Eingebettete Systeme
Echtzeitverhalten und Betriebssysteme

4. Aperiodische Tasks

Overview

- Scheduling with real-time constraints:
  - Table with some known algorithms:

<table>
<thead>
<tr>
<th></th>
<th>Equal arrival times</th>
<th>Arbitrary arrival times</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Independent tasks</strong></td>
<td>EDD (Jackson)</td>
<td>EDF (Horn)</td>
</tr>
<tr>
<td><strong>Dependent tasks</strong></td>
<td>LDF (Lawler)</td>
<td>EDF* (Chetto)</td>
</tr>
</tbody>
</table>

Earliest Deadline Due (EDD)

- **Jackson’s rule**: Given a set of $n$ tasks. Processing in order of non-decreasing deadlines is optimal with respect to minimizing the maximum lateness.
- **Proof concept**:

Example 1:

Earliest Deadline Due (EDD)
Earliest Deadline Due (EDD)

- **Example 2:**

<table>
<thead>
<tr>
<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>C_j</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>d_j</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Earliest Deadline First (EDF)

- **Horn’s rule:** Given a set of n independent tasks with arbitrary arrival times, any algorithm that at any instant executes the task with the earliest absolute deadline among the ready tasks is optimal with respect to minimizing the maximum lateness.

- **Concept of proof:** For each time interval \([t, t+1]\)

it is verified, whether the actual running task is the one with the earliest absolute deadline. If this is not the case, the task with the earliest absolute deadline is executed in this interval instead. This operation cannot increase the maximum lateness.

Earliest Deadline First (EDF)

- **Used quantities and terms:**
  - \(\sigma(t)\) identifies the task executing in the slice \([t, t+1]\)
  - \(E(t)\) identifies the ready task that, at time \(t\), has the earliest deadline
  - \(l_E(t)\) is the time \((\geq t)\) at which the next slice of task \(E(t)\) begins its execution in the current schedule
Earliest Deadline First (EDF)

Guarantee:
- worst case finishing time of task $i$: $f_i = \sum_{k=1}^{i} c_k(t)$
- EDF guarantee condition: $\forall i = 1, \ldots, n \sum_{k=1}^{i} c_k(t) \leq d_i$
- algorithm:

```
Algorithm: EDF_guarantee (J, Jnew)
{ J' = J \cup \{Jnew\}; /* ordered by deadline */
  t = current_time();
  f_0 = 0;
  for (each J_i \in J') {
    f_i = f_{i-1} + c_i(t);
    if (f_i > d_i) return(INFEASIBLE);
  }
  return(FEASIBLE);
}
```

Earliest Deadline First (EDF*)

- The problem of scheduling a set of $n$ tasks with precedence constraints (concurrent activation) can be solved in polynomial time complexity only if tasks are preemptable.
- The EDF algorithm minimizes the maximum lateness in the case of tasks with precedence constraints if the timing parameters are modified adequately.
- By the modification it is guaranteed that if there exists a valid schedule at all then
  - a task starts execution not earlier than its release time and not earlier than the finishing times of its predecessors (a task cannot preempt any predecessor)
  - all tasks finish their execution within their deadlines

Example:
- | J_1 | J_2 | J_3 | J_4 | J_5 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a_1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>c_1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
  | d_1 | 2   | 5   | 4   | 10  | 9

Modification of release times:
- 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 predecessor.
- Solution: $r_j^* = \max(r_j, \max(r_i^* + c_i : J_i \rightarrow J_j))$
Earliest Deadline First (EDF*)

- **Modification of deadlines:**
  - Task must finish the execution time within its deadline
  - Task must finish the execution not later than the maximum start time of its successor

  \[
  \text{task } b \text{ depends on task } a: \quad J_a \rightarrow J_b
  \]

  \[
  \begin{align*}
  f_a & \leq d_a \\
  f_a & \leq d_b - C_b
  \end{align*}
  \]

  \[
  J_a \rightarrow J_b
  \]

- **Solution:** \[d_i^* = \min(d_i, \min\{d_j^* - C_j : J_i \rightarrow J_j\})\]

- **Algorithm for modification of release times:**
  1. For any initial node of the precedence graph set \( r_i^* = r_i \)
  2. Select a task \( j \) such that its release time has not been modified but the release time of all immediate predecessors \( i \) have been modified. If no such task exists, exit.
  3. Set \( r_j^* = \max(r_j, \max(r_i^* + C_i : J_i \rightarrow J_j)) \)
  4. Return to step 2

- **Algorithm for modification of deadlines:**
  1. For any terminal node of the precedence graph set \( d_i^* = d_i \)
  2. Select a task \( i \) such that its deadline has not been modified but the release time of all immediate successors \( j \) have been modified. If no such task exists, exit.
  3. Set \( d_i^* = \min(d_i, \min\{d_j^* - C_j : J_i \rightarrow J_j\}) \)
  4. Return to step 2

Earliest Deadline First (EDF*)

- **Proof concept**
  - Show that if there exists a feasible schedule for the modified task set under EDF then the original task set is also schedulable.
  - To this end, show that the original task set meets the timing constraints. This can be done by using \( d_i^* \leq d_i \), \( r_i^* \geq r_i \).
  - In addition, show that the precedence relations in the original task set are not violated. In particular, show that
    - a task cannot start before its predecessor and
    - a task cannot preempt its predecessor.