Bilkent University

Department of Computer Engineering

CS342 Project 2 Report

Section 2

Multiprocessor Scheduling, Threads, Synchronization

Group Members:

Efe Beydoğan 21901548

Emir Melih Erdem 21903100

Experiments for Part A

1. Tests for RR with Q=20

N	Average Turnaround Times (ms)		
	Single-Queue	Multi-Queue	
		RM	LM
1	982	985	986
2	399	403	420
3	210	247	256
4	132	205	201
5	115	160	165

2. Tests for FCFS

N	Average Turnaround Times (ms)		
	Single-Queue	Multi-Queue	
		RM	LM
1	994	995	994
2	383	422	450
3	206	246	245
4	130	193	188
5	115	157	178

3. Tests for SJF

N	Average Turnaround Times (ms)		
	Single-Queue	Multi-Queue	
		RM	LM
1	589	589	589
2	262	256	293
3	185	206	219
4	129	177	183
5	114	150	178

Analysis for Part A

Here the average turnaround times obtained with single-queue and multi-queue approaches (both with round-robin and load balancing methods) are given for N from 1 to 5, and the experiment is repeated for the three scheduling algorithms—RR, FCFS, and SJF. In all of the cases, as the number of processors N increases from 1 to 5, turnaround times decrease gradually since more processors mean faster execution as the bursts can be run in parallel.

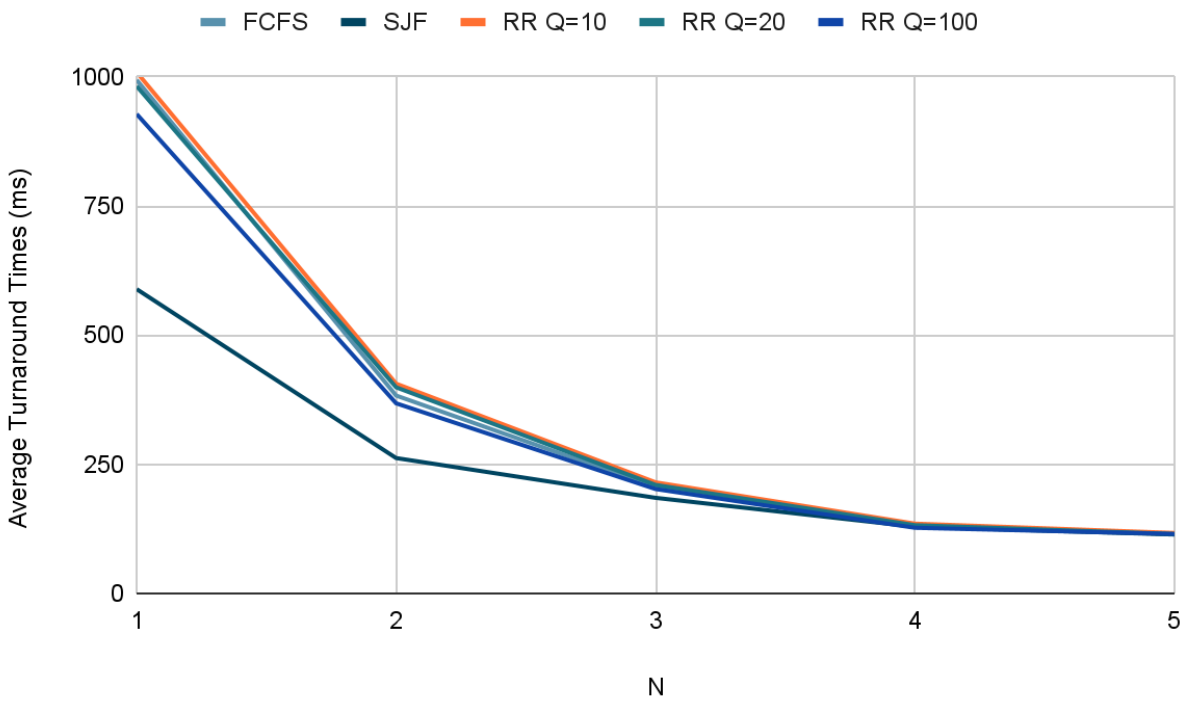
As expected, when there is only one processor (N=1), the single and multi-queue approaches give the same results since there is only one queue for scheduling. However, as N increases, single queue produces smaller turnaround times compared to multi-queue, independent of the queue selection method. The single-queue approach proved to be the most efficient method for all of the scheduling algorithms (RR with Q=20, FCFS, and SJF). This is because with a single queue, there is no need for load balancing, and processors can be used effectively. Whenever a processor becomes free, a waiting burst can directly be picked from the ready queue. However, with multi-queue, a processor can only select bursts from its own queue, resulting in an imbalance in loads and longer turnaround times.

When using the multi-queue approach, the load balancing method (LM) generally produced slightly longer turnaround times for all of the algorithms. Normally, load balancing might be expected to give better results. However, our load-balancing algorithm is not perfect (it does not consider the bursts running in the processors while calculating the load), it may place bursts in a busy processor's queue even when there are free CPUs. On the contrary, the round-robin method performs better, in this experiment, by distributing the load one by one across all processors. Additionally, the load balancing method comes with an overhead for computing the

loads of the queues during enqueue operations, which is also a factor causing the longer turnaround times.

Experiments for Part B

N	Average Turnaround Times (ms)				
	FCFS	SJF	RR Q=10	RR Q=20	RR Q=100
1	994	589	1009	982	928
2	383	262	406	399	368
3	206	185	215	210	202
4	130	129	135	132	127
5	115	114	117	115	115



Analysis for Part B

The above data compares different scheduling algorithms (FCFS, SJF, and RR with Q=10, 20, 100) in terms of average turnaround times while N increases from 1 to 5. Again, as the number of processors N increases from 1 to 5, turnaround times decrease as more processors mean faster execution since the bursts can be run in parallel.

Among the algorithms, SJF performs the best for any N, followed by FCFS and then RR. SJF enables the shorter bursts to not get blocked by the longer ones and, therefore, gives the shortest average turnaround times. On the other hand, RR is the worst in terms of turnaround times since bursts can only execute during short time slices and then wait for their turn again. It is also seen that time quantum Q affects the performance of the round-robin algorithm positively, with higher Q values giving shorter turnaround times. As Q increases, the bursts are less interrupted and can finish their executions in fewer rounds. When Q=100, since most bursts are shorter than 100 ms in our tests, almost no bursts are interrupted, and they can continue with their executions almost non-preemptively, as in FCFS. This explains why the results for RR approach FCFS as Q increases to 100.