# 

**Project Report**

CS-311-L

**Scheduling Algorithms**

Team Members:

Hamza Zaidi 2022379

Hassan Rais 2022212

Submitted to: Mam Mahrukh

#### **Title**

**A Comparative Analysis and Implementation of Process Scheduling Algorithms**

#### **Abstract**

This project explores the implementation and evaluation of five key process scheduling algorithms: First-Come-First-Serve (FCFS), Shortest Job First (SJF), Priority Scheduling, Round Robin (RR), and a proposed Hybrid Scheduler. The Hybrid Scheduler integrates elements of Priority Scheduling and Round Robin to balance efficiency, fairness, and adaptability. The report outlines the methodologies, performance metrics, and advantages of each algorithm, emphasizing why the Hybrid Scheduler offers superior performance in modern multitasking systems.

#### **Introduction**

Process scheduling is a cornerstone of operating system functionality, dictating the order in which processes access the CPU. An effective scheduling strategy can improve system performance by reducing waiting times, improving fairness, and ensuring efficient CPU utilization. This project aims to:

1. Implement and analyse key scheduling algorithms.
2. Design and develop a Hybrid Scheduler.
3. Compare the performance of traditional algorithms with the Hybrid Scheduler.

#### 

#### **Objectives**

The primary aims of this project include:

1. **Implementation**: Develop C-based implementations for FCFS, SJF, Priority Scheduling, Round Robin, and Hybrid Scheduler algorithms.
2. **Analysis**: Evaluate and compare algorithms based on metrics such as waiting time, turnaround time, and fairness.
3. **Innovation**: Create a Hybrid Scheduler that integrates the strengths of traditional algorithms while addressing their weaknesses.

#### **Methodology**

1. **Algorithm Implementation**:
   * All algorithms were implemented in C programming language to simulate a scheduling environment.
   * Multithreading techniques were used to mimic concurrent execution of processes.
2. **Performance Metrics**:
   * Average Waiting Time (AWT): Measures the average time processes spend in the ready queue.
   * Turnaround Time (TAT): Measures the total time taken from process arrival to completion.
   * CPU Utilization: Evaluates the efficiency of the CPU in executing processes.
   * Fairness: Assesses whether all processes receive adequate CPU time.
3. **Testing Environment**:
   * The algorithms were assessed on predefined workloads simulating diverse process types and arrival patterns.
   * Comparative analysis was conducted to confirm performance improvements.

#### **Scheduling Algorithms Overview**

1. **First-Come-First Serve (FCFS)**
   * **Description**: Executes processes in the order of their arrival.
   * **Advantages**:
     + Easy to implement.
     + Minimal overhead.
   * **Disadvantages**:
     + Convoy effect leads to high waiting times.
     + Inefficient for varying process sizes.
2. **Shortest Job First (SJF)**
   * **Description**: Prioritizes processes with the shortest burst time.
   * **Advantages**:
     + Minimizes average waiting time.
   * **Disadvantages**:
     + Starvation for long processes.
     + Requires prior knowledge of burst times.
3. **Priority Scheduling**
   * **Description**: Schedules processes based on priority levels.
   * **Advantages**:
     + High-priority tasks execute immediately.
   * **Disadvantages**:
     + Starvation risk for low-priority tasks.
     + Potential priority inversion.
4. **Round Robin (RR)**
   * **Description**: Allocates a fixed time quantum to processes in cyclic order.
   * **Advantages**:
     + Ensures fairness.
     + Suitable for interactive systems.
   * **Disadvantages**:
     + High context-switch overhead.
5. **Hybrid Scheduler**
   * **Description**: Combines Priority Scheduling for high-priority tasks and Round Robin for low-priority tasks, incorporating dynamic aging and adaptive time quantum.
   * **Advantages**:
     + Balances efficiency and fairness.
     + Prevents starvation.
     + Reduces context-switch overhead.
   * **Disadvantages**:
     + Slightly more complex to implement.

#### **Why the Hybrid Scheduler is Better**

The Hybrid Scheduler integrates strengths from traditional algorithms and addresses their limitations. Below is a detailed comparison:

1. **Comparison with FCFS**:
   * **Weakness**: High waiting times due to the convoy effect.
   * **Hybrid Advantage**: Prioritizes critical tasks and ensures fairness for all others.
2. **Comparison with SJF**:
   * **Weakness**: Starvation of long processes and dependency on burst time knowledge.
   * **Hybrid Advantage**: Incorporates dynamic aging to promote long-waiting processes.
3. **Comparison with Priority Scheduling**:
   * **Weakness**: Starvation of low-priority tasks.
   * **Hybrid Advantage**: Fairness is ensured through Round Robin for low-priority tasks.
4. **Comparison with Round Robin**:
   * **Weakness**: High context-switch overhead with fixed quantum.
   * **Hybrid Advantage**: Adapts quantum dynamically to workload characteristics.
5. **Dynamic Adaptability**:
   * Traditional algorithms are static.
   * Hybrid Scheduler adjusts to varying workloads using mechanisms like aging and adaptive quantum.

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### **Results**

The following table summarizes the performance metrics for all algorithms:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Metric** | **FCFS** | **SJF** | **Priority** | **RR** | **Hybrid** |
| **Fairness** | Poor | Poor | Moderate | Good | **Excellent** |
| **Waiting Time** | High | Low | Moderate | Moderate | **Low** |
| **Turnaround Time** | High | Low | Moderate | Moderate | **Low** |
| **Starvation Handling** | No | No | No | Yes | **Yes** |
| **Context Switching** | Low | Low | Low | High | **Moderate** |
| **Dynamic Adaptability** | No | No | No | No | **Yes** |

#### **Conclusion**

The Hybrid Scheduler demonstrates superior performance compared to traditional scheduling algorithms. By combining Priority Scheduling and Round Robin with features like dynamic aging and quantum adjustment, it ensures fairness, reduces waiting times, and improves CPU utilization. Its adaptability to varying workloads makes it the most effective solution for modern multitasking environments. Future work could explore integrating predictive models to further enhance scheduling efficiency.