Embedded Systems FS 2017

Solution to Exercise 2: Aperiodic Scheduling - RT Systems

Discussion Date: 22.03.2017 / 29.03.2017

Aufgabe 1: EDD

Check whether the Earliest Deadline Due (EDD) algorithm produces a feasible schedule for the following task set, given that all tasks are synchronous and arrive at time $t = 0$.

<table>
<thead>
<tr>
<th></th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
<th>$J_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_i$</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>$D_i$</td>
<td>8</td>
<td>15</td>
<td>3</td>
<td>11</td>
</tr>
</tbody>
</table>

Solution - Task 1

EDD (Earliest Deadline Due) is a scheduling algorithm that minimizes the maximum lateness. The Jackson’s rule says that: given a set of $n$ independent tasks, any algorithm that executes the tasks in order of non-decreasing deadlines is optimal with respect to minimizing the maximum lateness.

Assumptions about the task set for applying EDD:
- tasks have same arrival times (synchronous arrivals)
- tasks are independent

EDD is non-preemptive.

The EDD produces a feasible schedule given in the figure below.

The tasks are scheduled in the order of non-decreasing deadlines as follows: $J_3, J_1, J_4, J_2$. 
Aufgabe 2: LDF

Given the precedence graph in Figure 1 and the following table of task execution times and deadlines, determine the Latest Deadline First (LDF) schedule. Is the schedule feasible?

<table>
<thead>
<tr>
<th></th>
<th>J₁</th>
<th>J₂</th>
<th>J₃</th>
<th>J₄</th>
<th>J₅</th>
<th>J₆</th>
<th>J₇</th>
<th>J₈</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cᵢ</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Dᵢ</td>
<td>5</td>
<td>8</td>
<td>11</td>
<td>15</td>
<td>12</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 1: Precedence graph.

Solution - Task 2

LDF (Latest Deadline First) is a scheduling algorithm that minimizes the maximum lateness. It assumes synchronous task activations and is non-preemptive.

The algorithm to produce an LDF schedule proceeds in two stages: firstly, a precedence graph is constructed. Going from tail to head: among the tasks without successors or whose successors have been all selected, LDF selects the tasks with the latest deadline to be scheduled last. At runtime, tasks are extracted from the head of the queue: the first task inserted in the queue will be executed last.

The queue of tasks scheduled with LDF is as follows: J₁ – J₂ – J₃ – J₅ – J₄ – J₆ – J₇ – J₈ (on the next page, you can find a sketch of how the LDF algorithm proceeds). The LDF schedule is presented below:

Figure 2: LDF schedule.
The LDF algorithm proceeds as depicted above (follow the figures left to right).
Aufgabe 3: EDF

Given are five tasks with arrival times, execution times and deadlines according to the following table. (1)
Determine the Earliest Deadline First (EDF) schedule. Is the schedule feasible?

<table>
<thead>
<tr>
<th></th>
<th>J_1</th>
<th>J_2</th>
<th>J_3</th>
<th>J_4</th>
<th>J_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a_i</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>C_i</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>d_i</td>
<td>16</td>
<td>7</td>
<td>8</td>
<td>11</td>
<td>18</td>
</tr>
</tbody>
</table>

(2) At time \(t = 3\), a new task \(J_x\) arrives with execution time \(C_x = 2\) and deadline \(d_x = 10\). Can you guarantee the schedulability of the task set with this new task or do you have to reject it?

Solution - task 3

EDF (Earliest Deadline First) is optimal in the sense of feasibility (it minimizes the maximum lateness under following assumptions: scheduling algorithm is preemptive, the tasks are independent, and may have arbitrary arrival times). The Horn’s rule says that given a set of \(n\) independent tasks with arbitrary arrival times any algorithm that at any instant executes the task with earliest absolute deadlines among the ready tasks is optimal with respect to the maximum lateness.

(1) The EDF schedule is feasible, and the respective schedule is shown in the Figure 3.

(2) The arrival of a new task \(J_x\) at time point \(t = 3\) still maintains the schedule feasible. We can check this by computing ahead the finishing times of the tasks (with respect to the interesting time points). In the analysis, we consider the tasks in order of increasing deadlines.

We assume an on-line scenario, i.e. in the schedulability test we consider only tasks currently present in the system. The test is performed every time a new task arrives.

From the schedule in the preceding sub-problem, we know that before task \(J_x\) arrives at time \(t = 3\) we have:
- tasks \(J_1\), \(J_2\) and \(J_3\) are feasible
- task \(J_2\) finishes before its deadline at \(t = 3\)

That is, up to \(t = 3\) all active tasks in the system are feasible.

At \(t = 3\) we have three tasks in the system: \(J_1\), \(J_3\) and the new task, \(J_x\). For them we perform the schedulability test:

\[f_0 = t = 3\] (We have absolute deadlines in the problem specification.)

Task \(J_3\): \(f_1 = f_0 + c_3(3) = 3 + 4 = 7 \leq 8\) (ok)

Task \(J_x\): \(f_2 = f_1 + c_x(3) = 7 + 2 = 9 \leq 10\) (ok)
Task $J_1$: $f_3 = f_2 + c_1(3) = 9 + 3 = 12 \leq 16$ (ok)

Thus, at $t = 3$ all tasks in the system are feasible.

The next task to arrive is $J_4$. It arrives at $t = 8$. At $t = 8$, we have thee active tasks: $J_x$, $J_4$ and $J_1$. The schedulability test at $t = 8$ proceeds as follows:

Put $f_0 = t = 8$

Task $J_x$: $f_1 = f_0 + c_x(8) = 8 + 1 = 9 \leq 10$ (ok)

Task $J_4$: $f_2 = f_1 + c_4(8) = 9 + 2 = 11 \leq 11$ (ok)

Task $J_1$: $f_3 = f_2 + c_1(8) = 11 + 3 = 14 \leq 16$ (ok)

Thus, at $t = 8$ all tasks in the system are feasible.

Finally, task $J_5$ arrives at $t = 13$. At $t = 13$, we have two active tasks: $J_1$ and $J_5$. The schedulability test at $t = 13$ proceeds as follows:

Put $f_0 = t = 13$

Task $J_1$: $f_1 = f_0 + c_1(13) = 13 + 1 = 14 \leq 16$ (ok)

Task $J_5$: $f_2 = f_1 + c_5(13) = 14 + 3 = 17 \leq 18$ (ok)

Hence, the whole schedule is feasible. It is given in the figure below:

![Figure 4: EDF schedule (with $J_x$)](image)

**Aufgabe 4: EDF**

Given are seven tasks $A, B, C, D, E, F, G$ with following precedence constraints:

$A \rightarrow C, \ B \rightarrow C, \ C \rightarrow E, \ D \rightarrow F, \ B \rightarrow D, \ C \rightarrow F, \ D \rightarrow G$

All tasks arrive at time $t_0 = 0$, have a common deadline $d = 20$ and the following execution times:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_i$</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

(1) Construct the precedence graph for this task set. Then, modify the release times and deadlines so that EDF* can be used for its scheduling.

(2) Draw the resulting EDF* schedule and compute the average response time of the tasks.

(3) Assume the additional precedence constraint $E \rightarrow A$. Is there still a feasible schedule for the above task set? Justify your answer.
Solution - task 4

EDF* schedules tasks with precedence constraints. Release time and deadline of individual tasks are modified such that all the precedence constraints are satisfied. By doing this, the scheduling problem is transformed into a problem without precedence constraints, which can then be handled by a "normal" EDF scheduler.

There are several points to take into consideration while modifying tasks’ release times and deadlines:

**Release time modification:**
- task must start the execution not earlier than its release time
- task must start the execution not earlier than the minimum finishing time of its predecessors

**Deadline modification:**
- task must finish within its deadline
- task must finish not later than the maximum start time of its successors

(1) The precedence graph for the given task set is:

![Precedence Graph](image)

The modified release times and deadlines are:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1^*$</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>$d_1^*$</td>
<td>11</td>
<td>11</td>
<td>15</td>
<td>15</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

In particular, the release time modification proceeds as follows:

for initial nodes $\{A, B\}$ set $r_A^* = r_A$, $r_B^* = r_B$

$r_C^* = \max\{r_C, \max\{r_A^* + C_A, r_B^* + C_B\}\} = \max\{0, \max\{3, 2\}\} = 3$

$r_D^* = \max\{r_D, r_B^* + C_B\} = \max\{0, 0 + 2\} = 2$

$r_F^* = \max\{r_F, r_C^* + C_C, r_D^* + C_D\} = \max\{0, 3 + 4, 2 + 3\} = 7$

$r_E^* = \max\{r_E, r_C^* + C_C\} = \max 0, 3 + 4 = 7$

$r_G^* = \max\{r_G, r_D^* + C_D\} = \max 0, 2 + 3 = 5$

The deadline modification proceeds as follows:

for the terminal nodes $\{E, F, G\}$ set $d_E^* = d_F^* = d_G^* = 20$

$d_C^* = \min\{d_C, d_E^* - C_E, d_F^* - C_F\} = \min\{20, 20 - 2, 20 - 5\} = 15$

$d_D^* = 15$

$d_A^* = 11$

$d_B^* = 11$
(2) The resulting EDF⁺ schedule (based on the modified release times and deadlines) is presented below.

![EDF⁺ schedule](image)

The response time of a task is defined as the difference between its finishing time and release time. Therefore, the average response time for the above schedule is computed as follows:

\[
T_r = \frac{1}{7} \sum_{i=1}^{7} (f_i - r_i) = \frac{5 + 2 + 12 + 8 + 20 + 18 + 13}{7} = 11.14
\]

Note that for the computation of a task's response time we consider its original (not modified) release time.

(3) No, the task set is no longer schedulable. Under the new conditions, the constraints among tasks A, C and E introduce a cycle in the precedence graph. As a result, none of the three tasks can be executed as first and therefore, no feasible schedule exists.
Aufgabe 5: EDF with Precedence Constraints

Given are eight aperiodic tasks $J_1, ..., J_8$ with their arrival times, deadlines, and execution times as shown in the table below. Task precedence constraints are as follows: $J_1 \rightarrow J_2$, $J_2 \rightarrow J_3$, $J_3 \rightarrow J_4$, $J_5 \rightarrow J_6$, $J_6 \rightarrow J_7$, $J_6 \rightarrow J_8$, $J_2 \rightarrow J_7$, $J_7 \rightarrow J_4$ and $J_8 \rightarrow J_7$.

<table>
<thead>
<tr>
<th></th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
<th>$J_4$</th>
<th>$J_5$</th>
<th>$J_6$</th>
<th>$J_7$</th>
<th>$J_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_i$</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>$d_i$</td>
<td>3</td>
<td>8</td>
<td>15</td>
<td>15</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>$C_i$</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

(a) Construct the precedence graph.

Sample solution:

(b) Using EDF* algorithm, modify the arrival times and deadlines of the tasks in order to make the tasks schedulable under EDF. Enter the modified arrival times and deadlines in the table below.

<table>
<thead>
<tr>
<th></th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
<th>$J_4$</th>
<th>$J_5$</th>
<th>$J_6$</th>
<th>$J_7$</th>
<th>$J_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_i^*$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_i^*$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_i$</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Sample solution:
(c) Assume that the application is executed on a dual-core platform. At any time $t$, both cores execute the two ready tasks ($r_i^* \leq t$) with earliest deadlines (Note: A single task cannot be executed on two cores simultaneously). Using the arrival times and deadlines obtained in (b), construct the EDF schedule in Figure 6.

<table>
<thead>
<tr>
<th></th>
<th>$J_1$</th>
<th>$J_2$</th>
<th>$J_3$</th>
<th>$J_4$</th>
<th>$J_5$</th>
<th>$J_6$</th>
<th>$J_7$</th>
<th>$J_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_i^*$</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>$d_i^*$</td>
<td>3</td>
<td>8</td>
<td>12</td>
<td>15</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>$C_i$</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

![Figure 6: EDF schedule for part (c)](image)

Sample solution: There are several solutions since the tasks can run either on core 0 or core 1.

![Figure 7: EDF schedule solution for part (c)](image)

Makespan of all solutions should be 12.

(d) Now assume that the application is executed on a quad-core platform with the same scheduling rule (4 cores execute the four ready tasks with earliest deadlines). Will executing on the quad-core platform reduce the completion time of the application? Justify your answer with an explanation.

Sample solution: No. $J_4$ cannot be started earlier than time 9. Therefore, the minimum finish time of the application is 12 (finish time for the dual core platform).