Lotanna Akukwe

FEL Algorithm Simulation

Github Link

Assumptions

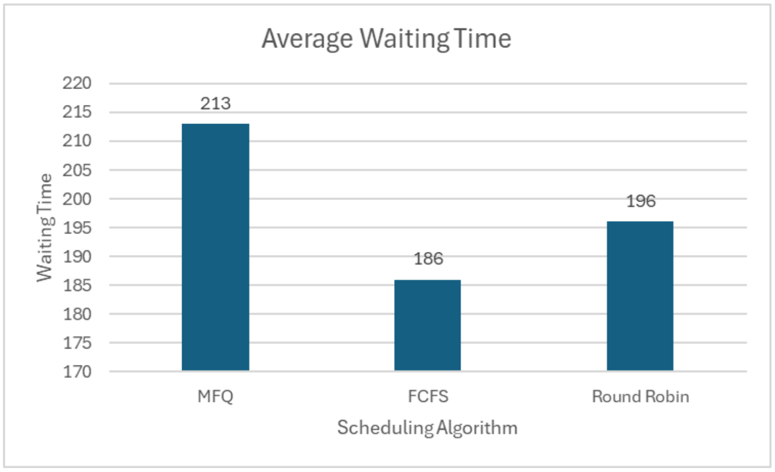
This event-based simulation holds different assumptions concerning timing, arrival policy, processor utilization, IO handling, scheduling algorithm, and more. The simulation is set to run for no more than 250ms, with a condition that frequently updates the current time based on the arrival time of the imminent event, despite whether it is departing or arriving. On the other hand, the arrival policy is determined by a fixed set of jobs that are predefined before the beginning of the simulation. This set holds the arrival and burst times for a specific job, and the simulation has 5 jobs. Additionally, another feature of the jobs is priority where the scheduling algorithm assigns jobs to processors based on its priority. Priority queue 0 determines the CPU burst time to be less than or equal to 20ms, priority queue 1 determines the job can stay in the CPU for a maximum of 40ms, while priority queue 2 determines that the job can complete its full burst time in the CPU without any interruptions. For instance, based on the priority (0-2), events will stay in the processor for the full duration of their burst time if the burst time is less than the priority queue's burst time or if it is of priority 2. Another assumption of the simulation is the handling of IO bursts in a job. It doesn't simulate an actual IO/blocked queue explicitly, assuming that the IO times are non-blocking and the jobs do not have to wait to be executed. Furthermore, there are two processors used in this simulation that have similar processing speeds. Finally, the simulation is built upon a Future Event List that manages all arrival and departure events that set the pace for the simulation.

Data Analysis

For further analysis of the data, I implemented 3 different scheduling algorithms, namely: Multilevel Feedback queue, First-come first-served, and round-robin. The MLFQ is such that jobs are assigned to multiple priority queues and are executed at different times based on their priority. Secondly, the FCFS had jobs executed in the order they arrived so that the CPU would always pick the first job in the single queue.

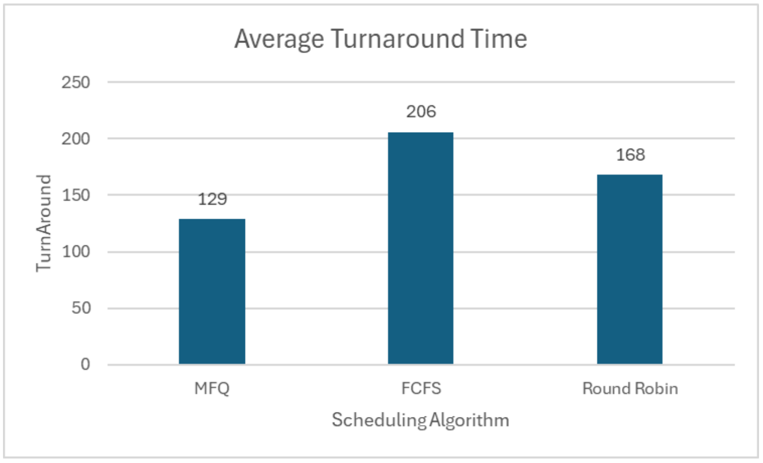
Lastly, the Round Robin executed jobs in a time-sliced manner. All this algorithm used a maximum simulation time of 250ms.

Average Waiting Time



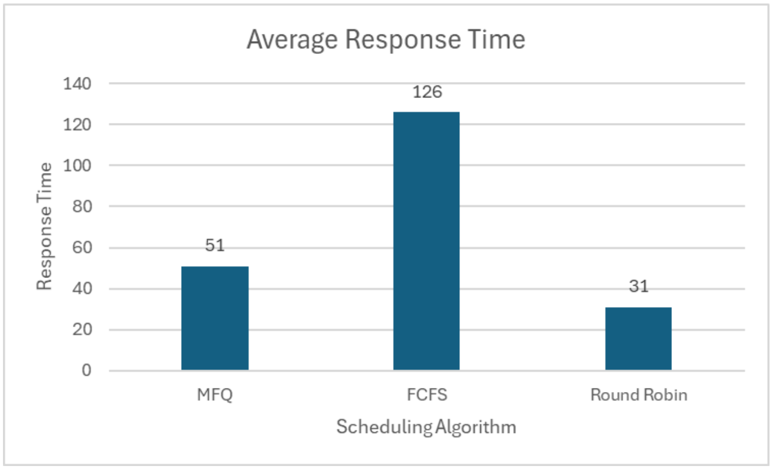
The chart above illustrates how the waiting time has changed according to different scheduling algorithms. The Multilevel Feedback Queue has the highest waiting time as the queue selection policies favor the jobs in the higher priority queues over the jobs in lower priority queues. This can lead to higher waiting time for those jobs in the lower-priority queues. Whereas, FCFS has the lowest average in waiting time because it doesn’t discriminate.

Average Turnaround Time



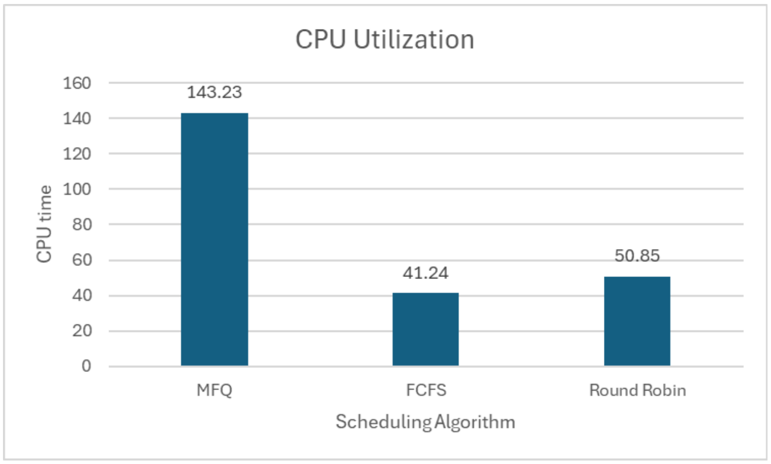
In the chart above, FCFS has the highest turnaround time for jobs followed closely by RoundRobin, with MLFQ having the least average in turnaround time.

Average Response Time



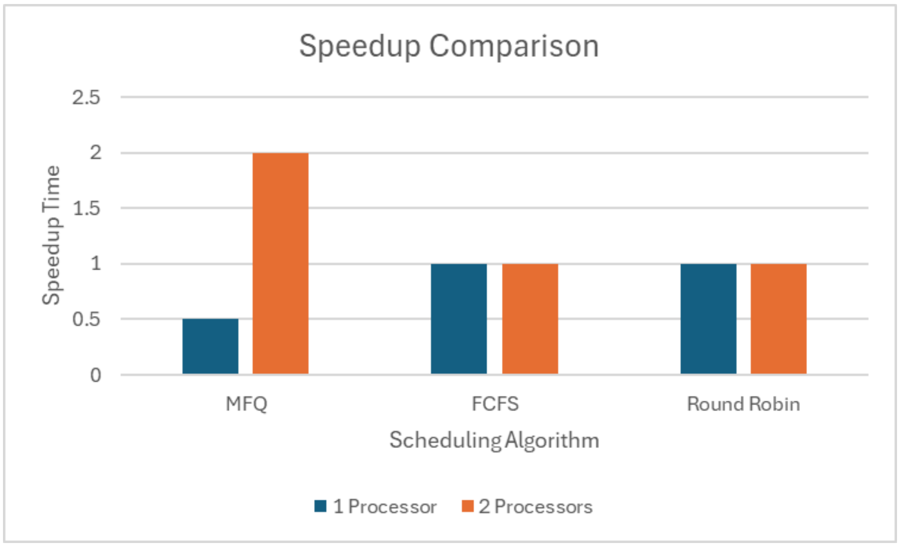
The chart above illustrates how the FCFS has the highest average in response time with more than a 50ms gap compared to MLFQ. It also shows that Round Robin has the lowest average which is a result of assigning a fixed time quantum of 30 ms to each process.

CPU Utilization



The chart above illustrates how the MLFQ has the highest CPU utilization among the scheduling algorithms with a utilization rate of 143.23%. It shows that the MLFQ can effectively utilize the CPUs to execute the events. However, exceeding 100% indicates that the simulation may be overcommitted and this is why it has longer wait times as shown in the waiting diagram. FCFS has a utilization rate of 41.24% suggesting that it cannot fully exploit the CPU processors, lowering the overall CPU utilization. Lastly, Round Robin has a utilization rate of 50.85% showing that it has moderate CPU utilization that strikes a balance between fairness and CPU efficiency. It ensures that the events receive regular CPU time without monopolizing the CPU processors.

SpeedUp Time



The MLFQ has a very high speedup time when utilizing two processors but significantly decreases with a single processor. It shows that the algorithm performance is better when multiprocessing. Meanwhile, FCFS and Round Robin show no improvement in speedup despite the change from one processor to two processors. It shows that their performance does not change significantly when switching to multiprocessing.