Assignment 3 - SYSC 4001 PART 1 (Simulation Report)

Aryan Kumar Singh (101299776)
Srivathsan Murali (101287534)

1. Introduction

This report analyzes the behavior of three CPU scheduling algorithms implemented in a custom simulator:

1. External Priority (EP)
2. Round Robin (RR)
3. External Priority + Round Robin (EP+RR)

A set of 20 scenarios with varying process characteristics were used to evaluate and compare algorithm performance. The goal of the simulation is to observe how scheduling strategies impact wait time, turnaround time, response time, and throughput under different workloads.

2. Simulation Method

Twenty input scenarios were created, each containing processes with different combinations of arrival times, CPU burst lengths, memory requirements, and I/O request patterns. Each scenario was executed using all three scheduling algorithms in the simulator. Following each run, execution traces were collected and analyzed to compute overall metrics. These metrics were then averaged across all scenarios to compare scheduler behavior under diverse workloads.

3. Summary Tables

Scenarios 1–5 (very small workloads - I/O and CPU)

Scenario	Algorithm	Average Response Time	Average Wait Time	Average Turnaround Time	Throughput
1	EP	0	0	30.0	0.033
1	RR	0	0	30.0	0.033
1	EPRR	0	0	30.0	0.033
2	EP	0	0	45.0	0.022
2	RR	0	0	45.0	0.022
2	EPRR	0	0	50.0	0.020
3	EP	20	20	52.5	0.031
3	RR	20	20	52.5	0.031
3	EPRR	20	20	52.5	0.031
4	EP	12.5	12.5	40.0	0.036
4	RR	12.5	12.5	40.0	0.036
4	EPRR	0	10	37.5	0.036
5	EP	4	12	77.5	0.021
5	RR	4	12	77.5	0.021
5	EPRR	4	8	79.0	0.020

Analysis of Group 1 (1-2):

For simple single-process workloads (Scenarios 1–2), all schedulers provide the same response and waiting time, since the single process always starts immediately and never waits in the ready queue. Differences in turnaround and throughput only appear when I/O is introduced. In

Scenario 2, the combined EP+RR scheduler slightly increases turnaround time compared to EP and RR, because of its additional scheduling overhead, but the effect is small.

Analysis of Group 2 (3-5):

In two-process workloads (Scenarios 3–5), EP, RR, and EP+RR are often very similar when both processes arrive at the same time and are purely CPU-bound (Scenario 3). When arrivals are staggered (Scenario 4), the combined EP+RR policy improves response time for some processes by allowing a higher-priority process to preempt earlier, producing a lower average response and waiting time than pure EP or RR. For I/O-bound workloads (Scenario 5), EP+RR reduces average waiting time compared to EP/RR, at the cost of a small increase in turnaround time due to extra context switches.

Scenarios 6-10 (Moderate workloads - I/O and CPU)

Scenario	Algorithm	Average Response Time	Average Wait Time	Average Turnaround Time	Throughput
6	EP	25.0	25.0	79.5	0.018
6	RR	25.0	25.0	79.5	0.018
6	EPRR	25.0	25.0	82.5	0.017
7	EP	2.5	25.5	80.0	0.019
7	RR	2.5	25.5	80.0	0.019
7	EPRR	2.5	25.0	82.5	0.018
8	EP	5.0	19.7	51.3	0.040
8	RR	7.0	18.7	50.3	0.039
8	EPRR	0.33	24.0	58.3	0.040
9	EP	22.0	30.0	58.0	0.050
9	EPRR	22.0	26.0	58.0	0.050
9	RR	6.0	46.0	74.0	0.050
10	EP	80.0	80.0	125.0	0.022

10	RR	80.0	80.0	125.0	0.022
10	EPRR	80.0	80.0	125.0	0.022

Analysis of Group 3 (6-9):

In this group the schedular behaviour is much different from each other requiring an analysis on each scenario.

Scenarios 6–7:

- All schedulers have the same response time.
- EP and RR have nearly identical waiting and turnaround.
- EP+RR sometimes slightly increases turnaround but reduces waiting for specific processes.

Scenario 8:

- EP: response 5, waiting ~19.7, turnaround ~51.3.
- RR: response 7 (slightly worse), waiting ~18.7, turnaround ~50.3.
- EP+RR: much smaller average response (0.33) but larger waiting and turnaround (24 and 58.3).

Interpretation: EP+RR quickly gives each process CPU time (good response), but frequent preemptions + I/O juggling delay completion (higher turnaround).

Scenario 9:

- EP & EPRR: response 22, waiting 30 vs 26, turnaround 58.
- RR: much better response (6) but very large waiting (46) and huge turnaround (74).
- RR here is “too fair” and keeps bouncing between processes, increasing delay until completion.

Scenario 10:

- All schedulers are identical — heavy job dominated; everybody waits long.
- Resp = 80, Wait = 80, Turnaround = 125, Throughput = low.

Thus for mixed workloads with both CPU- and I/O-bound processes (Scenarios 6–9), Round Robin tends to improve response time because processes get CPU time earlier, but it may significantly increase waiting and turnaround time when there are many competing jobs, especially if time slices overlap with frequent I/O (Scenario 9). The combined EP+RR strategy often achieves a middle-ground: it inherits good response behavior from RR while still

respecting priorities from EP. However, due to additional context switching, its turnaround time can be slightly worse than pure EP in some cases.

Scenarios 11–15 (Heavy Workload)

Scenario	Algorithm	Average Response Time	Average Wait Time	Average Turnaround Time	Throughput
11	EP	47.5	47.5	76.3	0.035
11	RR	47.5	47.5	76.3	0.035
11	EPRR	47.5	47.5	76.3	0.035
12	EP	48.4	66.0	110.0	0.031
12	EPRR	48.4	63.6	111.0	0.031
12	RR	23.6	59.2	103.2	0.029
13	EP	25.0	33.5	64.5	0.039
13	EPRR	25.0	31.8	64.8	0.038
13	RR	23.8	34.0	65.0	0.040
14	EP	31.7	31.7	66.7	0.029
14	RR	33.3	33.3	68.3	0.029
14	EPRR	0.0	31.7	66.7	0.029
15	EP	34.5	52.5	94.0	0.029
15	EPRR	34.5	50.5	94.3	0.029
15	RR	18.8	46.8	88.3	0.029

Analysis of Group 4 (11-15):

- Scenario 11: again, almost identical for all three.
- Scenario 12:

    - EP & EPRR share the same response (48.4) but different waiting and turnaround.
    - RR has much better response (23.6) and better turnaround (103.2 vs 110–111), but slightly higher waiting than EPRR.

Scenario 13–15:

- Modest differences:
    - RR tends to give slightly better turnaround in 12, 13, 15.
    - EP+RR often reduces waiting compared to EP but doesn't always win on turnaround.

In heavier multiprocess workloads (Scenarios 11–15), pure EP sometimes suffers from long response times for lower-priority processes, since a long high-priority job can block the others. Round Robin improves response time and can even reduce turnaround time when jobs are roughly similar (Scenario 12), at the cost of more context switches.

EP+RR generally smooths out extreme cases by combining priority awareness with preemption, which reduces waiting time for some lower-priority processes without sacrificing throughput.

Scenarios 16–20 (Heavy / Memory- stress - Bonus related : Scenarios 19 and 20)

Scenario	Algorithm	Average Response Time	Average Wait Time	Average Turnaround Time	Throughput
16	EP	93.0	109.2	169.4	0.019
16	EPRR	93.0	107.4	169.4	0.019
16	RR	74.6	104.4	164.6	0.019
17	(all)	60.0	60.0	90.0	0.033
18	EP	44.0	60.0	116.0	0.025
18	EPRR	44.0	56.0	116.0	0.025
18	RR	12.0	92.0	148.0	0.025
19	(all)	113.4	113.4	147.8	0.029
20	EP	64.7	80.3	116.4	0.034
20	EPRR	64.7	79.0	117.6	0.033

20	RR	40.0	96.9	133.0	0.034

Analysis of Group 5 (16-20) (Memory Stress / Bonus Oriented Scenarios):

- Scenario 16:
  - EP & EPRR: same response (93), high waiting and turnaround.
  - RR: better response (74.6) and better turnaround (164.6 vs 169.4).
- Memory logs (you saw them) show multiple processes competing for limited partitions; once memory is tight, queueing in READY is unavoidable.

- Scenario 17:
  - All three schedulers have the same metrics (resp = 60, wait = 60, turnaround = 90).
  - Suggests a very symmetric / balanced workload where policy does not matter.

- Scenario 18:
  - EP/EPRR: resp 44, wait ~56–60, turnaround 116.
  - RR: resp 12 (!) but waiting jumps to 92 and turnaround to 148.
  - Classic “RR gives good initial responsiveness but pays in completion time” example.

- Scenario 19 (fragmentation / cannot allocate):
  - All three schedulers have identical metrics (Resp ≈ 113.4, Wait ≈ 113.4, Turnaround ≈ 147.8).
  - In your memory logs, you can point out how some processes never get space until very late, even though total free memory exists but is unusable due to fragmentation.

- Scenario 20 (7-process mixed):
  - EP: resp ~64.7, wait ~80.3, TAT ~116.4.
  - EPRR: same response, slightly lower waiting (79), slightly higher turnaround (117.6).
  - RR: much better response (40) but much larger waiting (96.9) and highest turnaround (133).

The last group of scenarios (16–20) were designed to stress both the CPU scheduler and the memory management system. Under high memory pressure and fragmentation (Scenarios 19–20), all three schedulers can suffer from long response and waiting times because processes are delayed before they are even admitted to memory.

In Scenario 19 in particular, the metrics are nearly identical across EP, RR, and EP+RR, which confirms that the main bottleneck is memory allocation rather than CPU policy. Round Robin still tends to improve initial response time (e.g., Scenario 18 and 20), but this comes with significantly higher waiting and turnaround times due to frequent preemptions combined with delayed memory allocation. EP+RR generally tracks EP's throughput while slightly reducing waiting time by allowing preemption within the same priority band.

BONUS (Memory Management Analysis):

In our simulation codes, a summary of the status of all 6 partitions, total memory used by currently running processes, free memory, and usable memory that accumulates during external fragmentations that may occur if a process could be allocated into memory but would be spread into multiple partitions. A new entry gets added to the end of execution.txt every time a process transitions between states.

Memory Behavior in Scenario 19 – External Fragmentation Case

Observed Behavior

Early in the simulation, memory becomes highly fragmented across the 40MB, 25MB, 15MB, 10MB, 8MB, and 2MB partitions.

Several processes repeatedly enter the READY queue but **cannot be allocated memory**, even though the total free memory is high.

As a result, processes wait extremely long before ever starting their first CPU burst.

Because the CPU is idle until memory becomes available, the actual CPU scheduling policy (EP, RR, or EP+RR) has **almost no effect** on performance metrics.

Memory Behavior in Scenario 20 – Heavy 7-Process Mixed Load

Observed Behavior

- Memory partitions fill very quickly.
- New arrivals frequently must wait for large enough contiguous regions to free.

- Frequent context switches (especially under RR and EPRR) cause processes to leave and re-enter memory more often, increasing fragmentation pressure.

- EP tends to keep high-priority jobs in memory longer, reducing fragmentation but delaying lower-priority jobs.

- EPRR provides more balanced admission but with modest extra fragmentation.

- RR triggers the most memory movement because processes are swapped out more often due to time-slice expiration