RDCS423 Extra Tutorial Problems & Solutions #1 - Real-Time Scheduling Theory

1. Consider a more comprehensive industrial example (from Briand & Roy, 1999) for a new vital signs medical monitoring instrument. The plan is to use an off-the-shelf VMTR2a-100 board (PowerPC 604). The Ada code is compiled with Alsys Ada V5.7.2, and the Ada runtime will run on LynxOS 2.4. The task characteristics are:

   Task t_1: $C_1 = 2.0$ ms; $T_1 = 10$ ms
   Task t_2: $C_2 = 3.0$ ms; $T_2 = 11$ ms
   Task t_3: $C_3 = 2.0$ ms; $T_3 = 36$ ms
   Task t_4: $C_4 = 7.5$ ms; $T_4 = 40$ ms

   For this system, the overhead for the runtime and operating system is measured at 19% per KHz which translates to an execution time loss of 0.19 msec for every task.

   Apply the Utilization Bound Theorem (UBT) to determine if these tasks are schedulable using a rate monotonic scheduling strategy and, if not, apply the less conservative Completion Time Theorem (CTT).

2. Suppose that another task has to be added to check the hardware periodically, and this task has an execution time of 1 ms. What is the maximum invocation rate of this task so that the task set is still schedulable.

3. Suppose further that the rate determined above for the monitoring task isn't high enough and there is some concern about timely fault detection. What can de done to maintain schedulability of the task set when the additional monitoring task must have a minimum rate of 200Hz.

Solution:

1. Compute the utilizations for each task:

   Task t_1: $C_1 = 2.19$ ms; $T_1 = 10$ ms $\rightarrow$ $U_1 = 0.219$
   Task t_2: $C_2 = 3.19$ ms; $T_2 = 11$ ms $\rightarrow$ $U_2 = 0.290$
   Task t_3: $C_3 = 2.19$ ms; $T_3 = 36$ ms $\rightarrow$ $U_3 = 0.061$
   Task t_4: $C_4 = 7.69$ ms; $T_4 = 40$ ms $\rightarrow$ $U_4 = 0.192$

   i.e. $U_{total} = 0.762$. The upper bound from UBT is:
   $U(4) = 4(2^{1/4}-1) = 0.757$
   which is less than the total utilization of these tasks $\rightarrow$ all four tasks may not meet their deadlines.

   Before applying CTT we can just consider the three shortest period tasks and note that the utilisation is 0.219 + 0.290 + 0.061 = 0.472 so these three tasks must be schedulable as they meet the bound of U(3) = 0.780.

   We can now focus CTT on task 4, i.e. with $i = 4$ and checking scheduling points $p = 1$ to 4. Starting with a task set including all tasks from task 1, i.e. $k = 1$:

$$(k, p) = (1,1) \Rightarrow \; C_1\lceil (1)T_1/T_1 \rceil + C_2\lceil (1)T_1/T_2 \rceil + C_3\lceil (1)T_1/T_3 \rceil + C_4\lceil (1)T_1/T_4 \rceil \le (1)T_1$$

$$C_1\lceil (1)10/10 \rceil + C_2\lceil (1)10/11 \rceil + C_3\lceil (1)10/36 \rceil + C_4\lceil (1)10/40 \rceil \le (1)T_1$$
$$C_1 + C_2 + C_3 + C_4 \le T_1 \;\Rightarrow\; 2.19 + 3.19 + 2.19 + 7.69 > 10$$

$$(k, p) = (1,2) \Rightarrow \; C_1\lceil (2)T_1/T_1 \rceil + C_2\lceil (2)T_1/T_2 \rceil + C_3\lceil (2)T_1/T_3 \rceil + C_4\lceil (2)T_1/T_4 \rceil \le (2)T_1$$
$$C_1\lceil (2)10/10 \rceil + C_2\lceil (2)10/11 \rceil + C_3\lceil (2)10/36 \rceil + C_4\lceil (2)10/40 \rceil \le (2)T_1$$
$$2C_1 + 2C_2 + C_3 + C_4 \le 2T_1 \;\Rightarrow\; 4.38 + 6.38 + 2.19 + 7.69 > 20$$

$$(k, p) = (1,3) \Rightarrow \; C_1\lceil (3)T_1/T_1 \rceil + C_2\lceil (3)T_1/T_2 \rceil + C_3\lceil (3)T_1/T_3 \rceil + C_4\lceil (3)T_1/T_4 \rceil \le (3)T_1$$
$$C_1\lceil (3)10/10 \rceil + C_2\lceil (3)10/11 \rceil + C_3\lceil (3)10/36 \rceil + C_4\lceil (3)10/40 \rceil \le (1)T_1$$
$$3C_1 + 3C_2 + C_3 + C_4 \le 3T_1 \;\Rightarrow\; 6.57 + 9.57 + 2.19 + 7.69 < 30$$

We can stop here as we have found that at the third scheduling point the task set is schedulable (and if we hadn't ensured that the first three tasks where schedulable through application of UBT, then we should also apply the remainder of CTT).

2. We would be most interested to check the effect of this additional task on task 4. If we use the CTT again for the task 4 deadline (which is the same as the 4th scheduling point of task 1):

$$(k, p) = (1,4) \Rightarrow \; C_1\lceil (4)T_1/T_1 \rceil + C_2\lceil (4)T_1/T_2 \rceil + C_3\lceil (4)T_1/T_3 \rceil + C_4\lceil (4)T_1/T_4 \rceil \le (4)T_1$$
$$C_1\lceil (4)10/10 \rceil + C_2\lceil (4)10/11 \rceil + C_3\lceil (4)10/36 \rceil + C_4\lceil (4)10/40 \rceil \le (4)10$$
$$4C_1 + 4C_2 + 2C_3 + C_4 \le 4T_1 \;\Rightarrow\; 8.76 + 12.76 + 4.38 + 7.69 = 33.59$$

This is less than the deadline for task 4 (which is also 40 msec), and it shows that there is a 6.41 msec 'slack' which could be used to fit another task in. The additional task has an effective execution time of 1.19 msec (including overhead), and it could run 6.41/1.19 = 5.39 times (or 5, rounding appropriately). Running 5 times within 40 msec is an 8 msec re-invocation period which now places this task as the highest priority task. Having added this task and reordering we need to check the four highest priority tasks, which we can do via UBT first:

Task t_1: $C_1 = 1.19$ ms; $T_1 = 8$ ms $\rightarrow$ $U_1 = 0.149$
Task t_2: $C_2 = 2.19$ ms; $T_2 = 10$ ms $\rightarrow$ $U_2 = 0.219$
Task t_3: $C_3 = 3.19$ ms; $T_3 = 11$ ms $\rightarrow$ $U_3 = 0.290$
Task t_4: $C_4 = 2.19$ ms; $T_4 = 36$ ms $\rightarrow$ $U_4 = 0.061$
Task t_5: $C_5 = 7.69$ ms; $T_5 = 40$ ms $\rightarrow$ $U_5 = 0.192$

$U_{four\ top\ priority\ tasks} = 0.149 + 0.219 + 0.290 + 0.061 = 0.719$ which is less than the U(4) = 0.757 bound so all these are schedulable, and we know from limiting the number of times the new task 1 runs, that the lowest priority task should still be schedulable by CTT. As a final check now with all five tasks:

$(k, p) = (1,5) \Rightarrow$

$$C_1\lceil (5)T_1/T_1 \rceil + C_2\lceil (5)T_1/T_2 \rceil + C_3\lceil (5)T_1/T_3 \rceil + C_4\lceil (5)T_1/T_4 \rceil + C_5\lceil (5)T_1/T_5 \rceil \le (5)T_1$$
$$C_1\lceil (5)8/8 \rceil + C_2\lceil (5)8/10 \rceil + C_3\lceil (5)8/11 \rceil + C_4\lceil (5)8/36 \rceil + C_5\lceil (5)8/40 \rceil \le (5)8$$
$$5C_1 + 4C_2 + 4C_3 + 2C_4 + C_5 \le 5T_1 \;\Rightarrow\; 5.95 + 8.76 + 12.76 + 4.38 + 7.69 = 39.54 \le 40$$

3. There are essentially three measures (employed alone or in combination) that can be taken to maintain schedulability:

    a. Reduce execution times – starting with the task with largest execution time.
    b. Increase the task invocation periods – starting with the task with largest invocation periods.
    c. Allocate harmonic periods where possible.