School of Information Technologies and Engineering, ADA University

CSCI3510 – Principles of Operating Systems

Spring 2025 – 5/3/2025

Project Team 3 – Phase 2 (Final Project)

Project Title: CPU Scheduling Simulation & Comparative Analysis

Team member	Contribution to the homework	Estimated %
Laman Panakhova 16882	FCFS, Literature Review, PPT, Visuals, Uniform Distribution in Process Generation	20%
Aykhan Guliyev 12829	SJF, Project Proposal, PPT, Visuals, Random Distribution in Process Generation	20%
Emil Mirzazade 12584	RR, System Architecture, PPT, Visuals, Uniform Distribution in Process Generation	20%
Vusal Mammadov 16238	PS, Preliminary Results, PPT, Visuals, Random Distribution in Process Generation	20%
Huseyn Sadatkhanov 13770	RR, Conclusion, PPT, Visuals, Random Distribution in Process Generation	20%

1. Project proposal

1.1. Introduction

Nowadays, in modern operating systems, efficient CPU scheduling is considered very important for optimizing performance, ensuring fairness among processes, and minimizing waiting time. To follow the continuing trend in this field our project focuses on implementing and comparing different CPU scheduling algorithms to analyze their efficiency and impact on process execution. The primary goal of the project is to simulate and evaluate First Come, First Served (FCFS), Round Robin (RR), Shortest Job First (SJF), and Priority Scheduling (PS) algorithms.

1.2. Objectives

We intend to compare the performance based on average waiting time, turnaround time, and CPU utilization by implementing a simulation of the given CPU scheduling algorithms. Also, we try to visualize the sequential execution process through Gantt charts and demonstrate the impact of scheduling choices on process efficiency. Consequently, we provide a comparative analysis to determine the most efficient algorithm under various circumstances. We achieve this with several visualization tools by having the comparisons more readable.

1.3. Project Scope and Relevance

Considering the existing knowledge, we can verify that CPU scheduling plays a very important role in resource allocation and system responsiveness. Understanding and analyzing these algorithms grant us with significant understandings into real-world operating systems like windows and Linux, which use various versions of these scheduling methods. For example, Linux kernel uses the completely fair scheduler (CFS), which incorporates principles from RR and PS. Windows OS employs a preemptive priority-based scheduling algorithm by using a multi-level feedback queue (MLFQ). Similarly, embedded systems often use FCFS or PS due to simplicity and resource constraints.

2. Project Simulation

2.1 Research on Background & Literature Review

2.1.1 Overview

As it is asserted, scheduling is a key function of an operating system that directly impacts CPU performance by managing resource utilization. Different algorithms focus on ensuring fairness among processes in the ready queue, increasing throughput, and reducing issues like average waiting time (Omar, Jihad, & Hussein, 2021).

Moreover, adjustments and testing of CPU scheduling algorithms in an OS kernel is complex and time-consuming. It requires performance evaluation on real application assignments. It is clear that efficient CPU scheduling is essential for maximizing processor utilization by keeping multiple processes running simultaneously, aligning with OS design goals (Goel and Garg, 2012).

Thus, various scheduling algorithms, including the algorithms that we analyze - RR, FCFS, SJF, and PS, have been studied in the literature review, highlighting different optimization techniques to improve CPU performance metrics such as waiting time, response time, and turnaround time. However, no single algorithm fully satisfies all the criteria (Ali et al., 2021).

As it is discussed in the related papers, each CPU scheduling algorithm has its own unique properties, influencing how they prioritize processes based on several factors such as efficiency, throughput, and fairness. The short-term scheduler selects a process from the ready queue whenever the CPU is idle, aiming to optimize utilization, minimize waiting and response times, and ensure fair allocation (Goel and Garg, 2012).

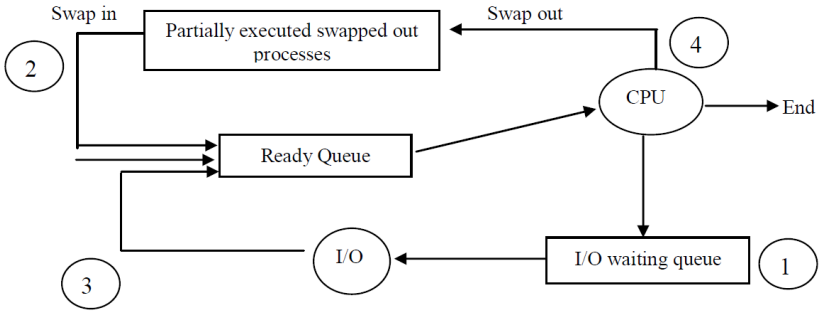
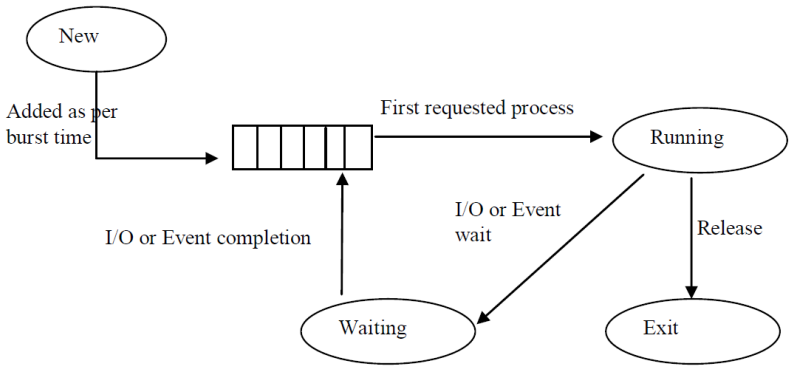


Figure 1: Shows the following states have been executed in the CPU Scheduler

(Goel and Garg, 2012)

2.1.2 First Come First Serve (FCFS)

From the articles it is shown that the simulation of the First Come First Serve (FCFS) scheduling algorithm results in low computational overhead but poor performance, with lower throughput and longer average waiting times. It is also mentioned that when short jobs are submitted after longer ones, they experience significantly longer waiting times (Saleem, U., & Javed, M. Y., 2000).



(Goel and Garg, 2012)

2.1.3 Shortest Job First (SJF)

From a different perspective, various studies compare CPU scheduling algorithms and also highlights that Shortest Job First (SJF) generally outperforms First-Come, First-Served (FCFS)

in reducing waiting time and improving efficiency. It demonstrates improvements in response time, resource utilization, and fairness (Ali et al., 2021).

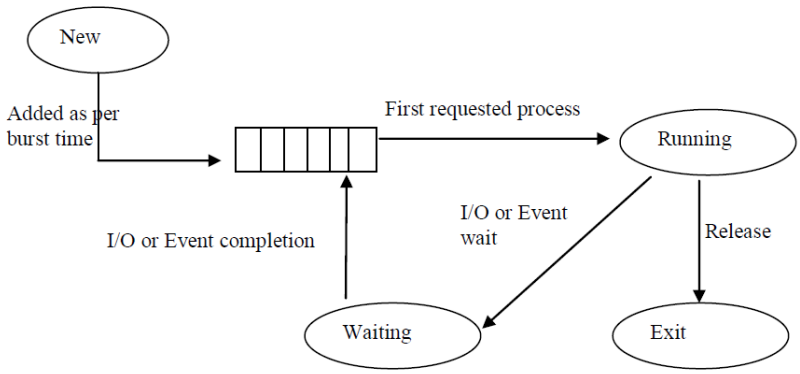


Figure 5: Shortest Job First Scheduling

(Goel and Garg, 2012)

2.1.4 Round Robin (RR)

With the similar analogy, Round Robin (RR) scheduling algorithm allocates CPU time to each process in a cyclic manner. The process is happening by using a fixed time slice, ensuring fairness but potentially increasing context switching overhead. Choice of an optimal time slice is important yet difficult, as a short quantum reduces efficiency due to frequent switches, while a long quantum results in poor response times, resembling First-Come, First-Served (FCFS) (International Journal of Emerging Technology and Advanced Engineering, 2014).

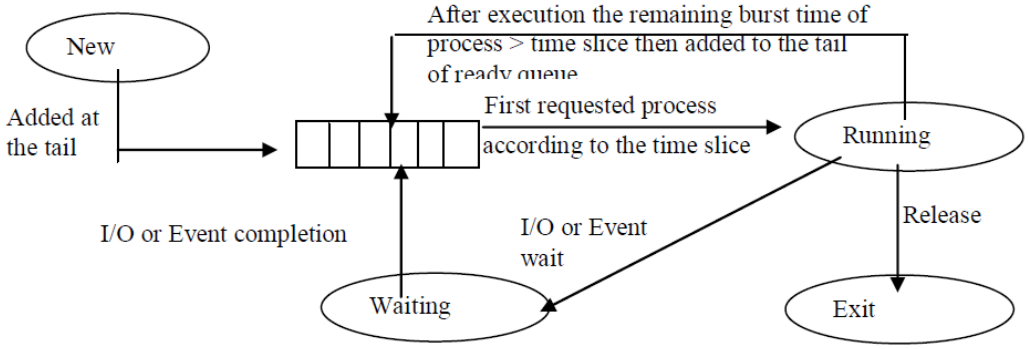


Figure 6: Round Robin Scheduling

(Goel and Garg, 2012)

2.1.5 Priority Scheduling (PS)

The articles claim that Priority Scheduling (PS) classifies processes based on predefined criteria, where higher-priority processes are allocated CPU time before lower-priority ones. In the preemptive version, lower-priority processes may experience starvation if higher-priority tasks continuously enter the ready queue, whereas the non-preemptive version allows a running process to complete before switching (Omar, Jihad, & Hussein, 2021).

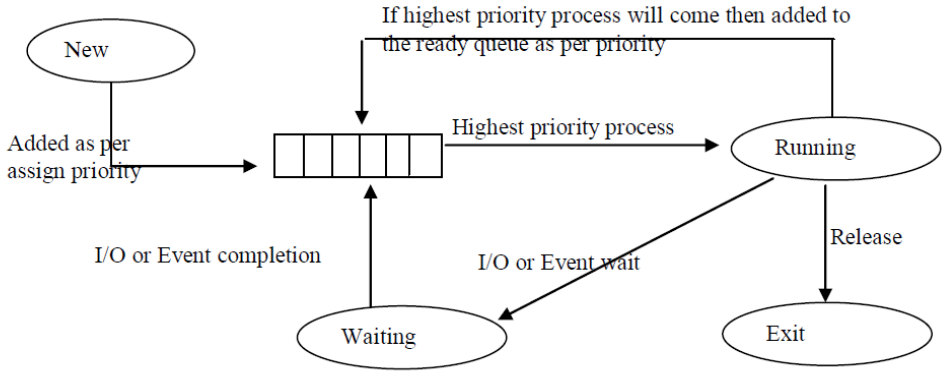


Figure 7: Priority Scheduling

(Goel and Garg, 2012)

2.1.6 Comparative Analysis

Summarizing the key takeaways from different articles we conclude our analysis with the comparison of different CPU scheduling algorithms. The main comparative reasoning sheds light on various traits regarding complexity, average waiting time (AWT), preemption, and overall performance. Firstly, FCFS shows the straightforward performance but results in high waiting time and low throughput. On the other hand, SJF minimizes waiting time but may cause starvation. Separately, Round Robin (RR) guarantees fairness with fixed time slices but can lead to large waiting times if the time quantum is not optimized. Last but not the least Priority Scheduling (PS), especially in its preemptive form, may cause starvation but offers good performance in particular cases, such as batch processing (Omar, Jihad, & Hussein, 2021).

2.1.7 Process Generation with Random Distribution

There is a specific paper that evaluates main four performance criteria—average waiting time (AWT), average turnaround time (ATAT), waiting time variance deviation (WTVD) and average response time (ART). The paper achieves this by simulating the scheduling of 500 randomly generated processes. Mainly, the study is divided into two categories: Shortest Job First (SJF) and Priority Scheduling (PS). The point of the task id to compare the traditional algorithms with new variants where processes are independent and CPU-bound (Mustapha, Abdullahi, & Junaidu, 2015).

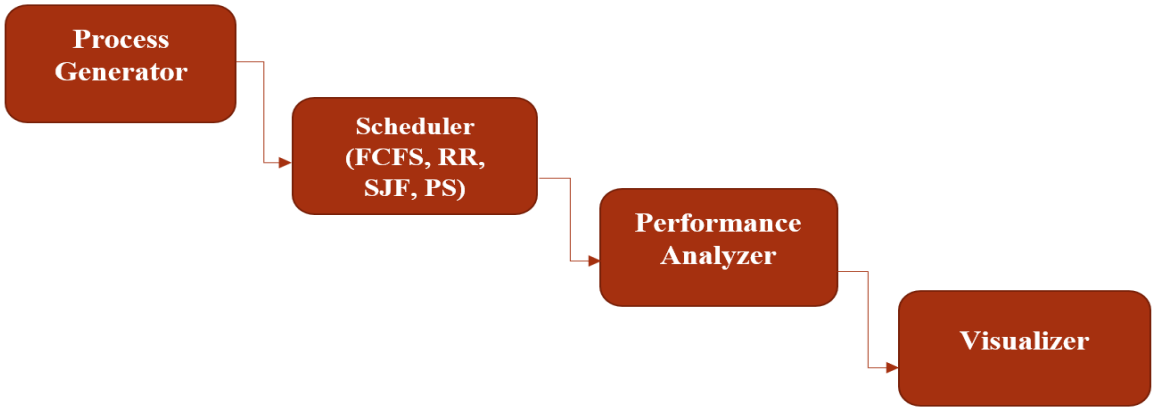
2.1.8 Process Generation with Uniform Distribution

Another interesting paper claims that they developed a process generator that creates processes as tuples, with attributes like arrival time, CPU time, and I/O occurrences. The arrival time follows a Poisson distribution, CPU time is uniformly distributed, and I/O times are based on an exponential distribution. These attributes achieve the purpose with parameters controlled by the user for the number of processes, and the mean and standard deviation of the distributions (Chelladurai, 2006).

As it is fully obvious from the mentioned articles, a process generator was developed to create process sets with attributes such as arrival time, CPU time, and priority, using Poisson distribution for arrivals, uniform distribution for burst times, and exponential distribution for priorities. The study evaluates CPU scheduling algorithms with 500 processes, where burst times range from 1 to 10, generated based on these distributions (Mustapha, Abdullahi, & Junaidu, 2015). This implies that random distribution and uniform distribution for scheduling algorithms is used in process generation frequently and highly practical.

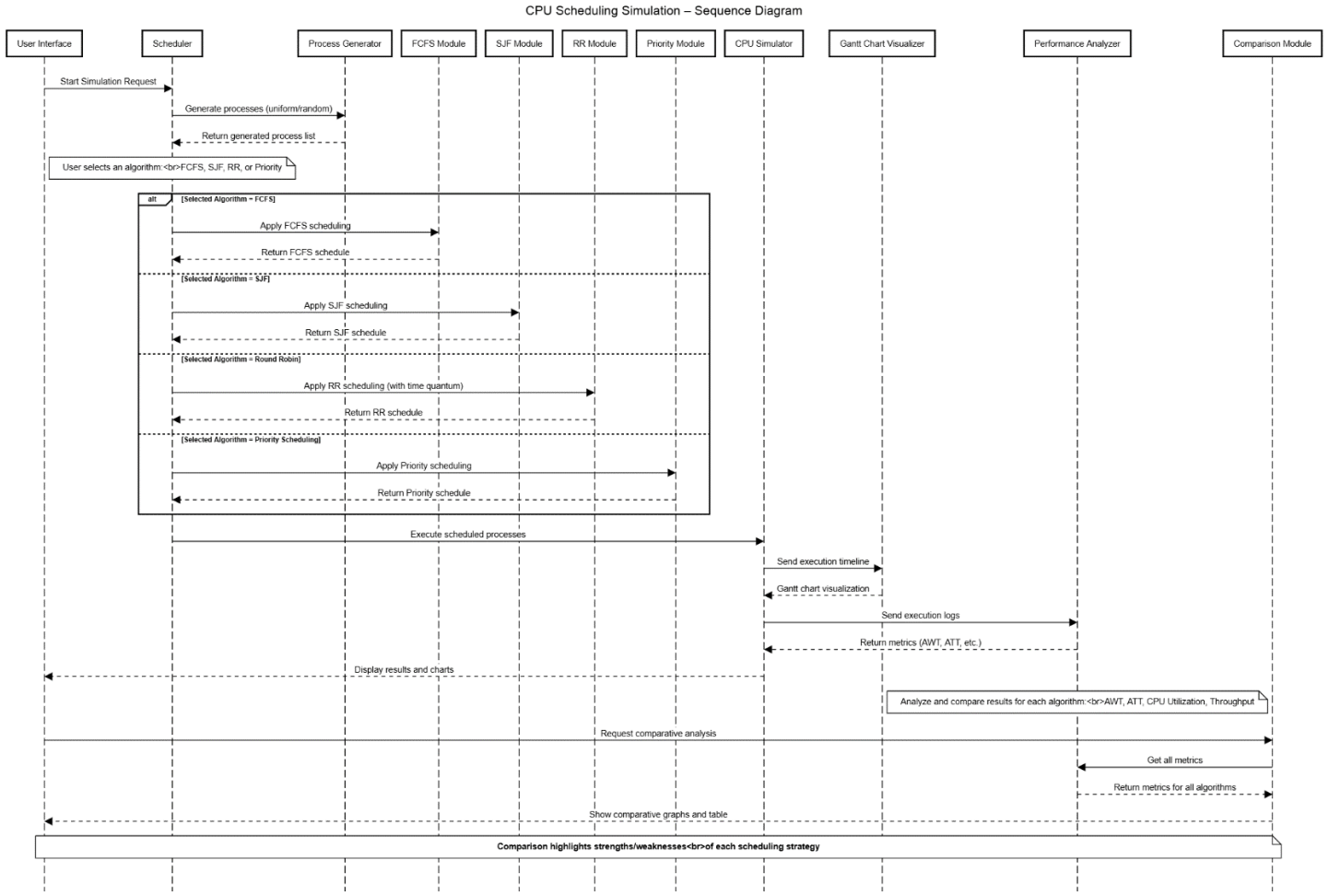
2.2. System Architecture

Our system consisting of simulation and comparative analysis will be designed as a sectional simulation environment where processes are scheduled based on different algorithms. The main components of the implementation will include *process generator* - where random processes with varying burst times and priorities will be generated, *scheduler* - where FCFS, RR, SJF, and PS will be implemented, *performance analyzer* - where key metrics such as waiting time and turnaround time will be measured, and *visualizer* - where scheduling results will be displayed graphically.



Diving deeper into more details, algorithmic simulations will mainly focus on storing incoming processes with attributes such as process id, arrival time, burst time, start time, end time, completion time, turnaround time, and waiting time. In the end we will compute the average waiting and the average turnaround time for each algorithm separately in order to analyze the overall performance. In order to achieve more precision and readable format “Gantt Chart” will be described for visualizing the process execution order. A final high-level system architecture

diagram is as follows.



3. Improvements and Adjustments on Code

We improved the current implementation and adjusted the code for better variability and efficiency from the point of comparative analysis. In addition to essential metrics like average waiting time (AWT) and average turnaround time (ATAT), we included two advanced performance metrics such as average response time (ART) which measures how quickly a process starts execution after arrival and waiting time variance deviation (WTVD) which calculates the standard deviation of waiting times. These enhancements provide a more holistic and quantitative evaluation of scheduling efficiency and fairness across algorithms.

4. Preliminary Results

As a last step of the project we implemented and tested the scheduling algorithms First Come First Served (FCFS), Round Robin (RR), Shortest Job First (SJF), and Priority Scheduling (PS). We manually enter the number of processes to generate and process generation type. In the end we obtain `process arrival time, burst time, start time, end time, completion time, turnaround time, and waiting time` as an output. Also, we consider `average waiting time, average turnaround time, average response rime, and waiting time variance deviation`. Each algorithm was tested with a set of processes, and the Gantt charts generated illustrate how they execute processes differently. Similarly, the performance metrics also were described with different visuals.

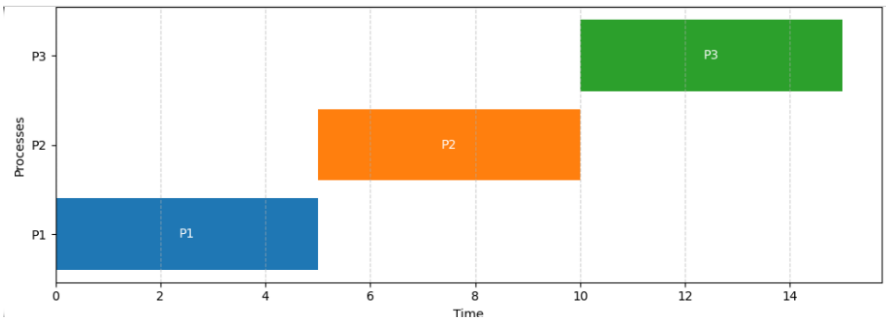
Initial results obtained for Uniform Distribution:

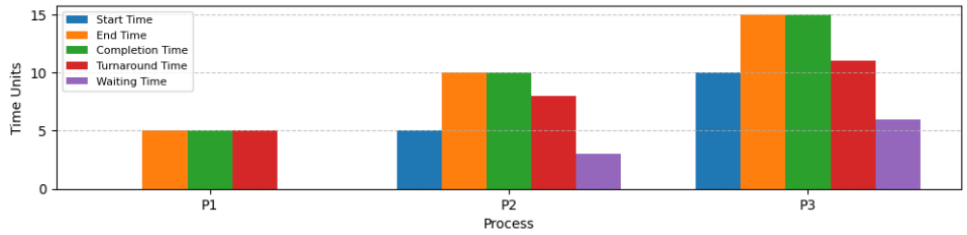
```
Enter number of processes (max 10): 3
Select process generation type:
1. Random
2. Uniform
Enter choice: 2
```

```
--- First-Come First-Served (FCFS) ---

Process Arrival   Burst   Start   End     Completion  Turnaround  Waiting
P1      0         5       0       5       5           5           0
P2      2         5       5       10      10          8           3
P3      4         5       10      15      15          11          6

Average Waiting Time: 3.00
Average Turnaround Time: 8.00
Average Response Time: 3.00
Waiting Time Variance Deviation: 2.45
```

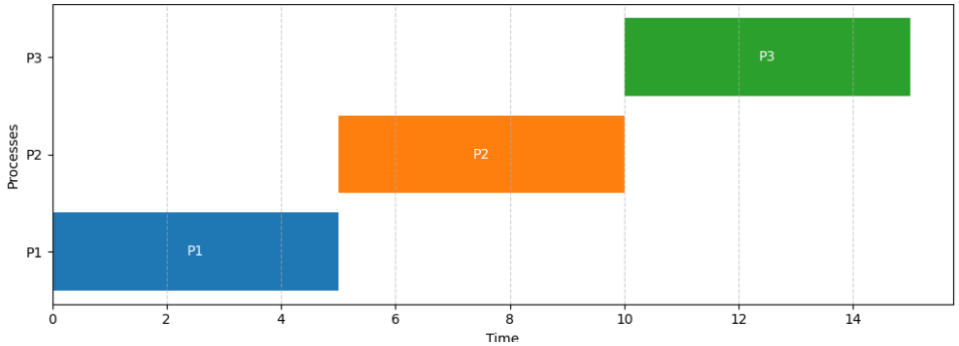


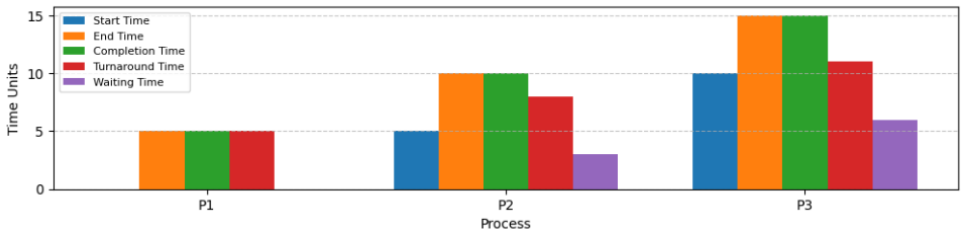



```
--- Shortest Job First (SJF, Non-preemptive) ---

Process Arrival   Burst   Start   End     Completion  Turnaround  Waiting
P1      0         5       0       5       5           5           0
P2      2         5       5       10      10          8           3
P3      4         5       10      15      15          11          6

Average Waiting Time: 3.00
Average Turnaround Time: 8.00
Average Response Time: 3.00
Waiting Time Variance Deviation: 2.45
```

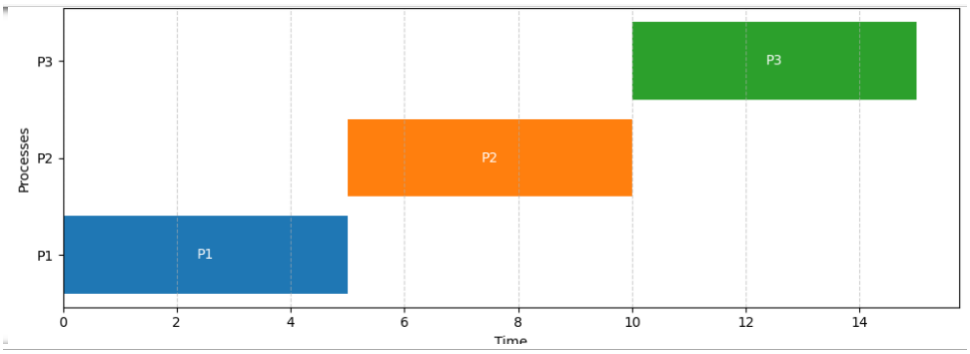


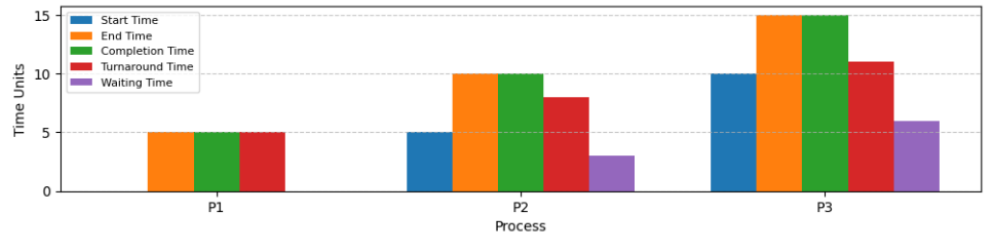



```
--- Priority Scheduling (Non-preemptive) ---

Process Arrival   Burst   Start   End     Completion  Turnaround  Waiting
P1      0         5       0       5       5           5           0
P2      2         5       5       10      10          8           3
P3      4         5       10      15      15          11          6

Average Waiting Time: 3.00
Average Turnaround Time: 8.00
Average Response Time: 3.00
Waiting Time Variance Deviation: 2.45
```

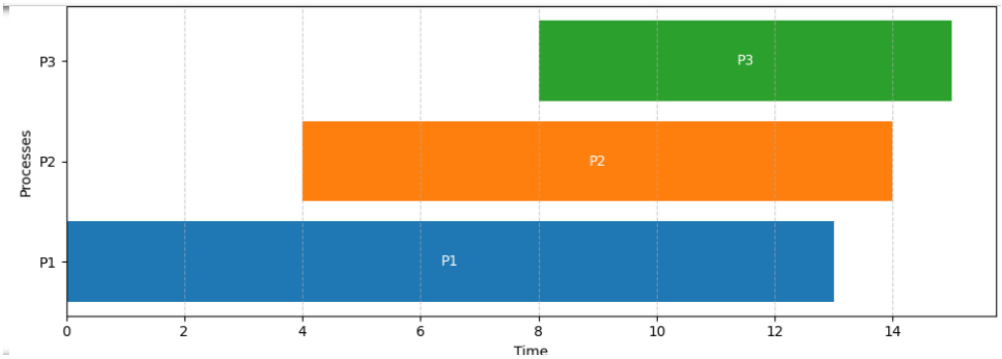


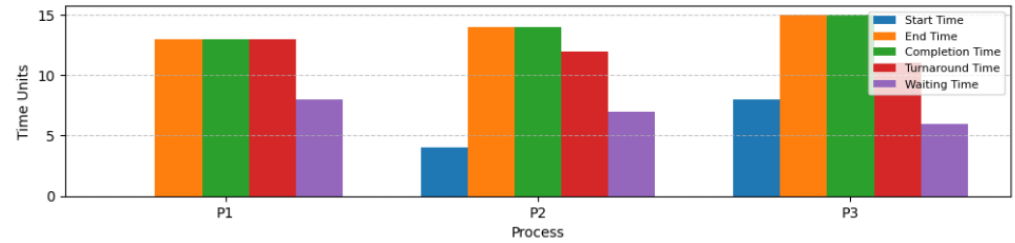



```
--- Round Robin (Quantum = 4) ---

Process Arrival   Burst   Start   End     Completion  Turnaround  Waiting
P1      0         5       0       13      13          13          8
P2      2         5       4       14      14          12          7
P3      4         5       8       15      15          11          6

Average Waiting Time: 7.00
Average Turnaround Time: 12.00
Average Response Time: 2.00
Waiting Time Variance Deviation: 0.82
```





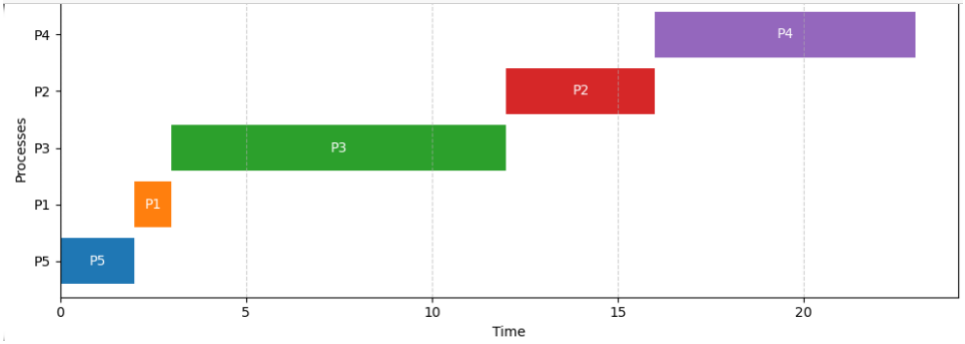
Initial results obtained for Random Distribution:

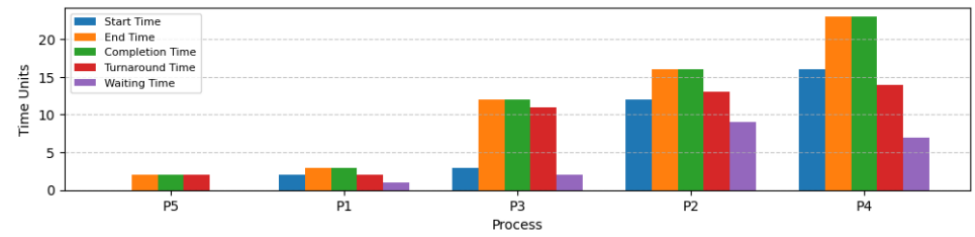
```
Enter number of processes (max 10): 5
Select process generation type:
1. Random
2. Uniform
Enter choice: 1
```

```
--- First-Come First-Served (FCFS) ---

Process Arrival   Burst   Start   End     Completion  Turnaround  Waiting
P4      1         6       1       7       7           6           0
P5      5         1       7       8       8           3           2
P3      7         5       8       13      13          6           1
P2      7         8       13      21      21          14          6
P1      7         9       21      30      30          23          14

Average Waiting Time: 4.60
Average Turnaround Time: 10.40
Average Response Time: 4.60
Waiting Time Variance Deviation: 5.12
```

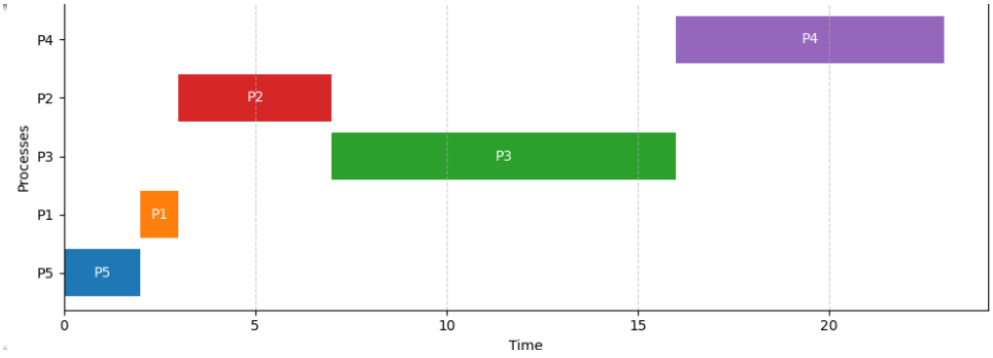


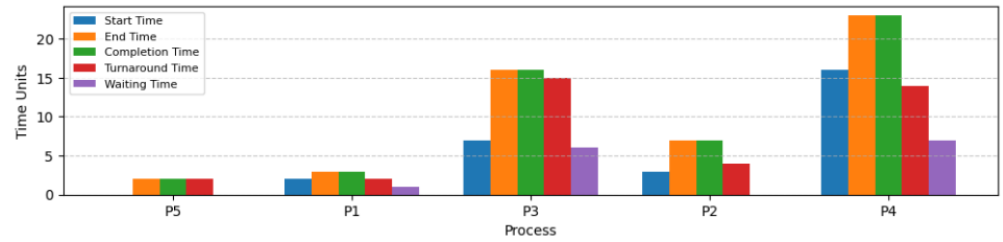


```
--- Shortest Job First (SJF, Non-preemptive) ---

Process Arrival   Burst   Start   End     Completion  Turnaround  Waiting
P4      1         6       1       7       7           6           0
P5      5         1       7       8       8           3           2
P3      7         5       8       13      13          6           1
P2      7         8       13      21      21          14          6
P1      7         9       21      30      30          23          14

Average Waiting Time: 4.60
Average Turnaround Time: 10.40
Average Response Time: 4.60
Waiting Time Variance Deviation: 5.12
```

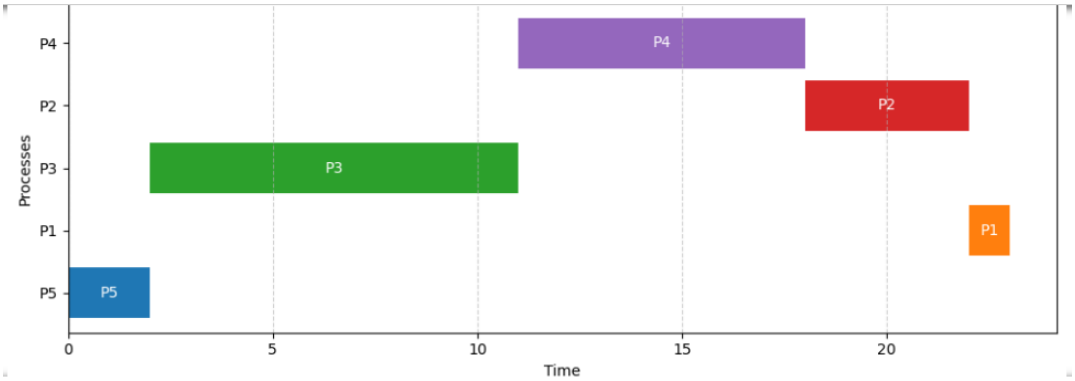


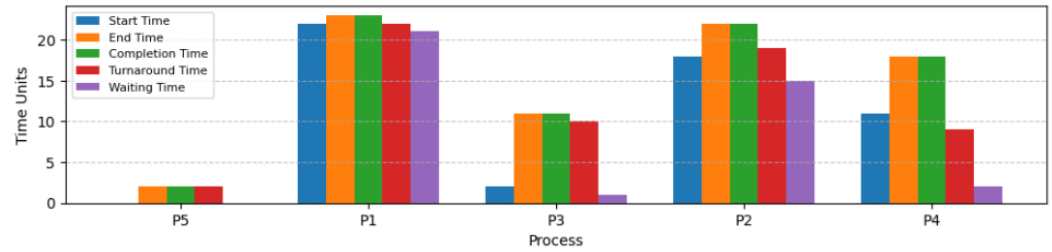



```
--- Priority Scheduling (Non-preemptive) ---

Process Arrival   Burst   Start   End     Completion  Turnaround  Waiting
P4      1         6       1       7       7           6           0
P5      5         1       7       8       8           3           2
P3      7         5       8       13      13          6           1
P2      7         8       13      21      21          14          6
P1      7         9       21      30      30          23          14

Average Waiting Time: 4.60
Average Turnaround Time: 10.40
Average Response Time: 4.60
Waiting Time Variance Deviation: 5.12
```

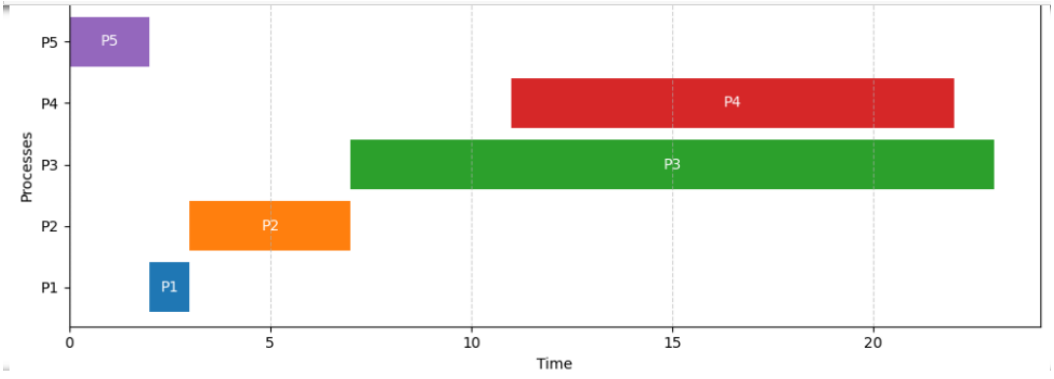


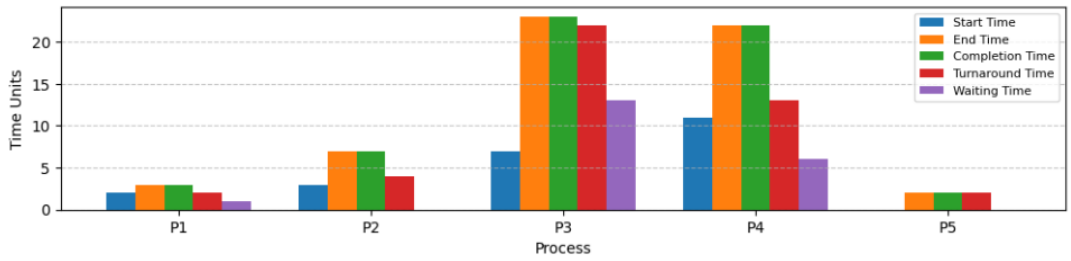



```
--- Round Robin (Quantum = 4) ---

Process Arrival   Burst   Start   End     Completion  Turnaround  Waiting
P1      7         9       8       30      30          23          14
P2      7         8       12      28      28          21          13
P3      7         5       16      29      29          22          17
P4      1         6       1       8       8           7           1
P5      5         1       5       6       6           1           0

Average Waiting Time: 9.00
Average Turnaround Time: 14.80
Average Response Time: 3.00
Waiting Time Variance Deviation: 7.07
```





4. Conclusion

Our preliminary results demonstrate the strengths and weaknesses of each CPU scheduling algorithm. It is obvious from the charts that each algorithm differs from each other for its different metrics for the use of time. In the next phase, we will expand our implementation, improve our analysis, and document trade-offs in scheduling policies. By completing this project, we gained theoretical and practical insights into OS design and scheduling efficiency.

Algorithm	Best For	Weaknesses
FCFS	Simple batch processing	Requires high waiting time
RR	Time-sharing systems	If quantum is too small, then too many context switches
SJF	Optimizing wait times	Happens starvation of long processes
PS	Real-time systems	Happens starvation of lower-priority processes

Recourses:

Tanenbaum, A. S., & Bos, H. (2015). *Modern operating systems* (4th ed.). Pearson.

N. Goel and R. B. Garg, *"A Comparative Study of CPU Scheduling Algorithms," International Journal of Graphics & Image Processing*, India, vol. 2, no. 4, Nov. 2012.

S. M. Ali, R. F. Alshahrani, A. H. Hadadi, T. A. Alghamdi, F. H. Almuhsin, and E. E. El-Sharawy, *"A Review on the CPU Scheduling Algorithms: Comparative Study," IJCSNS International Journal of Computer Science and Network Security*, vol. 21, no. 1, pp. 19-XX, Jan. 2021. doi: 10.22937/IJCSNS.2021.21.1.4.

International Journal of Emerging Technology and Advanced Engineering, vol. 4, no. 1, Jan. 2014. Available: www.ijetae.com.

Omar, H. K., Jihad, K. H., & Hussein, S. F. (2021). Comparative analysis of the essential CPU scheduling algorithms. *Bulletin of Electrical Engineering and Informatics, 10*(5), 2742–2750. https://doi.org/10.11591/eei.v10i5.2812

M. Umar Saleem and M. Younus Javed, *"Computer Engineering Department, College of Electrical and Mechanical Engineering, Peshawar Road, Rawalpindi - 46000, National University of Sciences and Technology - PAKISTAN,"* Email: myjaved@yahoo.com, Tel: 051-478783, 051-561-32966, ISBN 0-7803-6355-8.

Chelladurai, J. (2006). *Fair and efficient CPU scheduling algorithms* (Master's thesis, University of Northern British Columbia). Library and Archives Canada. Available: https://core.ac.uk/download/pdf/84872662.pdf

Mustapha, M. A., Abdullahi, S. E., & Junaidu, S. B. (2015). *Optimization of priority-based CPU scheduling algorithms to minimize starvation of processes using an efficiency factor*. *International Journal of Computer Applications, 132*(11). Available: https://www.ijcaonline.org/research/volume132/number11/mustapha-2015-ijca-907574.pdf