

HW #6 Solutions

1. 3 flows, f_1 , f_2 , and f_3 all pass through the same router R . Packets from flows f_1 and f_2 arrive at R in bursts of 100 with a period of one second, packets from f_3 arrive at the router one at a time with a period of .01 seconds. Packets from all flows contain the the same number of bytes). Suppose packets from f_1 start arriving at R at time $t = 0$, flow f_2 at time $t = 5$, and flow f_3 at time $t = 0.5$. If each flow transmits a total of 1000 packets, and the router processes a packet every 0.005 seconds, what are the completion times of the various flows when

- (a) R uses FIFO (FCFS) queueing?

Answer:

Time (sec)	Buffered Packets	Processed Packets
0.0	100 (f_1)	0 (f_1) , 0 (f_3) , 0 (f_2)
0.5	1 (f_3)	100 (f_1) , 0 (f_3) , 0 (f_2)
1.0	100 (f_1), 1 (f_3)	100 (f_1) , 50 (f_3) , 0 (f_2)
1.5	50 (f_3)	200 (f_1) , 50 (f_3) , 0 (f_2)
2.0	100 (f_1), 1 (f_3)	200 (f_1) , 150 (f_3) , 0 (f_2)
2.5	50 (f_3)	300 (f_1) , 150 (f_3) , 0 (f_2)
3.0	100 (f_1), 1 (f_3)	300 (f_1) , 250 (f_3) , 0 (f_2)
3.5	50 (f_3)	400 (f_1) , 250 (f_3) , 0 (f_2)
4.0	100 (f_1), 1 (f_3)	400 (f_1) , 350 (f_3) , 0 (f_2)
4.5	50 (f_3)	500 (f_1) , 350 (f_3) , 0 (f_2)
5.0	100 (f_1), 100 (f_2), 1 (f_3)	500 (f_1) , 450 (f_3) , 0 (f_2)
5.5	100 (f_2), 50 (f_3)	600 (f_1) , 450 (f_3) , 0 (f_2)
6.0	100 (f_3), 100 (f_1), 100 (f_2), 1 (f_3)	600 (f_1) , 450 (f_3) , 100 (f_2)
6.5	100 (f_1), 100 (f_2), 50 (f_3)	600 (f_1) , 550 (f_3) , 100 (f_2)
7.0	100 (f_2), 100 (f_3), 100 (f_1), 100 (f_2), 1 (f_3)	700 (f_1) , 550 (f_3) , 100 (f_2)
7.5	100 (f_3), 100 (f_1), 100 (f_2), 50 (f_3)	700 (f_1) , 550 (f_3) , 200 (f_2)
8.0	100 (f_1), 100 (f_2), 100 (f_3), 100 (f_1), 100 (f_2), 1 (f_3)	700 (f_1) , 650 (f_3) , 200 (f_2)
8.5	100 (f_2), 100 (f_3), 100 (f_1), 100 (f_2), 50 (f_3)	800 (f_1) , 650 (f_3) , 200 (f_2)
9.0	100 (f_3), 100 (f_1), 100 (f_2), 100 (f_3), 100 (f_1), 100 (f_2), 1 (f_3)	800 (f_1) , 650 (f_3) , 300 (f_2)
9.5	100 (f_1), 100 (f_2), 100 (f_3), 100 (f_1), 100 (f_2), 50 (f_3)	800 (f_1) , 750 (f_3) , 300 (f_2)
10.0	100 (f_2), 100 (f_3), 100 (f_1), 100 (f_2), 100 (f_3), 100 (f_2), 1 (f_3)	900 (f_1) , 750 (f_3) , 300 (f_2)
10.5	100 (f_3), 100 (f_1), 100 (f_2), 100 (f_3), 100 (f_2), 50 (f_3)	900 (f_1) , 750 (f_3) , 400 (f_2)
11.0	100 (f_1), 100 (f_2), 50 (f_3), 100 (f_2), 50 (f_3), 100 (f_2)	900 (f_1) , 850 (f_3) , 400 (f_2)
11.5	100 (f_2), 100 (f_3), 100 (f_2), 50 (f_3), 100 (f_2)	1000 (f_1) , 850 (f_3) , 400 (f_2)
12.0	100 (f_3), 100 (f_2), 50 (f_3), 100 (f_2), 100 (f_2)	1000 (f_1) , 850 (f_3) , 500 (f_2)
12.5	100 (f_2), 50 (f_3), 100 (f_2), 100 (f_2)	1000 (f_1) , 950 (f_3) , 500 (f_2)
13.0	50 (f_3), 100 (f_2), 100 (f_2), 100 (f_2)	1000 (f_1) , 950 (f_3) , 600 (f_2)
13.5	50 (f_2), 100 (f_2), 100 (f_2)	1000 (f_1) , 1000 (f_3) , 650 (f_2)
14.0	50 (f_2), 100 (f_2), 100 (f_2)	1000 (f_1) , 1000 (f_3) , 750 (f_2)
15.0	50 (f_2)	1000 (f_1) , 1000 (f_3) , 950 (f_2)
15.25	End of processing the flows	1000 (f_1) , 1000 (f_3) , 1000 (f_2)

For all the following exercise we assume that when packets arrive at the same time they arrive in order f_1, f_2, f_3 (i.e packets from f_1 arrive first and packets from f_3 arrive last. This assumption doesn't affect the solution significantly as similar results are obtained if we assume another ordering in arrival times. For the FIFO(First Come First Serve) queueing we have to compute the times of arrival for the packets for the different flows. Then we have to compute the packets that are ahead the last packet of flow f_1, f_2, f_3 . We will compute the processing time after subtracting the time that the router was idle if any. Also we have to remember that our processing power is $(1/0.005)(\text{packets/sec}) = 200$ packets/sec. We have completions times of 11.5 sec , 15.25 sec and 13.25 sec for f_1, f_2 and f_3 respectively.

(b) R uses round robin?

Answer:

Time (sec)	Buffered Packets	Processed Packets
0.0	100 (f_1)	0 (f_1) , 0 (f_3) , 0 (f_2)
0.25	50 (f_1), 0 (f_2), 0 (f_3)	50 (f_1), 0 (f_2), 0 (f_3)
0.50	0 (f_1), 0 (f_2), 0 (f_3)	100 (f_1), 0 (f_2), 0 (f_3)
0.75	0 (f_1), 0 (f_2), 0 (f_3)	100 (f_1), 0 (f_2), 25 (f_3)
1.00	0 (f_1), 0 (f_2), 0 (f_3)	100 (f_1), 0 (f_2), 50 (f_3)
1.25	75 (f_1), 0 (f_2), 0 (f_3)	125 (f_1), 0 (f_2), 75 (f_3)
1.50	50 (f_1), 0 (f_2), 0 (f_3)	150 (f_1), 0 (f_2), 100 (f_3)
1.75	25 (f_1), 0 (f_2), 0 (f_3)	175 (f_1), 0 (f_2), 125 (f_3)
2.00	0 (f_1), 0 (f_2), 0 (f_3)	200 (f_1), 0 (f_2), 150 (f_3)
2.25	75 (f_1), 0 (f_2), 0 (f_3)	225 (f_1), 0 (f_2), 175 (f_3)
2.50	50 (f_1), 0 (f_2), 0 (f_3)	250 (f_1), 0 (f_2), 200 (f_3)
2.75	25 (f_1), 0 (f_2), 0 (f_3)	275 (f_1), 0 (f_2), 225 (f_3)
3.00	0 (f_1), 0 (f_2), 0 (f_3)	300 (f_1), 0 (f_2), 250 (f_3)
3.25	75 (f_1), 0 (f_2), 0 (f_3)	325 (f_1), 0 (f_2), 275 (f_3)
3.50	50 (f_1), 0 (f_2), 0 (f_3)	350 (f_1), 0 (f_2), 300 (f_3)
3.75	25 (f_1), 0 (f_2), 0 (f_3)	375 (f_1), 0 (f_2), 325 (f_3)
4.00	0 (f_1), 0 (f_2), 0 (f_3)	400 (f_1), 0 (f_2), 350 (f_3)
4.25	75 (f_1), 0 (f_2), 0 (f_3)	425 (f_1), 0 (f_2), 375 (f_3)
4.50	50 (f_1), 0 (f_2), 0 (f_3)	450 (f_1), 0 (f_2), 400 (f_3)
4.75	25 (f_1), 0 (f_2), 0 (f_3)	475 (f_1), 0 (f_2), 425 (f_3)
5.00	0 (f_1), 0 (f_2), 0 (f_3)	500 (f_1), 0 (f_2), 450 (f_3)
5.25	83 (f_1), 83 (f_2), 9 (f_3)	517 (f_1), 17 (f_2), 466 (f_3)
5.50	66 (f_1), 67 (f_2), 17 (f_3)	534 (f_1), 33 (f_2), 483 (f_3)
5.75	50 (f_1), 50 (f_2), 25 (f_3)	550 (f_1), 50 (f_2), 500 (f_3)
6.00	33 (f_1), 33 (f_2), 34 (f_3)	567 (f_1), 67 (f_2), 516 (f_3)
6.25	116 (f_1), 117 (f_2), 42 (f_3)	584 (f_1), 83 (f_2), 533 (f_3)
6.50	100 (f_1), 100 (f_2), 50 (f_3)	600 (f_1), 100 (f_2), 550 (f_3)
6.75	83 (f_1), 83 (f_2), 59 (f_3)	617 (f_1), 117 (f_2), 566 (f_3)
7.00	66 (f_1), 67 (f_2), 67 (f_3)	634 (f_1), 133 (f_2), 583 (f_3)
7.25	150 (f_1), 150 (f_2), 75 (f_3)	650 (f_1), 150 (f_2), 600 (f_3)
7.50	133 (f_1), 133 (f_2), 84 (f_3)	667 (f_1), 167 (f_2), 616 (f_3)

Time (sec)	Buffered Packets	Processed Packets
7.50	133 (f_1), 133 (f_2), 84 (f_3)	667 (f_1), 167 (f_2), 616 (f_3)
7.75	116 (f_1), 117 (f_2), 92 (f_3)	684 (f_1), 183 (f_2), 633 (f_3)
8.00	100 (f_1), 100 (f_2), 100 (f_3)	700 (f_1), 200 (f_2), 650 (f_3)
8.25	183 (f_1), 183 (f_2), 109 (f_3)	717 (f_1), 217 (f_2), 666 (f_3)
8.50	166 (f_1), 167 (f_2), 117 (f_3)	734 (f_1), 233 (f_2), 683 (f_3)
8.75	150 (f_1), 150 (f_2), 125 (f_3)	750 (f_1), 250 (f_2), 700 (f_3)
9.00	133 (f_1), 133 (f_2), 134 (f_3)	767 (f_1), 267 (f_2), 716 (f_3)
9.25	216 (f_1), 217 (f_2), 142 (f_3)	784 (f_1), 283 (f_2), 733 (f_3)
9.50	200 (f_1), 200 (f_2), 150 (f_3)	800 (f_1), 300 (f_2), 750 (f_3)
9.75	183 (f_1), 183 (f_2), 159 (f_3)	817 (f_1), 317 (f_2), 766 (f_3)
10.00	166 (f_1), 167 (f_2), 167 (f_3)	834 (f_1), 333 (f_2), 783 (f_3)
10.25	150 (f_1), 250 (f_2), 175 (f_3)	850 (f_1), 350 (f_2), 800 (f_3)
10.50	133 (f_1), 233 (f_2), 184 (f_3)	867 (f_1), 367 (f_2), 816 (f_3)
10.75	116 (f_1), 217 (f_2), 167 (f_3)	884 (f_1), 383 (f_2), 833 (f_3)
11.00	100 (f_1), 200 (f_2), 150 (f_3)	900 (f_1), 400 (f_2), 850 (f_3)
11.25	83 (f_1), 283 (f_2), 134 (f_3)	917 (f_1), 417 (f_2), 866 (f_3)
11.50	66 (f_1), 267 (f_2), 117 (f_3)	934 (f_1), 433 (f_2), 883 (f_3)
11.75	50 (f_1), 250 (f_2), 100 (f_3)	950 (f_1), 450 (f_2), 900 (f_3)
12.00	33 (f_1), 233 (f_2), 84 (f_3)	967 (f_1), 467 (f_2), 916 (f_3)
12.25	16 (f_1), 317 (f_2), 67 (f_3)	984 (f_1), 483 (f_2), 933 (f_3)
12.50	0 (f_1), 300 (f_2), 50 (f_3)	1000 (f_1), 500 (f_2), 950 (f_3)
12.75	0 (f_1), 275 (f_2), 25 (f_3)	1000 (f_1), 525 (f_2), 975 (f_3)
13.00	0 (f_1), 250 (f_2), 0 (f_3)	1000 (f_1), 550 (f_2), 1000 (f_3)
13.25	0 (f_1), 300 (f_2), 0 (f_3)	1000 (f_1), 600 (f_2), 1000 (f_3)
13.50	0 (f_1), 250 (f_2), 0 (f_3)	1000 (f_1), 650 (f_2), 1000 (f_3)
13.75	0 (f_1), 200 (f_2), 0 (f_3)	1000 (f_1), 700 (f_2), 1000 (f_3)
14.00	0 (f_1), 150 (f_2), 0 (f_3)	1000 (f_1), 750 (f_2), 1000 (f_3)
14.25	0 (f_1), 200 (f_2), 0 (f_3)	1000 (f_1), 800 (f_2), 1000 (f_3)
14.50	0 (f_1), 150 (f_2), 0 (f_3)	1000 (f_1), 850 (f_2), 1000 (f_3)
14.75	0 (f_1), 100 (f_2), 0 (f_3)	1000 (f_1), 900 (f_2), 1000 (f_3)
15.00	0 (f_1), 50 (f_2), 0 (f_3)	1000 (f_1), 950 (f_2), 1000 (f_3)
15.25	0 (f_1), 0 (f_2), 0 (f_3)	1000 (f_1), 1000 (f_2), 1000 (f_3)

In Round Robin we have equal share amount of processing power among all non-empty buffers. In other worlds, if some buffers are empty the processing power is shared between the rest non-empty buffers equally. f_1 time of completion is: 12.490 sec , f_2 time of completion is: 15.250 sec, f_3 time of completion is: 13.000 sec.

(c) R uses virtual clock?

Answer:

Time (sec)	Buffered Packets	Processed Packets
0.0	100 (f_1)	0 (f_1) , 0 (f_3) , 0 (f_2)
0.25	50 (f_1), 0 (f_2), 0 (f_3)	50 (f_1), 0 (f_2), 0 (f_3)
0.50	0 (f_1), 0 (f_2), 0 (f_3)	100 (f_1), 0 (f_2), 0 (f_3)
0.75	0 (f_1), 0 (f_2), 0 (f_3)	100 (f_1), 0 (f_2), 25 (f_3)
1.00	0 (f_1), 0 (f_2), 0 (f_3)	100 (f_1), 0 (f_2), 50 (f_3)
1.25	75 (f_1), 0 (f_2), 0 (f_3)	125 (f_1), 0 (f_2), 75 (f_3)
1.50	50 (f_1), 0 (f_2), 0 (f_3)	150 (f_1), 0 (f_2), 100 (f_3)
1.75	25 (f_1), 0 (f_2), 0 (f_3)	175 (f_1), 0 (f_2), 125 (f_3)
2.00	0 (f_1), 0 (f_2), 0 (f_3)	200 (f_1), 0 (f_2), 150 (f_3)
2.25	75 (f_1), 0 (f_2), 0 (f_3)	225 (f_1), 0 (f_2), 175 (f_3)
2.50	50 (f_1), 0 (f_2), 0 (f_3)	250 (f_1), 0 (f_2), 200 (f_3)
2.75	25 (f_1), 0 (f_2), 0 (f_3)	275 (f_1), 0 (f_2), 225 (f_3)
3.00	0 (f_1), 0 (f_2), 0 (f_3)	300 (f_1), 0 (f_2), 250 (f_3)
3.25	75 (f_1), 0 (f_2), 0 (f_3)	325 (f_1), 0 (f_2), 275 (f_3)
3.50	50 (f_1), 0 (f_2), 0 (f_3)	350 (f_1), 0 (f_2), 300 (f_3)
3.75	25 (f_1), 0 (f_2), 0 (f_3)	375 (f_1), 0 (f_2), 325 (f_3)
4.00	0 (f_1), 0 (f_2), 0 (f_3)	400 (f_1), 0 (f_2), 350 (f_3)
4.25	75 (f_1), 0 (f_2), 0 (f_3)	425 (f_1), 0 (f_2), 375 (f_3)
4.50	50 (f_1), 0 (f_2), 0 (f_3)	450 (f_1), 0 (f_2), 400 (f_3)
4.75	25 (f_1), 0 (f_2), 0 (f_3)	475 (f_1), 0 (f_2), 425 (f_3)
5.00	0 (f_1), 0 (f_2), 0 (f_3)	500 (f_1), 0 (f_2), 450 (f_3)
5.25	100 (f_1), 50 (f_2), 25 (f_3)	500 (f_1), 50 (f_2), 450 (f_3)
5.50	100 (f_1), 0 (f_2), 50 (f_3)	500 (f_1), 100 (f_2), 450 (f_3)
5.75	100 (f_1), 0 (f_2), 25 (f_3)	500 (f_1), 100 (f_2), 500 (f_3)
6.00	100 (f_1), 0 (f_2), 0 (f_3)	500 (f_1), 100 (f_2), 550 (f_3)
6.25	200 (f_1), 50 (f_2), 25 (f_3)	500 (f_1), 150 (f_2), 550 (f_3)
6.50	200 (f_1), 0 (f_2), 50 (f_3)	500 (f_1), 200 (f_2), 550 (f_3)
6.75	200 (f_1), 0 (f_2), 25 (f_3)	500 (f_1), 200 (f_2), 600 (f_3)
7.00	200 (f_1), 0 (f_2), 0 (f_3)	500 (f_1), 200 (f_2), 650 (f_3)
7.25	300 (f_1), 50 (f_2), 25 (f_3)	500 (f_1), 250 (f_2), 650 (f_3)
7.50	300 (f_1), 0 (f_2), 50 (f_3)	500 (f_1), 300 (f_2), 650 (f_3)
7.75	300 (f_1), 0 (f_2), 25 (f_3)	500 (f_1), 300 (f_2), 700 (f_3)
8.00	300 (f_1), 0 (f_2), 0 (f_3)	500 (f_1), 300 (f_2), 750 (f_3)
8.25	400 (f_1), 50 (f_2), 25 (f_3)	500 (f_1), 350 (f_2), 750 (f_3)
8.50	400 (f_1), 0 (f_2), 50 (f_3)	500 (f_1), 400 (f_2), 750 (f_3)
8.75	400 (f_1), 0 (f_2), 25 (f_3)	500 (f_1), 400 (f_2), 800 (f_3)
9.00	400 (f_1), 0 (f_2), 0 (f_3)	500 (f_1), 400 (f_2), 850 (f_3)
9.25	500 (f_1), 50 (f_2), 25 (f_3)	500 (f_1), 450 (f_2), 850 (f_3)
9.50	500 (f_1), 0 (f_2), 50 (f_3)	500 (f_1), 500 (f_2), 850 (f_3)
9.75	499 (f_1), 0 (f_2), 26 (f_3)	501 (f_1), 500 (f_2), 899 (f_3)
10.00	499 (f_1), 0 (f_2), 1 (f_3)	501 (f_1), 500 (f_2), 949 (f_3)

Time (sec)	Buffered Packets	Processed Packets
10.00	499 (f_1), 0 (f_2), 1 (f_3)	501 (f_1), 500 (f_2), 949 (f_3)
10.25	474 (f_1), 75 (f_2), 26 (f_3)	526 (f_1), 525 (f_2), 949 (f_3)
10.50	449 (f_1), 50 (f_2), 51 (f_3)	551 (f_1), 550 (f_2), 949 (f_3)
10.75	424 (f_1), 25 (f_2), 51 (f_3)	576 (f_1), 575 (f_2), 949 (f_3)
11.00	399 (f_1), 0 (f_2), 51 (f_3)	601 (f_1), 600 (f_2), 949 (f_3)
11.25	374 (f_1), 75 (f_2), 51 (f_3)	626 (f_1), 625 (f_2), 949 (f_3)
11.50	349 (f_1), 50 (f_2), 51 (f_3)	651 (f_1), 650 (f_2), 949 (f_3)
11.75	324 (f_1), 25 (f_2), 51 (f_3)	676 (f_1), 675 (f_2), 949 (f_3)
12.00	299 (f_1), 0 (f_2), 51 (f_3)	701 (f_1), 700 (f_2), 949 (f_3)
12.25	274 (f_1), 75 (f_2), 51 (f_3)	726 (f_1), 725 (f_2), 949 (f_3)
12.50	249 (f_1), 50 (f_2), 51 (f_3)	751 (f_1), 750 (f_2), 949 (f_3)
12.75	224 (f_1), 25 (f_2), 51 (f_3)	776 (f_1), 775 (f_2), 949 (f_3)
13.00	199 (f_1), 0 (f_2), 51 (f_3)	801 (f_1), 800 (f_2), 949 (f_3)
13.25	174 (f_1), 75 (f_2), 51 (f_3)	826 (f_1), 825 (f_2), 949 (f_3)
13.50	149 (f_1), 50 (f_2), 51 (f_3)	851 (f_1), 850 (f_2), 949 (f_3)
13.75	124 (f_1), 25 (f_2), 51 (f_3)	876 (f_1), 875 (f_2), 949 (f_3)
14.00	99 (f_1), 0 (f_2), 51 (f_3)	901 (f_1), 900 (f_2), 949 (f_3)
14.25	74 (f_1), 75 (f_2), 51 (f_3)	926 (f_1), 925 (f_2), 949 (f_3)
14.50	50 (f_1), 50 (f_2), 50 (f_3)	950 (f_1), 950 (f_2), 950 (f_3)
14.75	33 (f_1), 33 (f_2), 34 (f_3)	967 (f_1), 967 (f_2), 966 (f_3)
15.00	16 (f_1), 17 (f_2), 17 (f_3)	984 (f_1), 983 (f_2), 983 (f_3)
15.25	0 (f_1), 0 (f_2), 0 (f_3)	1000 (f_1), 1000 (f_2), 1000 (f_3)

f_1 time of completion is: 15.240 sec, f_2 time of completion is: 15.245 sec, f_3 time of completion is: 15.250 sec.

- (d) R uses weighted fair queueing where the slack (i.e., the maximum reserve that a flow can build up) is 200 packets?

Answer:

Time (sec)	Buffered Packets	Processed Packets	Processed Packets(with slack of 200)
0.0	100 (f_1)	0 (f_1), 0 (f_3), 0 (f_2)	0 (f_1), 0 (f_3), 0 (f_2)
0.25	50 (f_1), 0 (f_2), 0 (f_3)	50 (f_1), 0 (f_2), 0 (f_3)	50 (f_1), 0 (f_2), 0 (f_3)
0.50	0 (f_1), 0 (f_2), 0 (f_3)	100 (f_1), 0 (f_2), 0 (f_3)	100 (f_1), 0 (f_2), 0 (f_3)
0.75	0 (f_1), 0 (f_2), 0 (f_3)	100 (f_1), 0 (f_2), 25 (f_3)	100 (f_1), 0 (f_2), 25 (f_3)
1.00	0 (f_1), 0 (f_2), 0 (f_3)	100 (f_1), 0 (f_2), 50 (f_3)	100 (f_1), 0 (f_2), 50 (f_3)
1.25	75 (f_1), 0 (f_2), 0 (f_3)	125 (f_1), 0 (f_2), 75 (f_3)	125 (f_1), 0 (f_2), 75 (f_3)
1.50	50 (f_1), 0 (f_2), 0 (f_3)	150 (f_1), 0 (f_2), 100 (f_3)	150 (f_1), 0 (f_2), 100 (f_3)
1.75	25 (f_1), 0 (f_2), 0 (f_3)	175 (f_1), 0 (f_2), 125 (f_3)	175 (f_1), 0 (f_2), 125 (f_3)
2.00	0 (f_1), 0 (f_2), 0 (f_3)	200 (f_1), 0 (f_2), 150 (f_3)	200 (f_1), 0 (f_2), 150 (f_3)

Time (sec)	Buffered Packets	Processed Packets	Processed Packets(with slack of 200)
2.00	0 (f_1), 0 (f_2), 0 (f_3)	200 (f_1), 0 (f_2), 150 (f_3)	200 (f_1), 0 (f_2), 150 (f_3)
2.25	75 (f_1), 0 (f_2), 0 (f_3)	225 (f_1), 0 (f_2), 175 (f_3)	200 (f_1), 0 (f_2), 175 (f_3)
2.50	50 (f_1), 0 (f_2), 0 (f_3)	250 (f_1), 0 (f_2), 200 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
2.75	25 (f_1), 0 (f_2), 0 (f_3)	275 (f_1), 0 (f_2), 225 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
3.00	0 (f_1), 0 (f_2), 0 (f_3)	300 (f_1), 0 (f_2), 250 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
3.25	75 (f_1), 0 (f_2), 0 (f_3)	325 (f_1), 0 (f_2), 275 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
3.50	50 (f_1), 0 (f_2), 0 (f_3)	350 (f_1), 0 (f_2), 300 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
3.75	25 (f_1), 0 (f_2), 0 (f_3)	375 (f_1), 0 (f_2), 325 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
4.00	0 (f_1), 0 (f_2), 0 (f_3)	400 (f_1), 0 (f_2), 350 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
4.25	75 (f_1), 0 (f_2), 0 (f_3)	425 (f_1), 0 (f_2), 375 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
4.50	50 (f_1), 0 (f_2), 0 (f_3)	450 (f_1), 0 (f_2), 400 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
4.75	25 (f_1), 0 (f_2), 0 (f_3)	475 (f_1), 0 (f_2), 425 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
5.00	0 (f_1), 0 (f_2), 0 (f_3)	500 (f_1), 0 (f_2), 450 (f_3)	200 (f_1), 0 (f_2), 200 (f_3)
5.25	100 (f_1), 50 (f_2), 25 (f_3)	500 (f_1), 50 (f_2), 450 (f_3)	200 (f_1), 50 (f_2), 200 (f_3)
5.50	100 (f_1), 0 (f_2), 50 (f_3)	500 (f_1), 100 (f_2), 450 (f_3)	200 (f_1), 100 (f_2), 200 (f_3)
5.75	100 (f_1), 0 (f_2), 25 (f_3)	500 (f_1), 100 (f_2), 500 (f_3)	200 (f_1), 100 (f_2), 250 (f_3)
6.00	100 (f_1), 0 (f_2), 0 (f_3)	500 (f_1), 100 (f_2), 550 (f_3)	200 (f_1), 100 (f_2), 300 (f_3)
6.25	200 (f_1), 50 (f_2), 25 (f_3)	500 (f_1), 150 (f_2), 550 (f_3)	200 (f_1), 150 (f_2), 300 (f_3)
6.50	200 (f_1), 0 (f_2), 50 (f_3)	500 (f_1), 200 (f_2), 550 (f_3)	200 (f_1), 200 (f_2), 300 (f_3)
6.75	199 (f_1), 0 (f_2), 26 (f_3)	501 (f_1), 200 (f_2), 599 (f_3)	201 (f_1), 200 (f_2), 349 (f_3)
7.00	199 (f_1), 0 (f_2), 1 (f_3)	501 (f_1), 200 (f_2), 649 (f_3)	201 (f_1), 200 (f_2), 399 (f_3)
7.25	274 (f_1), 75 (f_2), 26 (f_3)	526 (f_1), 225 (f_2), 649 (f_3)	226 (f_1), 225 (f_2), 399 (f_3)
7.50	249 (f_1), 50 (f_2), 51 (f_3)	551 (f_1), 250 (f_2), 649 (f_3)	251 (f_1), 250 (f_2), 399 (f_3)
7.75	224 (f_1), 25 (f_2), 76 (f_3)	576 (f_1), 275 (f_2), 649 (f_3)	276 (f_1), 275 (f_2), 399 (f_3)
8.00	199 (f_1), 0 (f_2), 101 (f_3)	601 (f_1), 300 (f_2), 649 (f_3)	301 (f_1), 300 (f_2), 399 (f_3)
8.25	274 (f_1), 75 (f_2), 126 (f_3)	626 (f_1), 325 (f_2), 649 (f_3)	326 (f_1), 325 (f_2), 399 (f_3)
8.50	249 (f_1), 50 (f_2), 151 (f_3)	651 (f_1), 350 (f_2), 649 (f_3)	351 (f_1), 350 (f_2), 399 (f_3)
8.75	224 (f_1), 25 (f_2), 176 (f_3)	676 (f_1), 375 (f_2), 649 (f_3)	376 (f_1), 375 (f_2), 399 (f_3)
9.00	200 (f_1), 0 (f_2), 200 (f_3)	700 (f_1), 400 (f_2), 650 (f_3)	400 (f_1), 400 (f_2), 400 (f_3)
9.25	283 (f_1), 83 (f_2), 209 (f_3)	717 (f_1), 417 (f_2), 666 (f_3)	417 (f_1), 417 (f_2), 416 (f_3)
9.50	266 (f_1), 67 (f_2), 217 (f_3)	734 (f_1), 433 (f_2), 683 (f_3)	434 (f_1), 433 (f_2), 433 (f_3)
9.75	250 (f_1), 50 (f_2), 225 (f_3)	750 (f_1), 450 (f_2), 700 (f_3)	450 (f_1), 450 (f_2), 450 (f_3)
10.00	233 (f_1), 33 (f_2), 234 (f_3)	767 (f_1), 467 (f_2), 716 (f_3)	467 (f_1), 467 (f_2), 466 (f_3)
10.25	216 (f_1), 117 (f_2), 242 (f_3)	784 (f_1), 483 (f_2), 733 (f_3)	484 (f_1), 483 (f_2), 483 (f_3)
10.50	200 (f_1), 100 (f_2), 250 (f_3)	800 (f_1), 500 (f_2), 750 (f_3)	500 (f_1), 500 (f_2), 500 (f_3)
10.75	183 (f_1), 83 (f_2), 234 (f_3)	817 (f_1), 517 (f_2), 766 (f_3)	517 (f_1), 517 (f_2), 516 (f_3)
11.00	166 (f_1), 67 (f_2), 217 (f_3)	834 (f_1), 533 (f_2), 783 (f_3)	534 (f_1), 533 (f_2), 533 (f_3)
11.25	150 (f_1), 150 (f_2), 200 (f_3)	850 (f_1), 550 (f_2), 800 (f_3)	550 (f_1), 550 (f_2), 550 (f_3)
11.50	133 (f_1), 133 (f_2), 184 (f_3)	867 (f_1), 567 (f_2), 816 (f_3)	567 (f_1), 567 (f_2), 566 (f_3)
11.75	116 (f_1), 117 (f_2), 167 (f_3)	884 (f_1), 583 (f_2), 833 (f_3)	584 (f_1), 583 (f_2), 583 (f_3)
12.00	100 (f_1), 100 (f_2), 150 (f_3)	900 (f_1), 600 (f_2), 850 (f_3)	600 (f_1), 600 (f_2), 600 (f_3)
12.25	83 (f_1), 183 (f_2), 134 (f_3)	917 (f_1), 617 (f_2), 866 (f_3)	617 (f_1), 617 (f_2), 616 (f_3)
12.50	66 (f_1), 167 (f_2), 117 (f_3)	934 (f_1), 633 (f_2), 883 (f_3)	634 (f_1), 633 (f_2), 633 (f_3)
12.75	50 (f_1), 150 (f_2), 100 (f_3)	950 (f_1), 650 (f_2), 900 (f_3)	650 (f_1), 650 (f_2), 650 (f_3)
13.00	33 (f_1), 133 (f_2), 84 (f_3)	967 (f_1), 667 (f_2), 916 (f_3)	667 (f_1), 667 (f_2), 666 (f_3)

Time (sec)	Buffered Packets	Processed Packets	Processed Packets(with slack of 200)
13.00	33 (f_1), 133 (f_2), 84 (f_3)	967 (f_1), 667 (f_2), 916 (f_3)	667 (f_1), 667 (f_2), 666 (f_3)
13.25	16 (f_1), 217 (f_2), 67 (f_3)	984 (f_1), 683 (f_2), 933 (f_3)	684 (f_1), 683 (f_2), 683 (f_3)
13.50	0 (f_1), 200 (f_2), 50 (f_3)	1000 (f_1), 700 (f_2), 950 (f_3)	700 (f_1), 700 (f_2), 700 (f_3)
13.75	0 (f_1), 199 (f_2), 1 (f_3)	1000 (f_1), 701 (f_2), 999 (f_3)	700 (f_1), 701 (f_2), 749 (f_3)
14.00	0 (f_1), 150 (f_2), 0 (f_3)	1000 (f_1), 750 (f_2), 1000 (f_3)	700 (f_1), 750 (f_2), 750 (f_3)
14.25	0 (f_1), 200 (f_2), 0 (f_3)	1000 (f_1), 800 (f_2), 1000 (f_3)	700 (f_1), 800 (f_2), 750 (f_3)
14.50	0 (f_1), 150 (f_2), 0 (f_3)	1000 (f_1), 850 (f_2), 1000 (f_3)	700 (f_1), 850 (f_2), 750 (f_3)
14.75	0 (f_1), 100 (f_2), 0 (f_3)	1000 (f_1), 900 (f_2), 1000 (f_3)	700 (f_1), 900 (f_2), 750 (f_3)
15.00	0 (f_1), 50 (f_2), 0 (f_3)	1000 (f_1), 950 (f_2), 1000 (f_3)	700 (f_1), 900 (f_2), 750 (f_3)
15.25	0 (f_1), 0 (f_2), 0 (f_3)	1000 (f_1), 1000 (f_2), 1000 (f_3)	700 (f_1), 950 (f_2), 750 (f_3)

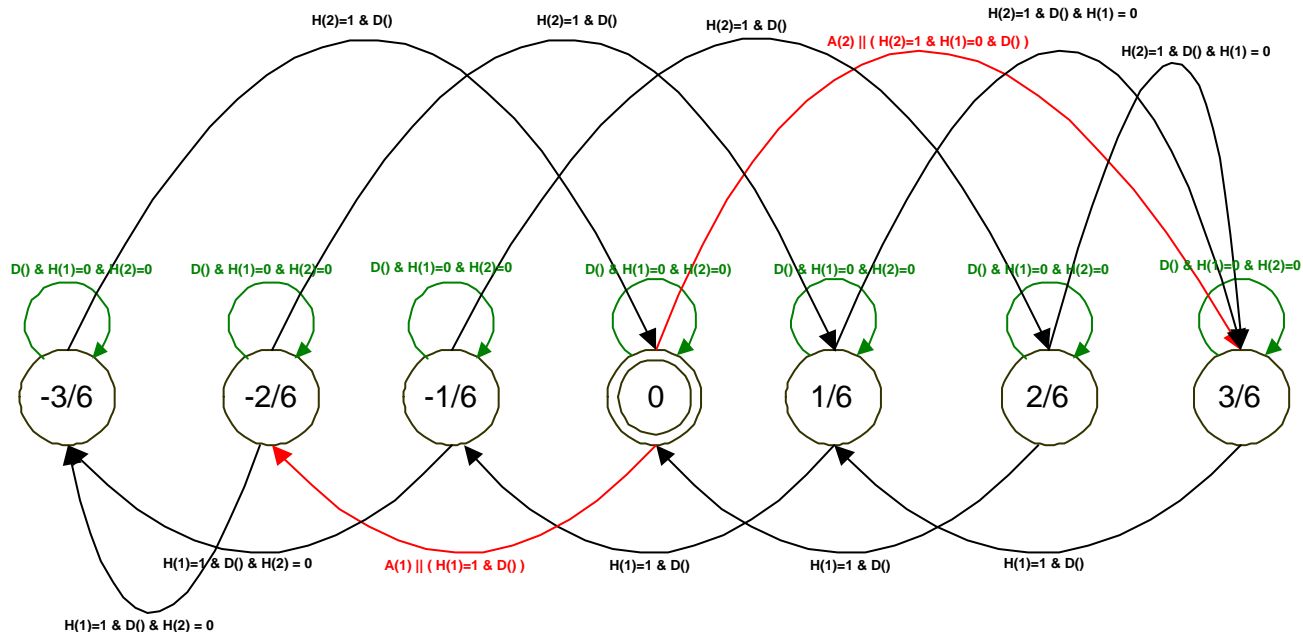
f_1 time of completion is: 13.490 sec, f_2 time of completion is: 15.250 sec, f_3 time of completion is: 13.755 sec.

- Construct a FSM for a router that implements weighted fair queueing for two flows f_1 and f_2 where $w_1 = 3$, $w_2 = 2$, and the slack is 0.5 (where the processing of a packet from f_1 adds $1/3$ to its virtual clock, and the processing of a packet from f_2 adds $1/2$ to its virtual clock). Your state machine can depend on the following functions and events to simplify its design:

- $IE()$: the queue is empty (prior to an arrival)
- $H(i)$: $H(i)$ equals 0 when no packets from f_i are queued in the system, and equals 1 otherwise
- $D()$: triggered when the router completes processing its current packet.
- $A(i)$: triggered when an arrival to the router from f_i occurs (only needs to be used when the router is not processing any packets).

With these functions and events, you should build your FSM so that it indicates clearly whose packet should be processed next whenever such a decision needs to be made. (Hint: each state should indicate the current difference between the two flow's clocks)

Answer:



3. Consider a queue where packets arrive at rate λ and whose processing times are exponentially distributed with rate μ . When the queue is empty, the processor processes “pretend” packets at rate μ . When a real packet arrives, the pretend packet is immediately aborted (discarded) and the real packet is processed. The completion of a “pretend” packet is counted as an event.

Recall that $S(n)$ equals the number of packets in the system after the n th event. Here, an event is a packet arrival or a service completion, including the completion of pretend packets.

- (a) What are $\Pr(S(n) = 1 | S(n-1) = 0)$ and $\Pr(S(n) = 0 | S(n-1) = 0)$?

Answer:

The $\Pr(S(n) = 1 | S(n-1) = 0)$ is the transition probability from state 0 to state 1 and it is:

$$\Pr(S(n) = 1 | S(n-1) = 0) = \frac{\lambda}{\lambda + \mu}$$

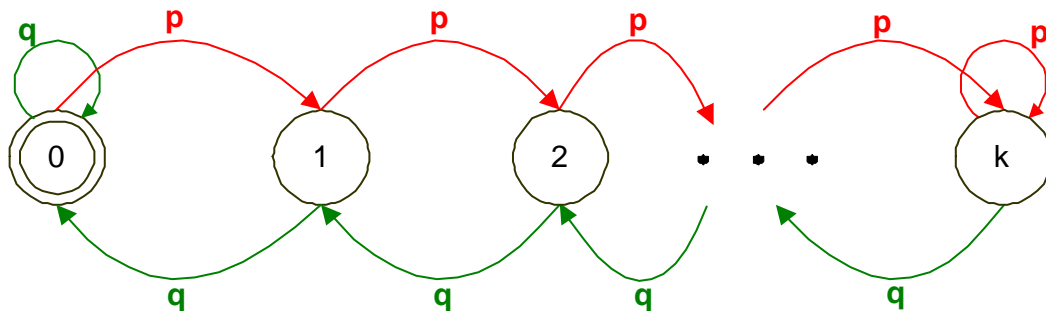
Also $\Pr(S(n) = 0 | S(n-1) = 0)$ is the transition probability from state 0 to state 0 and it is:

$$\Pr(S(n) = 0 | S(n-1) = 0) = \frac{\mu}{\lambda + \mu}$$

- (b) Construct the Markov model for this queueing system.

Answer:

Let $p = \frac{\lambda}{\lambda + \mu}$ and $q = \frac{\mu}{\lambda + \mu}$ then the Markov model is the following:



- (c) Solve for the steady-state distribution, $\pi_i = \lim_{n \rightarrow \infty} \Pr(S(n) = i)$ for all $i, 0 \leq i \leq k$.

For this part we have to use two equations. The first is that $\sum_{i=0}^k \pi_i = 1$ and the second comes from the fact that for each state the inflow must equal the outflow. We have that for state 0, $\pi_1 = \frac{p}{q} \cdot \pi_0$ (2). In general for state i we have $\pi_{i-1} \cdot p + \pi_{i+1} \cdot q = \pi_i(p + q) = \pi_i$. This for $i = 1$ gives us $p \cdot \pi_0 + q \cdot \pi_2 = \pi_1 \Rightarrow q \cdot \pi_2 = \pi_1 - p \cdot \pi_0$ and using (2) we get:

$$q \cdot \pi_2 = \frac{p}{q} \cdot \pi_0 - p \cdot \pi_0 \Rightarrow q \cdot \pi_2 = \frac{p - p \cdot q}{q} \cdot \pi_0 \Rightarrow \pi_2 = \frac{p - p \cdot q}{q^2} = \frac{p \cdot (1 - q)}{q^2} \pi_0 = \frac{p^2}{q^2} \pi_0 \text{ as } p = 1 - q$$

So $\pi_2 = \left(\frac{p}{q}\right)^2 \pi_0$. Similarly we can show that in general:

$$\pi_i = \left(\frac{p}{q}\right)^i \pi_0 \quad (3)$$

Now we can use (1) to compute π_0 . Now (1) using (3) becomes :

$$\sum_{i=0}^k \pi_i = 1 \Rightarrow \sum_{i=0}^k \left(\frac{p}{q}\right)^i \pi_0 = 1 \Rightarrow \frac{1 - \left(\frac{p}{q}\right)^{k+1}}{1 - \frac{p}{q}} \cdot \pi_0 = 1 \Rightarrow \pi_0 = \frac{1 - \frac{p}{q}}{1 - \left(\frac{p}{q}\right)^{k+1}}$$

So

$$\pi_i = \left(\frac{p}{q}\right)^i \cdot \frac{1 - \frac{p}{q}}{1 - \left(\frac{p}{q}\right)^{k+1}}$$

- (d) Solve for π_i as $k \rightarrow \infty$.

Answer:

As $k \rightarrow \infty$ we get (assuming $p < q$):

$$\lim_{k \rightarrow \infty} \sum_{i=0}^k \left(\frac{p}{q}\right)^i \pi_0 = 1 \Rightarrow \sum_{i=0}^{\infty} \left(\frac{p}{q}\right)^i \pi_0 = 1 \Rightarrow \frac{1}{1 - \frac{p}{q}} \cdot \pi_0 = 1 \Rightarrow \pi_0 = \left(1 - \frac{p}{q}\right) \text{ and } \pi_i = \left(\frac{p}{q}\right)^i \cdot \left(1 - \frac{p}{q}\right)$$

In case $p \geq q$ the queue is unstable.

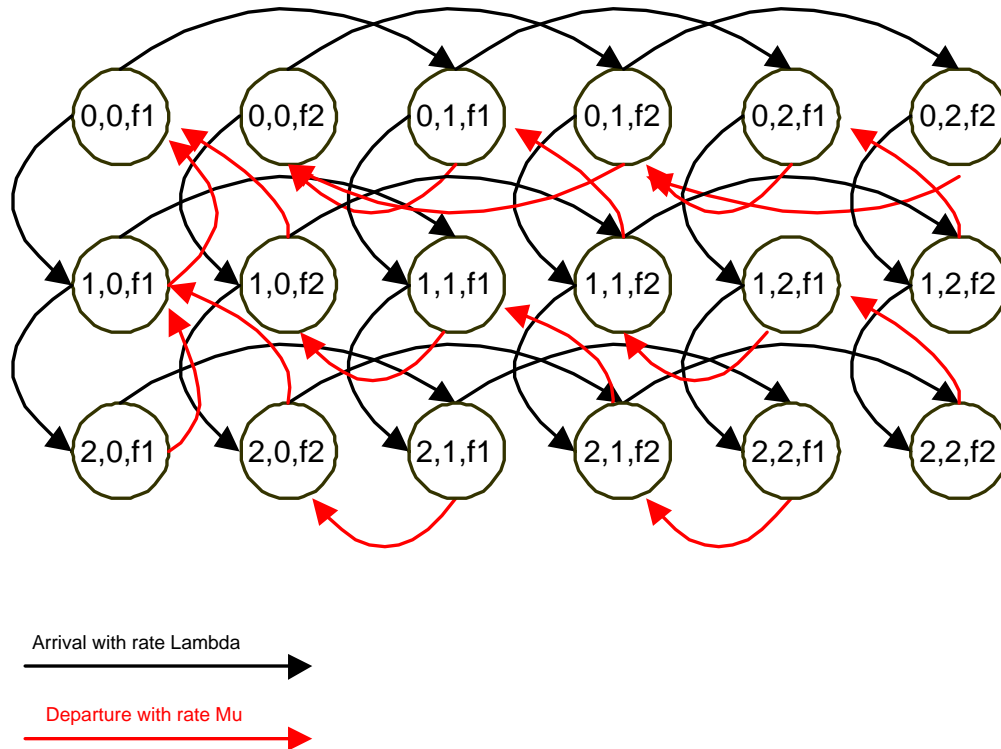
- (e) π_i is different than the value computed using the Markov model formulated in class. Explain why this is the case.

Answer:

The model we had in the class did not have the loop in state 0. That causes a change in the steady state distribution.

4. Construct a Markov model for a round-robin queueing system that can store up to 4 packets for processing, where no more than 2 packets from a single flow is ever stored. Assume there are two flows in the system, where both flow's packets are processed at rate μ , and flow f_i 's packets arrive at rate λ_i . Label transitions with their transition probabilities.

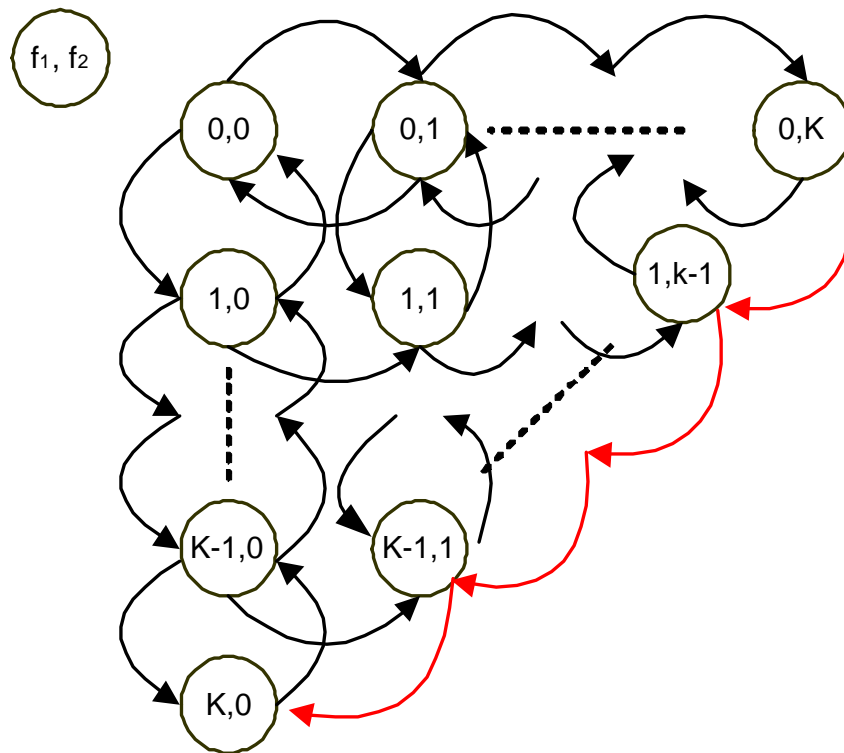
Answer:



5. Construct Markov models for each of two priority queueing systems, each processing packets from two flows. Each system can hold a total of k packets from each flow. A packet from f_1 should always be processed before any packets from f_2 in the queue. Assume both flow's packets are processed at rate μ , and flow f_i 's packets arrive at rate λ_i .

- (a) In the first system, a packet from f_1 arriving to a full queue will replace a packet in the queue from f_2 (if one exists).

Answer:



- (b) In the second system, packets in the queue are not removed except when their processing is complete (i.e., a packet arriving from f_1 to a full queue will be turned away, even if there are packets from flow f_2 in the queue).

Answer:

