Common Approaches to Real-Time Scheduling

- **Clock-driven** (time-driven) schedulers
- **Priority-driven** schedulers
- Examples of priority driven schedulers
- Effective timing constraints
- The Earliest-Deadline-First (EDF) Scheduler and its optimality

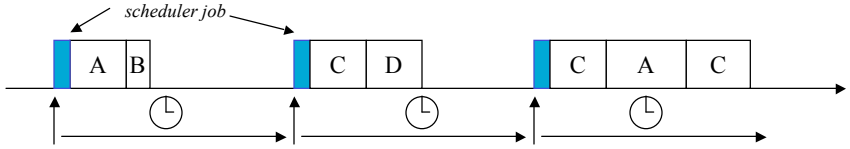
Common Approaches to Real-Time Scheduling

- 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.



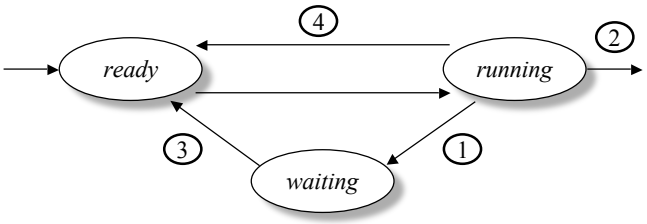
- Special case: When job parameters are known *a priori*, schedule can be pre-computed off-line, and stored as a table (table-driven schedulers).

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.

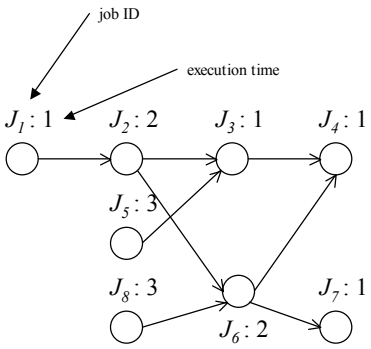
- Possible **implementation** of preemptive priority-driven scheduling:
  - Assign priorities to jobs.
  - Scheduling decisions are made when
    - Job becomes ready
    - Processor becomes idle
    - Priorities of jobs change
  - At each scheduling decision time, choose ready task with highest priority.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

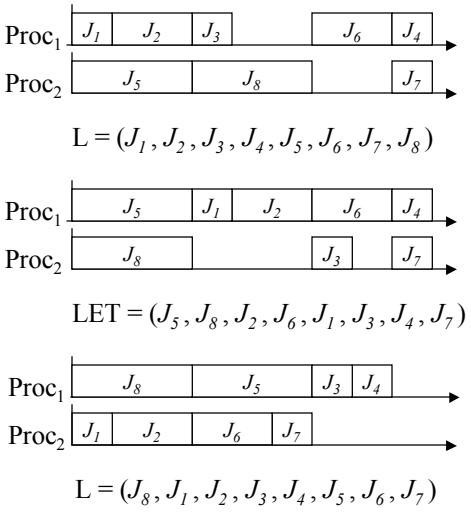
Scheduling Decisions



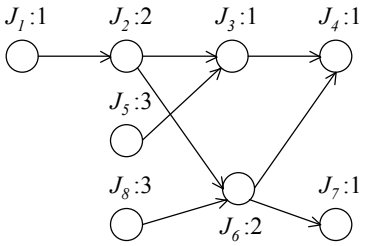
- Scheduling decision points:
  1. The running process changes from *running* to *waiting* (current CPU burst of that process is over).
  2. The running process terminates.
  3. A waiting process becomes ready (new CPU burst of that process begins).
  4. The current process switches from *running* to *ready* .

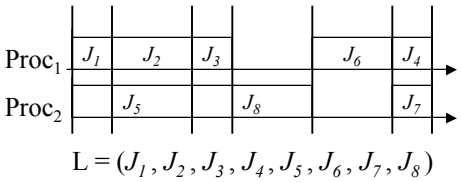
Example: Priority-Driven Non-Preemptive Schedules





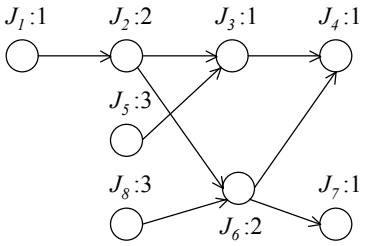
Example: Priority-Driven Non-Preemptive Schedules

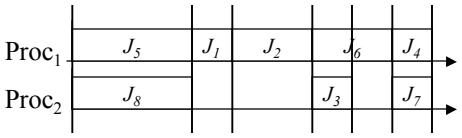




L = $(J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8)$

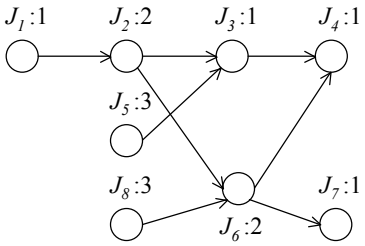
Example: Priority-Driven Non-Preemptive Schedules

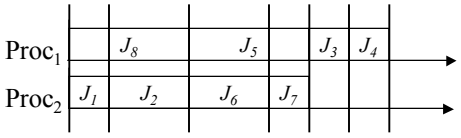




LET = $(J_5, J_8, J_2, J_6, J_1, J_3, J_4, J_7)$

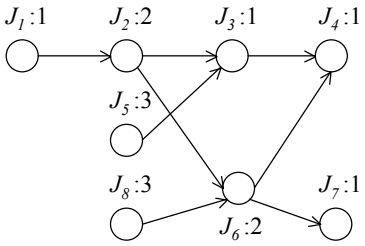
Example: Priority-Driven Non-Preemptive Schedules

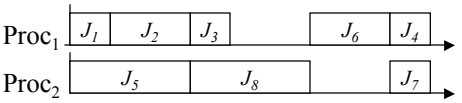




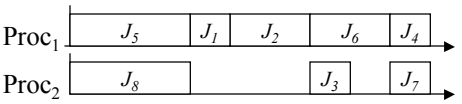
$L = (J_8, J_1, J_2, J_3, J_4, J_5, J_6, J_7)$

Example: Priority-Driven Non-Preemptive Schedules

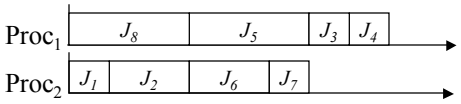




$L = (J_1, J_2, J_3, J_4, J_5, J_6, J_7, J_8)$



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



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

Effective Timing Constraints

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

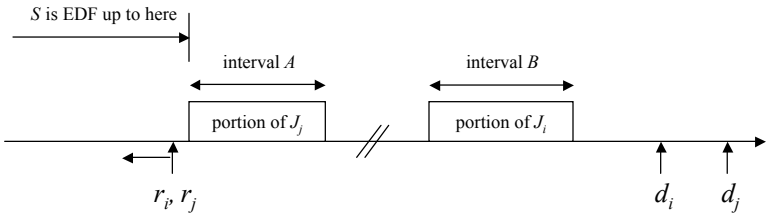
- **Theorem:**

> (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.

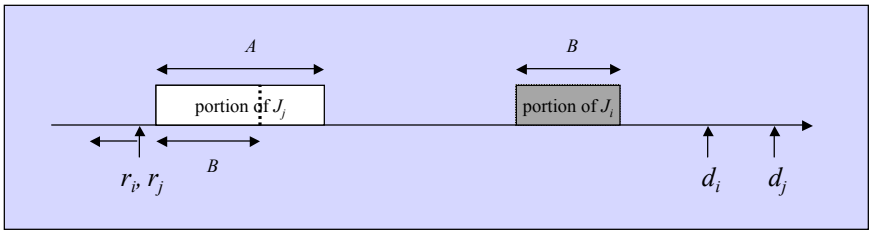
Proof of Optimality of EDF

- Assume that arbitrary schedule *S* meets timing constraints.
- For *S* to not be an EDF schedule, we must have the following situation:



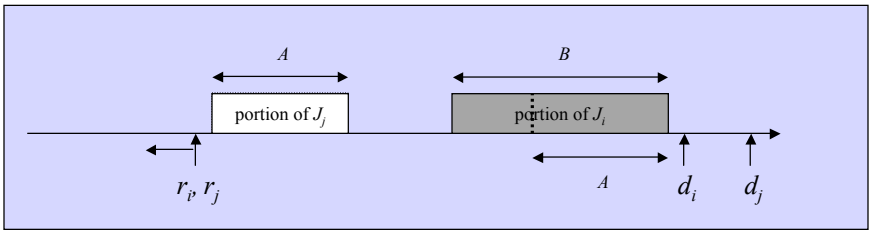
Proof of Optimality of EDF (2)

- We now have two cases.
- Case 1: *L(A) > L(B)*



Proof of Optimality of EDF (3)

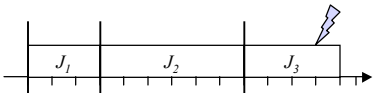
- We now have two cases.
- Case 2: *L(A) <= L(B)*



EDF Not Always Optimal

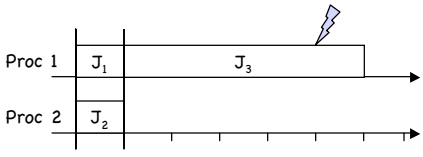
- Case 1: When preemption is not allowed:

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



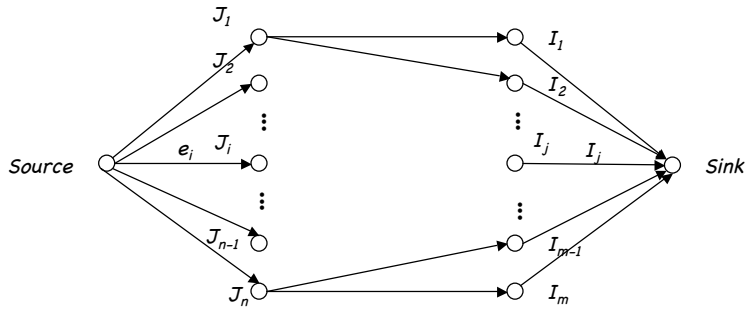
- Case 2: On more than one processor:

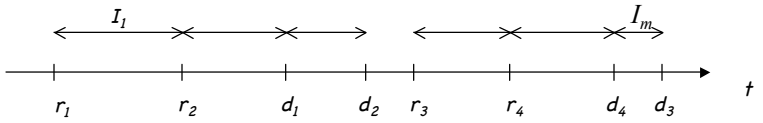
		r_i	d_i	e_i
J_1	= (	0,	4,	1)
J_2	= (	0,	4,	1)
J_3	= (	0,	5,	5)



Preemptive Scheduling of Jobs with Arbitrary Release Times, Deadlines, Execution Times

- Determine schedule over a hyperperiod.
- Formulate scheduling problem as network flow problem.





NP Completeness of Non-Preempt Deadline Scheduling

- **Theorem:** The problem of scheduling a non-preemptable set of jobs $J_1, \ldots, J_i, \ldots J_n$, each with release time r_i, deadline d_i, and execution time e_i is *NP*-complete.

- **Proof:** Transformation from PARTITION [Garey/Johnson,1979]

  Given: Finite set $A = \{A_1, \ldots, A_i, \ldots, A_m\}$, each element of size a_i.

  Let $B = \sum_{i=1}^{m} a_i$

  Partition A into two sets, each of same size.

  Define a job set $J_1, \ldots, J_{m+1}$, as follows:

$$\text{for } i \le i \le m, \text{ define } J_i = \begin{cases} r_i = 0 \\ d_i = B+1 \\ e_i = a_i \end{cases}, J_{m+1} = \begin{cases} r_{m+1} = \lceil B/2 \rceil \\ d_{m+1} = \lceil (B+1)/2 \rceil \\ e_{m+1} = 1 \end{cases}$$