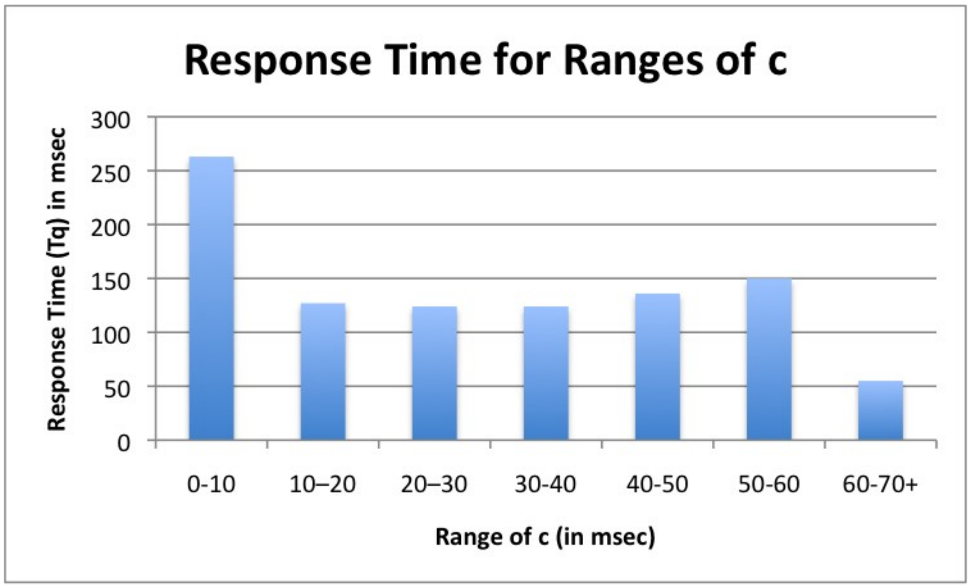
CS350 Homework 05 Tim Duffy

1) a. Write a simulator that would evaluate the performance of a round-robin scheduler. Measure the average response time for requests submitted to your round-robin scheduler when c = 1, 10, 30, and 60 milliseconds. Compare the results from your simulator to those you calculate using an M/M/1 approximation. What conclusions can you make?

Quantum	Average response time (Tq)
c = 1	293.8 msecs
c = 10	115.7 msecs
c = 30	127.6 msecs
c = 60	149.4 msecs

When the quantum is small (say 1 msec) then the average response time is very slow. This is because requests for large amounts of time must wait in the queue several times in order to finish. The closer the size of the quantum gets to the average service time, the faster the system seems to respond. Furthermore, when the size of the quantum becomes larger than the average service time, the system becomes closer to an M/M/1 system. This is because more and more requests are completed with only 1 quantum. Thus, after the quantum becomes much greater than average Ts, the system hits a bound in response time as it would if it were an M/M/1.

b.



2) a. Pass
b. Pass
c. With the preemptive shortest-Remaining-First method, the slowdown for small jobs is smaller than it would be in round-robin scheduling. However, the slowdown for events that need the resource for a long time is greater than it would be for round-robin scheduling. This is because longer jobs see more interruptions by short jobs than with round-robin scheduling. With this scheduling method, starvation of a process is possible, especially with a busy system.

3) Compute the slowdown for jobs A, B, and C for the following algorithms:

FIFO:

Tq(A) = 1000 msec
Tq(B) = 2000 msec
Tq(C) = 2000 + 5* (10 + 100) = 2000 + 550 = 2550 msec

FIFO Slowdown for A = 1000/1000 = **1**
FIFO Slowdown for B = 2000/1000 = **2**
FIFO Slowdown for C = 2550/550 = **4.636**

Shortest-Remaining-Time-First: (Preemptive)

- C is chosen first, and uses the CPU for 10 msec, then goes on to I/O
- Now A and B remain. Since A is first, it will be served next.
- At t = 110 msec, C returns and is selected.
  A has been served for 100 msec, C has completed 1 cycle

- At t = 120, C moves to the I/O and service of A resumes
- At t = 220, C returns and begins service
  A has been served for 200 msec, C has completed 2 cycles

- At t = 230, C moves to the I/O and service of A resumes
- At t = 330, C returns and begins service
  A has been served for 300 msec, C has completed 3 cycles

- At t = 340, C moves to the I/O and service of A resumes
- At t = 440, C returns and begins service
  A has been served for 400 msec, C has completed 4 cycles

- At t = 450, C moves to the I/O and completes at t = 550
- At t = 450, A begins service, and goes to completion
  Tq(C) = 550 msec
  Tq(A) = 450 + 600 = 1050 msec

- Finally, B begins service to its completion Tq(B) = 1050 + 1000 = 2050

STR Slowdown for A = 1050/1000 = **1.05**
STR Slowdown for B = 2050/1000 = **2.05**
STR Slowdown for C = 550/550 = **1**

Round-Robin with a 200 msec quantum:

- A is served for 200 msec	t = 200
- B is served for 200 msec	t = 400
- C is served for 10 msec (1 of 5)	t = 410
- A is served for 200 msec	t = 610
- B is served for 200 msec	t = 810

- C is served for 10 msec (2 of 5)	t = 820	
- A is served for 200 msec	t = 1020	
- B is served for 200 msec	t = 1220	
- C is served for 10 msec (3 of 5)	t = 1230	
- A is served for 200 msec	t = 1430	
- B is served for 200 msec	t = 1630	
- C is served for 10 msec (4 of 5)	t = 1640	
- A is served for 200 msec	t = 1840	Tq(A) = 1840
- B is served for 200 msec	t = 2040	Tq(B) = 2040
- C is served for 10 msec (5 of 5)	t = 2050	Tq(C) = 2050

RR Slowdown for A = 1840/1000 = **1.84**
RR Slowdown for B = 2040/1000 = **2.04**
RR Slowdown for C = 2050/550 = **3.72**

Virtual Round Robin with 200-milliseconds quantum

- A, B and C are all in the ready queue - A is selected first
- A is served for 200 msec, kept in High Priority queue   t = 200
- B is served for 200 msec, kept in HP queue   t = 400
- C is served for 10 msec (1 of 5)
  Upon it return it is put in the Low priority queue   t = 410
  From here on, A and B will both finish since they will remain in the HP queue
  However, they will still alternate taking 200 msec quantums

- A is served for 200 msec (400 total)	t = 610	
- B is served for 200 msec (400 total)	t = 810	
- A is served for 200 msec (600 total)	t = 1010	
- B is served for 200 msec (600 total)	t = 1210	
- A is served for 200 msec (800 total)	t = 1410	
- B is served for 200 msec (800 total)	t = 1610	
- A is served for 200 msec (completed)	t = 1810	Tq(A) = 1810
- B is served for 200 msec (completed)	t = 2010	Tq(B) = 2010

The HP queue is now empty, so service of C begins

- C is served for 10 msec (2 of 5)	t = 2020	
- C completes I/O job	t = 2120	
- C is served for 10 msec (3 of 5)	t = 2230	
- C completes I/O job	t = 2330	
- C is served for 10 msec (4 of 5)	t = 2340	
- C completes I/O job	t = 2440	
- C is served for 10 msec (5 of 5)	t = 2450	
- C completes I/O job	t = 2550	Tq(C) = 2550

VRR Slowdown for A = 1840/1000 = **1.84**
VRR Slowdown for B = 2010/1000 = **2.01**
VRR Slowdown for C = 2050/550 = **3.72**

4) There are 3 classes of jobs in a computer system. Jobs of class A arrive to the system very infrequently, but are of the highest possible priority. Jobs of class C arrive to the system very infrequently and have the lowest possible priority. Jobs of class B arrive to the system quite frequently and have medium priority. Assume that all three classes of processes need the CPU. To manage the CPU, priority scheduling is used, whereby the highest priority Job that needs the CPU is scheduled (preempting any other job of lower priority that may be using the CPU, if necessary). Ties are broken arbitrarily. Answer the following questions:

a. Is starvation possible? Which of the three classes of jobs is most susceptible to starvation?

Yes, starvation is possible. Class C is susceptible to starvation. If a job of class C arrives, but then a steady stream of jobs of class A or B (which is possible do to the high frequency of class B jobs), then the jobs on the lowest priority queue will starve for resources.

b. Jobs of Class A will have worst-case response time of **5 seconds**.
Jobs of Class B will have worst-case response time of 5 + (1/2 * 5) = **7.5 seconds**
Jobs of Class C will have worst-case response time when:

- There is a Class A event just before it
    - Which, in the worst-case will cause a backup of Class B events
        5 B events will join the queue while the A event uses CPU

But, higher priority events will interrupt a class C job.

So it will take 7.5 seconds (worst-case) for no A or B class jobs scheduled. At that point, C may use the CPU until an A or B job arrives (0.5 seconds later). The C class can then use the 0.5 seconds between each future B events until it completes.

7.5s wait + 0.5s service + 38*(0.5s wait + 0.5s service) + 0.5s service
= 20s service + (7.5s + 19s wait) = **46.5 sec worst-case response time for C class**

c. What is the maximum amount of CPU time per period for jobs of class C beyond which the system will never reach steady state?

The system will grow unstable when the time to process one class C job exceeds the rate at which they arrive. More specifically when worst-case Ts for class C jobs exceeds 5 min.

7.5 wait + 0.5 served+ 52*(0.5 wait + 0.5 served) = 1 min - Then a new class A job arrives, and the process begins again

So, for every 1 min, a class C job only is served for 26.5 seconds.
Therefore, if the class C job needs to be served for more than 5(26.5) = 132.5

seconds, another class C job would arrive before the first finishes. Once this occurs, the queue for class C jobs will overflow!

132.5 seconds is the maximum amount of CPU for class C jobs.

d. As seen above, if the amount of CPU time needed by a class C job is more than 26.5 seconds, it will take more than 1 minute to process. Note: since B does not use the resource R, it can still compete for CPU. At this time, a new class A will arrive. However, since the first class C job is not done using R, the new class A job will be unable to compete for CPU. Thus, this priority scheme is not adequate when sharing R.

e. The **worst-case response time for a class A job** occurs when:

- A class C job arrives just before and uses R

Worst-case response time for the C job above is 40 seconds (alternating service between the C job and incoming B class jobs).

Once R is free, the A job takes priority over anything else, so worst-case response time **is 40 seconds**

f. Define what is meant by priority inversion.

Priority inversion occurs when a lower priority event preempts a higher priority e event, which essentially makes the lower priority event seem as something with higher priority.

g. Define what is meant by priority inheritance.

When a one or more high priority jobs are blocked by a lower priority job (priority inversion), a solution that can be implemented is priority inheritance. This method will set these high priority jobs to the highest level of priority, ignoring the previous scheduling method. Once this is complete, the original scheduling selection process resumes.

h. What is the worst-case response time for jobs of class A if priority inheritance is used?

If priority scheduling is used, then when a class A job arrives, it will use the resource needed regardless of whether or not another job currently has it. Thus, the worst-case response time for class A jobs with priority inheritance is **5 seconds.**