John Anderson

CS 149 Section 3

HW #2

Part 2:

Problem 1:

P0 = red P1 = blue p2 = green p3 = yellow p4 = purple

A) FIFO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P0 | p1 | P2 | P3 | P4 |

0s 80s 105s 120s 145s 155s

B) SJF (non- preemptive)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P0 | P2 | P4 | P1 | P3 |

0 80s 95s 105s 130s 155s

C) RR (1 bursts)

P0 =0s – 155 p1= 15 -- 95 p2=15s – 60s p3=85 -- 141 p4= 90 – 124

(I think, if the bursts in the RR work with adding at the end)

D)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | P1 | P0 | P3 | P4 | P3 | P0 | P2 |

0 15 40 85 90 100 120 140 155

E) (non-preemptive)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P0 | P2 | P1 | P4 | P3 |

0 80 95 120 130 155

Problem 2:

A i. efficiency is = T / (T / S)

ii. Something like this I think. = S/ (t + S)

iii. If q approaches 0 then T is big enough that there is no effect compared to the context switching. So efficiency would be Q/(Q+S)

B. IF the arrival rate is Half of the service rate then the utilization would be 50%.

Problem 3:

Use littles law N = XR, N = 5, X = 1000 per second, R = N/X

So average Response time = 5 milliseconds

Problem 4:

Yes, as long as bounded average response time fits correctly creating the correct amount of arrival rate and service rate needed for the utilization to be 100%. Also as long as the service time keeps it together using the R = (S/(1-U)) formula

Problem 5:

The response time for the task to complete is the (que time + task time). So No matter how big or small the task is. It still has to wait for the tasks in the que before it even starts to execute itself. So, having a variance service time will make the response time every scattered because the que time will be constently changing as well.

Problem 6:

The Single que FIFO would be more optimal then the FIFO que per server, because when a large task comes to the single que it will be sent to one of the servers, but the other servers still are working on the other tasks in the que. When its FIFO per server having a large task will complete stop one of the FIFO ques and make the average sky rocket because the que for the single FIFO server will stop and wait for this long task to finish.

Problem 7:

Worst cause would be like

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P0 = 100s | P1 = 50s | P2 = 25s | P3 = 2s | P4 = 1s |

Then because It is running FIFO it will have to run the larger ones before running the really fast ones. Making even the FIFO worst case is if the longest causes where done at the start.

Problem 8:

If you do your HW in SJF you can finish all of the short problems first and think that the homework assignment fast and under estimate how long it will take, because you haven’t hit the problems that could take longer than a couple minutes to complete.

Problem 9:

RR : This algorithm would be best used when all tasks need to be completed with an average response time. Like in a streaming service because they want to split the tasks usage, so each user gets the same quality use.

SJF: This would be better to be used for a server that runs a game that needs to check a lot of small things often and larger tasks can be ignored for a while and their response time is not important compared to small tasks. But also it has to be able to be able to deal with some starvation for the large tasks.