Networks and Operating System Essentials 2

Assessed Exercise 2 Report

Scheduling algorithms can be preemptive or non-preemptive in nature. Preemptive scheduling is employed when a process needs to be switched from waiting to ready state or running to ready state. The CPU allocates its resources to this scheduling kind in limited amounts and then takes it away. Shall there be remaining CPU burst time, the process is placed back in the queue. On the other hand, Non-preemptive scheduling is employed when a process wants to switch from running to waiting state. This scheduling makes the process hold onto CPU resources until completion or until the process is in waiting state. It waits for the process to fully utilize its CPU burst time and then allocates it to another process. Here's an insight into the Scheduling algorithms being used for this scheduling exercise:

1. **FCFS/FIFO (Non – Preemptive) –** The First Come First Served, or First In First Out scheduler works in a way in which a process is added to queue in the order of its arrival. The process added first in the queue is executed to completion before other processes hence, average time taken to implement all processes tend to get higher, but this scheduling algorithm is also the easiest in terms of implementation.

2. **SJF (Non – Preemptive) –** The Shortest Job First scheduler works in a way where all processes are added to a queue which is sorts the processes in ascending order of time taken, and then executes the process taking the shortest time to completion. Although this method can be unrealistic for large number of processes, it can be the most efficient in certain cases.

3. **RR (Preemptive) –** The Round Robin scheduler works much like FCFS however in a preemptive manner where the CPU is allotted specific amount of time for a process. It works based on time slicing. Processes are added to queue in the order they come in however they're executed only for the duration of the time slice, and not to completion. If the process is executed to completion before the time ends, then its successfully terminated otherwise the remaining process is added to the end of the queue.

4. **SRTF (Preemptive) -** The Shortest Remaining Time First scheduler works much like SJF, where the processes are added to the queue in ascending order of time taken. This also largely depends upon the remaining CPU burst time hence, the scheduler is called every time a new process arrives and make an attempt to execute it to completion, but if not, then it is added back to the queue much like RR.

- *Analysing Different Seed Numbers* -

1. Seed Number 1797410758

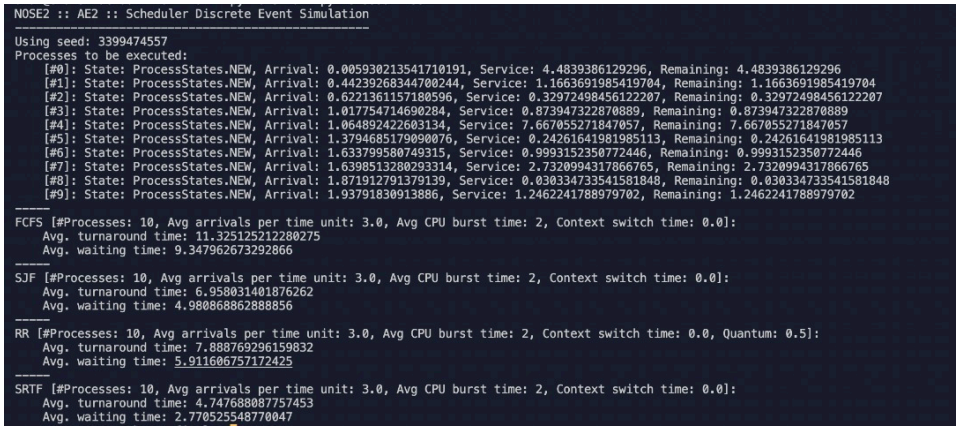
Continued on next page

From the given Terminal result, its visible that the SRTF scheduler has performed the best out of 4 schedulers, it's also performed significantly better than its preemptive half RR. One of the reasons for this is because Shortest Remaining Time First scheduler sorts the processes first which thereby results in lower wait times as all the processes taking less time are getting out of the way first without being held back by the longer processes. Round Robin's quantum time is greater as well which made it slower than SRTF

2. Seed Number 2688744162

From the given Terminal result it is clear that Shortest Job First and Shortest Remaining Time First schedulers did best. This is interesting because unlike the previous seed result where preemptive schedulers performed best, here we have one preemptive and one non-preemptive scheduler that performed better than its other halves. SRTF sorts the processes by remaining time thereby executing the quicker ones first, therefore significantly reducing the Average turnaround and Average waiting time. In the schedulers where the processes aren't sorted, First Come First Serve performed better than Round Robin. This is because the quantum time is greater than processes in RR which means that the RR scheduler would need to wait for the quantum time to terminate hence, take longer to move onto the next process. This would result in a much higher Average turnaround and Average waiting time.

3. Seed Number 3399474557



As visible in the Terminal results, this seed number's output follows a similar trend as seed 2688744162 where Shortest Job First and Shortest Remaining Time First schedulers performed best out of the 4. The thing changed is that the difference between average times among the 2 is greater than difference observed between SRTF and SJF in seed 2688744162. The commonality between SJF and SRTF is that both these schedulers order the queue

keeping in mind the time factor, both, remaining and time taken for service which outperformed the schedulers that did not follow the same prioritizing method- FCFS and RR. When SJF and SRTF prioritize the shorter tasks and get them out of the way, FCFS and RR would make easier tasks wait while processing heavier ones just because they came earlier. This makes the overall turnaround time significantly greater. RR scheduler has a much shorter average turnaround and average waiting time than FCFS as well because the size of processing is greater than the quantum.

To conclude, as we can see in all 3 executions of the different seed numbers, Shortest Remaining Time First scheduler performed the best consistently. It truly has the best of both worlds while being preemptive and efficiently sorting the list of processes before execution. First Come First Serve scheduler is the simplest to implement because it executes processes as they come, which makes it give the highest average turnaround and average waiting time regularly. The performance of Round Robin and Shortest Job First really depends upon the size of processes to be executed, SJF tends to perform better since it prioritises based on resources leading to an overall lower average turnaround and waiting time. However, RR can also do better because it is preemptive in nature, thereby making the average turnaround and waiting time shorter because it'll spend the same time executing all processes, and queue up the remaining unfinished processes.

- -