Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Real-Time Systems

3d-Earliest Deadline First

Antonio Camacho Santiago
antonio.camacho.santiago@upc.edu

Objective

To understand Earliest Deadline First (EDF) schedulers…

To design Earliest Deadline First schedulers…

To analyze Earliest Deadline First …

To evaluate the pros and cons of Earliest Deadline First schedulers…

```
at design stage:
  nothing to do (just check feasibility)

at runtime each Sys_Tick:
  for each active task
    dispatch the task with shorter absolute deadline
```

EDF scheduler

At run-time, absolute deadlines of active tasks are checked and priorities are online modified

$$\forall t, \tau_i, \tau_j: d_i < d_j \Rightarrow P_i > P_j$$

It can also be used the following rule saying that priorities are assigned proportionally to the inverse of the absolute deadline

$$P_i \propto \frac{1}{d_i}$$

Note that absolute deadline is $d_i = \boldsymbol{\phi}_i + kT_i + D_i$ where $\boldsymbol{\phi}_i$ is the initial phase, k is a positive integer, T_i is the task period and D_i is the relative deadline

At each system tick, the scheduler looks for the existing active tasks to dispatch the task with higher priority. Thus, preemption is allowed at each system tick

More details can be found in S.K.Baruah, L. E: Rosier and R. R Howell, “Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor,” Journal of Real Time Systems, n. 2, 1990

Requisites

The approach for the earliest deadline first scheduler is based on periodic tasks as follows:

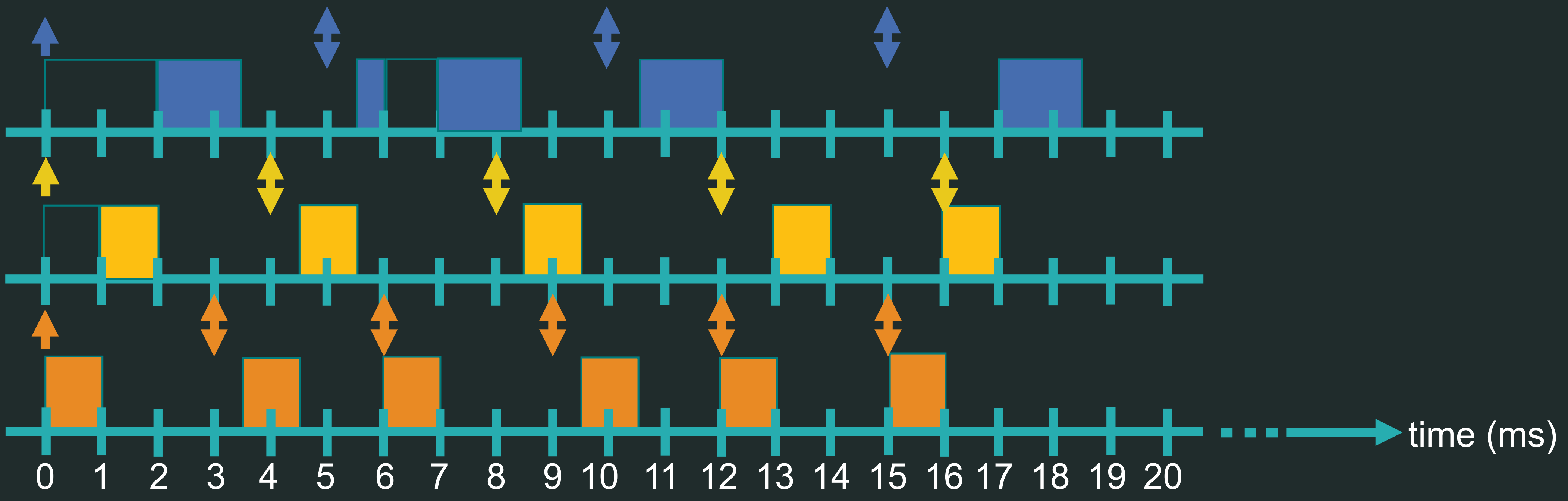
- 1 microprocessor
- Tasks may arrive at any time
- The WCET for each task is known, fitted and less than its deadline
- Deadlines of each task are less or equal to their periods
- Periodic (simple feasibility test) or aperiodic tasks (processor demand test)
- EDF for no precedence among tasks or EDF* when precedence exists
- **Tasks can be preempted**
- **RT kernel uses dynamic priorities**

The schedulability analysis tries to know in advance if all the release times for each task occurs before its deadline.

EDF-Periodic tasks T_i=D_i

Typical periodic task set with T_i=D_i

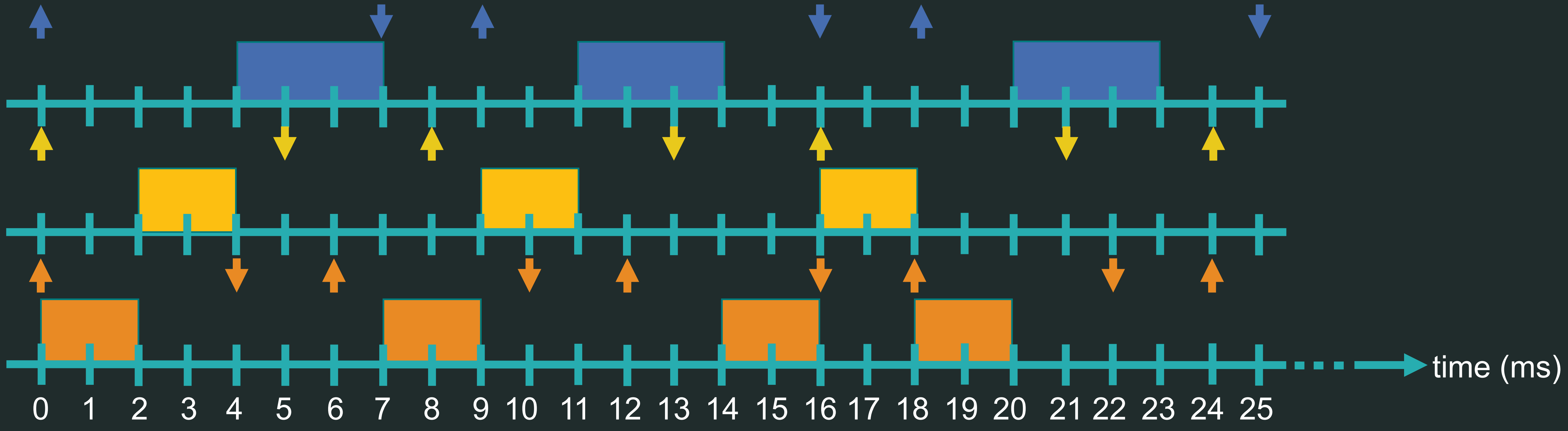
Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1	1	3
τ_2	1	4
τ_3	1.5	5



EDF-Periodic tasks $T_i \leq D_i$

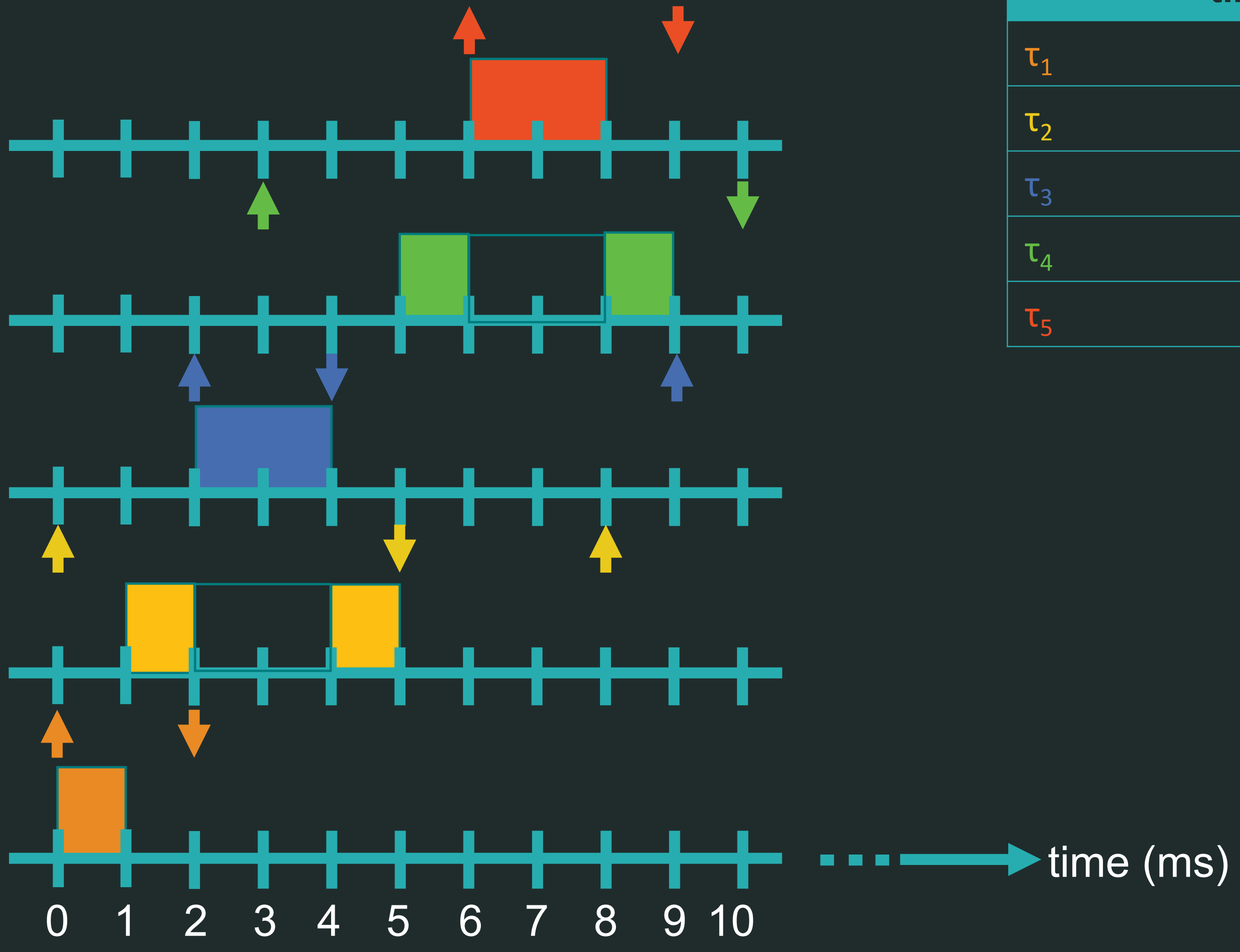
Periodic task set with $T_i \leq D_i$

Task τ_i	Computing time c_i (ms)	Deadline D_i (ms)	Period T_i (ms)
τ_1	2	4	6
τ_2	2	5	8
τ_3	3	7	9



EDF-Aperiodic tasks

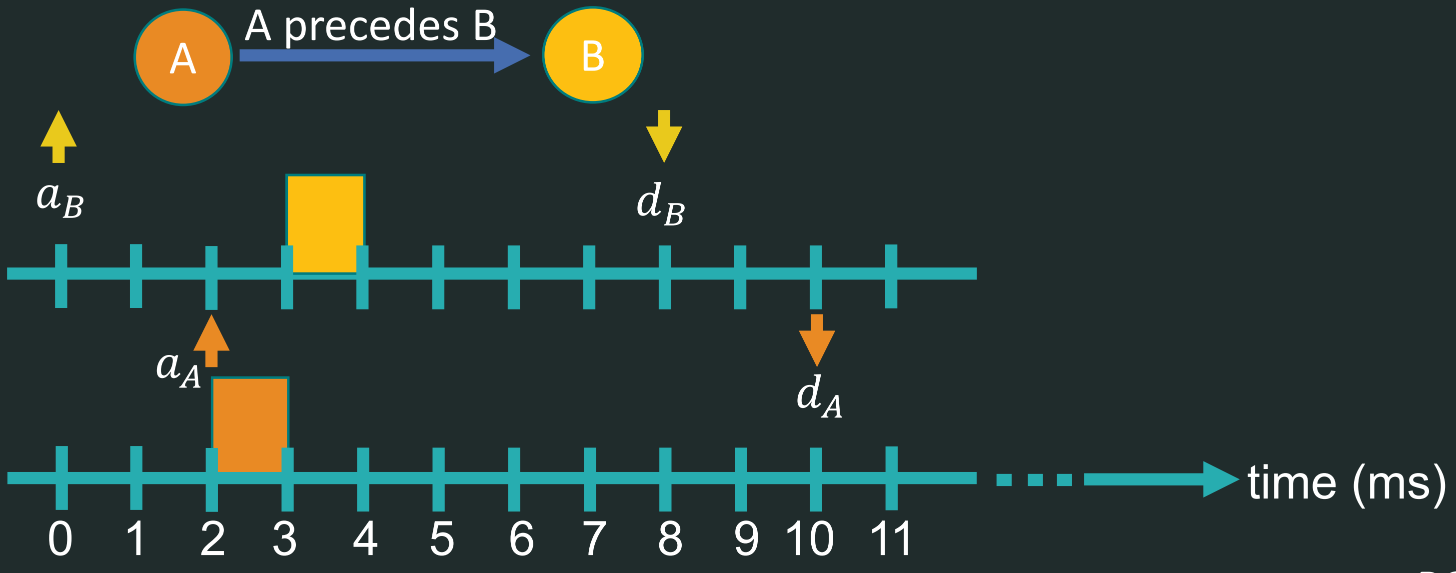
Aperiodic tasks with different arrival times



Task τ_i	Computing time c_i (ms)	Deadline d_i (ms)	Arrival a_i time (ms)
τ_1	1	2	0
τ_2	2	5	0
τ_3	2	4	2
τ_4	2	10	3
τ_5	2	9	6

EDF*-precedence constraints

Until now EDF has no precedence constraints. In case $\tau_A \rightarrow \tau_B$ then EDF becomes EDF*



POSTPONE SUCCESSOR ARRIVAL TIME:

$$a_B^* = a_A + C_A$$

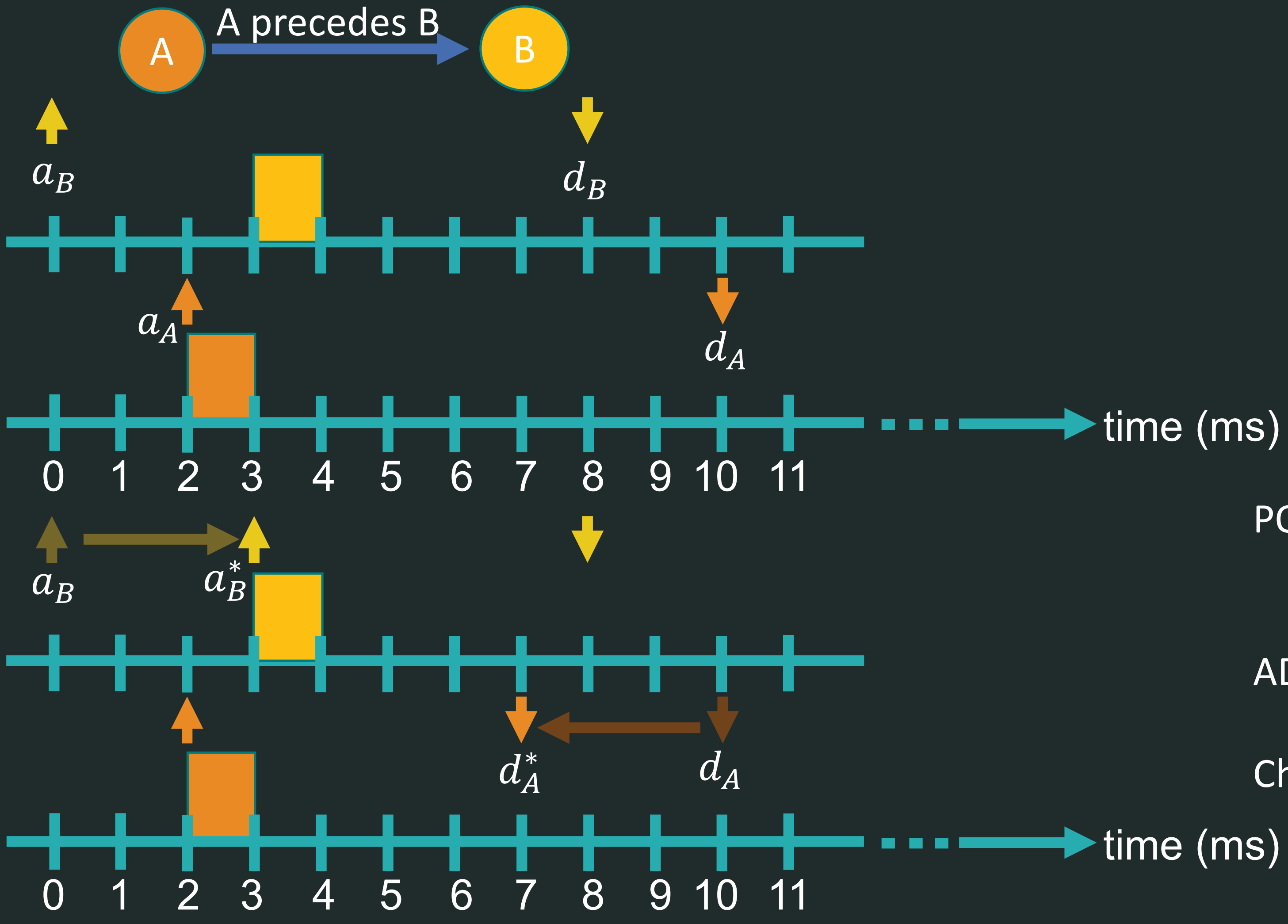
ADVANCE DEADLINE OF PRECESSOR:

$$d_A^* = d_B - C_B$$

Check feasibility of the new task set

EDF*-precedence constraints

Until now EDF has no precedence constraints. In case $\tau_A \rightarrow \tau_B$ then EDF becomes EDF*



POSTPONE SUCCESSOR ARRIVAL TIME:

$$a_B^* = a_A + C_A$$

ADVANCE DEADLINE OF PRECESSOR:

$$d_A^* = d_B - C_B$$

Check feasibility of the new task set

EDF*-precedence constraints

Generic approach for complex precedence paths:
Transform a set J of dependent tasks into a set J* of independent tasks

Arrival time modification:

1 For all root nodes, set $a_i^* = a_i$
2 Select a task τ_i such that all its immediate predecessors have been modified, else exit.
3 Set $a_i^* = \max\left\{a_i, \max_{\tau_k \to \tau_i}(a_k^* + C_k)\right\}$
4 Repeat from line 2.

Deadline modification

1 For all leaves, set $d_i^* = d_i$
2 Select a task τ_i such that all its immediate successors have been modified, else exit.
3 Set $d_i^* = \min\left\{d_i, \min_{\tau_k \to \tau_i}(d_k^* - C_k)\right\}$
4 Repeat from line 2.

Online test for accepting a new task

How to online check if a new task can be scheduled at runtime?

Let *J* be the set of current guaranteed task

Let J_{new} be the new task that has arrived

To accept J_{new} U J = J' must be schedulable

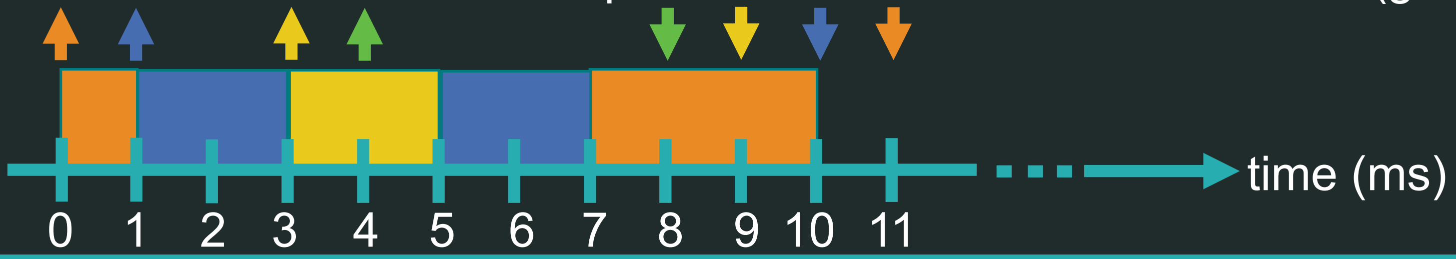
J' is schedulable iff $\forall\, i = 1, 2, .., n$: $f_i \leq d_i$

where $f_i = \sum_{k=1}^{n} c_k$, being c_i the remaining Worst Case Execution Time of task J_i

The online test consists of checking the n-conditions at the arrival times, ordering the tasks by increasing deadlines

Task τ_i	Computing time c_i (ms)	deadline d_i (ms)	Arrival time a_i (ms)
τ_1	4	11	0
τ_2	2	9	3
τ_2	4	10	1
τ_{new}	2	8	4

There is no space to accommodate the new task (green) without missing deadline for task 1



Online test for accepting a new task

3d-Earliest Deadline First

How to online check if a new task can be scheduled at runtime?

$t = 0, J_1: C_1(0) = 4 \leq 11 - 0 = d_1 - t$

$t = 1, J_3: C_3(1) = 4 \leq 10 - 1 = d_3 - t$

$J_1: C_3(1) + C_1(1) = 4 + 3 = 7 \leq 10$

$t = 3, J_2: C_2(3) = 2 \leq 9 - 3 = d_2 - t$

$J_3: C_2(3) + C_3(3) = 2 + 2 \leq 10 - 3 = d_3 - t$

$J_1: C_2(3) + C_3(3) + C_1(3) = 2 + 2 + 3 \leq 11 - 3 = d_1 - t$

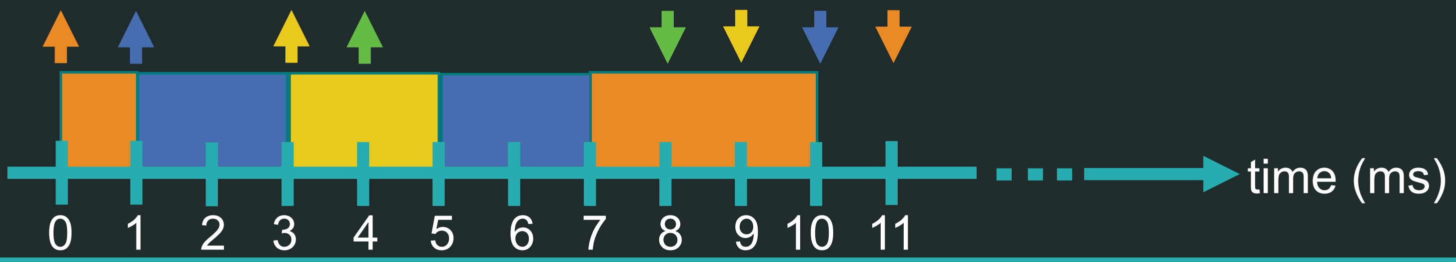
$t = 4, J_4: C_4(4) = 2 \leq 8 - 4 = d_4 - t$

$J_2: C_4(4) + C_2(4) = 2 + 1 \leq 9 - 4 = d_2 - t$

$J_3: C_4(4) + C_2(4) + C_3(4) = 2 + 1 + 2 \leq 10 - 4 = d_3 - t$

$J_1: C_4(4) + C_2(4) + C_3(4) + C_1(4) = 2 + 1 + 2 + 3 = 8 \geq 7 = 11 - 4 = d_1 - t$ → Not feasible (as shown in the plot)

Task τ_i	Computing time c_i (ms)	deadline d_i (ms)	Arrival time a_i (ms)
τ_1	4	11	0
τ_2	2	9	3
τ_3	4	10	1
τ_{new}	2	8	4



Offline feasibility analysis

For periodic tasks with D_i=T_i check feasibility as

That's all folks!! $$U_{\text{total}} = \sum_{i=1}^{n} U_i \leq 1$$ (necessary and sufficient condition)

For periodic tasks with D_i<T_i check feasibility using processor demand as

$$L \geq \sum_{i=1}^{n} \left(\left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1 \right) C_i$$ (necessary and sufficient condition)

Note that condition $L \geq \sum_{i=1}^{n} \left\lfloor \frac{L}{T_i} \right\rfloor C_i$ based on processor demand analysis is the same as $U_{\text{total}} = \sum_{i=1}^{n} U_i \leq 1$ when D_i=T_i

Processor Demand Analysis

Processor demand criterion: In any interval, the computation demanded by the task set must be no greater than the available time.

$$g(t_1, t_2) = \sum_{\substack{r_{i,k} \geq t_1 , \\ d_{i,k} \leq t_2}} C_i$$

The set of tasks is feasible if in any interval of time, the processor demand does not exceed the available time

$$\forall\, t_1, t_2: \quad g(t_1, t_2) \leq t_1 - t_2$$

Problem: how to perform the test. Solution: find instances of τ_i contributing in $[t_1, t_2]$

$$g(t_1, t_2) = \sum_{i=1}^{n} \eta_i(t_1, t_2)\, C_i = \sum_{i=1}^{n} \max\left(0, \left\lfloor \frac{t_2 + T_i - D_i}{T_i} \right\rfloor - \left\lceil \frac{t_1}{T_i} \right\rceil\right) C_i$$

Assumption: initial phase $\forall i$: ϕ_i=0

$$g(0, L\,) = \sum_{i=1}^{n} \eta_i(0, L)\, C_i = \sum_{i=1}^{n} \left\lfloor \frac{L + T_i - D_i}{T_i} \right\rfloor C_i$$

where L is the final time interval. Schedulabiliy is ensured iff $g(0, L) \leq L$

Processor Demand Analysis

The test with assumption ϕ_i=0 is simplified for D_i=T_i

$$g(0, L) = \sum_{i=1}^{n} \left\lfloor \frac{L}{T_i} \right\rfloor C_i \leq L$$

How many test need to be analyzed for L? The function $g(0, L)$ is stepwise constant between two deadlines. Therefore, check only at deadline intervals. Moreover check along the hyperperiod.

A set of tasks with ϕ_i=0, D_i≤T_i is EDF-feasible iff

$$g(0, L) \leq L \text{ with } D = \{d_k | d_k \leq \min(H, L^*)\}$$

$$H = \mathrm{lcm}(T_1, T_2, , \ldots, T_n)$$

$$L^* = \frac{\sum_{i=1}^{n} (T_i - D_i) U_i}{1 - U}$$

Processor Demand Analysis

Task τ_i	Computing time c_i (ms)	Deadline D_i (ms)	Period T_i (ms)
τ_1	2	4	6
τ_2	2	5	8
τ_3	3	7	9

Processor utilization

$$U = \sum_{i=1}^{n} U_i = \frac{2}{6} + \frac{2}{8} + \frac{3}{9} = \frac{11}{12} \leq 1$$

$$H = \mathrm{lcm}(T_1, T_2, , \dots, T_n) = 72$$

$$L^* = \frac{\sum_{i=1}^{n} (T_i - D_i) U_i}{1 - U} = 25$$

Conditions should be evaluated at

$$D = \{d_k | d_k \leq \min(H, L^*)\} \leq 25 = \{4, 5, 7, 10, 13, 16, 21, 22, 25\}$$

Test is $g(0, L) = \sum_{i=1}^{n} \left\lfloor \frac{L + T_i - D_i}{T_i} \right\rfloor C_i \leq L$

L	4	5	7	10	13	16	21	22	25
g(0,L)	2	4	7	9	11	16	18	20	23
Test *g(0,L)* ≤ *L*	OK	OK	OK	OK	OK	OK	OK	OK	OK

EDF scheduler

Pros:

The earliest deadline first scheduler is based on dynamic priorities configured at runtime according to the absolute deadline of each task

Sufficient condition for T_i=D_i and periodic tasks $U_{\text{total}} \leq 1$

For aperiodic tasks, check demand processor criterion

Optimality: among all the algorithms, EDF is optimal in terms of feasibility

if some feasible schedule exists, EDF will find it. EDF minimizes the maximum latency.

Note that EDF is not optimal in systems where no preemption is allowed

Cons:

EDF is not provided by any commercial RTOS!

It is less predictable and less controllable when trying to reduce response time

It requires higher overhead compared with fixed priorities schedulers

During overload, all tasks can miss their deadline (domino effect)