Student 1: Manoj Kuppuswamy Thyagarajan, **ID:** 101166589
Student 2: Shabesa Kohilavani Arunprakash. **ID:** 101258619

Assignment 3 Report - Concurrency, Shared Memory, Virtual Memory, Files

Part 1

EP Scheduler

File	**Throughput**	**Avg Turnaround (ms)**	**Avg I/O Spacing (ms)**
ep_1	1	10.00	0.00
ep_2	1	11.00	0.00
ep_3	2	11.00	0.00
ep_4	2	9.50	8.00
ep_5	2	19.00	3.60
ep_6	2	27.50	5.29
ep_7	2	7.00	3.50
ep_8	1	12.00	0.00
ep_9	1	8.00	0.00
ep_10	2	16.50	2.67
ep_11	2	13.00	5.00
ep_12	2	10.00	0.00
ep_13	2	11.50	0.00
ep_14	2	15.50	10.00
ep_15	1	25.00	6.00
ep_16	2	10.50	0.00
ep_17	1	7.00	0.00
ep_18	2	13.00	0.00
ep_19	2	13.50	0.00

ep_20	2	14.50	3.17

RR Scheduler

File	**Throughput**	**Avg Turnaround (ms)**	**Avg I/O Spacing (ms)**
rr_1	1	9.00	0.00
rr_2	1	5.00	0.00
rr_3	2	13.50	4.00
rr_4	2	8.00	0.00
rr_5	1	10.00	0.00
rr_6	2	4.50	0.00
rr_7	2	17.50	5.00
rr_8	1	12.00	5.00
rr_9	1	7.00	0.00
rr_10	1	17.00	4.00
rr_11	1	9.00	0.00
rr_12	1	8.00	0.00
rr_13	1	4.00	0.00
rr_14	1	5.00	0.00
rr_15	1	5.00	0.00
rr_16	2	16.00	4.67
rr_17	1	26.00	6.00
rr_18	1	10.00	0.00
rr_19	1	4.00	0.00
rr_20	1	11.00	0.00

EPRR Scheduler

Student 1: Manoj Kuppuswamy Thyagarajan, **ID:** 101166589
Student 2: Shabesa Kohilavani Arunprakash. **ID:** 101258619

File	**Throughput**	**Avg Turnaround (ms)**	**Avg I/O Spacing (ms)**
eprr_1	2	13.00	4.00
eprr_2	2	11.00	3.00
eprr_3	1	25.00	6.00
eprr_4	2	10.00	0.00
eprr_5	1	3.00	0.00
eprr_6	2	14.00	4.67
eprr_7	1	4.00	0.00
eprr_8	2	13.50	4.00
eprr_9	2	9.50	0.00
eprr_10	2	14.50	0.00
eprr_11	2	16.50	4.25
eprr_12	1	4.00	0.00
eprr_13	1	5.00	0.00
eprr_14	1	15.00	2.00
eprr_15	2	17.00	6.67
eprr_16	1	8.00	0.00
eprr_17	2	11.50	0.00
eprr_18	1	4.00	0.00
eprr_19	1	8.00	0.00
eprr_20	2	6.50	0.00

Final Scheduler Performance Summary

Scheduler	**Avg Throughput**	**Avg Turnaround (ms)**	**Avg IO Spacing (ms)**

Student 1: Manoj Kuppuswamy Thyagarajan, **ID:** 101166589
Student 2: Shabesa Kohilavani Arunprakash. **ID:** 101258619

EP	1.55	13.53	1.86
RR	1.25	10.98	2.00
EPRR	1.55	11.55	2.33

Analysis of Simulation Results

Across 60 simulation scenarios, three CPU schedulers were evaluated: EP, RR, and EPRR. For each run, three metrics were recorded:

- Throughput
- Average Turnaround Time
- Average I/O Spacing (used as a proxy for responsiveness)

1. Throughput Analysis

Average throughput

- EP: 1.55
- RR: 1.25
- EPRR: 1.55

In EP, 14 out of 20 files achieved throughput = 2, while 6 files achieved throughput = 1 (14×2 + 6×1) / 20 = 1.55

In RR, only 5 files had throughput = 2, and 15 files had throughput = 1 (5×2 + 15×1) / 20 = 1.25

In EPRR, 14 files had throughput = 2, 6 files had throughput = 1. Same avg as EP = 1.55

From these values, EP and EPRR clearly achieve the highest throughput, while RR lags behind.

EP and EPRR are able to finish more processes within the same time window because they incur less context-switching overhead. Under RR, the fixed time quantum forces the CPU to switch between processes frequently, even when a process could continue using the CPU productively. These extra context switches add overhead and slow down overall process completion, which is especially harmful for CPU-bound tasks.

EPRR matches EP's throughput because its hybrid design avoids the worst parts of pure RR. By scheduling based on priority first, and then using RR only within the same priority level, EPRR keeps high-priority or short jobs from being trapped behind long queues. As a result, it reduces unnecessary context switches while still preserving some fairness.

2. Turnaround Time Analysis

Average turnaround time (lower is better):

- EP: 13.53 ms
- RR: 10.98 ms

- EPRR: 11.55 ms

EP has several high values such as:

- ep_6 = 27.50 ms
- ep_15 = 25.00 ms

These long jobs delay all others → higher overall average.

RR has consistently low turnaround:

- rr_6 = 4.50 ms
- rr_13 = 4.00 ms
- rr_14 = 5.00 ms

These low results pull the average down → best turnaround.

EPRR shows mixed but mostly good turnaround:

- Low examples: eprr_5 = 3.00 ms, eprr_7 = 4.00 ms
- High examples: eprr_3 = 25.00 ms, eprr_15 = 17.00 ms

Here, RR performs best. Because processes are rotated frequently, no single process can sit in the ready queue for a long time without running. This prevents starvation and keeps individual waiting times relatively low. Short jobs benefit in particular, since they do not need to wait for a long job to complete before they get CPU time.

EPRR also achieves good turnaround time, but is slightly slower than RR. The reason is that enforcing priorities can delay lower‑priority tasks: if many higher‑priority processes are present, low‑priority ones may need to wait longer between their CPU slices.

EP gives the worst turnaround time. Since it behaves like FCFS, processes are handled strictly in arrival order. If a long or I/O‑bound job happens to arrive early, all later processes must wait behind it, even if they are very short. This "convoy effect" significantly increases the average turnaround time.

To summarize the mechanisms:

- EP penalizes short processes that arrive after long ones (classic convoy effect).
- RR distributes CPU time evenly, so waiting time is shared more fairly across all processes.
- EPRR tries to balance fairness and priority, improving turnaround for important or short tasks while still preventing complete starvation of others.

3. I/O Spacing (Response Time Proxy)

Average I/O spacing

- EP: 1.86 ms
- RR: 2.00 ms
- EPRR: 2.33 ms

EP has several fast I/O returns, such as:

- ep_1 = 0.00 ms
- ep_3 = 0.00 ms
- ep_10 = 2.67 ms

RR shows more fragmented bursts:

- rr_7 = 5.00 ms
- rr_10 = 4.00 ms
- rr_16 = 4.67 ms

These contribute to a slightly higher spacing.

EPRR delays lower-priority processes:

- eprr_11 = 4.25 ms
- eprr_15 = 6.67 ms

Lower I/O spacing indicates that a process reaches its next I/O operation sooner, so this metric reflects how responsive a scheduler is to I/O-bound workloads. In the results, EP and RR are relatively close, with EP slightly better, because it allows processes to run in long, uninterrupted CPU bursts. For I/O-bound tasks, these uninterrupted bursts let the process reach its I/O request point quickly, producing smaller gaps between I/O operations.

EPRR, however, shows the largest I/O spacing. Since it must enforce priority levels in addition to Round Robin within each level, some processes, especially low-priority I/O-bound ones, experience longer delays between CPU bursts. As a result, these tasks wait longer before issuing their next I/O request, reducing responsiveness slightly.

Effect of Process Type

I/O-bound processes

- Prefer EP, because uninterrupted CPU bursts help them reach their next I/O request sooner.

- Under RR, their CPU bursts are divided into small time slices, which increases the spacing between successive I/O operations.

CPU-bound processes

- Prefer RR, since regular, repeated CPU access prevents them from being stuck behind one long job.

- Under EP, a single long CPU-bound process can dominate the CPU and block all other processes, reducing responsiveness system-wide.

4. Algorithm Behaviour Across Different Workload Types

EP (Earliest Priority / FCFS-like)

Favors:

- Long CPU-burst processes
- I/O-bound tasks that benefit from large, continuous CPU time
- Homogeneous or lower-load scenarios

Why:

- Long, uninterrupted bursts let processes—especially I/O-bound ones—reach I/O faster.
- Predictable behaviour due to no preemption.

Weaknesses:

- Long jobs delay all others (convoy effect).
- Performs poorly on mixed workloads with large variations in burst lengths.

RR (Round Robin)

Favors:

- Short CPU-burst processes
- Mixed workloads with varying burst lengths
- Scenarios requiring fairness

Why:

- Regular time slicing prevents starvation.
- Short tasks tend to finish after only a few quantum cycles.

Weaknesses:

- Frequent context switching reduces throughput.
- Fragmented CPU bursts worsen I/O spacing for I/O-bound tasks.

EPRR (Hybrid Priority + Round Robin)

Favors:

- A mix of short, medium, and long processes
- Systems requiring priority enforcement or quality-of-service separation
- Balanced, general-purpose workloads

Why:

- Priority scheduling allows important or short tasks to run sooner.
- RR within each priority level maintains fairness among same-priority tasks.
- Fewer unnecessary context switches than pure RR → higher throughput.

Weaknesses:

- Low-priority I/O-bound tasks may see larger gaps between CPU bursts.
- More complex behaviour than either EP or RR.

Student 1: Manoj Kuppuswamy Thyagarajan, **ID:** 101166589
Student 2: Shabesa Kohilavani Arunprakash. **ID:** 101258619

5. Overall Comparison and Interpretation

Putting all three metrics together, each scheduler is “best” in a different dimension:

- Best turnaround time: RR
- Best throughput: EP and EPRR (tie)
- Best responsiveness (I/O spacing): EP
- Most balanced overall behaviour: EPRR

The differences come from their core design principles:

- EP is essentially sequential. This usually gives good throughput but leads to poor fairness and long waits for some processes.
- RR is designed to be fair. It improves turnaround time and prevents starvation but pays for this with lower throughput and slightly worse I/O spacing.
- EPRR combines ideas from both. It keeps throughput close to EP while improving fairness and turnaround time compared to EP, though not quite matching RR on fairness.

Metric	**Best Scheduler**	**Why (with data)**
Turnaround	RR	Many tasks < 6ms (rr_6, rr_13, rr_14)
Throughput	EP & EPRR	14/20 files completed 2 processes
Responsiveness	EP	Multiple files have I/O spacing = 0
Balanced	EPRR	Avoids extreme highs like ep_6 (27.5 ms) and rr_17 (26 ms)

6. Why Each Algorithm Favours Different Processes

The table below summarizes how each scheduler behaves for different process types:

Process Type	**EP Performance**	**RR Performance**	**EPRR Performance**
CPU-bound	Poor → long waits for others ep_15 = 25 ms (bad)	Best → fair, frequent CPU slices rr_13 = 4 ms (good)	Good → priority helps reduce waits eprr_13 = 5 ms (good)
I/O-bound	Best → uninterrupted bursts reach I/O faster ep_1 = 0 spacing	Moderate → fragmented bursts delay I/O rr_8 = 5 ms spacing (worse)	Moderate → depends on priority eprr_11 = 4.25 ms
Mixed workload	Inconsistent ep_6 = 27.5 ms (worst)	Good rr_6 = 4.5 ms	Best (balanced behaviour) eprr_6 = 14 ms, balanced