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

Proc1: $J_1, J_2, J_3, J_5, J_6, J_7, J_8$
Proc2: $J_6, J_7, J_8$

$L = (J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8)$

Proc1: $J_1, J_2, J_3, J_6, J_4$
Proc2: $J_5, J_7, J_8$

LET = $(J_5, J_8, J_2, J_6, J_1, J_3, J_4, J_7)$

Proc1: $J_6, J_7, J_8$
Proc2: $J_1, J_5, J_3$

$L = (J_8, J_1, J_2, J_3, J_4, J_5, J_6, J_7)$
Effective Timing Constraints

- Timing constraints often inconsistent with precedence constraints.
  Example: $d_1 > d_2$, but $J_1 < J_2$

- Effective timing constraints on single processor:
  - Effective release time: $r_i^{eff} := \max\{r, \{r_j^{eff} | J_j < J_i\}\}$
  - Effective deadline: $d_i^{eff} := \min\{d, \{d_j^{eff} | J_j < J_i\}\}$

- 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 if and only if such a schedule exists.

- Proof: by schedule transformation.
## EDF Not Always Optimal

- **Case 1: When preemption is not allowed:**

  \[
  r_i \quad d_i \quad e_i \\
  J_1 = (0, 10, 3) \\
  J_2 = (2, 14, 6) \\
  J_3 = (4, 12, 4)
  \]

- **Case 1: On more than one processor:**

  \[
  r_i \quad d_i \quad e_i \\
  J_1 = (0, 4, 1) \\
  J_2 = (0, 4, 1) \\
  J_3 = (0, 5, 5)
  \]