Lecture 19: Real-Time Scheduling II

Seyed-Hosein Attarzadeh-Niaki

Based on the Slides by Edward Lee and Rodolfo Pellizzoni

Review

- Simple RTOS: Microkernel scheduler
- Task model and periodicity
- Preemptive scheduling
- Rate monotonic scheduling
  - Feasibility
  - Optimality

Outline

- Earliest Due Date (EDD) and Earliest Deadline First (EDF) scheduling
  - Optimality
- Precedence Constraints
  - Latest Deadline First (LDF) scheduling
  - EDF* scheduling

Deadline Driven Scheduling: Jackson's Algorithm: EDD (1955)

Given n independent *one-time* tasks with deadlines $d_1 , \ldots , d_n$, schedule them to minimize the maximum *lateness*, defined as

$$L_{\max} = \max_{1 \le i \le n} \{ f_i - d_i \}$$

where f_i is the finishing time of task i. Note that this is negative iff all deadlines are met.

Earliest Due Date (EDD) algorithm: Execute them in order of non-decreasing deadlines.

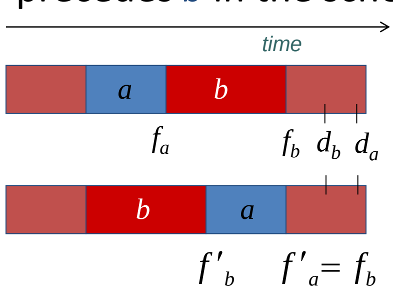
Note that this does not require preemption.

Theorem: EDD is Optimal in the Sense of Minimizing Maximum Lateness

- To prove, use an interchange argument:
  - Given a schedule *S* that is not EDD, there must be tasks *a* and *b* where
    - *a* immediately precedes *b* in the schedule but
    - $d_a > d_b$.
  - Why?
- We can prove that this schedule can be improved by interchanging *a* and *b*.
  - Thus, no non-EDD schedule achieves smaller max lateness than EDD,
  - so the EDD schedule must be optimal.

Consider a non-EDD Schedule S

There must be tasks *a* and *b* where *a* immediately precedes *b* in the schedule but $d_a > d_b$



$$L_{\max} = \max\{f_a - d_a, f_b - d_b\} = f_b - d_b$$

$$L'_{\max} = \max\{f'_a - d_a, f'_b - d_b\}$$

Case 1: $f'_a - d_a > f'_b - d_b$. ($L'_{max} = f'_a - d_a$)
Then: $L'_{max} \leq f'_a - d_a = f_b - d_a \leq L_{max}$
(because: $d_a > d_b$).

Case 2: $f'_a - d_a \leq f'_b - d_b$. ($L'_{max} = f'_b - d_b$)
Then: $L'_{max} \leq f'_b - d_b \leq L_{max}$
(because: $f'_b < f_b$).

Theorem: $L'_{max} \leq L_{max}$.
Hence, S' is no worse than S.

Deadline Driven Scheduling: Horn's algorithm: EDF (1974)

- Extend EDD by allowing tasks to "arrive" (become ready) at any time.

- Earliest deadline first (EDF)
  - 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 all arrived tasks
  - is optimal w.r.t. minimizing the maximum lateness.

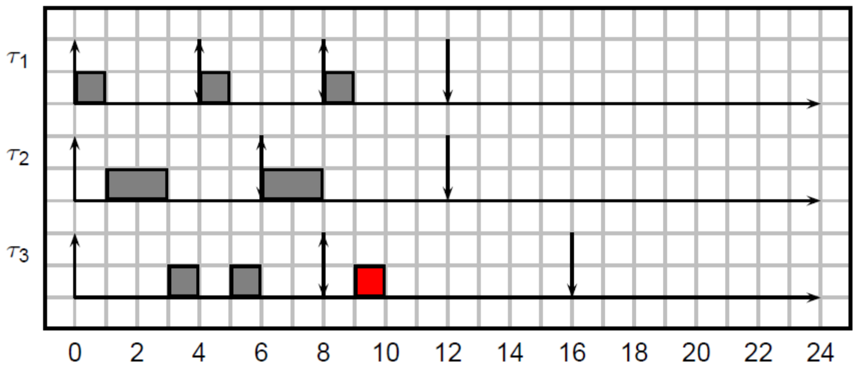
- Proof uses a similar interchange argument.

Using EDF for Periodic Tasks

- The EDF algorithm can be applied to periodic tasks as well as aperiodic tasks.
  - Simplest use: Deadline is the *end of the period*.
  - Alternative use: *Separately specify* deadline (relative to the period start time) and period.

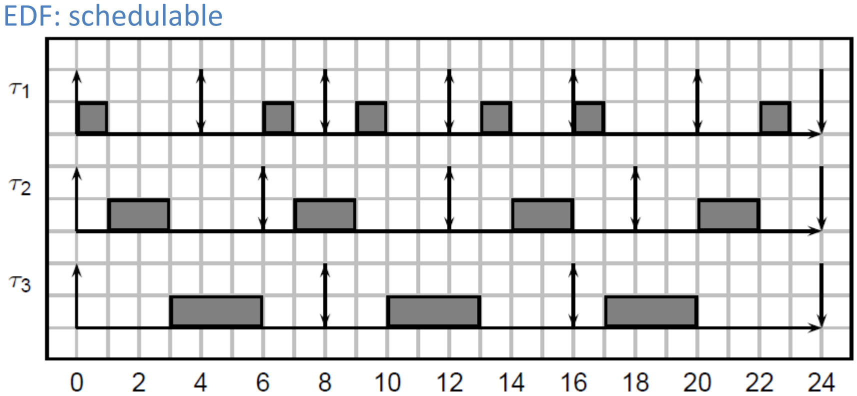
A Comparative Example

- τ_1 (e1 = 1, p1 = 4), τ_2 (e2 = 2, p1 = 6), τ_3 (e3 = 3, p3 = 8)
- Utilization: U = 1/4 + 2/6 + 3/8 = 23/24
- RM: don't know (utilization bound), in reality: not schedulable



A Comparative Example

- τ_1 (e1 = 1, p1 = 4), τ_2 (e2 = 2, p1 = 6), τ_3 (e3 = 3, p3 = 8)
- Utilization: U = 1/4 + 2/6 + 3/8 = 23/24
- EDF: schedulable



RMS vs. EDF? Which one is better?

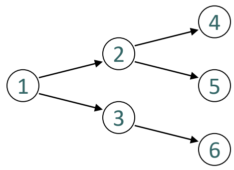
- Favoring RMS
  - Scheduling decisions are simpler
    - fixed priorities vs. the dynamic priorities required by EDF.
    - EDF scheduler must maintain a list of ready tasks that is sorted by priority.
- Favoring EDF
  - Since EDF is optimal w.r.t. maximum lateness, it is also optimal w.r.t. feasibility.
    - RMS is only optimal w.r.t. feasibility.
    - For infeasible schedules, *RMS completely blocks lower priority tasks*, resulting in unbounded maximum lateness.
  - EDF can achieve full utilization where RMS fails to do that
  - EDF results in fewer preemptions in practice, and hence less overhead for context switching.
  - Deadlines can be different from the period.

The Simplest Model

- So far we looked at the *simplest possible model*, where
  - All tasks are periodic
  - Single processor
  - Tasks do not share any resource
- Not very realistic, but instructive (we have simple results)!

- Questions happening in reality:
  - What if there are task *interdependencies*?
  - What happens when you start *sharing resources*?
  - What happens if you schedule a *mix of periodic/aperiodic tasks*?
  - What happens if you use a *multiprocessor*?
  - More *complex task models*?
- We briefly look at some of these issues.

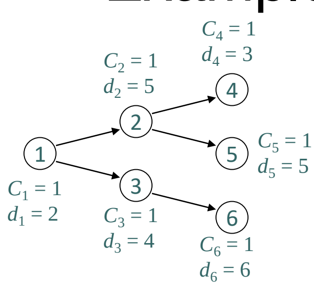
Precedence Constraints

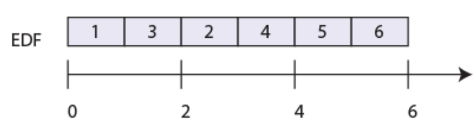


DAG, showing that task 1 must complete before tasks 2 and 3 can be started, etc.

A directed acyclic graph (DAG) shows precedences, which indicate which tasks must complete before other tasks start.

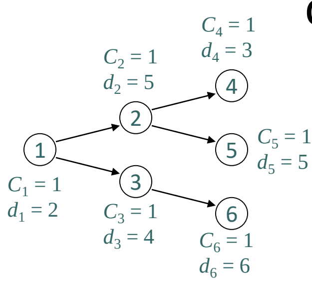
Example: EDF Schedule

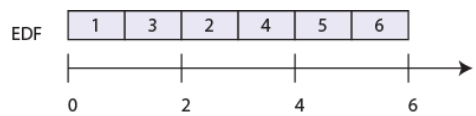




Is this feasible?

EDF is not optimal under precedence constraints



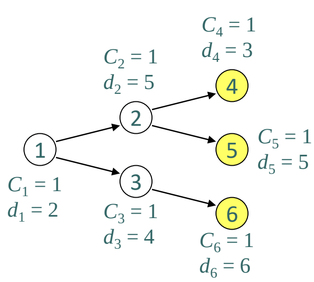


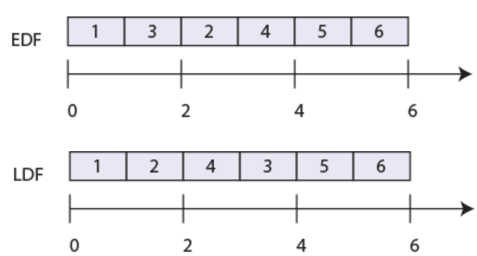
- The EDF schedule chooses task 3 at time 1 because it has an earlier deadline.
- This choice results in task 4 missing its deadline.

Is there a feasible schedule?

Latest Deadline First (LDF)

(Lawler, 1973)

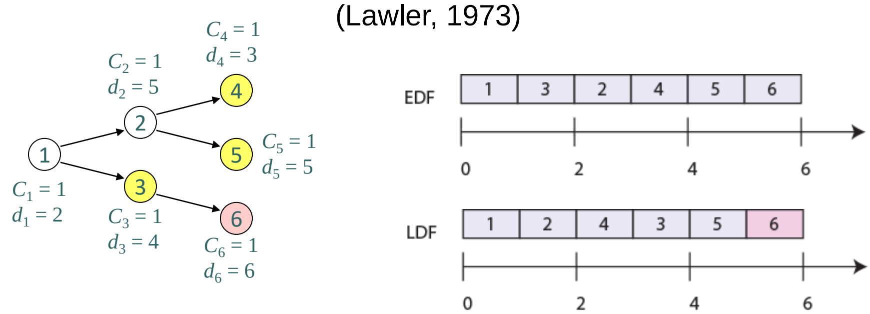




- The LDF scheduling strategy builds a schedule backwards.
- Given a DAG,
  - choose the leaf node with the latest deadline to be scheduled last,
  - and work backwards.

Latest Deadline First (LDF)

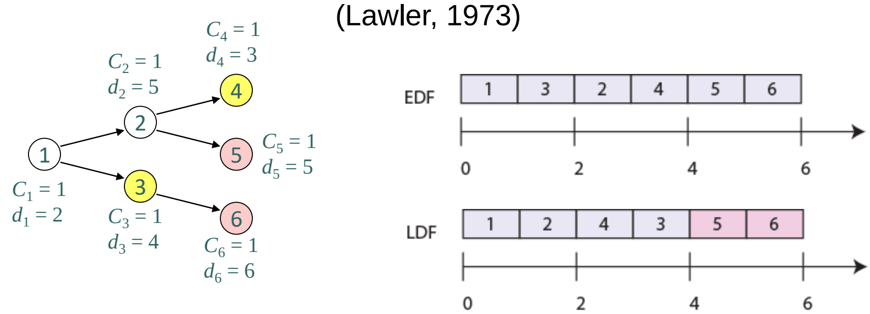
(Lawler, 1973)



- The LDF scheduling strategy builds a schedule backwards.
- Given a DAG,
  - choose the leaf node with the latest deadline to be scheduled last,
  - and work backwards.

Latest Deadline First (LDF)

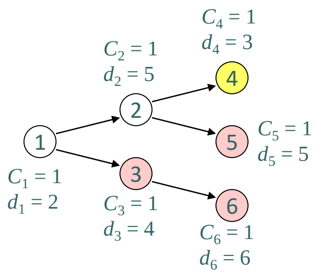
(Lawler, 1973)

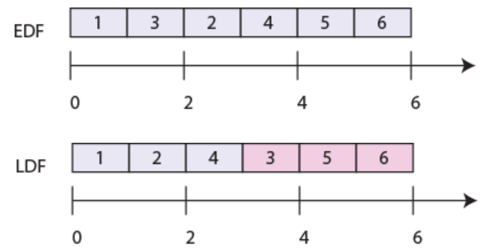


- The LDF scheduling strategy builds a schedule backwards.
- Given a DAG,
  - choose the leaf node with the latest deadline to be scheduled last,
  - and work backwards.

Latest Deadline First (LDF)

(Lawler, 1973)

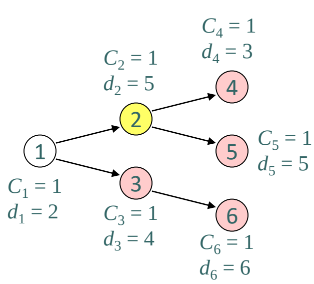


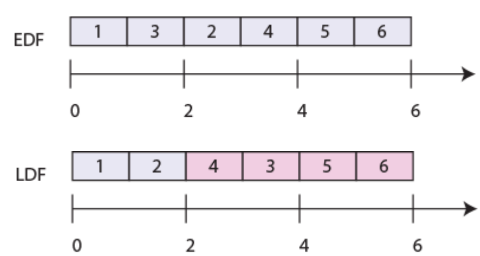


- The LDF scheduling strategy builds a schedule backwards.
- Given a DAG,
  - choose the leaf node with the latest deadline to be scheduled last,
  - and work backwards.

Latest Deadline First (LDF)

(Lawler, 1973)

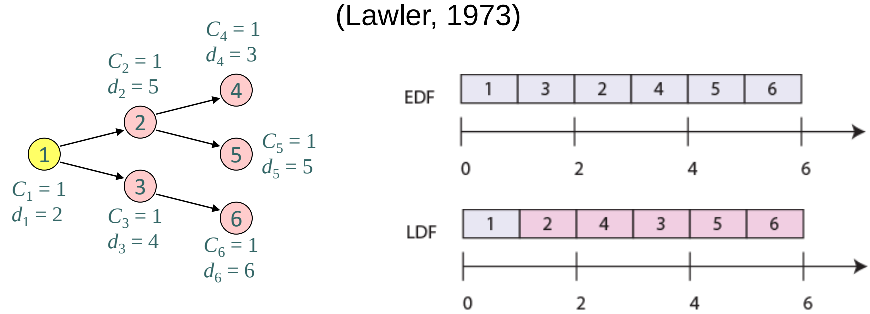




- The LDF scheduling strategy builds a schedule backwards.
- Given a DAG,
  - choose the leaf node with the latest deadline to be scheduled last,
  - and work backwards.

Latest Deadline First (LDF)

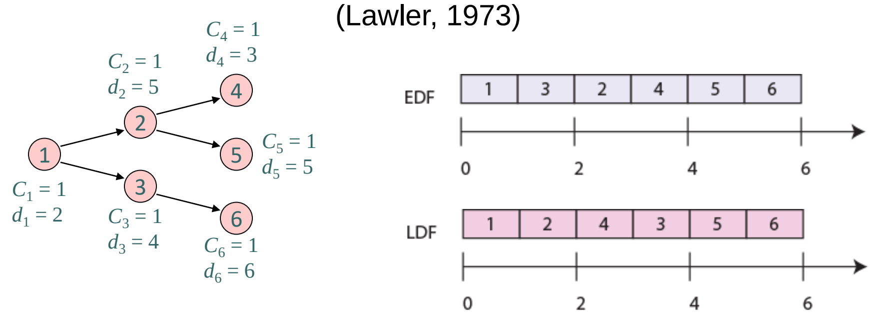
(Lawler, 1973)



- The LDF scheduling strategy builds a schedule backwards.
- Given a DAG,
  - choose the leaf node with the latest deadline to be scheduled last,
  - and work backwards.

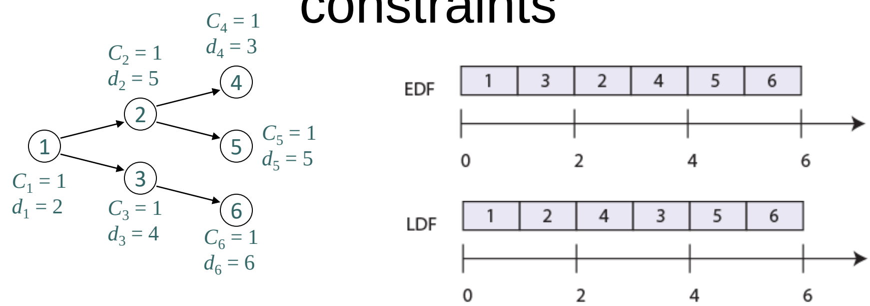
Latest Deadline First (LDF)

(Lawler, 1973)



- The LDF scheduling strategy builds a schedule backwards.
- Given a DAG,
  - choose the leaf node with the latest deadline to be scheduled last,
  - and work backwards.

LDF is optimal under precedence constraints



- The LDF schedule shown at the bottom respects all precedences and meets all deadlines.

✓ Also minimizes maximum lateness

Latest Deadline First (LDF)

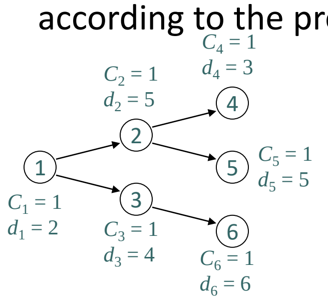
(Lawler, 1973)

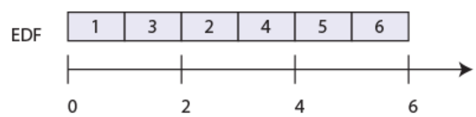
- LDF is optimal in the sense that it minimizes the maximum lateness.
- It does not require preemption.
  - We'll see that EDF can be made to work with preemption.
- However, it requires that
  - all tasks be *available* and
  - their *precedences known*

  before any task is executed.

EDF with Precedences (EDF*)

- With a preemptive scheduler, EDF can be modified to
  - account for precedences and
  - allow tasks to arrive at arbitrary times.
- Simply *adjust the deadlines and arrival times* according to the precedences.





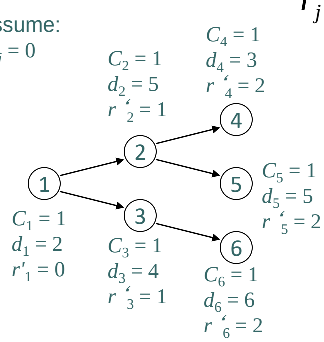
Recall that for the tasks at the left, EDF yields the schedule above, where task 4 misses its deadline.

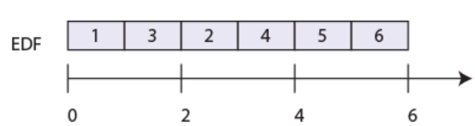
EDF with Precedences Modifying release times

Given n tasks with precedences and release times r_i, if task i immediately precedes task j, then modify the release times as follows:

$$r_j' = \max(r_j, r_i + C_i)$$

assume:
$r_i = 0$

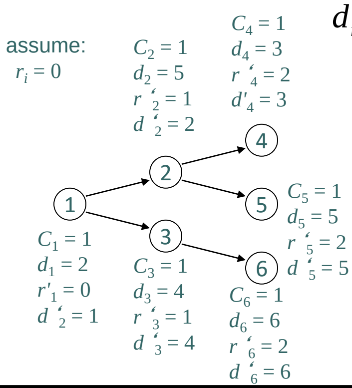


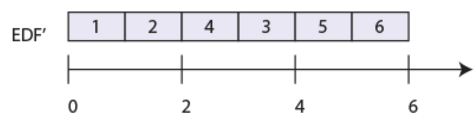


EDF with Precedences Modifying deadlines

Given n tasks with precedences and deadlines d_i, if task i immediately precedes task j, then modify the deadlines as follows:

$$d'_i = \min(d_i, d'_j - C_j)$$





Using the revised release times and deadlines, the above EDF schedule is optimal and meets all deadlines.

EDF* Optimality

EDF with precedences is optimal in the sense of minimizing the maximum lateness.