Project 2
CS342 - Operating Systems
Bilkent University

Gokhan Simsek - 21401407

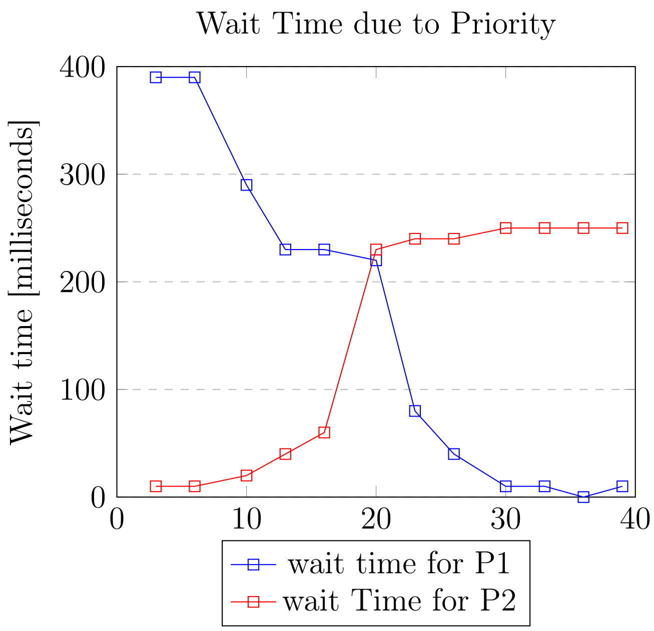
November 16, 2017

In this project, different experiments are run to measure different statistics using Completely Fair Scheduler. Two processes with the exact same bursts and start times, but with different priorities are created, and different statistics are measured.

In this experiment, I kept the priority of P1 fixed, at 20, and changed the priority of P2. The bursts and start times of both processes are exactly the same. I will measure wait time, response time and turnaround time differences.

0.1 Wait Time

```
Example experiment, with
   prios 20 and 30:

1 start 0 prio 20
1 cpu 30
1 io 100
1 cpu 50
1 io 20
1 cpu 200
1 end
2 start 0 prio 30
2 cpu 30
2 io 100
2 cpu 50
2 io 20
2 cpu 200
2 end
```

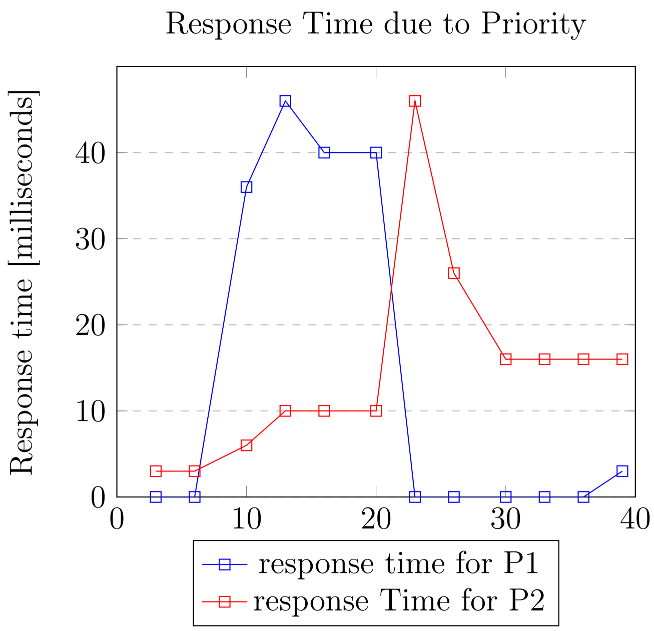


We can see from the plot the spike that happens when priority of P2 changes from 16 to 20. This is the overtaking step. Timeslices and vruntimes are calculated with respect to the weight of the process and the total load. i.e.
$timeslice = targeted\_latency * load\_of\_process/total\_load$ and
$vruntime+ = t * nice\_load/load\_of\_process$.
So, timeslices are calculated relative to each other. When a process gets the higher priority (smaller priority coefficient), it becomes more prevalent and runs in the CPU more than the other for a given targeted latency. Thus, when $priority\_of\_P2 < 20$, P2 has the upper hand and runs more. When the priorities get closer, a balance is

reached, and each process gets to run similar amounts in a given targeted latency. This is the reason for the spike between 16 and 20.

0.2 Response Time

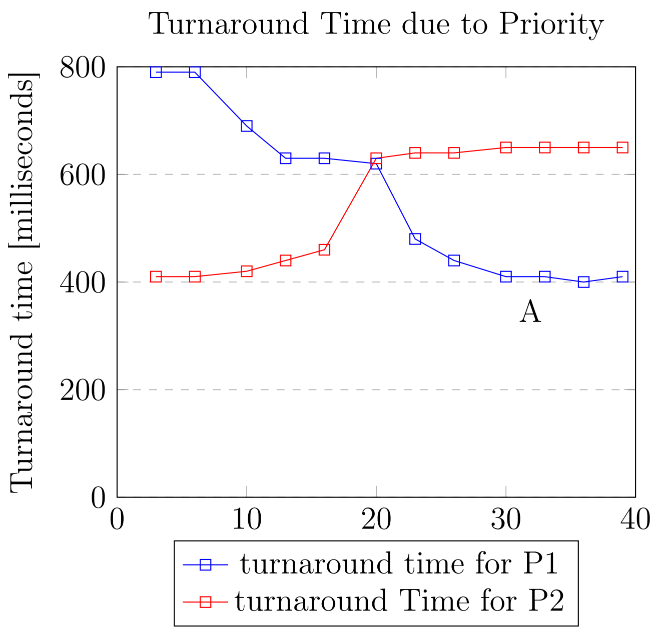
```
P1,P2:
Experiment Results
<pid> <prio> <starttime>
   <finishtime> <turnaround>
   <waittime> <responsetime>

20 3
1 20 0 790 790 390 0
2 3 0 410 410 10 3

20 6
1 20 0 790 790 390 0
2 6 0 410 410 10 3

20 10
1 20 0 690 690 290 36
2 10 0 420 420 20 6

20 13
1 20 0 630 630 230 46
2 13 0 440 440 40 10

20 16
1 20 0 630 630 230 40
2 16 0 460 460 60 10

20 20
1 20 0 620 620 220 0
2 20 0 630 630 230 10
```



A similar spike can be seen in this graph as well, perhaps more dramatically. The interesting thing in this graph is the behavior of the response time of P1. This is due to P1 having the smaller process ID, i.e. 1, compared to P2. When they come out of I/O, their vruntimes are the same so the processor breaks the tie by selecting the process with the smaller ID.

0.3 Turnaround Time

```
P1,P2:
Experiment Results
<pid> <prio> <starttime>
   <finishtime> <turnaround>
   <waittime> <responsetime>

20 23
1 20 0 480 480 80 0
2 23 0 640 640 240 46

20 26
1 20 0 440 440 40 0
2 26 0 640 640 240 26

20 30
1 20 0 410 410 10 0
2 30 0 650 650 250 16

20 33
1 20 0 410 410 10 0
2 33 0 650 650 250 16

20 36
1 20 0 400 400 0 0
2 36 0 650 650 250 16

20 39
1 20 0 410 410 10 3
2 39 0 650 650 250 16
```



similar change occurs in this comparison as well. As P2 gets closer to 20, P1 starts taking over. This results in lower and lower turnaround times for P1.

As a general discussion for the plots above, the reason that they are not perfectly symmetric is the difference between their IDs. When all else is the same, the implemented CFS prefers processes with smaller IDs. Thus, the two processes have small differences.