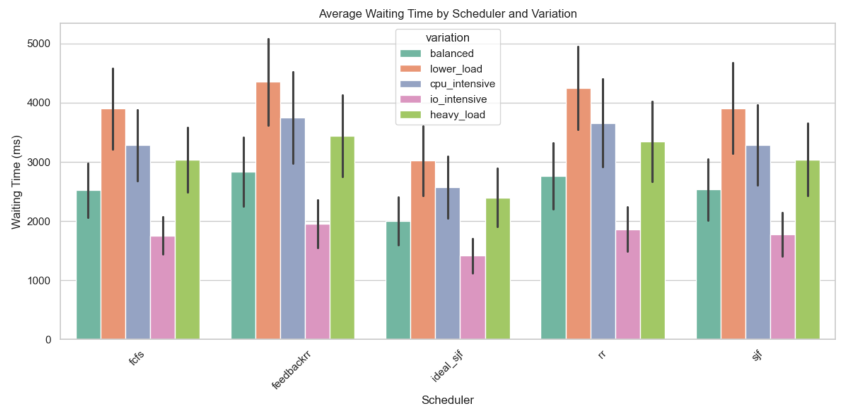
Seeds	111111, 222222, 333333, 444444, 555555	Multiple distinct seeds mitigate the influence of random variability in workload generation, ensuring reproducibility and statistical reliability of the performance results.
Simulator Parameters	**Time Limit:** 10000 ms **Interrupt Time:** 1 (Fixed) **Time Quantum (RR, MLFQ):** 5 ms **Alpha Burst (SJF):** 0.5 **Initial Burst (SJF):** 10	Short Time Quantum (5ms) highlights the impact of frequent context switches on overhead vs. responsiveness, while Alpha and Initial Burst values provide moderate predictiveness in SJF.
Performance Metrics	Turnaround Time, Waiting Time, CPU Utilization	Through these metrics, we provide a comprehensive and detailed evaluation of each scheduler, measuring efficiency through turnaround and waiting time, while response time is used to assess scheduler interactivity and fairness. Additionally, CPU utilization allows us to measure the effectiveness of resource management.
Data Collection	***.in*** Generated Input Files (25 total: 5 seeds * 5 workload scenarios (Balanced, CPU-Intensive, Heavy Load, Lower Load, I/O Intensive)) ***.out*** Generated Output Files (125 total: 25 for each seed (25 input files * 5 schedulers) Aggregated results for all the seeds in a **.csv** file for analysis	This clear and systematic structure enables easy validation, reproduction, and comparison of data, while the aggregated file facilitates straightforward statistical analysis.

Validity:

- **Averaging through 5 distinct seeds** reduces the impact of random variance, ensuring statistical reliability and consistency.
- **Structured Workload Scenarios:** The experiment examines five distinct workload scenarios (Baseline, CPU-Intensive, I/O-Intensive, Lower Load, and Higher Load), isolating specific contexts and providing insight into how each scheduler manages varying intensities of CPU and I/O demand.
- **Consistency with Theory: Observed trends (MLFQ/RR's rapid response, higher overhead, and FCFS's vulnerability under CPU-bound loads) that align with what we consider expected scheduler behaviours**.

Results:

The experiment measures: **waiting time**, **turnaround time**, **response time**, and CPU **utilization**, across five different schedulers—**FCFS**, **RR**, **Ideal SJF**, **SJF (Exponential Averaging)**, and **FeedbackRR(MLFQ)**. The experiment carries out under a set number of processes (50) across different workloads/process types (Balanced, CPU-Intensive, I/O-Intensive, Lower Load, Heavy Load).

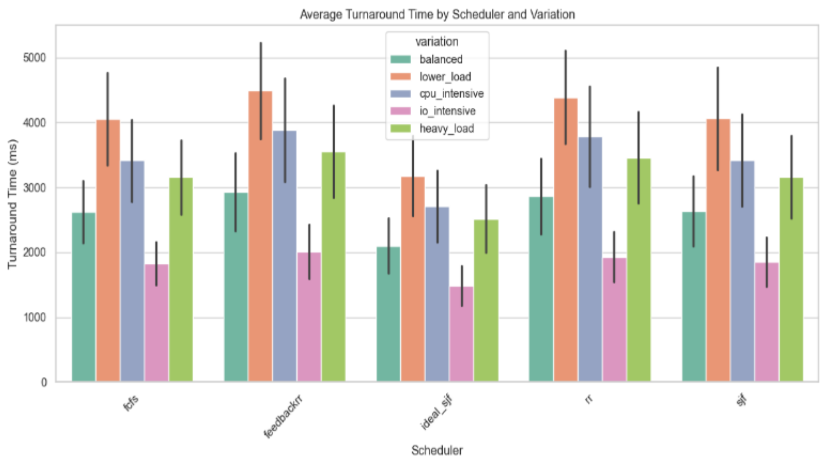


🟩	I/O Intensive
🟪	Lower Load
🟦	CPU Intensive
🟩	Balanced
🟧	Heavy Load

6. Average Waiting Time by Scheduler and Variation (ms)

Scheduler / Variation	Fcfs	FeedbackRR	IdealSJFS	RR	SJFS
Balanced	2519.6	2834.5	1997.5	2765.5	2535.3
CPU - Intensive	3281.2	3754.7	2571.3	3659.6	3284.5
I/O – Intensive	3037.0	3439.3	2399.1	3343.3	3041.7
Lower Load	1756.2	1949.1	1413.6	1862.1	1776.2
Heavy Load	3901.1	4353.2	3019.3	4253.1	3908.9

Table 6

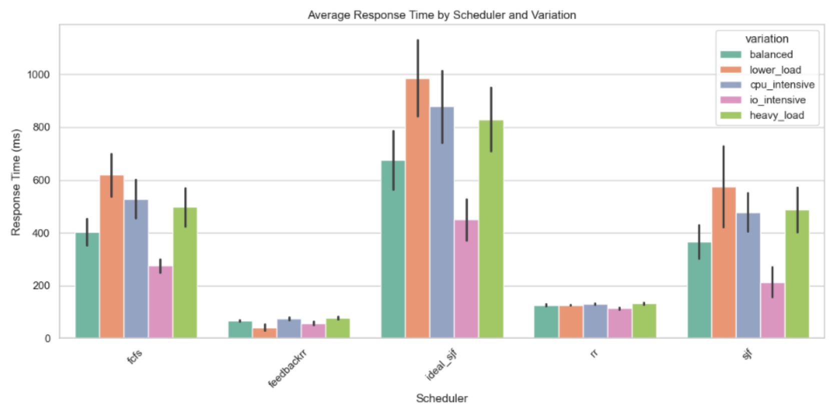


🟩	I/O Intensive
🟪	Lower Load
🟦	CPU Intensive
🟩	Balanced
🟧	Heavy Load

7. Average Turnaround Time by Scheduler and Variation (ms)

Scheduler / Variation	Fcfs	FeedbackRR	IdealSJFS	RR	SJFS
Balanced	2619.4	2928.5	2097.5	2860.6	2635.3
CPU - Intensive	3411.4	3878.9	2701.5	3784.7	3414.7
I/O – Intensive	3155.0	3551.4	2517.1	3456.4	3159.7
Lower Load	1825.9	2012.8	1483.1	1927.4	1845.7
Heavy Load	4055.4	4488.1	3177.3	4388.6	4064.0

Table 7

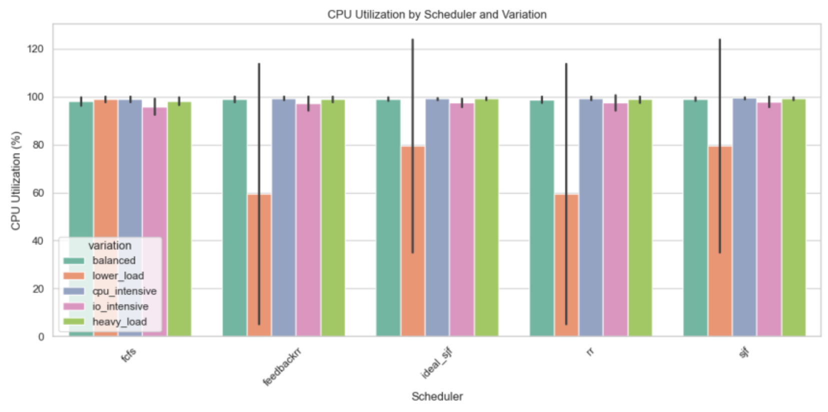


🟩	I/O Intensive
🟪	Lower Load
🟦	CPU Intensive
🟩	Balanced
🟧	Heavy Load

8. Average Response Time by Scheduler and Variation (ms)

Scheduler / Variation	Fcfs	FeedbackRR	IdealSJFS	RR	SJFS
Balanced	403.9	66.45	675.79	125.32	366.83
CPU - Intensive	528.3	74.81	878.69	130.4	477.39
I/O – Intensive	497.36	78.29	830.01	132.83	487.74
Lower Load	275.03	57.25	449.82	113.48	213.62
Heavy Load	618.77	41.51	986.3	126.17	574.97

Table 8



🟩	I/O Intensive
🟪	Lower Load
🟦	CPU Intensive
🟩	Balanced
🟧	Heavy Load

9. CPU Utilization by Scheduler and Variation (%)

Scheduler / Variation	Fcfs	FeedbackRR	IdealSJFS	RR	SJFS
Balanced	98.28	99.06	99.21	98.87	99.19

CPU - Intensive	99.06	99.44	99.34	99.49	99.57
I/O – Intensive	98.29	99.07	99.34	98.96	99.24
Lower Load	96.01	97.29	97.72	97.63	97.93
Heavy Load	99.02	59.56	79.54	59.5	79.61

Table 9

Discussion:

This experiment investigates how context switch overhead affects the performance of each scheduler under five different workload scenarios: Balanced, CPU-Intensive, I/O-Intensive, Lower Load, and Heavy Load, each comprising 50% CPU-bound and 50% I/O-bound applications. The ideal SJF has the lowest average waiting and turnaround times (e.g., 1,413 ms and 2,097.5 ms in Lower Load) by minimizing unnecessary switches (Figures 6 and 7). However, it falls behind in responsiveness, reaching up to 986 ms in the Heavy Load scenario, as shown in Table 8. On the other hand, MLFQ (FeedbackRR) and Round Robin (RR) exhibit rapid response times, reaching up to 41.51 ms, due to high-density preemptions that benefit recently arrived jobs (Figure 8). Their context-switching overhead is evident in Table 9, where CPU utilization drops to extremely low points for MLFQ and RR under Heavy Load (59%), i.e., a significant percentage of CPU time is spent switching between processes rather than executing them. As FCFS is non-preemptive, it consistently achieves full CPU utilization but performs poorly in CPU-bound scenarios, with average waiting and turnaround times that exceed 3,900 ms when burst activity accumulates (Figures 6 and 7). Furthermore, SJF (Exponential Averaging) achieves a balanced performance, but still never surpassing Ideal SJF in efficiency nor MLFQ/RR in responsiveness, while neither incurring excessive overhead nor experiencing prolonged waiting times. In general, these findings define the essential trade-offs between responsiveness and CPU utilization: under light loads, higher context switching rates (MLFQ/RR) yield rapid response at low cost in throughput; under heavy loads, high preemption rates severely impair efficiency. Schedulers thus need to be chosen based on comparable workload intensity and the desired trade-off between interactivity and total resource utilization.

Threats to Validity:

1. **Lack of Variation in Priority** (Construct Validity)

2. **Aggregation and Averaging of Metrics** sometimes conceals meaningful outliers and extreme scenarios that could risk exposing hidden scheduler behavior.

3. **Fixed Time Quantum:** The use of a fixed short time quantum (5ms) may skew the results in favor of non-preemptive algorithms by overemphasizing context-switch overhead for specific workloads.

Conclusion:

Experiment 2 demonstrates that even a slight context switch overhead has a significant impact on scheduling decisions, as evident from the apparent fluctuations in performance metrics across schedulers reflecting different workloads (also compared to the baseline). Preemptive schedulers, such as MLFQ and RR, operate most effectively in reducing response time, particularly for I/O-bound processes; however, they may experience degraded CPU utilization with increasing loads due to excessive context switching. Fewer non-preemptive or preemptive schedulers (such as SJF-based and FCFS) have better overall efficiency at the expense of interactivity. Finally, these findings confirm that workload intensity and tolerance to overhead in the system should be used as determinants of the scheduler choice, with an emphasis on the trade-off between rapid response and effective utilization.

Experiment 3:

Introduction:

This experiment examines the impact of the time quantum—the duration of each CPU time slice—on the performance of preemptive scheduling algorithms. In previous experiments, we saw how scheduler behaviours change under varying workload sizes (Experiment 1) and context-switch overheads (Experiment 2). In this study, we build on those findings by systematically varying the time quantum for Round Robin (RR) and Multi-Level Feedback Queue (MLFQ), while continuing to compare them against non-preemptive baselines (FCFS, SJF, and Ideal SJF). Since the time quantum is small, processes receive frequent CPU preemptions, which improves responsiveness but increases context switching overhead to a high level. Contrarily, having a considerable time quantum reduces overhead but may increase waiting time for interactive or short-running processes. Suppose we measure waiting time, turnaround time, and response time with different quantum values. In that case, we can conclude that, based on these measurements, the scheduler's settings are the fairest and efficient in preemptive scheduling.

Methodology:

Scope and Objective: This experiment utilizes four time quantum values (5, 25, 50, 100 ms) applied to the five schedulers (FCFS, RR, SJF, Ideal SJF, MLFQ), noting that only RR and MLFQ depend on the quantum value. The rest of the simulator parameters are identical across schedulers (Identical to experiment 2). The aim is to compare how different size quanta affect RR and MLFQ, assess time quantum's impact on waiting time, turnaround time, and response time, benchmark results against non-preemptive schedulers, and identify any near-optimal quantum that balances responsiveness with minimal overhead.

Data Values Experiment 3:

Aspect	Values	Rationale
Schedulers Tested	FCFS, RR, Ideal SJF, SJF, MLFQ (FeedbackRR)	Use of all schedulers to capture diverse scheduling behaviours and trade-offs clearly outlined by theory.
Process Types	Balanced	CPU-bound and I/O-bound processes are balanced proportionally, allowing for fair assessment of how each scheduler manages different processing demands.
Workload Sizes	50 processes	Moderate size to highlight performance differences.
Input Generation	**Static Priority:** 0 **Inter-Arrival**:5.0 **CPU Burst:** 20.0 **I/O Burst:** 10.0 **Number of Bursts: 9.0**	Realistic arrival time with balanced PCU and I/O bursts focusing on the impact of different schedulers and time quanta.
Seeds	111111, 222222, 333333, 444444, 555555	Multiple distinct seeds mitigate the influence of random variability in workload generation, ensuring reproducibility and statistical reliability of the performance results.
Simulator Parameters	**Time Limit:** 10000 ms **Interrupt Time:** 1 (Fixed) **Time Quantum (RR, MLFQ):** 5 , 25, 50, 100 ms **Alpha Burst (SJF):** 0.5 **Initial Burst (SJF):** 10	Short Time Quantum (5ms) highlights the impact of frequent context switches on overhead vs. responsiveness, while Alpha and Initial Burst values provide moderate predictiveness in SJF.
Performance Metrics	Turnaround Time, Waiting Time, Response Time	Through these metrics, we provide a comprehensive and detailed evaluation of each scheduler, measuring efficiency through turnaround and waiting time, while response time is used to assess scheduler interactivity and fairness.
Data Collection	***.in*** Generated Input Files 5 (5 seeds) ***.out*** Generated Output Files (65 total: 25 for each seed for RR and MLFQ and 5 for SJF, IdealSJF, and Fcfs) Aggregated results for all the seeds in a **.csv** file for analysis	This clear and systematic structure enables easy validation, reproduction, and comparison of data, while the aggregated file facilitates straightforward statistical analysis.

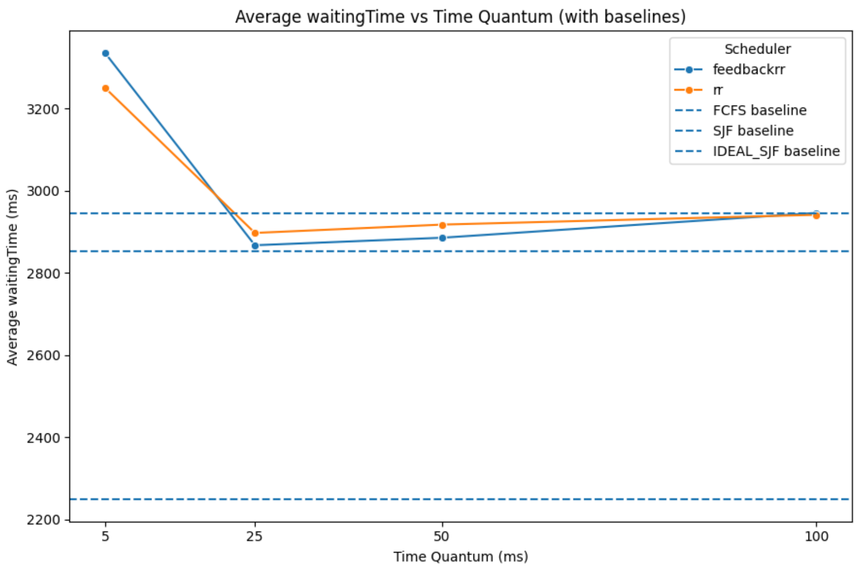
Validity:

- **Fixed Parameters:** Time quantum was varied (5, 25, 50, 100 ms), other parameters (e.g., alpha for SJF = 0.5) stayed constant, thus results are specific to the chosen alpha and workload settings but valid within that controlled scope.
- **Finding consistent with scheduling theory/expectations**

Results:

This experiment investigates how varying the time quantum (5, 25, 50, 100 ms) affects the performance of preemptive schedulers—RR and FeedbackRR (MLFQ)—by measuring waiting time, turnaround time, and response time. Non-preemptive schedulers (FCFS, SJF, Ideal SJF) are included as baselines. All simulations were run on a balanced workload of 50 processes

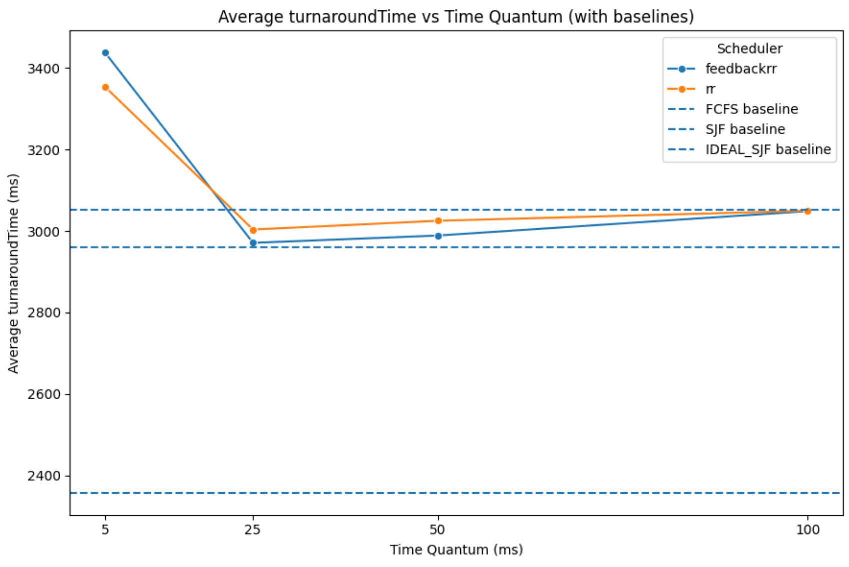
(50% CPU-bound, 50% I/O-bound) to isolate the impact of time quantum on responsiveness and scheduling overhead.



10. Average Waiting Time vs Time Quantum

Scheduler	Time Quantum	Waiting Time
FCFS	N/A	2943.9
SJF	N/A	2852.5
Ideal SJF	N/A	2250.1
RR	5	3249.4
RR	25	2896.9
RR	50	2917.2
RR	100	2941.1
FeedbackRR	5	3334.1
FeedbackRR	25	2867.1
FeedbackRR	50	2885.3
FeedbackRR	100	2945.7

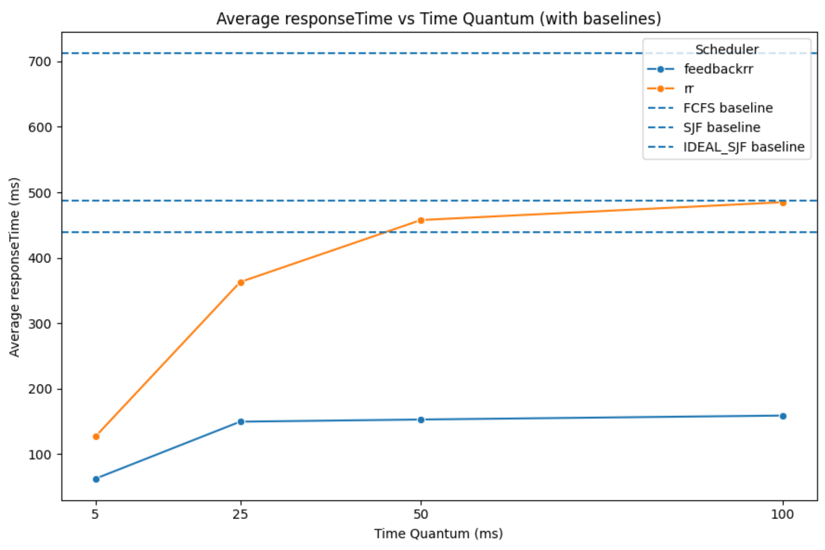
Table 10



11. Average Turnaround Time vs Time Quantum

Scheduler	Time Quantum	Turnaround Time
FCFS	N/A	3052.1
SJF	N/A	2960.1
Ideal SJF	N/A	2357.73
RR	5	3353.2
RR	25	3003.4
RR	50	3024.9
RR	100	3049.4
FeedbackRR	5	3437.3
FeedbackRR	25	2970.8
FeedbackRR	50	2988.6
FeedbackRR	100	3048.4

Table 11



12. Average Response Time vs Time Quantum

Scheduler	Time Quantum	Response Time
FCFS	N/A	486.9
SJF	N/A	438.4
Ideal SJF	N/A	712.1
RR	5	127.0
RR	25	362.9
RR	50	457.6
RR	100	484.8
FeedbackRR	5	62.6
FeedbackRR	25	149.7
FeedbackRR	50	153.0
FeedbackRR	100	159.0

Table 12

Discussion:

This experiment explores the impact of varying the time quantum (5, 25, 50, 100 ms) for preemptive schedulers (RR, MLFQ) compared to non-preemptive baselines (FCFS, SJF, Ideal SJF). The results show that small quanta (e.g., 5 ms) yield excellent response times for RR (127.0 ms) and especially MLFQ (62.6 ms) (Figure 12), as frequent preemptions enable newly arriving jobs to reach the CPU quickly. However, the same high switch frequency results in longer average waiting and turnaround times (Table 10) (e.g., a 3,334.1 ms MLFQ waiting time compared to 2,867.1 ms with 25 Time Quanta). Overhead is reduced to the highest quantum (100), but responsiveness suffers (e.g., RR response time increases to 484.8 ms) (Table 12). MLFQ consistently has the lowest response of any quantum tried, indicating that multi-level priority queues, when augmented with time-slicing, can aggressively favor interactive jobs. Meanwhile, Ideal SJF has the best waiting and turnaround times overall (Figures 10, 11)—due to its knowledge of burst lengths—but a worse response time (712.1 ms) (Figure 12), consistent with its non-preemptive nature. Standard SJF and FCFS are both insensitive to quantum changes, offering efficiency-focused baselines. SJF again trades off middling waiting and turnaround times for a modest response, while FCFS has lower overhead but poorer queuing performance if CPU bursts overlap. Finally, these findings validate that an intermediate quantum (i.e., 25 or 50 ms) is generally a stable trade-off: it prevents the high overhead of a very low quantum while still allowing for greater responsiveness than the largest quantum. Therefore, every system must determine whether low response times or reduced overhead and queuing are most important, as adjusting the quantum can significantly alter this trade-off.

Threats to Validity:

- **Fixed Parameters**: Other scheduler parameters (e.g., alpha for SJF = 0.5, static priority = 0) so the observed results might not generalize if those parameters change significantly.
- **Limited Quantum Values:** Four quanta (5, 25, 50, 100 ms) were tested; real systems might reveal different sweet spots or require finer-grained increments.

Conclusion:

Alteration of the time quantum demonstrates a clear compromise between responsiveness and overall efficiency in preemptive scheduling. Smaller quanta favor interactive responsiveness at the risk of inflating waiting/turnaround via too much context switching, whereas larger quanta reduce overhead at the expense of responding late for brief tasks. MLFQ and RR illustrate these trends, with Ideal SJF, SJF, and FCFS being quantum-insensitive—offering baselines that show how predictability of bursts or absence of preemption can optimize throughput at the expense of response time. An intermediate quantum is thus usually optimal for trading off user-oriented responsiveness against system-wide efficiency, and time-slice settings must be tuned to workload demands.