Name: Solutions (Bonus: 2 points)

18-648: Embedded Real-Time Systems

Quiz #4

Fall 2017

60 minutes

Instructions

1. Use the scratch paper at the end for your convenience.
2. Be brief and to the point.
3. Show 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. **Please watch the screen for any clarifications**.
8. Don't get stuck on any single question. Come back to any difficult questions later.

For Graders' Use Only

Name:	_____ / 2 (*Bonus*)
1.	_____ / 20
2.	_____ / 10
3.	_____ / 24
4.	_____ / 22
5.	_____ / 24

TOTAL : _____ / 100

Question 1. True or False. (20 points – 2 points each)

i. False — The Dhall taskset can be globally scheduled by RMS (or EDF) (*fill in this blank before saying True/False*) scheduling policy.

ii. False — Global scheduling for multiprocessors utilizes caches better than partitioned scheduling.

iii. False — Using a single queue ("bank") policy in a multi-server environment scheduling increases average waiting times.

iv. True — Using partitioned scheduling, each node cannot always be scheduled up to 100% using fixed-priority preemptive scheduling.

v. True — Scheduling anomalies can happen with non-preemptive scheduling policies.

vi. True — Period transformation introduces overhead.

vii. False — There are computationally efficient algorithms that yield optimal bin-packing solutions.

viii. True — Known heuristics can only yield near-optimal solutions for the bin-packing problem.

ix. True — Tasks can be split to run on more than one processor with a precedence constraint.

x. False — In a taskset where some tasks are period-transformed, the (Liu & Layland) least upper bound of schedulable utilization is lower than $ln\ 2$.

Question 2. Dhall's Taskset (10 points)

i. *Give the basic parameters of Dhall's taskset for a 3-processor configuration.* ***(3 points)***

An infinite number of task-sets can be given; one example is as shown:

τ_1: $\{C_1 = 2\varepsilon,\ T_1 = 100\}$

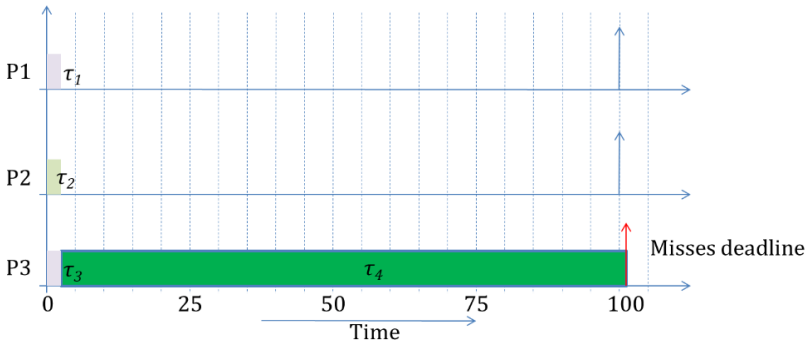
τ_2: $\{C_2 = 2\varepsilon,\ T_2 = 100\}$

τ_3: $\{C_3 = 2\varepsilon,\ T_3 = 100\}$

τ_4: $\{C_4 = 100,\ T_4 = 100+\varepsilon\}$

ii. *Plot the timeline below for either RMS or EDF (state your assumption). Use your own scale.* ***(4 points)***

An RMS schedule for the above example, with assumption that all tasks arrive at same time would look as below:



iii. *What is the lowest utilization at which the Dhall's taskset has feasibility problems in this configuration?* ***(3 points)***

For the above task-set,

Total utilization $= \frac{C_1}{T_1} + \frac{C_2}{T_2} + \frac{C_3}{T_3} + \frac{C_4}{T_4} = \frac{2\epsilon}{100} + \frac{2\epsilon}{100} + \frac{2\epsilon}{100} + \frac{100}{100+\epsilon} \sim =$

1

The per processor utilization is 1.03/3 = 0.3433. You can of course supply very small values for e(=ε) and other (large) values for T_i.

Question 3. What a Pack Job! (24 points)

Consider the following task set with 6 tasks (Assume $D_i=T_i$). Allocate the tasks to processors when tasks cannot be split. Mission: Minimize bin count. Hint: Make sure that the task set in each bin is schedulable by RMS.

τ_1: $\{C_1 = 60, T_1 = 200\}$

τ_2: $\{C_2 = 120, T_2 = 260\}$

τ_3: $\{C_3 = 140, T_3 = 400\}$

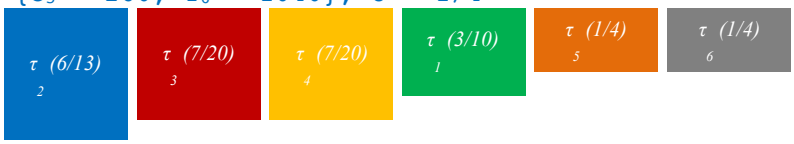
τ_4: $\{C_4 = 280, T_4 = 800\}$

τ_5: $\{C_5 =260, T_5 = 1040\}$

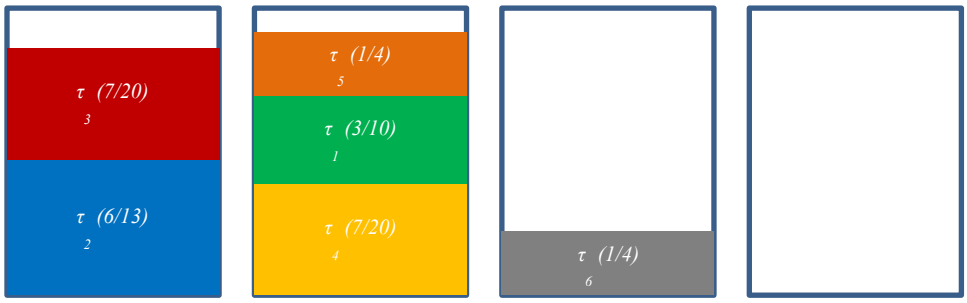
τ_6: $\{C_5 = 260, T_6 = 1040\}$

a) *Show how the tasks can be allocated to processors according to the **First-Fit Decreasing** Heuristic. **(8 points)***

Arranging the tasks in decreasing order of utilization (sizes):
τ_2: $\{C_2 = 120, T_2 = 260\}$, U = 6/13
τ_3: $\{C_3 = 140, T_3 = 400\}$, U = 7/20
τ_4: $\{C_4 = 280, T_4 = 800\}$, U = 7/20
τ_1: $\{C_1 = 60, T_1 = 200\}$, U = 3/10
τ_5: $\{C_5 = 260, T_5 = 1040\}$, U = 1/4
τ_6: $\{C_5 = 260, T_6 = 1040\}$, U = 1/4



81% 90% 25%



Important Notes:

- *Bin 1 is schedulable. Task 2 will complete at 120. Task 3 will complete at time 260 when task 2 comes back.*
- *Bin 2 is schedulable. Task 1 will complete at 60. Task 4 will complete at 400. Task 5 will complete at 780.*
- *Bin 3 is schedulable with only one task having U < 1.0.*

When we add τ_5 to the second bin, it raises the total utilization on this processor to 90%, which is well above the conservative Least-upper-bound for three tasks. So, we compute the exact response time test to see if we can have τ_5 in this bin (you can always the timeline starting with the critical instant where all tasks arrive together):

$$A_5^0 = C_5 = 260$$

$$A_5^1 = 260 + \left\lceil \frac{260}{200} \right\rceil \times 60 + \left\lceil \frac{260}{800} \right\rceil \times 280 = 660$$

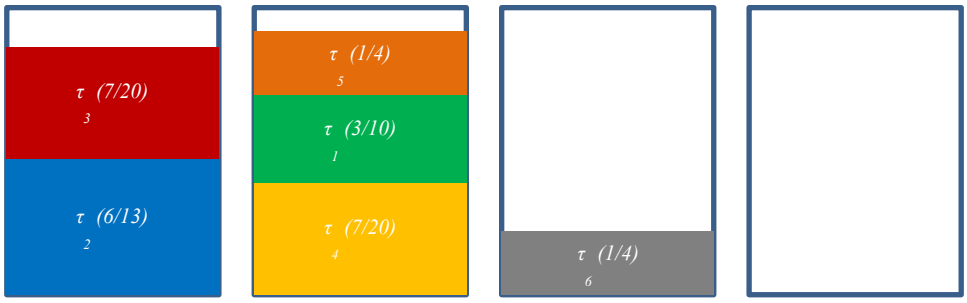
$$A_5^2 = 260 + \left\lceil \frac{660}{200} \right\rceil \times 60 + \left\lceil \frac{660}{800} \right\rceil \times 280 = 780$$

$$A_5^3 = 260 + \left\lceil \frac{780}{200} \right\rceil \times 60 + \left\lceil \frac{780}{800} \right\rceil \times 280 = 780$$

Since τ_5 meets its deadline, we can schedule τ_5 in this bin.

b) *Show how the tasks can be allocated to processors according to* ***Best-Fit Decreasing*** *Heuristic. (****8 points****)*

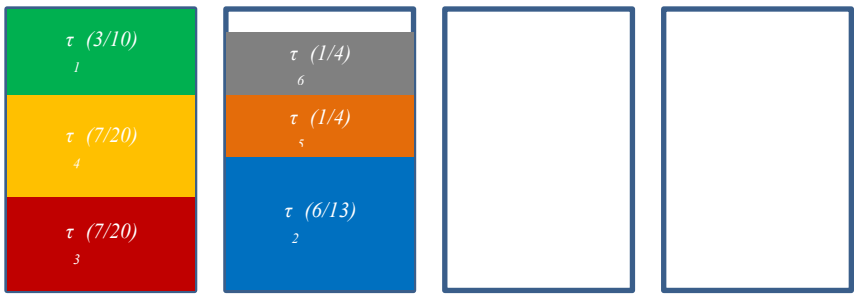
Best-fit decreasing also produces the same allocation as FFD (note that this is NOT always true – can you come up with such an example?).



c) *Using your global "cognitive fit" scheme (i.e. using the amazing power of the neural network in your brain), what is the (best possible)* ***optimal*** ***packing*** *for this taskset, which minimizes the number of required processors? (****8 points****)*

Please see configuration below.

Tasks τ_1, τ_3, τ_4 on Bin 1 have harmonic periods. The tasks on the Bin 2 are also harmonic. So, each bin can go up to 100% utilization. Only two bins are needed now. Three cheers for your brain's neural networks!

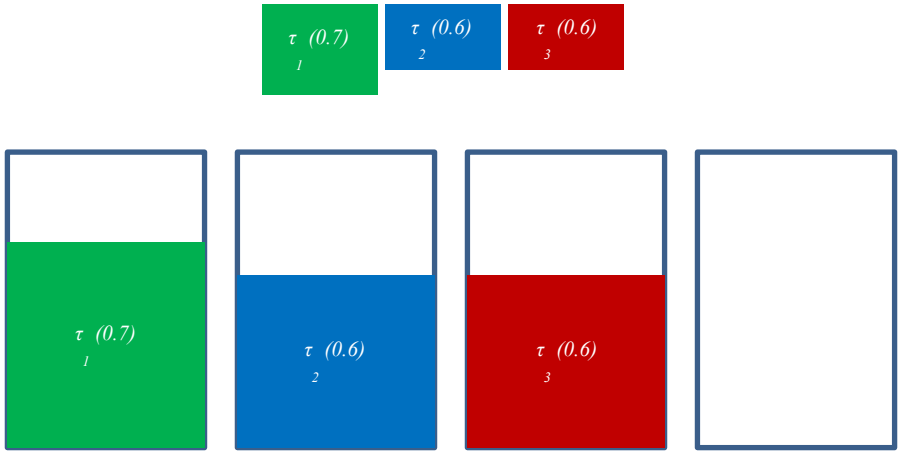

τ (3/10)
1
τ (7/20)
4
τ (7/20)
3
τ (1/4)
6
τ (1/4)
5
τ (6/13)
2

Question 4. Got a Splitting Headache? (22 points)

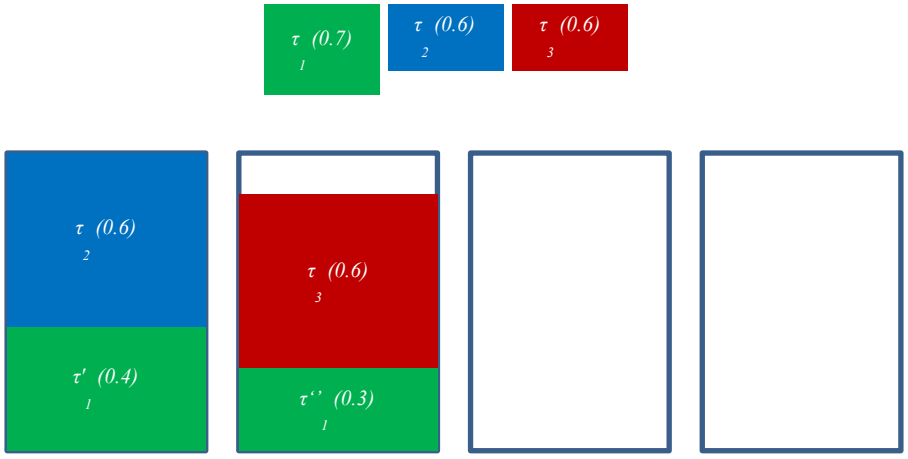
Consider the following task-set with 3 tasks (Assume $D_i = T_i$).

τ_1: {$C_1 = 35, T_1 = 50$} τ_2: {$C_2 = 60, T_2 = 100$} τ_3: {$C_3 = 120, T_3 = 200$}

(a) *Show the task assignment with **First-Fit Decreasing** Heuristics when splitting is not allowed. Use only as many bins as required. **(6 points)***



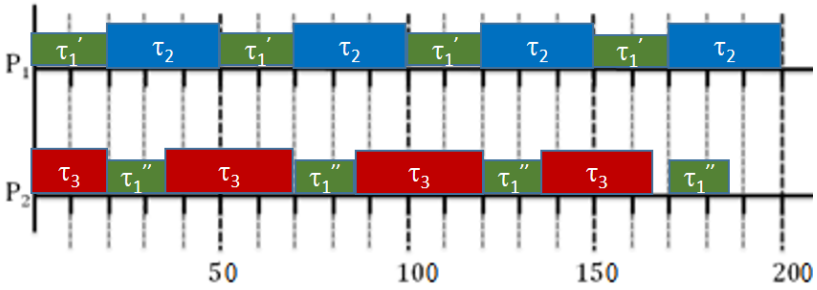
(b) *Show the task assignment with First-Fit Decreasing Heuristics when splitting is allowed (i.e. you "fill" each bin 'completely' as you go, splitting if need be. Decreasing means order by size). Use only as many bins as required. And minimize the number of splits to minimize migration overheads. (State any assumptions you are forced to make). **(8 points)***



Task 1 gets allocated first to bin 1. Task 2 goes into bin 1, but causes an "overflow". We pick Task 1 to split since it is the highest priority task. We can make τ_1' be 40% (20/50 with a deadline of 20) and the remaining piece τ_1'' of 30% (15/50 with an offset of 20, and a relative deadline of 30) goes into bin 2. Task 3 can now be safely allocated to bin 2 as well.

Note that the allocation scenario assumes that there is no context-switching overhead, no migration overhead and no splitting overhead.

(c) *Draw the timeline for the hyperperiod of the taskset assuming that all tasks arrive at time 0.* ***(8 points)***



Alternatively, instead of task 1, task 2 could have been split into two pieces (30,100, D_2')and (30,100, 100-D_2') but the first piece would have consumed the entire deadline of task 2. To remediate, we would have to period-transform task 2 into (30,50), and then (giving it higher priority in breaking the tie with task 1) split it into (15,50,15) and (15,50,35).

Question 5. Oh, the Transformation We Seek! (24 points)

Suppose that I_i represents the importance of a task τ_i. Consider the following taskset:

$\tau_1 = \{C_1 = 15, T_1 = 50\}$ $I_1 = Low$

$\tau_2 = \{C_2 = 50, T_2 = 100\}$ $I_2 = Medium$

$\tau_3 = \{C_3 = 40, T_3 = 200\}$ $I_3 = High$

(a) *Apply the period transformation technique to re-prioritize these tasks such that the transformed task priorities are consistent with task importance. Specify tie-breaking policy if required.* ***(8 points)***

Tasks can be transformed in multiple ways as long as the transformed periods satisfy the following:

$$T_3^* \leq T_2^* \leq T_1^*$$

For, example, one option could be:

τ_1^* = {C_1 = 15, T_1 = 50} I_1 = Low (No transformation)

τ_2^* = {C_2 = 25, T_2 = 50} I_2 = Medium (Transformed by a factor of 2)

τ_3^* = {C_3 = 5, T_3 = 25} I_3 = High (Transformed by a factor of 8)

Another more interesting and efficient option from the point of view of minimizing context-switching overhead could be:

τ_1^* = {C_1 = 15, T_1 = 50} I_1 = Low (No transformation)

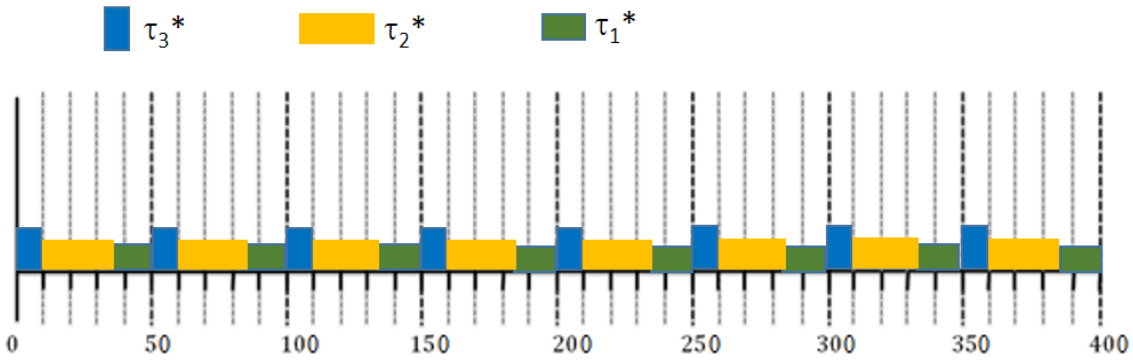
τ_2^* = {C_2 = 25, T_2 = 50} I_2 = Medium (Transformed by a factor of 2)

τ_3^* = {C_3 = 10, T_3 = 50} I_3 = High (Transformed by a factor of 4)

We can break the tie to facilitate $P_3^* \geq P_2^* \geq P_1^*$ where P's represent corresponding task priorities.

(b) *Draw the timeline of the <u>period-transformed taskset</u> WHEN in fact C_1 turns out to be 20 time-units instead of the expected 15 time-units .* ***(8 points)***

Using RMS, as shown below, task τ_1 misses all its deadlines. But thanks to period transformation, this task originally with the shortest period ended up having the lowest priority, and the more important tasks continue to meet their deadlines.



(c) *Can τ_1 ever meet any of its deadlines under your period transformation? Hint: Always know your assumptions.* ***(6 points)***

Task τ_1 can meet its deadlines if tasks τ_2 and τ_3 execute for less than their worst-case execution times. Similarly, instances ("jobs") of task τ_1 can also meet its deadlines if its actual execution duration is shorter than its worst-case execution time.