CIS 721 - Real-Time Systems
Homework #3
Fall 2015

Due: Monday, October 12, 2015, by 1:20 pm.

Twenty-five points. Please turn in your answers on a separate sheet attached to this sheet in class. Solutions will be posted online after class.

1. A system consists of five periodic tasks scheduled rate-monotonically with the following properties:

Task	Resource Requirements	Run-Time (C_i)	Deadline (D_i)	Period (T_i)	Blocking Time (B_i)	Response Time (R_i)
τ_1	[X;2]	2	6	6		
τ_2	none	2	12	12		
τ_3	[Y;4]	4	20	20		
τ_4	[X;3 [Z;1]]	5	200	200		
τ_5	[Z;2 [X;1]]	6	210	210		

For example, the resource requirements for task τ_4 indicate a nested critical section in which resource X is accessed for 3 units of time and Z is accessed for 1 unit of time (the lock on X is held for a total of three time units). Suppose that the non-preemptable critical sections protocol (NPCS) is used.

   (a) Draw the resource requirement graph associated with this set of tasks and resources.
   (b) Compute the worst-case blocking time for each task (complete the table shown above).
   (c) Apply the Utilization-Based Test to the task set. Can any conclusion be made; if so, what is your conclusion?
   (d) Compute the worst-case response time for each task (complete the table shown above). Is the task set feasible based on your Response Time Analysis?

2. Compute the worst case blocking time for the same problem for the Basic (Original) Priority Ceiling Protocol (PCP) – only complete part (b) from the previous problem.

Task	Resource Requirements	Run-Time (C_i)	Deadline (D_i)	Period (T_i)	Blocking Time (B_i)
τ_1	[X;2]	2	6	6	
τ_2	none	2	12	12	
τ_3	[Y;4]	4	20	20	
τ_4	[X;3 [Z;1]]	5	200	200	
τ_5	[Z;2 [X;1]]	6	210	210	

3. Three tasks are said to be ***transitively blocked*** when τ_3 blocks τ_2 which in turn blocks τ_1. Show that the priority ceiling protocol prevents transitive blocking, and consequently prevents deadlock among two or more tasks. In particular, first prove that the priority ceiling protocol prevents transitive blocking. Then, using this fact, show that deadlock cannot occur. Hint: recall that deadlock requires a circular wait – tasks waiting on each other in a cycle.

4. Show that under the priority-inheritance protocol, a task can be blocked directly by any lower-priority task for **at most once** for the duration of one outermost critical section, in the absence of deadlock.