Yiping Wang

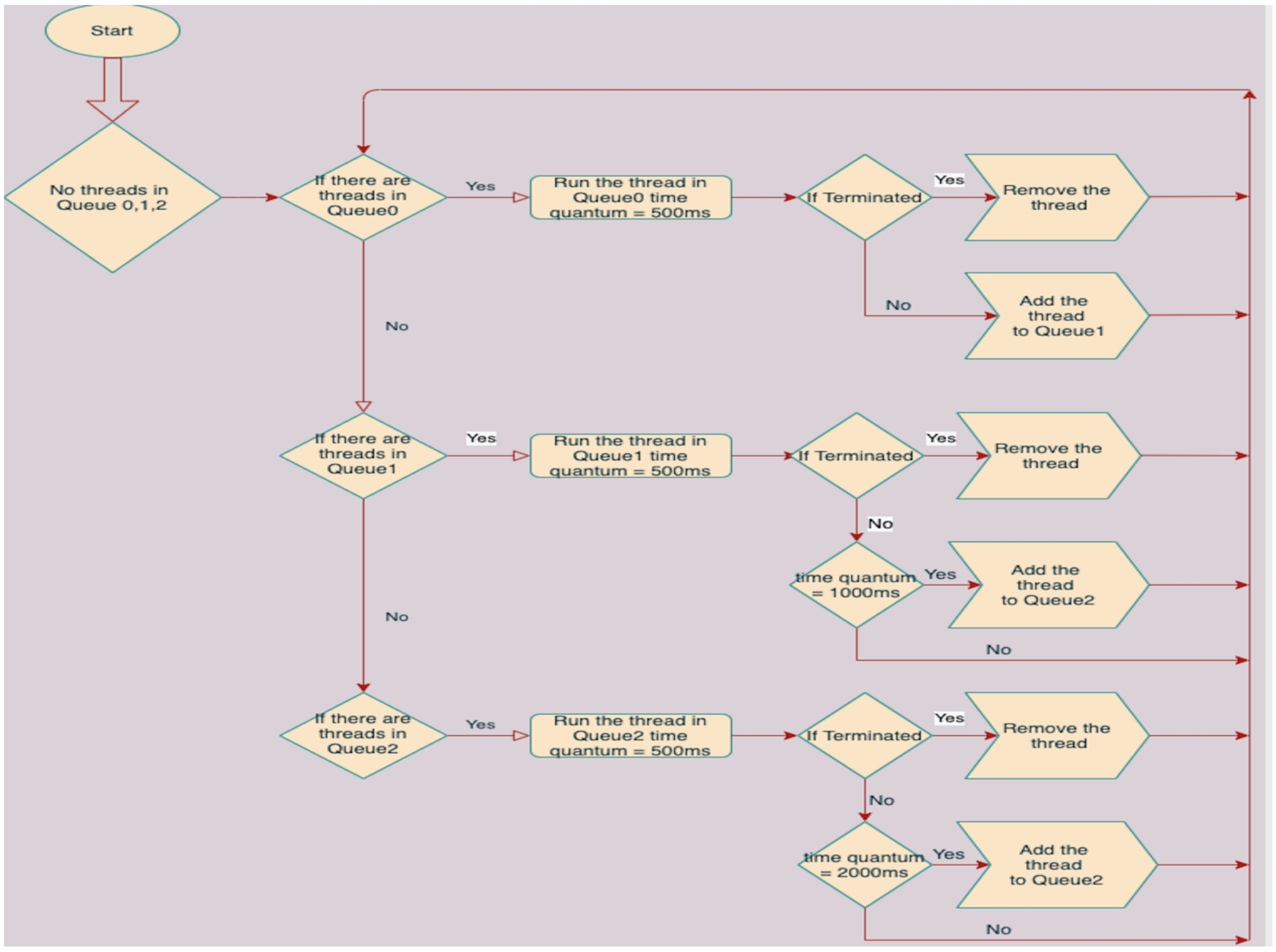
CptS 370 Program 3: Operating System Scheduler

Report

Explanation of the design and algorithm for Part 2

Back-queue (MFQS) scheduler is divided into three separate queues, Queue0,1 and 2.

As shown in the figure below, the new thread enters Queue0 first. Time Quantum = 500 ms. If the thread completes, it is removed from Q0. If the thread does not finish executing, put it on the next queue.


Start
No threads in Queue 0,1,2
If there are threads in Queue0
Yes
Run the thread in Queue0 time quantum = 500ms
If Terminated
Yes
Remove the thread
No
Add the thread to Queue1
No
If there are threads in Queue1
Yes
Run the thread in Queue1 time quantum = 500ms
If Terminated
Yes
Remove the thread
No
time quantum = 1000ms
Yes
Add the thread to Queue2
No
No
If there are threads in Queue2
Yes
Run the thread in Queue2 time quantum = 500ms
If Terminated
Yes
Remove the thread
No
time quantum = 2000ms
Yes
Add the thread to Queue2
No

By the same logic, the thread continues into Queue1 and Queue2. The specified time Quantum = 1000 ms and time Quantum = 2000 ms. The queue will first run the current thread and then the last waiting thread.

Test Performance Results

	Round-robin Scheduler			Multilevel Feed Back-Queue(MFQS) Scheduler		
Thread Name	Response time	Turnaround time	Execution time	Response time	Turnaround time	Execution time
Thread[a]	1998	29003	27005	498	24505	24007
Thread[b]	2997	9999	7002	998	5507	4509
Thread[c]	3998	21001	17003	1499	16503	15004
Thread[d]	4998	33003	28005	1999	31503	29504
Thread[e]	5999	6499	500	2500	8001	5501
Total	19990	99505	79515	7494	86019	78525

Analysis of the data tables revealed that, the Multilevel Feed Back-Queueue (MFQS) Scheduler performs better than the Round-robin Scheduler, especially in terms of response time, as compared to 7494 and 19990. It's nearly a third the speed of the competition. This is somewhat surprising, as it passes requests directly to the next queue, rather than assigning them to internal servers in turn each time. During the execution process, although there is a sleep time, it does not have to wait for the time to be initialized, so the overall efficiency is still better than Round-robin Scheduler, and the Excution time will definitely be faster. This result leads to the shortening of the Turnaround time.

	FCFS-based Queue2			Multilevel Feed Back-Queue(MFQS) Scheduler		
Thread Name	Response time	Turnaround time	Execution time	Response time	Turnaround time	Execution time
Thread[a]	4999	34009	29010	498	24505	24007
Thread[b]	7500	8501	1001	998	5507	4509
Thread[c]	9501	24506	15005	1499	16503	15004
Thread[d]	12002	42510	30508	1999	31503	29504
Thread[e]	14503	15003	500	2500	8001	5501
Total	48505	124529	76024	7494	86019	78525

By comparing the data in the table, I found that the turnaround time of FCFS is much larger than that of MFQS, my guess is that it is due to the large amount of time added after thread termination. Also, since it takes 2000ms or more for each queue to go to the next thread, it results in longer response times. In terms of execution time is almost the same, but FCFS is slightly faster, which just shows that the two implementations of FCFS and MFQS have some shortcomings, FCFS slightly to make up for the first two shortcomings.