Real-time Concepts for Embedded Systems
WT 18/19

Assignment 6

Submission Deadline: Sunday, 03.02.2019, 23:55

- *Submit the solution in PDF via Ilias (only one solution per team).*
- *Respect the submission guidelines (see Ilias).*

1 Multiprocessor Priority Ceiling Protocol [10 points]

a) [10 points] Consider a system with four processors (P_1, P_2, P_3, $and P_4$) and six tasks (T_1 to T_6), as shown in Figure 1. The order of task priorities are determined in the order of task indices, i.e. T_1 has the highest priority and T_6 has the lowest priority. In the figure, X_i represents local resources and G_i represents global resources.

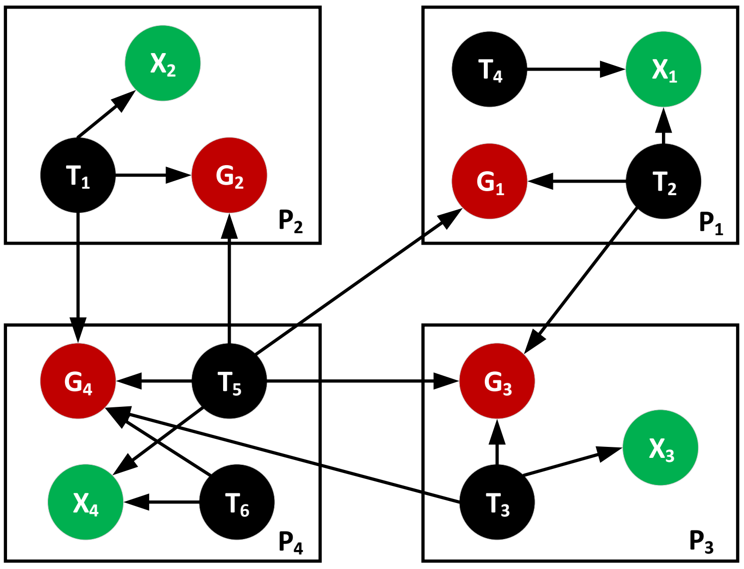

Figure 1: Multiprocessor System

The critical section and other relevant parameters of the system are given below.

T_1 : $[X_2{:}1][G_2{:}2][G_1{:}0.5][G_4{:}1][G_2{:}5];3p_1 = p_5;\ e_{1,L} = 2$

T_2 : $[G_1{:}3][X_1{:}5][G_3{:}1];p_1 = p_5$

T_3 : $[X_3{:}3][G_4{:}2][G_3{:}2];2p_1 = 5p_5$

T_4 : $[X_1{:}2][X_1{:}6]$

$\boxed{T_5 : [X_4{:}1][G_1{:}3][X_4{:}2][G_2{:}2][X_4{:}1][G_3{:}2][X_4{:}2][G_4{:}2]}$

T_6 : $[G_4{:}2][X_4{:}1][X_4{:}4]$

Now, Calculate the blocking time $\mathbf{b_5(rc)}$ of task T_5?

2 Synchronization Protocols [9 points]

a) [4 points] Consider a multiprocessor system with three processors (P_1, P_2, and P_3). A task-set $\mathbb{T}_1 = \{T_1, T_2, T_3, T_4\}$ along with its visit sequence is given in Table 1 and is to be scheduled with the Greedy Synchronization Protocol. The order of task priorities are determined in the order of task indices as shown below. Figure 2 also shows the precedence constraints among subtasks of the task-set. Show using a timing diagram whether each of the tasks in $\mathbb{T}_1$ is schedulable.

$$\pi_1 > \pi_{2,1}, \pi_{2,2} > \pi_{3,1}, \pi_{3,2}, \pi_{3,3} > \pi_4$$

Note: Show your scheduling for the tasks upto 15 time units.

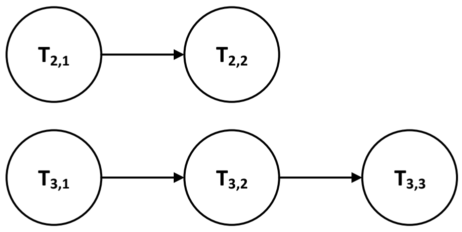

Figure 2: Precedence constraints

Table 1: Task Set $\mathbb{T}_1$

Task	**Release Time**	**Execution Time**	**Period/Relative Deadline**	**Visit Sequence**
T_1	0	2	6	P_1
$T_{2,1}$	0	3	6	P_2
$T_{2,2}$	0	3	6	P_3
$T_{3,1}$	0	1	8	P_1
$T_{3,2}$	0	2	8	P_2
$T_{3,3}$	0	1	8	P_3
T_4	2	4	11	P_3

b) [5 points] Consider the same multiprocessor system and the task-set from the previous part. Now, show using a timing diagram whether the task-set is schedulable using the *Release Guard Protocol*. Indicate the values of $RG(T_{i,k})$ whenever it changes.

Note: Show your scheduling for the tasks upto 15 time units.

Hint: Definition of Idle state: For this question, a processor is said to be idle, if there is no task/sub-task waiting in the ready queue.

3 Task Assignment [31 points]

In this question, task assignment in multiprocessor systems is addressed, using the ILP formulation presented in the lecture. We want to compute an optimal task assignment for the task set $\mathbb{T}_2$ from Table 2 considering the communication costs and the interference costs. Communication costs between the various tasks are are given in in Figure 5. If there is no edges between any two tasks, they do not communicate with each other.

Table 2: Task Set $\mathbb{T}_2$

Task	**Execution Time**	**Period/Deadline**
1	1	3
2	1.5	7.6
3	2.3	8.3
4	1.6	9.3
5	2.2	7.6
6	1.5	9.8
7	2.1	12.3
8	2.3	8.7
9	2.6	11.3
10	2.1	10.9
11	1.8	7.3
12	0.4	5

We will again use **zimpl** to model the ILP formulation. In the provided zip-file, you find the following files:

(1) communication_cost.lst Each line contains the non-zero communication cost between a pair of tasks in different modules in the following format:

```
task_i task_j communication_cost_between_i_j
...
```

(2) interference_cost_finite.lst Each line contains the non-zero interference cost between a pair of tasks in the same module in the following format:

```
task_i task_j interference_cost_between_i_j
...
```

If not noted otherwise, interference costs are 0, i.e. this file can be left blank. Note, that only finite values are allowed.

(3) m.dat Contains the number of modules. The number of modules is imported into the parameter `m`.

(4) max_util.dat Contains the maximum utilization per module. Note: In this question the maximum allowed utilization is equal for all modules and imported into the parameter `max_util`.

(5) task_util.lst Contains the utilization of the tasks in the following format:

```
task_i u_i
task_j u_j
...
```

(6) multi_processor_task_assignment.zpl Skeleton of the zimpl model of the ILP for task assignment, contains import statements for the files mentioned above.

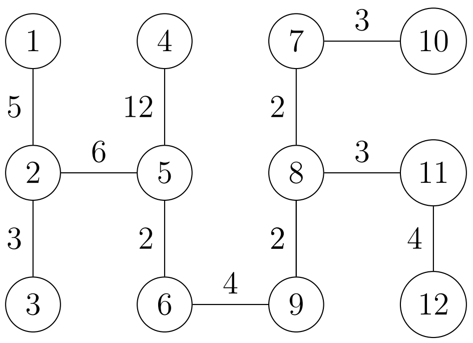

Figure 5: Communication costs between tasks in different modules.

a) [2 points] In the lecture, bin-packing heuristics such as First-Fit, Best-Fit or Rate-Monotonic First Fit for task assignment have been presented. These heuristics have much lower runtime, albeit are not guaranteed to provide optimal solutions. Which aspects of a scheduling/task assignment problem (execution time, precedence constraints, communication costs, resource access cost) are considered directly by these heuristics? Justify your answer.

b) [2 points] Complete `task_util.lst` with the remaining task utilizations. Round each utilization to two trailing digits.

c) [2 points] Complete `communication_cost.lst` with the remaining values for inter-task communication costs (cf. Figure 5). Only non-zero values are required.

d) Complete the zimpl ILP model for task assignment as follows

   i. [2 points] Add an auxiliary function `kronecker_delta(i,k)` for the Kronecker delta $\delta_{i,k}$ which returns 1, if $i = k$ and returns 0 otherwise. This function shall be used later on in the definition of the objective. *Hint: helper functions can be defined in zimpl using the keyword* `defnumb`.

   ii. [3 points] Add ILP variables to encode the assignment of modules to tasks. As in the ILP provided in the lecture, define a set of binary variables $A[i,k]$. We define that task i is assigned to module k, if $A[i,k] = 1$, else $A[i,k] = 0$. Additionally, define a set of (auxiliary) binary variables $A[i,k,j,l]$ with i, j in the set of task indices and k, l in the set of module indices. Variable $A[i,k,j,l]$ shall be used later on in the definition of the objective, since the expression $A[i,k] * A[j,l]$ from the objective function provided in the lecture is not linear but quadratic.

   iii. [4 points] Add the constraint which ensures that *every* task is assigned to exactly one module.

iv. [4 points] Add the constraint which ensures that the total utilization of each module is less or equal to the maximum allowed utilization of each module.

v. [3 points] Add the constraint which ensures that

$$A[i,k,j,l] = A[i,k] * A[j,l].$$

That is, $A[i,k,j,l] = 1$ if task i is assigned to module k and task j is assigned to module l, otherwise $A[i,k,j,l] = 0$.
Hint: The `vif`-construct provided by zimpl may be helpful.

vi. [3 points] Add the objective function which optimizes interference costs and communication costs. Use the helper function `kronecker_delta(i,k)` and $A[i,k,j,l]$ from the previous question parts.

e) [3 points] For this question part, consider only three modules and no interference cost for tasks in the same module. What is the lowest value for `max_util` with which a feasible solution can be obtained?

Note: It is sufficient to give the value `max_util` with precision of two trailing digits.

f) [3 points] Assume the task assignment shall assign task 1 and task 2 to different modules. However, assume that your ILP toolchain does not allow parameters with value ∞, thus you use "very large values" for the interference cost of $I(T_1, T_2)/I(T_2, T_1)$ and an additional (auxiliary constraint).

Briefly describe this auxiliary constraint and how to choose the values of $I(T_1, T_2)/I(T_2, T_1)$ (no formula required), such that you can ensure that the solver will either return a solution with a task assignment where task 1 and task 2 are assigned to different modules, or the problem is infeasible.