Common Approaches to Real-Time Scheduling

- **Clock-driven** (time-driven) schedulers
  - Scheduling decisions are made at *specific time instants*, which are typically chosen *a priori*.

- **Priority-driven** schedulers
  - Scheduling decisions are made when particular events in the system occur, *e.g.*
    - a job becomes available
    - processor becomes idle
  - **Work-conserving**: processor is busy whenever there is work to be done.

Clock-Driven (Time-Driven) -- Overview

- **Scheduling decision time**: point in time when scheduler decides which job to execute next.
  - Scheduling decision time in clock-driven schedulers is defined *a priori*.
  - For example: Scheduler periodically wakes up and generates a portion of the schedule.

- **Special case**: When job parameters are known *a priori*, schedule can be precomputed off-line, and stored as a table (table-driven schedulers).
Priority-Driven -- Overview

- Basic rule: Never leave processor idle when there is work to be done. (such schedulers are also called work conserving)
- Based on list-driven, greedy scheduling.
- Examples: FIFO, LIFO, SET, LET, EDF.

- Possible implementation of preemptive priority-driven scheduling:
  - Assign priorities to jobs.
  - Scheduling decisions are made when
    - Job becomes ready
    - Processor becomes idle
    - Priorities of jobs change
  - At each scheduling decision time, chose ready task with highest priority.

- In non-preemptive case, scheduling decisions are made only when processor becomes idle.

Example: Priority-Driven Non-Preemptive Schedules
Effective Timing Constraints

- Timing constraints often inconsistent with precedence constraints. Example: \( d_1 > d_2 \), but \( J_1 \rightarrow J_2 \)
- Effective timing constraints on single processor:
  - Effective release time: \( r_i^{\text{eff}} := \max\{r_i, \{r_j^{\text{eff}} | J_j \rightarrow J_i \} \} \)
  - Effective deadline: \( d_i^{\text{eff}} := \min\{d_i, \{d_j^{\text{eff}} | J_j \rightarrow J_i \} \} \)

- Theorem: A set of Jobs \( J \) can be feasibly scheduled on a processor if and only if it can be feasibly scheduled to meet all effective release times and deadlines.

Interlude: The EDF Algorithm

- The EDF (earliest-deadline-first) algorithm:
  At any time, execute that available job with the earliest deadline.

- Theorem: (Optimality of EDF) In a system one processor and with preemptions allowed, EDF can produce a feasible schedule of a job set \( J \) with arbitrary release times and deadlines iff such a schedule exists.

- Proof: by schedule transformation.
EDF Not Always Optimal

- Case 1: When preemption is not allowed:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_i$</td>
<td>$d_i$</td>
<td>$e_i$</td>
</tr>
<tr>
<td>$J_1$</td>
<td>(0, 10, 3)</td>
<td></td>
</tr>
<tr>
<td>$J_2$</td>
<td>(2, 14, 6)</td>
<td></td>
</tr>
<tr>
<td>$J_3$</td>
<td>(4, 12, 4)</td>
<td></td>
</tr>
</tbody>
</table>

- Case 1: On more than one processor:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_i$</td>
<td>$d_i$</td>
<td>$e_i$</td>
</tr>
<tr>
<td>$J_1$</td>
<td>(0, 4, 1)</td>
<td></td>
</tr>
<tr>
<td>$J_2$</td>
<td>(0, 4, 1)</td>
<td></td>
</tr>
<tr>
<td>$J_3$</td>
<td>(0, 5, 5)</td>
<td></td>
</tr>
</tbody>
</table>