# Processor Scheduling Algorithms
# Homework #2
# June 28th, 2016

| Student | Completed |
|---|---|
| Tyler Jones (009645610) | X |
| Scot Matson (009602502) | X |
| Francisco McGee (008973445) | X |
| John Kennedy (001462826 | X |
| Daniel Tam (009238632) | X |

**INTRODUCTION**

Homework #2 requires creation of code to implement different process scheduling algorithms, measure their performance against randomly chosen input data, and use the data to determine strengths, weaknesses, and suitabilities for particular process workflows. Below are the results obtained and subsequent conclusions drawn.

The method of testing specified by the homework assignment gave guidelines for the testing process. A summary of those guidelines follows:

1) Generate a set of simulated processes to be scheduled by each algorithm. Each simulated process must have:
    a) An arrival time
    b) An expected run time
    c) A process priority
2) Test the following algorithms
    a) First Come First Served (FCFS)
    b) Shortest Job First (SJF)
    c) Shortest Remaining Time (SRT)
    d) Round robin (RR)
    e) Highest Priority First (HPF)
        i. HPF_non-preemptive (HPFN)
        ii. HPF_preemptive (HPFP)
        iii. HPFN-Aging
        iv. HPFP-Aging
3) Assume only one processing queue
4) Four priority queues must be used for HPFP
5) Calculate the following statistics
    a) Average turnaround time for the process
    b) Average waiting time for the process
    c) Average response time for the process
    d) Throughput for the algorithm

**RESULTS**

The test code was run 5 times for each algorithm, creating a new set of simulated processes each time. The Java code for the project is included in the bundle (.zip) with this report, along with output of the algorithm test runs. Java was chosen for this project due its object oriented nature, which lends itself to simulations of real processes and algorithms, as well as the team's familiarity with the language.

The tabularized summary of the runs and the calculated data is contained in Table 1 below, with best times highlighted in green.

Table 1

| | **Turnaround** | **Wait** | **Response** | **Throughput** |
|---|---|---|---|---|
| **FCFS** | 45.94271 | 40.207317 | 40.207317 | 0.15941638 |
| **SJF** | 8.483307 | 5.75 | 5.75 | 0.3016416 |
| **SRT** | 6.46689 | **4.690944** | 4.7090907 | 0.33207062 |
| **RR** | **1.0446482** | 31.502737 | **0.5489796** | 0.9861984 |
| **HPFN** | 27.80099 | 21.373333 | 21.373333 | 0.14368077 |
| **HPFP** | 1.4060634 | 5.189919 | 0.9178296 | **0.9947695** |
| **HPFN-Aging** | 45.69306 | 39.97561 | 39.97561 | 0.15983263 |
| **HPFP-Aging** | 1.4050694 | 5.324791 | 0.9868357 | **0.9947695** |

**ANALYSIS**

Our examination of this data yields the following conclusions:

1) Best Turnaround: RR

RR is very common in industry. By giving a fraction of compute time to each process at a time, the shorter ones get finished faster and so turnaround time is high. It makes sense that the HPFP and HPFP-Aging are close to RR for Turnaround because they are both based on RR as in keeping with the assignment instructions.

2) Best Wait Time: SRT

It makes sense that SRT has best wait time. Because SRT finishes the processes with the shortest remaining time first, then processes are spending less time waiting. High wait time makes sense with FCFS because it is simply going in order of arrival time, and so throughput is not optimized. FCFS has no way to optimize throughput because it does not try to find shorter processes and attend to them first. FCFS is followed by HPFN and HPFN-Aging because they are both based on FCFS, as per assignment instructions.

3) Best Response Time: RR

It makes sense that RR has the best response time. This is because it is attending to each process at a set interval in the queue, and so it reaches a newly added process faster than any of the other algorithms. Again, because both the HPFP and HPFP-Aging implementations are based on RR, it makes sense that they would have similar response times, but a little slower due to the priority-evaluation overhead.

4) Best Throughput: HPFP/HPFP-Aging/RR

In benchmarking tests, RR usually outperforms in throughput. This is partially because all the shortest jobs end up getting finished first evenly throughout the queue. The impact of

arrival time on process completion for shorter processes is minimized because RR moves through the queue at set intervals.

**CAUTION**: These results do not take into account high run time jobs. Although more jobs may be scheduled on the processor than would be in a First Come First Serve system, if the number of processes to be performed is high, and arrival times are less than the quanta, and the run times are greater than the quanta, then the chances of a process being completed decline. In such a scenario, FCFS may well be a better choice for a batch system.

**CONCLUSION**

In conclusion, we find the Round Robin algorithm to be the best overall choice due to its performance, ease of implementation, and simple management. Highest Priority First preemptive is the best choice for interactive systems, but requires more management and regular adjustment in order to yield optimal gains. As well, HPFP and HPFP-Aging performance are application-dependent; while throughput might ultimately be low, HPFP performance would be optimal if it is processing all the highest priority processes more quickly. It is up to the application or system setup to correctly prioritize the processes for HPFP to perform optimally.