Assignment 3 Part 1 Report
SYSC4001 Operating Systems

Adrian Joaquin
Thawkir Choudhury

1. Introduction

This experiment evaluates three CPU scheduling algorithms, External Priority (EP), Round Robin (RR), and a hybrid External Priority paired with Round Robin (EP+RR), using a simulated multiprogramming system with memory partitions and I/O blocking derived from work done in the previous two assignments. The goal is to measure and compare their effects on key performance metrics:

- Throughput
- Average Waiting Time
- Average Turnaround Time
- Average Response Time

A batch of 20 diverse test cases was executed under all three schedulers. Transitions were logged and analyzed using a Python-based visualization and metrics script. Comparison charts highlight the quantitative performance of each scheduler.

2. Overview of Scheduling Algorithms

2.1 External Priority (EP)

A non-preemptive priority scheduler where lower PID = higher priority.
Once a process enters RUNNING, it keeps the CPU until:

- It blocks on I/O, or
- It terminates.

2.2 Round Robin (RR)

A preemptive time-sliced scheduler using 100 ms quantum.

This ensures fairness among processes but can increase context switching overhead.

2.3 External Priority + RR (EP+RR)

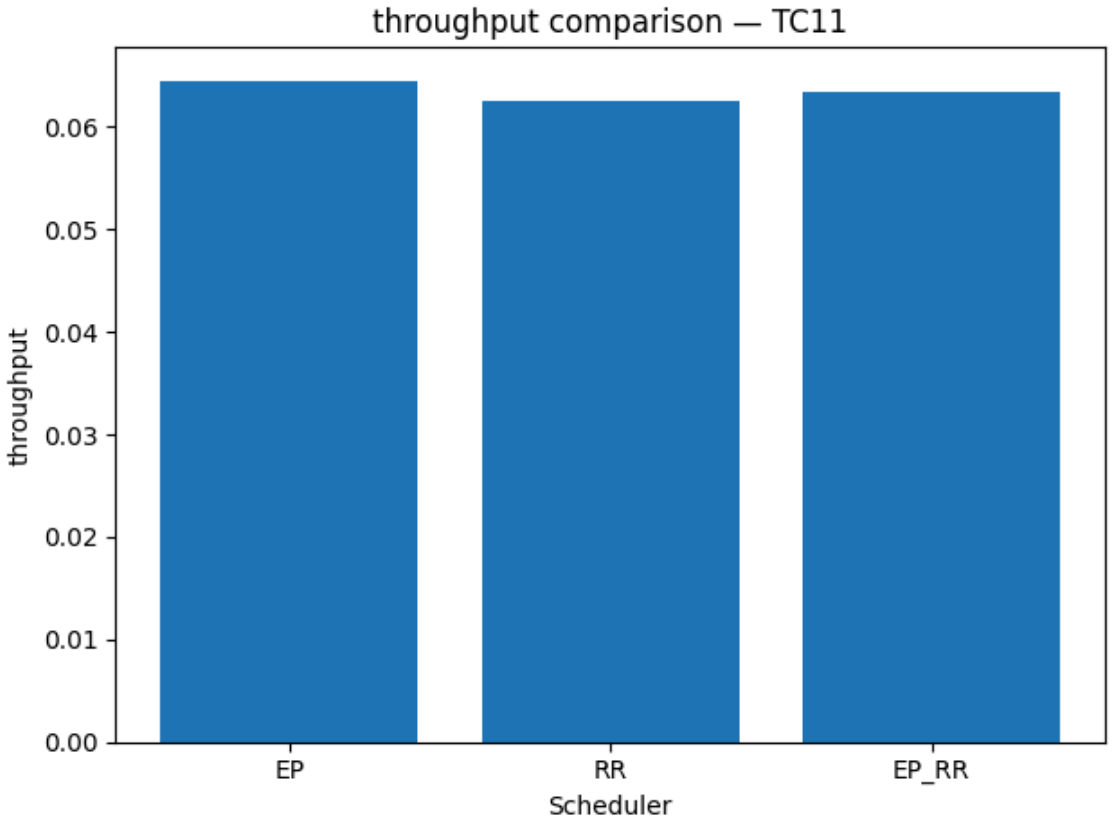
A hybrid scheduler:

- Highest-priority READY process always preempts.
- Processes of equal priority share CPU via RR.
- Offers the responsiveness of RR and predictability of priority scheduling.

3. Results and Analysis

3.1 Throughput

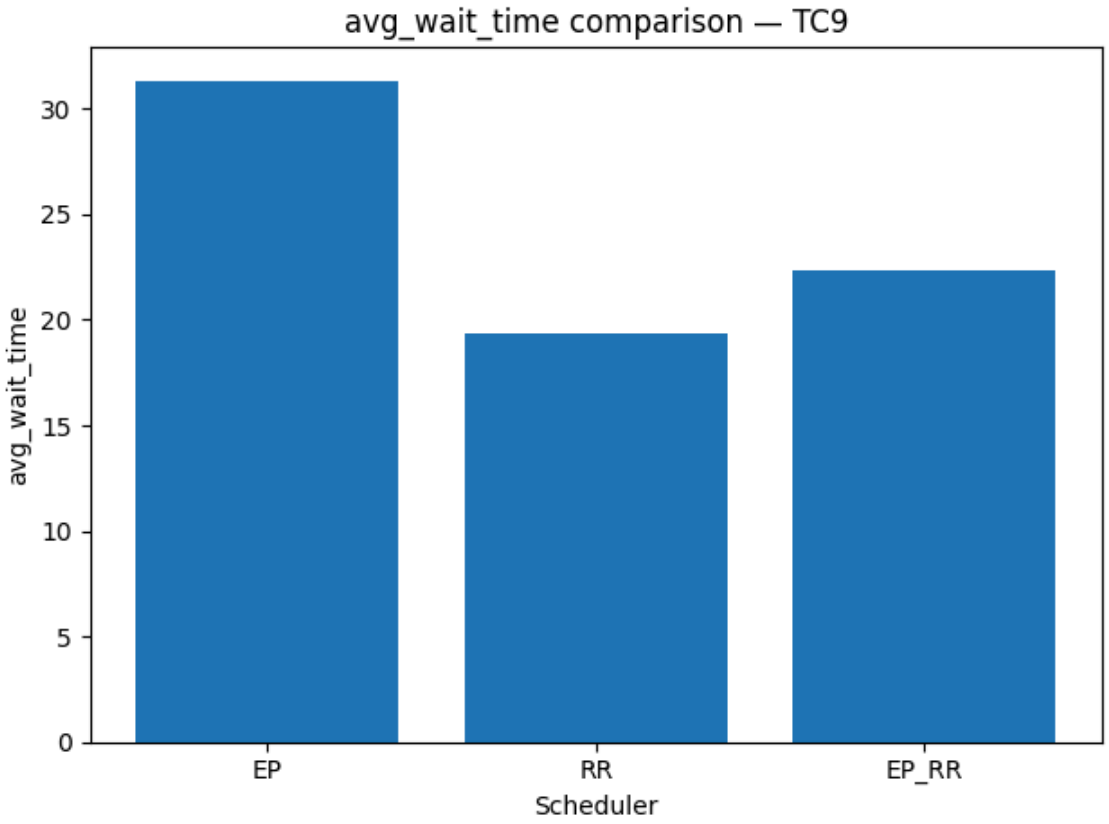
- RR generally achieves the highest throughput, especially in test cases involving long CPU bursts or many similarly sized processes.
- EP achieves the lowest throughput because long high-priority jobs monopolize the CPU, reducing the system's ability to complete other jobs.
- EP+RR throughput falls between the two, combining priority responsiveness with RR's CPU sharing.



This trend is shown in the above comparison generated using test case automation.

3.2 Average Wait Time

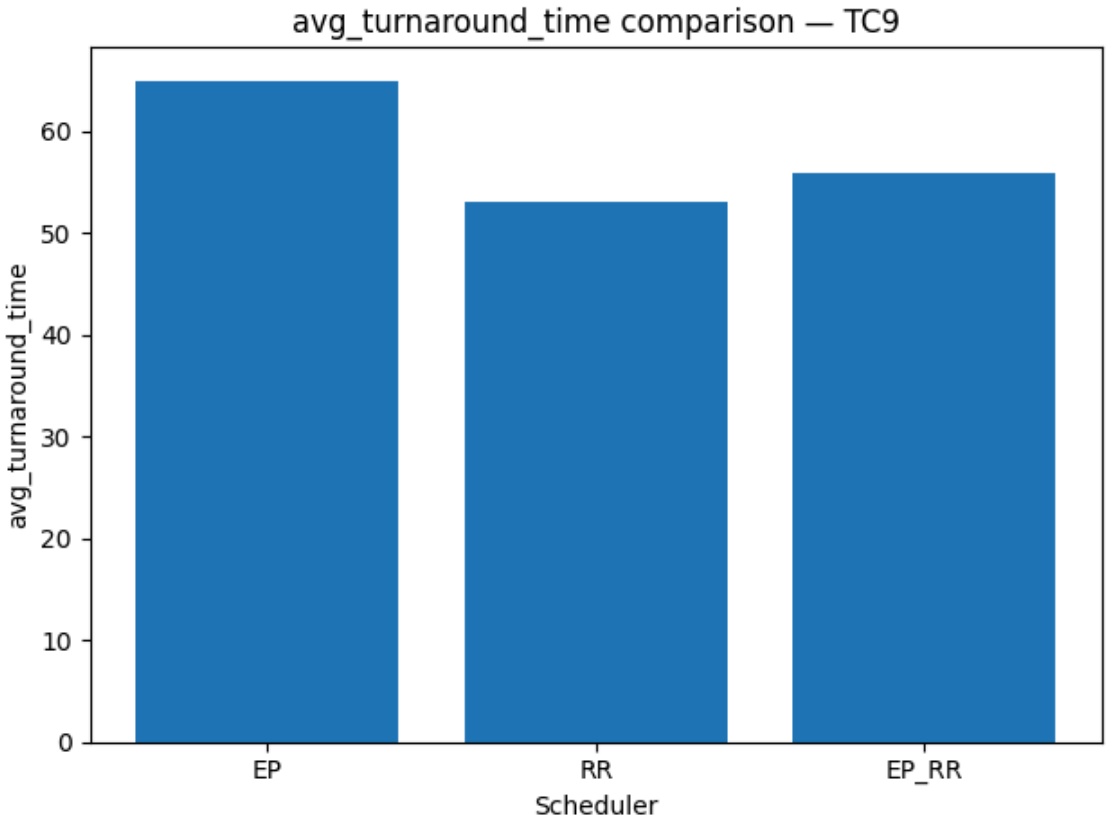
- EP creates very high waiting times for low-priority processes, especially when a single high-priority process runs for a long time.
- RR produces the lowest waiting times overall because time slicing prevents starvation and forces the CPU to circulate among jobs.
- EP+RR reduces waiting time compared to EP, but waiting times will still be skewed for low-priority PIDs.



The above conclusion is supported by the automated test cases as shown above.

3.3 Average Turnaround Time

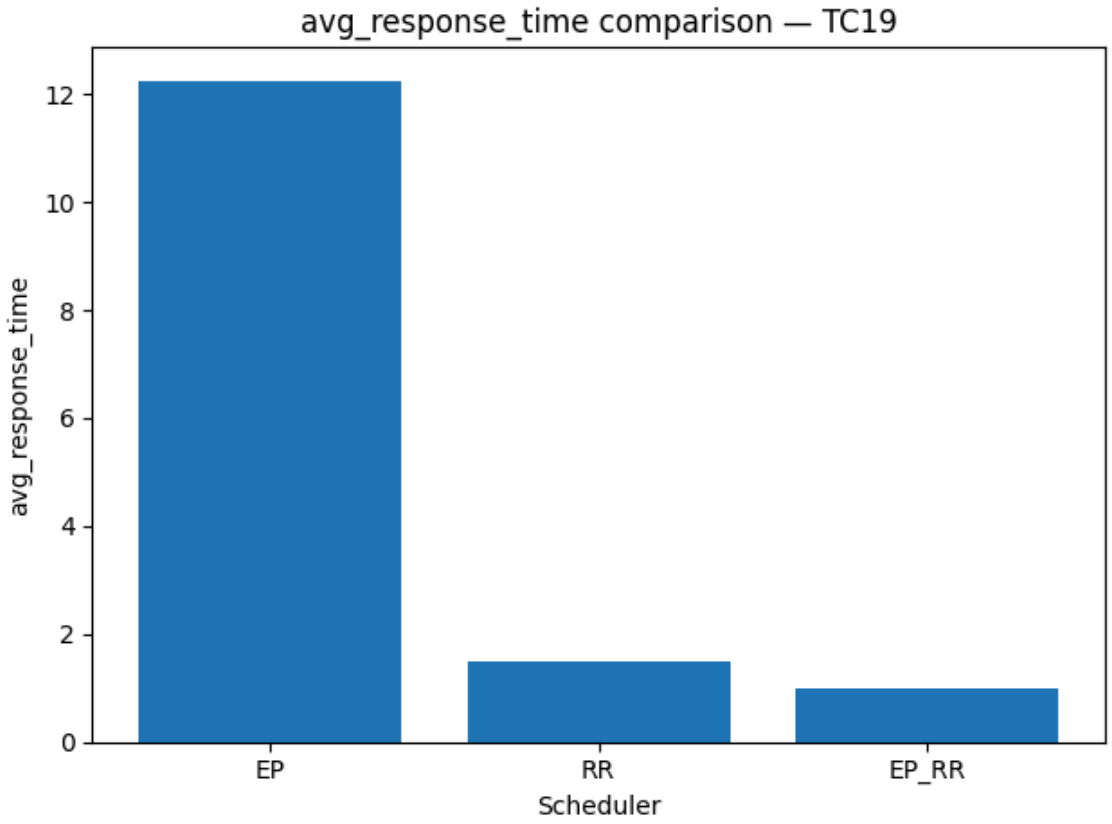
- EP can lead to long turnaround times because it is non-preemptive and each new process must wait for a free partition
- RR tends to improve turnaround time consistency, although it may increase the turnaround time of short, high-priority tasks because they are forced to wait for quantum cycles.
- EP+RR gives the best balance, offering lower turnaround times for important processes while still preventing starvation.



These conclusions are visualized by the above test case, showing how a hybrid approach can offer a balance between the advantages and drawbacks of EP and RR.

3.4 Average Response Time

- EP has the fastest response time for high-priority processes, because the highest priority job immediately runs as soon as it arrives. However, this means that low priority processes see very slow response times.
- RR offers lower overall response time variance, because every process gets CPU time quickly.
- EP+RR combines both features, highest priority jobs get the CPU first, but lower-priority processes still get prompt time slices leading the fastest possible response time



The above conclusions are shown in this diagram as EP is shown to have the longest response time in a scenario where the priority of processes is varied. The preemptive nature allows the RR scheduler to improve on this and the hybrid approach improves even further.

4. Recommendations / Final Discussion

EP is useful for strict priority scheduling but harms fairness. RR improves fairness and throughput but may delay high-priority tasks. EP+RR provides the best balance. These conclusions align with real-world OS scheduling strategies.

5. Conclusion

EP favors important tasks but causes starvation. RR maximizes fairness but ignores priority. EP+RR balances both concerns. The metrics and visualizations confirm expected theoretical behavior.