**Group 3: Katherine Nguyen, Dishank Oza, Shreyas Parimi, Arya Pefourque, Lucas Peters**

In this project, we examine the efficiency of different process scheduling algorithms by averaging 5 iterations and comparing their performance based on turnaround time, waiting time, response time, and throughput. Here is a summary of our results:

| Algorithm | Avg TAT | Avg WT | Avg RT | Avg Throughput |
| --- | --- | --- | --- | --- |
| FCFS | 7.0 | 1.1 | 0.6 | 5.0 |
| SJF | 7.0 | 1.1 | 0.6 | 5.0 |
| SRT | 7.2 | 1.3 | 0.4 | 5.0 |
| RR | 8.0 | 2.1 | 0.2 | 5.0 |
| HPF (Non-Pree) | 6.9 | 1.2 | 0.8 | 5.0 |
| HPF (Pree) | 16.9 | 11.2 | 5.2 | 32.0 |

units are computer ticks

**FCFS:**

First Come First Serve is a simple approach that works well when processes are spaced out and not vastly different in size. In this assignment, FCFS has a consistent result comparable to the other processes. However, if there are a larger number of processes or processes that take longer or shorter in time, FCFS may get stuck as short jobs get stuck behind long ones.

**SJF:**

Shortest Job First will compare the length of a process and prioritizes the one with the smaller time. It performed identically to FCFS because all processes arrived at similar intervals and did not overlap, but in real systems long jobs may be stuck behind many short jobs and starve.

**SRT:**

Shortest Remaining time is a preemptive version of SJF with a slightly increased TAT but a lower RT. This means that it starts responding to a new job the fastest out of all of the processes. It works well for time-sensitive workloads or interactive systems, but SRT could have a higher overhead from constant context switching.

**RR:**

Round Robin had the second highest TAT and WT out of all of the algorithms, but a much lower RT. This makes sense considering that RR context switches often to keep fairness between the processes, and it works well in time-sharing environments.

**HPF (Non-Pree):**Highest Priority First will run the highest priority job until it finishes, and it has the lowest TAT out of all of the processes. Overall, it handles small batches very efficiently when priorities are well distributed. However, if workloads are larger or unevenly distributed in terms of priority, then low priority processes may starve.

**HPF (Pree):**This algorithm had the worst TAT and WT because of the constant context switching which adds significant overhead. It could work in systems that must respond immediately to high-priority events.

**Overall Analysis:**

Given the data and comparing the statistics, HPF (NP) had the best overall performance, with the lowest TAT and WT. RR had the best RT, but a higher TAT and WT. Overall, all processes had around the same throughput, besides HPF (P) which lagged behind in every category. These results show us that certain scheduling algorithms are better in certain contexts, depending on workload, arrival pattern, and the overall goal of the system. Every algorithm will have its own strengths and tradeoffs, and it all depends on what the OS prioritizes, such as responsiveness, throughput, or fairness.