Name: *SOLUTIONS* **(Bonus: 2 points)**

18-648: Embedded Real-Time Systems

Quiz #3 Solutions

Fall 2017

60 minutes

Instructions

1. This is a **CLOSED-BOOK and CLOSED-NOTES quiz.**
2. There are 10 pages to this quiz and 4 questions. There are two empty "scratch" pages at the end.
3. Show all relevant work.
4. Partial credit may be given for some questions.
5. The use of a calculator is allowed.
6. The time limit will be *strictly* enforced.
7. Watch the screen for any clarifications.

For Graders' Use Only

Name: _____ / 1

1._____ / 16

2._____ / 34

3._____ / 25

4._____ / 25

TOTAL. _____ / 100

Question 1. True/False Questions (16 points)

a) ____TRUE____ There does not exist an alternative single-frequency assignment which consumes less power than the SysClock assignment under fixed-priority preemptive scheduling of periodic tasks.

b) ____FALSE___ There might exist an alternative single-frequency assignment which consumes less power than the PM-Clock assignment under fixed-priority preemptive scheduling of periodic tasks.

c) ____FALSE___ Assume PM-Clock assigns a frequency of f_1 to task τ_1 and f_2 to τ_2 where τ_1 has higher priority than τ_2. Then, it is guaranteed that $f_1 < f_2$.

d) ____TRUE____ Given a 2-task set of τ_1: (C = 1, T = D = 4) and τ2: (5, 10) under static voltage/frequency scaling with EDF scheduling, no deadlines will be missed if the processor clock frequency is set to f_{max} * 0.8.

e) ____FALSE___ Increasing the voltage at which a CPU is operating lengthens its maximum gate delays.

f) ____TRUE____ A background server will perform no worse than a sporadic server running at the *lowest* priority among all tasks.

g) ____TRUE____ If a deferrable server with a budget of *C* and a period of *T* is schedulable, so is a sporadic server with the same parameters.

h) ____FALSE___ If a sporadic server with a budget of *C* and a period of *T* is schedulable, so is a deferrable server with the same parameters.

Question 2. Is it 'a periodic' server or an 'aperiodic server'? (34 points)

Let each task τ be represented by (C, T=D). Assume that rate-monotonic scheduling is used. Consider two periodic tasks given by τ_1 = (5,10) and τ_2 = (2,20). A server is used to service aperiodic tasks at the highest priority in the system: the server task parameters are (C = 2, T = 5). **Note: The server's budget is set to be 2 at time 0 in all cases,** i.e. the release time of the server is 0.

Please answer the following questions. (Duplicate timelines are provided in the scratch paper if you want to try different things – you may want to think first rather than trial and error). Make sure you draw the budget changes of each server with respect to the timeline.

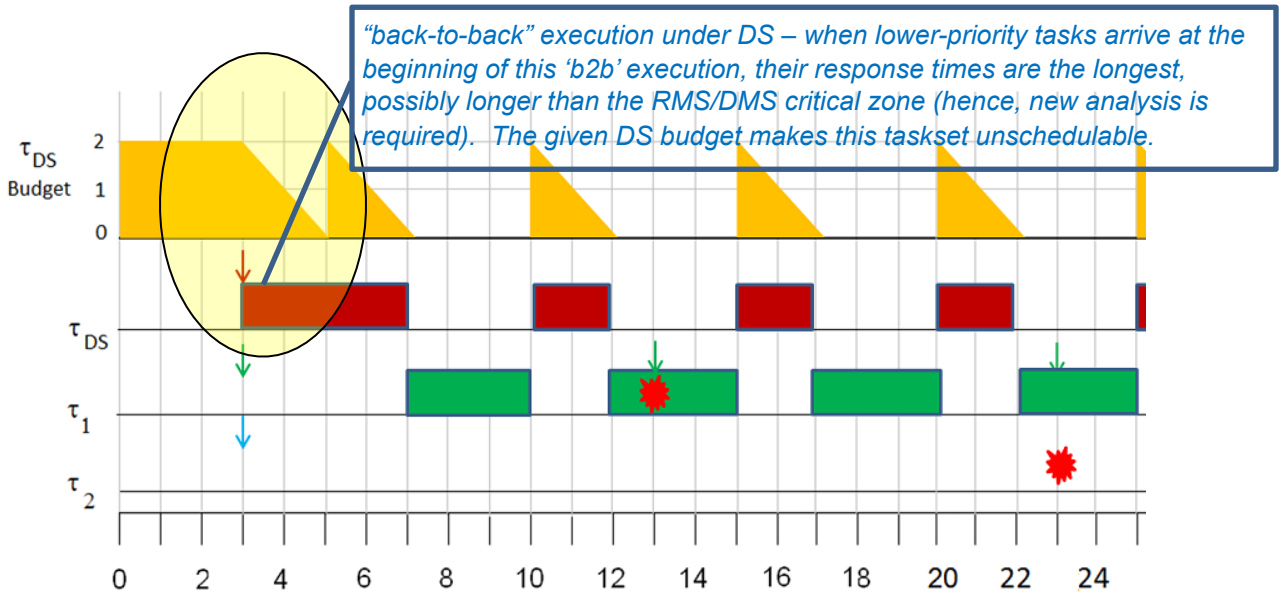
(a) Suppose that the aperiodic server is a **Deferrable Server**. Your goal is to have the DS service an aperiodic task but make both periodic tasks miss their deadlines. If it cannot be done, say so and say why. Else, generate one aperiodic task with an appropriate service (computation) time. Specify its release time r and its computation time. And specify the release times for *periodic* tasks τ_1 and τ_2. *Hint*: go back and see the length of the timeline below. **(14 points)**

Aperiodic task A_0 has a release time of r_0 = *3*, computation time C_0 = *minimum of 10*.

Task τ_1 has a release time of r_1 = *3*.

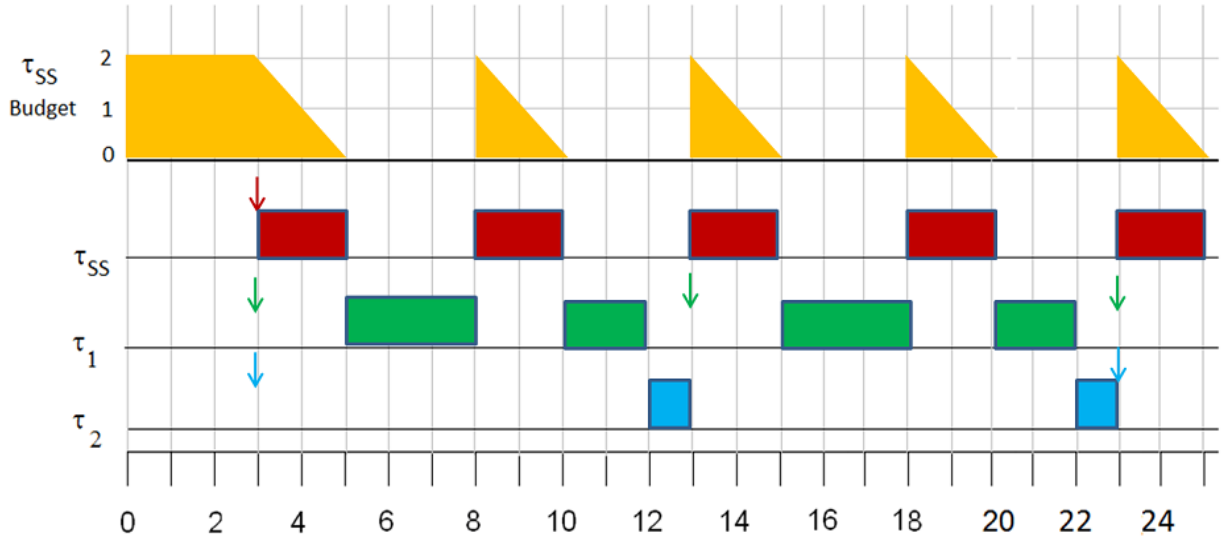
Task τ_2 has a release time of r_2 = *3*

Plot the execution timeline below for the above parameters. Assume that a task that misses its deadline just continues to execute under the normal scheduling policy until completion with later task instances getting lower priority than earlier task instances.



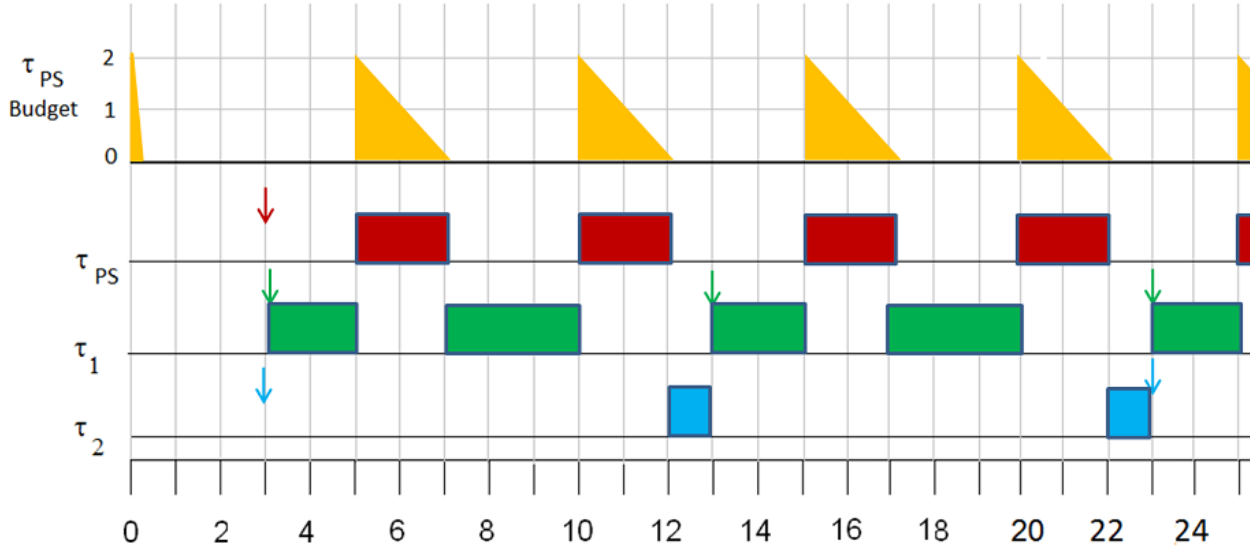
Multiple other answers are possible (3 <= r < 4 , r <= r1 < 4, r1 <= r2 <=5) for forcing τ_1 and τ_2 to miss their deadlines (represented by red exploding stars).

(b) For the parameters you specify in (a) above, draw the corresponding timeline until time 25 if the server is a **sporadic server** (same *C* and *T* as the DS). **(10 points)**



Note that there cannot be any back-to-back execution of τ_{SS} *. The execution of* τ_{SS} *looks like a normal periodic task to* τ_1 *and* τ_2*. Hence, all normal DMS and RMS formulas continue to apply for schedulability tests. At the same time, the aperiodic task started being serviced as soon as it came in at* t = 3.

(c) Repeat the same exercise except that the server is a **polling server** with the same DS parameters **(10 points)**



Again, no back-to-back execution of τ_{PS}. The execution of τ_{PS} looks like a normal periodic task to τ_1 and τ_2. Hence, all normal DMS and RMS formulas continue to apply for schedulability tests. However the aperiodic task had to wait for 2 units of time before it could being execution since it missed the polling at t = 0. Think of the computation times when C_0 is (say) 1.

Question 3. Time to slow down a tad? (25 points)

Consider the following taskset.

Task	*C (ms)*	*T (ms)*	*D(ms)*	*U*
τ_1	2	8	4	0.25
τ_2	4	20	20	0.20
τ_3	8	40	40	0.20
		Total		0.65

a) *Compute the frequency to run the processor at for each task according to the SysClock algorithm for the above taskset. (6 points)*

Task 1:
At t=4, C = ceil(4/8)*2 = 2; Ω = 2/4 = 0.5
Ω_1 = min(0.5) = **0.5**

Task 2:
At t=8: C = ceil(8/8)*2+ceil(8/20)*4 = 6; Ω = 6/8 = 0.75
At t=16: C = ceil(16/8)*2+ceil(16/20)*4 = 8; Ω = 8/16 = 0.5
At t=20: C = ceil(20/8)*2+ceil(20/20)*4 = 10; Ω = 10/20 = 0.5
Ω 2 = min(0.75,0.5,0.5) = **0.5**

Task 3:
t=8: C = ceil(8/8)*2+ceil(8/20)*4+ceil(8/40)*8 = 14; Ω = 14/8 = 1.75
t=16: C = ceil(16/8)*2+ceil(16/20)*4+ceil(16/40)*8 = 16; Ω = 16/16 = 1
t=20: C = ceil(20/8)*2+ceil(20/20)*4+ceil(20/40)*8 = 18; Ω = 18/20 = 0.9
t=24: C = ceil(24/8)*2+ceil(24/20)*4+ceil(24/40)*8 = 22; Ω = 22/24 = 0.9167
t=32: C = ceil(32/8)*2+ceil(32/20)*4+ceil(32/40)*8 = 24; Ω = 24/32 = 0.75
t=40: C = ceil(40/8)*2+ceil(40/20)*4+ceil(40/40)*8 = 26; Ω = 26/40 = 0.65
Ω 3=min(1.75,1,0.9,0.9167,0.75,0.65) = **0.65**

SysClock frequency: V = max(0.5,0.5,0.65) = **0.65**

b) *Compute the frequency to run the processor at for each task in the above taskset according to the PM-Clock algorithm. (10 points)*

The bottleneck task for SysClock is task 3, so there are no lower-priority tasks to slow down and for task 3 to meet its deadlines, all higher-priority tasks must run at a frequency no lower than 0.65

Hence, **PM-Clock**: V1 = V2 = V3 = **0.65**

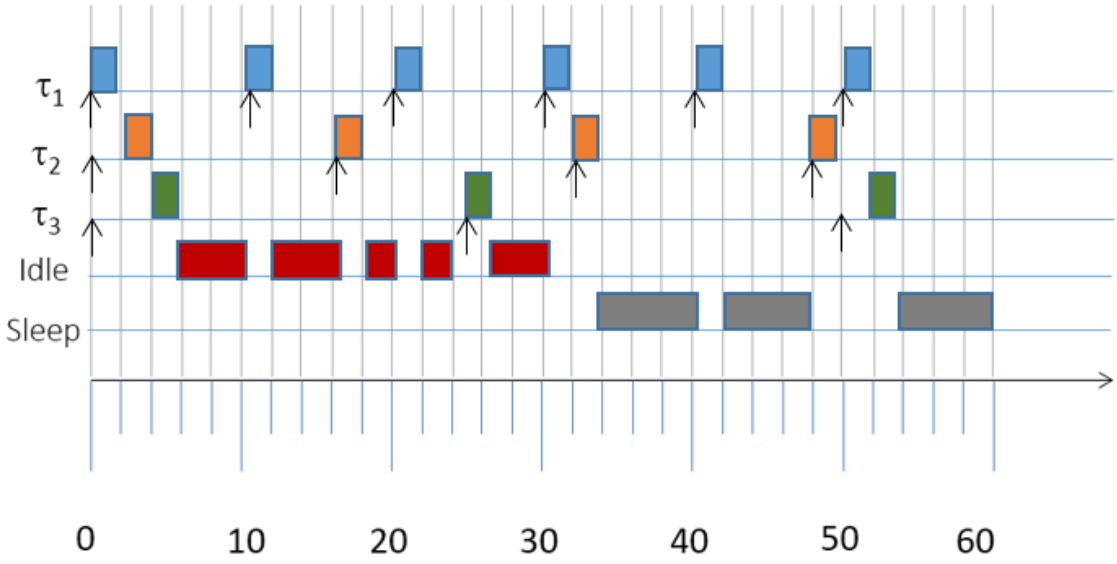
Please show all your work above and fill in your final answers in the following table:

Scheme	**Task τ_1**	**Task τ_2**	**Task τ_3**
SysClock frequency	0.65	0.65	0.65
PM-Clock frequency	0.65	0.65	0.65

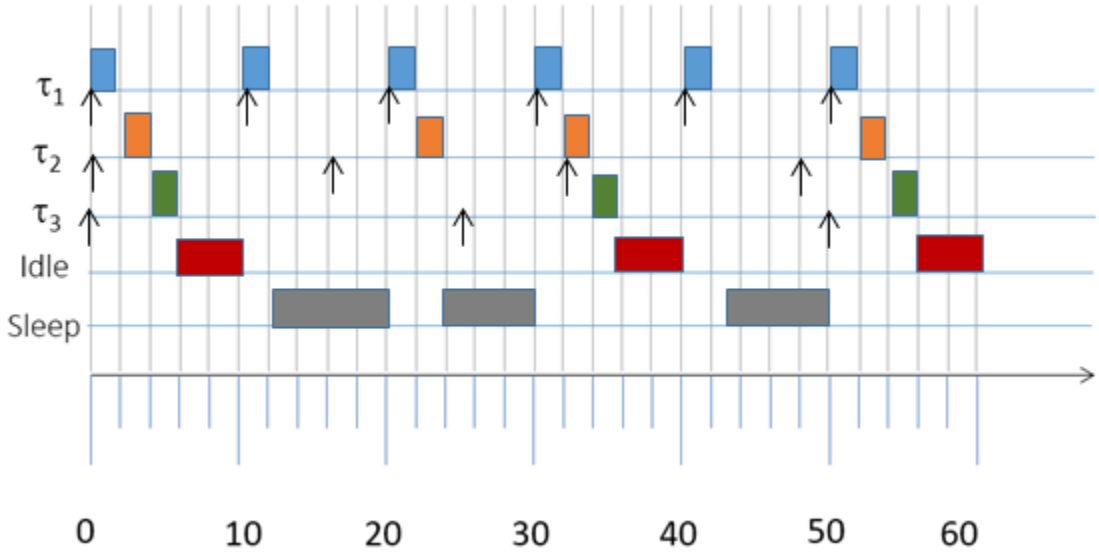
Question 4. Harmonizing Rates (25 points)

Consider the task set {C=2, T=10}, {2, 16} and {2, 25} running on a processor which has an idle state and a sleep state which consume progressively less power than the active state when instructions are executed. The maximum time to get into and come out of the sleep state is 5 time units. **Note**: Assume that all tasks arrive at time 0.

(a) Draw the timeline under a normal RMS schedule from t=0 to 60. ***(5 points)***

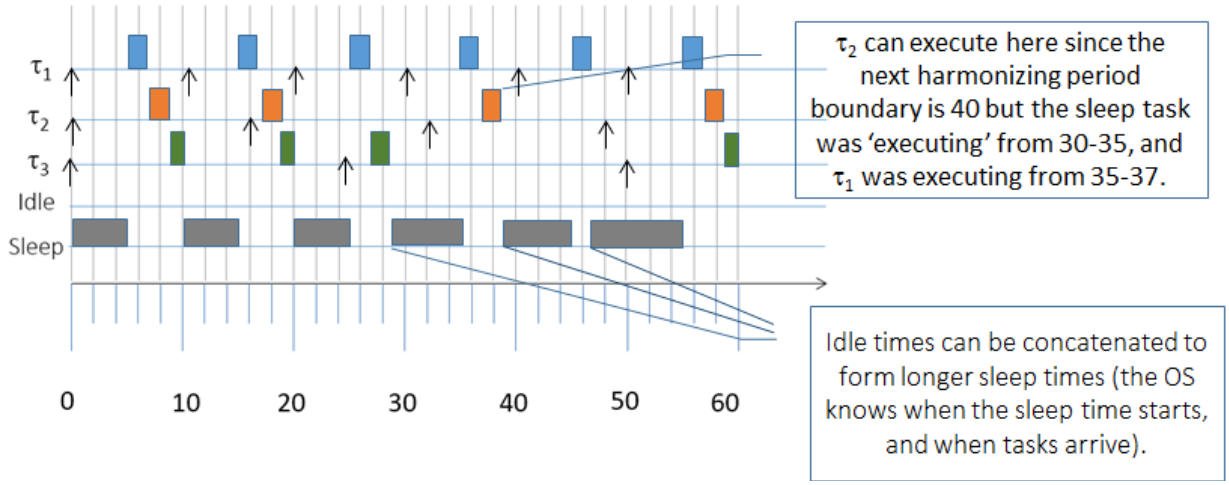


(b) Draw the timeline under rate-harmonized scheduling. Assume a rate-harmonization period of 10 time-units (= T_1). ***(8 points)***



(c) Draw the timeline under energy-saving harmonized scheduling. Assume the same rate-harmonization period as (b) above. ***(10 points)***

- Budget/"execution time" for the Energy Saver task is **5 time units**.



(d) Compare and contrast the time spent in idle/sleep states among the above three schedules. ***(2 points)***

- RMS has the fewest slots used for the sleep state.
- RHS has more sleep state slots than RMS.
- With ES-RHS, all "idle" time is spent in the sleep state. (Any idle slots that show up are *always* contiguous with the execution of a subsequent Energy Save task 'execution', and hence can be combined by the OS by 'looking ahead').