Loren Howard, Yujian Zhang, Ken Wakaba

Professor Ahmed Ezzat

COEN 383

13 October 2017

Process Scheduling Algorithms Report

First-Come-First-Serve is a non-preemptive policy that selects and runs a process based on FIFO. Although it is the easiest to implement, it does not perform well when it comes to the metrics we have discussed. With FCFS, longer processes are going to fare better than shorter ones. FCFS serves no problem if all processes run the same length of time and arrive at the same moment, but this is never the case. If a long process were to arrive before, short processes, the short processes would have to wait until the long one is done in its entirety. This is what is known as the Convoy Effect. Due to this, the wait time as well as the turnaround time of the processes within FCFS could be poor. Response time too will be high if we have a large discrepancy in process execution time.

To solve the problem of FCFS, another policy is used known as Shorted Job First (SJF). As the name suggest, SJF runs the next shortest job within the queue. It is still a non-preemptive job so it will allow all processes to complete in its entirety before choosing the next shortest job. Response time for SJF is excellent but only for short processes. Meaning that even with this priority in place, processes could still end up waiting a while to complete, as well as have potentially high turnaround time, although it is an improvement from FCFS.

Shortest Remaining Time (SRT) is much like SJF, except it is a preemptive policy. A job is chosen based on the shortest remaining run time of a process within the queue. When a new process comes int, it is compared with the remaining time of the current process and if it is less, the current job is stopped and the new job is run. Due to the preemptive policy, SRT has very similar response time to that of SPN, but it has more consistent shorter response times. Although it is a slight improvement, SRT as well as SJF still do not have particularly good response time. If multiple processes arrive at the same time, the processes on the tail end of the queue still must wait for all the others before to run completely before being scheduled. Although great for turnaround time, the wait time and response time are poor.

Round Robin, the solution to the poor response time problem with the above policies, is also one that uses preemption. Each process is run for a certain time, or quanta, and once the quanta is complete, the scheduler runs the next process in the queue. The shorter the quanta, the better the response time is for each process. While this policy is great for response time, it has awful wait time and turnaround time. This issue comes from the fact that RR only runs each process for a short amount of time before giving it to the next process. Having a small quantum also presents the problem of constant context switches, affecting the cost and performance. On the other hand, setting the quantum to be too long may cause poor response time, almost close to that of FCFS.

For Highest Priority First, the scheduler assigns a priority to each process, ranking them from highest priority to lowest. Depending on whether it is a preemptive or non-preemptive policy, the lower priority processes will get interrupted if a higher priority algorithm comes in. When looking at our metrics, higher priority processes have smaller wait times and response times than lower priority.

Based on the results we received from running our simulation, the concepts discussed above were confirmed. In terms of Response Time, RR performed the best with an average time of 1.800, as opposed to FCFS which had an average time of 10.938. These number make sense due to the fact that RR as a preemptive policy give each job a fair chance at the CPU. When looking at Turnaround, however, RR performed the worst with an average time of 28.650 as opposed to SRT with an average of 15.304. For the Wait Time, RR again performed the poorly with an average of 19.133compared to SRT at 6.448. Again, these values make sense because since RR gives every process a fair chance, each process must wait a significant amount of time before completion, even if it only needs a short amount of time with the CPU. SRT and SJF performed well when it came to turnaround and waiting time since they take into consideration the length of each job. When comparing them all together, SRT performed the best overall.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Response (ms)** | **Turnaround (ms)** | **Wait (ms)** | **Throughput (per quantum)** |
| **FCFS** | 10.938 | 20.581 | 10.938 | 0.0967 |
| **SJF** | 7.685 | 16.540 | 7.685 | 0.1036 |
| **SRT** | 5.402 | 15.304 | 6.448 | 0.1036 |
| **RR** | 1.800 | 28.650 | 19.133 | 0.0980 |
| **HPF (Non-P)** | 8.420 | 17.781 | 8.420 | 0.0995 |
| **HPF (P)** | 5.080 | 22.033 | 12.513 | 0.0981 |
| **HPF (Non-P) Aging** | 9.225 | 18.876 | 9.225 | 0.0969 |
| **HPF (P) Aging** | 4.390 | 27.247 | 17.510 | 0.0965 |