Matthew Huynh
CS350 HW #5
Analysis

Exercise 1

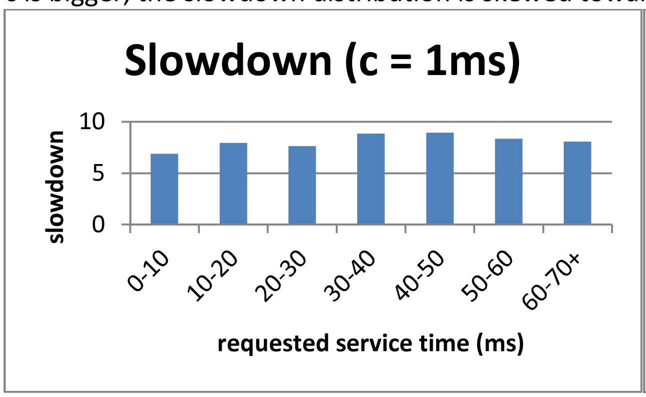
a)
M/M/1 approximation calculations:
rho: 0.9
Tq: 0.3 s

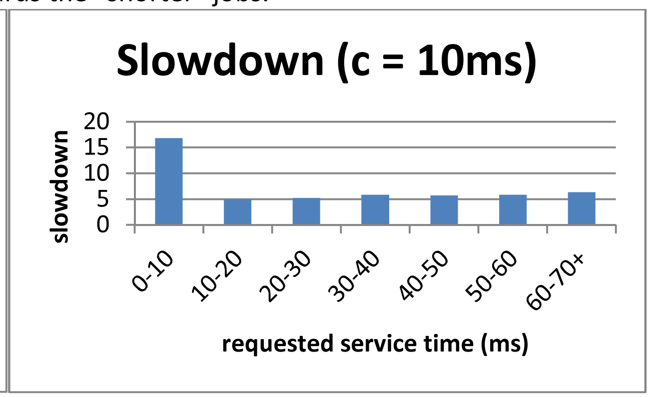
Empirical results:

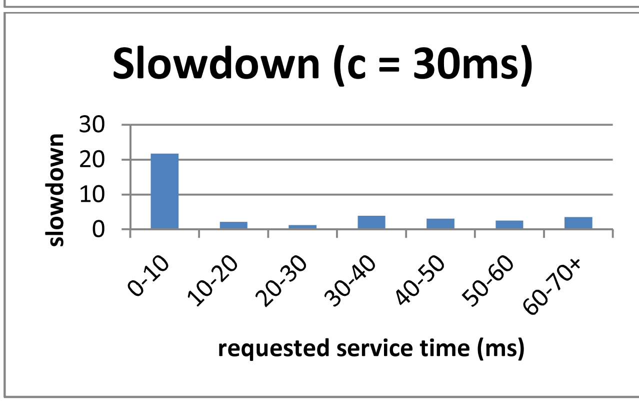
Time quantum c (ms)	Average response time (s)
1	0.28
10	0.15
30	0.1
60	0.1

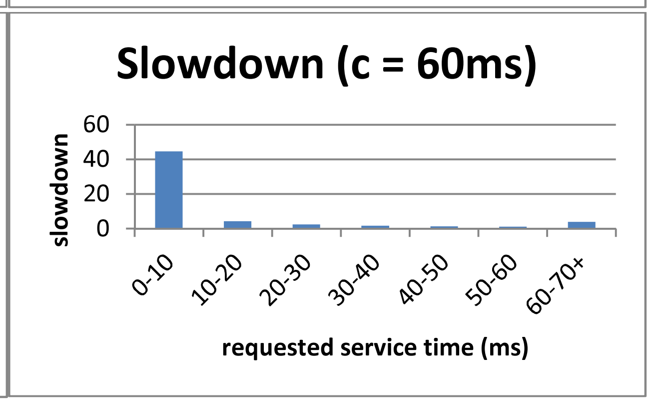
As the time quantum c approaches 0, a round-robin scheduler emulates general processor sharing.

b) Slowdown does indeed depend on c. As c gets smaller, the slowdown distribution is more uniform whereas if c is bigger, the slowdown distribution is skewed towards the “shorter” jobs.







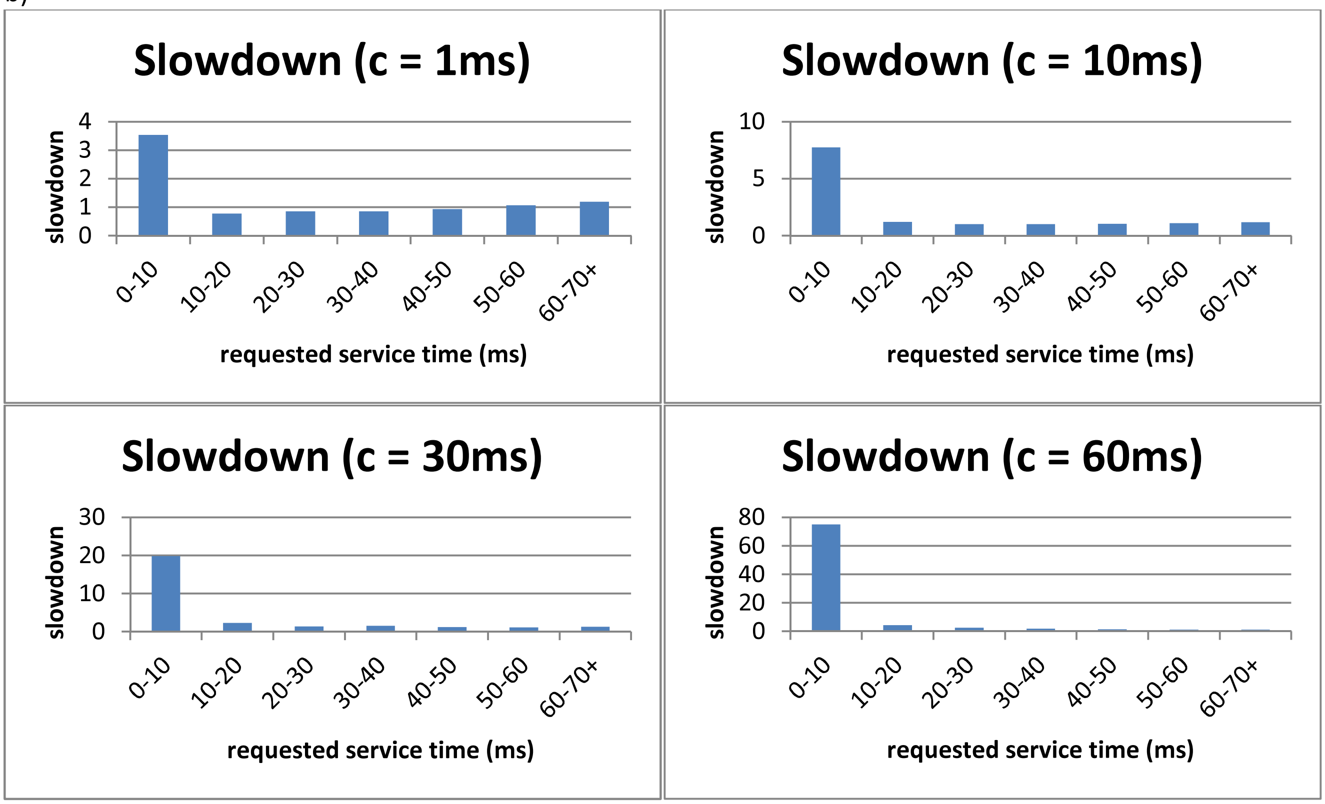


Exercise 2

a)

Time quantum c (ms)	Average response time (s)	Average slowdown
1	0.0308	1.080
10	0.0359	6.375
30	0.0469	6.875
60	0.0696	25.116

b)



c) The observed slowdown profile of this scheduler is similar to the round-robin scheduler because the slowdown skews more towards the shorter jobs as c increases. When c is 1ms, even though the slowdown is still skewed towards the 0-10ms jobs, overall slowdown is reduced significantly across the board (compare this scheduler's max slowdown of below 4x with RR scheduler's min slowdown of above 5x).

Exercise 3

FIFO

Job	Requested Service Time	Response Time	Slowdown
A	1	1	1
B	1	2	2
C	0.05	2.05	41

SRTF

Job	Requested Service Time	Response Time	Slowdown
A	1	1.01	1.01

B	1	2.02	2.02
C	0.05	2.05	41

RR with 200ms quantum

Job	Requested Service Time	Response Time	Slowdown
A	1	1.84	1.84
B	1	2.04	2.04
C	0.05	2.05	41

VRR with 200ms quantum

Job	Requested Service Time	Response Time	Slowdown
A	1	1.85	1.85
B	1	2.05	2.05
C	0.05	1.25	1.25

Exercise 4

a) Starvation is possible for Class C jobs if the combined utilization of Class A and B jobs exceeds 100% because then Class C will never gain access to the CPU simply because there is always a Class B or Class A job needing the CPU.
b) worst-case response time for Class A job: **5 seconds** (since it would preempt any current job on the CPU)
worst-case response time for Class B job: **5.5 seconds** (waits for a Class A job to complete)
worst-case response time for Class C job: **50 seconds** (class A, B, C all arrive at same time)
c) 5/60+0.5/1+20/300 = 0.65, which is less than 3*(2^(1/3)-1) = ~0.77
The maximum amount of CPU time per period for a class C job is about 58 seconds.
d) If a class C job is holding the CPU, it is also holding R. When a class A job arrives, it won't get the CPU simply because R is busy. This means that the highest-priority job is being blocked by the lowest-priority job.
e) 45 seconds because class B jobs will keep interrupting the class C job, and this goes on for 40 seconds until the class C job is done, at which time the class A job will run for 5 seconds.
f) Priority inversion is when a higher priority task is blocked because it is waiting on a shared resource and that shared resource is being held by a lower priority task.
g) Priority inheritance is a strategy to prevent priority inversion. A task that is holding onto a shared resource will temporarily "inherit," i.e. gain, the priority of the highest-priority task waiting for that shared resource so that the shared resource will eventually be released.
h) If priority inheritance is used to schedule the CPU, the worst-case response time for class A jobs would be 25 seconds because the class C job would run for 20 seconds, followed by the class A job for 5 seconds.