Inovace tohoto kurzu byla v roce 2011/12 podpořena projektem CZ.2.17/3.1.00/33274 financovaným Evropským sociálním fondem a Magistrátem hl. m. Prahy.

OPP A PRAHA PRAGUE PRAGA PRAG

EVROPSKÁ UNIE

Evropský sociální fond

Praha & EU: Investujeme do vaší budoucnosti

Department of Distributed and Dependable Systems

Embedded and Real-time Systems

Periodic Task Scheduling II

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems
D3S

Tomáš Bureš

<bures@d3s.mff.cuni.cz>

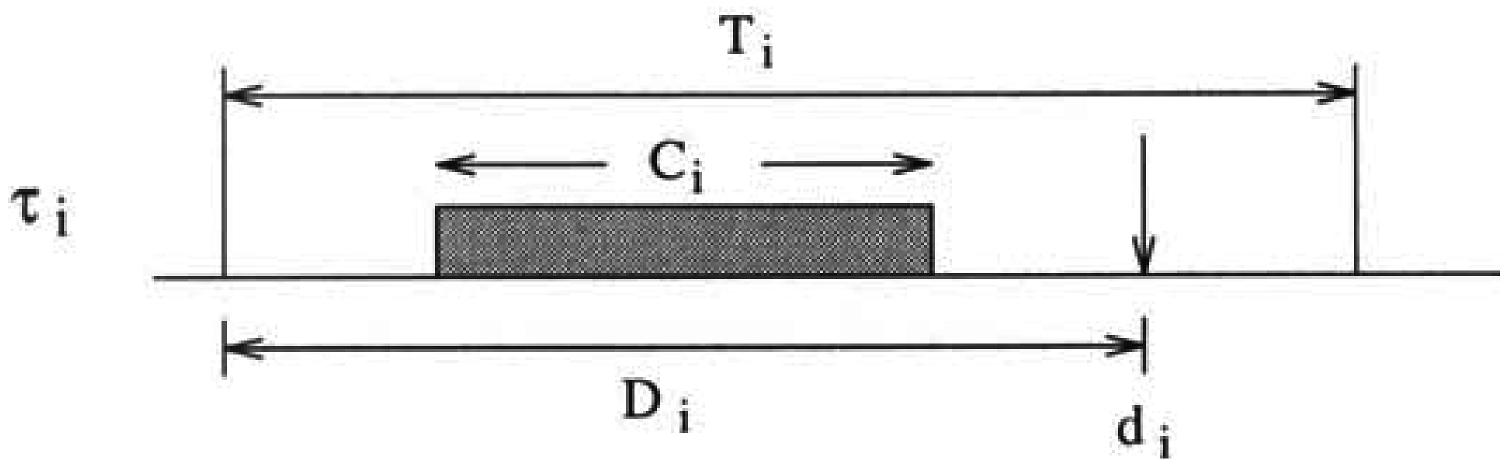
CHARLES UNIVERSITY IN PRAGUE
Faculty of Mathematics and Physics

Deadline Monotonic

- A generalization of Rate Monotonic
  - Task deadlines are allowed to be shorter than periods $D < T$
- Each task characterized by
  - A phase Φ_i
  - A worst-case computation time C_i (constant for each instance)
  - A relative deadline D_i (constant for each instance)
  - A period T_i
- A task is assigned a priority inversely proportional to its relative deadline

Deadline Monotonic

$$C_i \leq D_i \leq T_i$$
$$r_{i,k} = \Phi_i + (k-1)T_i$$
$$d_{i,k} = r_{i,k} + D_i$$



Deadline Monotonic

- DM is optimal among static priority scheduling algorithms, which allow relative deadlines less or equal to periods.

- Proof similar to RM

DM Schedulability Analysis

- The feasibility of a set of tasks with deadlines unequal to their periods could be guaranteed using the Rate-Monotonic schedulability test, by reducing tasks' periods to relative deadlines

$$\sum_{i=1}^{n} \frac{C_i}{D_i} \leq n(2^{\frac{1}{n}} - 1)$$

- This is however a big overestimation since it does not reflect the periods which can be relatively large

DM Schedulability Analysis

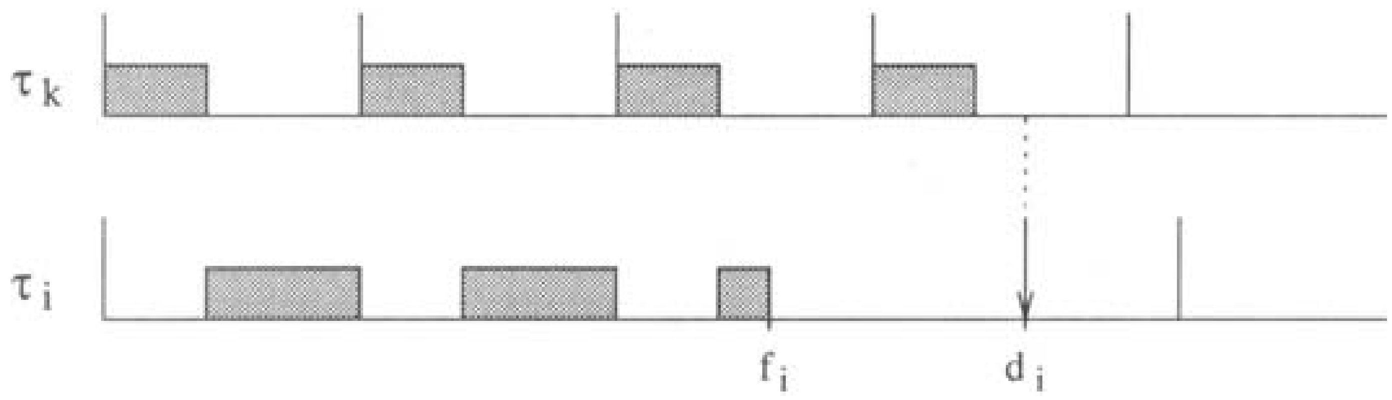
- Better test may be derived by noting that:
  - The worst-case processor demand occurs when all tasks are released simultaneously; that is, at their critical instants;
  - For each task the sum of its processing time and the interference (preemption) imposed by higher priority tasks must be less than or equal to its relative deadline

$$\forall i, 1 \le i \le n: C_i + I_i \le D_i$$

$$I_i = \sum_{j=1}^{i-1} \left\lceil \frac{D_i}{T_j} \right\rceil C_j$$

Schedulability Analysis

- This is still an overestimation since it assumes that every higher priority task interferes exactly $\lceil D_i / T_j \rceil$ times, which is not necessarily true



- Thus the test is still only sufficient but not necessary

Response Time Analysis

- Sufficient and necessary test for schedulability:

The longest response time R_i of a periodic task τ_i (both in RM and DM) is computed, at the critical instant, as the sum of its computation time and the interference due to preemption by higher-priority tasks: $R_i = C_i + I_i$, where

$$I_i = \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Hence,

$$R_i = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Department of Distributed and Dependable Systems

Response Time Analysis

- Since R_i is on both sides, the solution is to find the smallest value of R_i which satisfies the equation.
- Only a subset of points in the interval $[0, D_i]$ need to be examined for feasibility.
  - The interference on a task i only increases when there is a release of a higher-priority task.

$$R_i = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \qquad \text{\#1}$$

Department of Distributed and Dependable Systems

Response Time Analysis

- Let R_i^k be the k^{th} estimate of R_i and let I_i^k be the interference on task τ_i in the interval

$$[0, R_i^k]: I_i^k = \sum_{j=1}^{i-1} \left\lceil \frac{R_i^k}{T_j} \right\rceil C_j$$

- The calculation of R_i is performed as follows:
  1. Iteration starts with $R_i^0 = C_i$, which is the first point in time that τ_i could possible complete.
  2. The actual interference I_i^k in the interval $[0, R_i^k]$ is computed by equation **#1**.
  3. If $I_i^k + C_i = R_i^k$, then R_i^k is the actual worst-case response time of task τ_i; that is, $R_i = R_i^k$. Otherwise, the next estimate is given by $R_i^{k+1} = I_i^k + C_i$, and the iteration continues to step 2.

Department of Distributed and Dependable Systems

Response Time Analysis

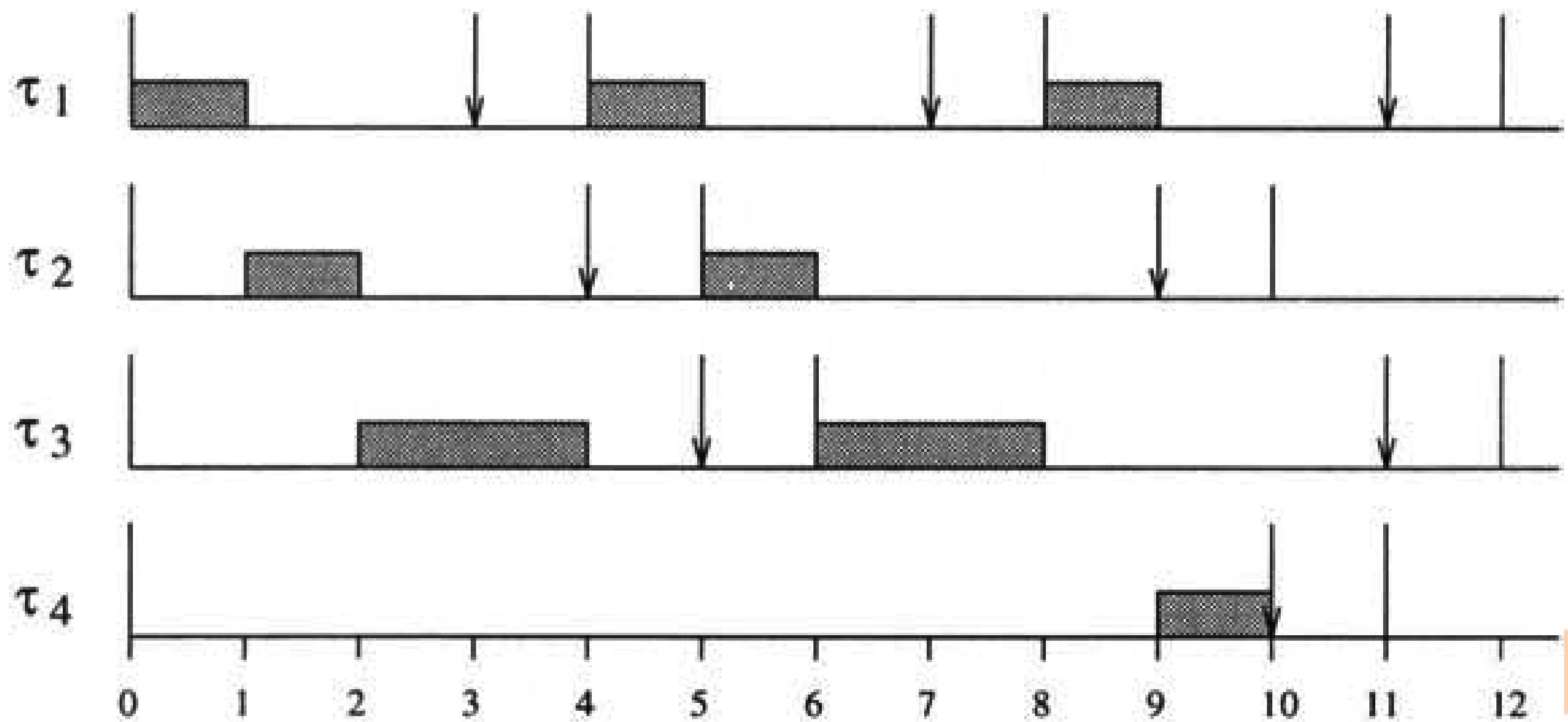
- Once calculated, the feasibility of task i is guaranteed if and only if:

$$R_i \leq D_i$$

Department of Distributed and Dependable Systems

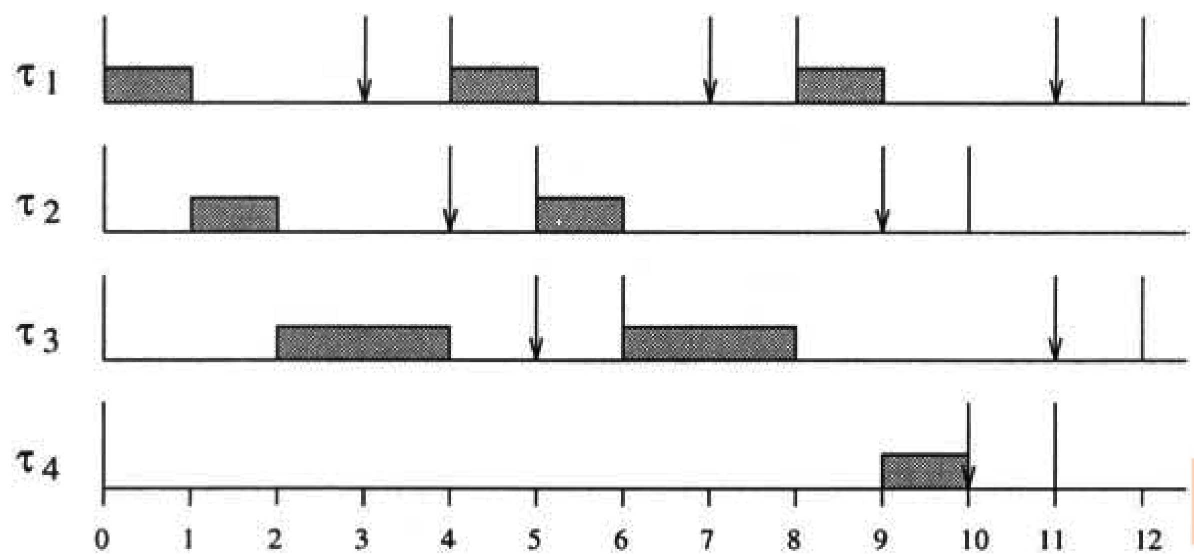
Response Time Analysis – Example

	C_i	T_i	D_i
τ_1	1	4	3
τ_2	1	5	4
τ_3	2	6	5
τ_4	1	11	10



Response Time Analysis – Example

- Step 0: $R_4^0 = C_4 = 1$, but $I_4^0 = 4$ and $I_4^0 + C_4 > R_4^0$
- Step 1: $R_4^1 = I_4^0 + C_4 = 5$, but $I_4^1 = 5$ and $I_4^1 + C_4 > R_4^1$
- Step 2: $R_4^2 = I_4^1 + C_4 = 6$, but $I_4^2 = 6$ and $I_4^2 + C_4 > R_4^2$
- Step 3: $R_4^3 = I_4^2 + C_4 = 7$, but $I_4^3 = 7$ and $I_4^3 + C_4 > R_4^3$
- Step 4: $R_4^4 = I_4^3 + C_4 = 9$, but $I_4^4 = 9$ and $I_4^4 + C_4 > R_4^4$
- Step 5: $R_4^5 = I_4^4 + C_4 = 10$, but $I_4^5 = 9$ and $I_4^5 + C_4 > R_5^4$,
  - hence τ_4 finishes at $R_4 = 10$.



Response Time Analysis – Example

- Since $R_4 \leq D_4$, τ_4 is schedulable within its deadline.
- If $R_i \leq D_i$ for all tasks, we conclude the task is schedulable by DM.

Earliest Deadline First

- Selects the task with the shortest absolute deadline

- Preemptive, with dynamic priority assignment
- Conditions:
  - independent tasks
  - deadline = period
  - release time = start of period

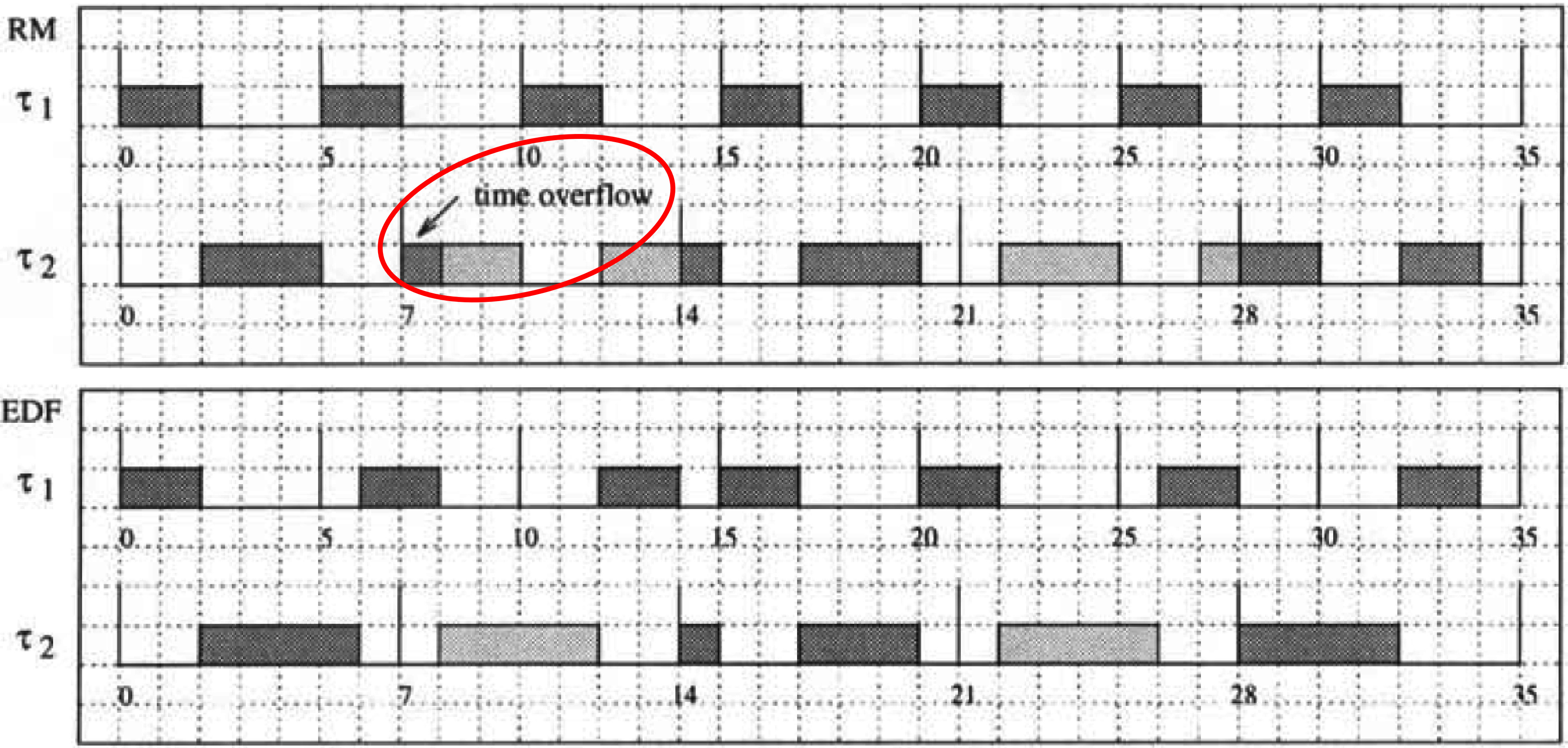
- Optimal, as proved in aperiodic case

- Schedulability analysis:
  - All tasks meet their deadline if $U \leq 1$

Department of Distributed and Dependable Systems

Example

$$U = \frac{2}{5} + \frac{4}{7} \simeq 0.97$$



Schedulability Analysis

- **Theorem:** A set of period tasks is schedulable with EDF if and only if

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

- *Proof of only if:* The task set cannot be scheduled if $U > 1$. By defining $T = T_1 T_2 \dots T_n$, the total demand of computation time requested by all tasks in T can be calculated as

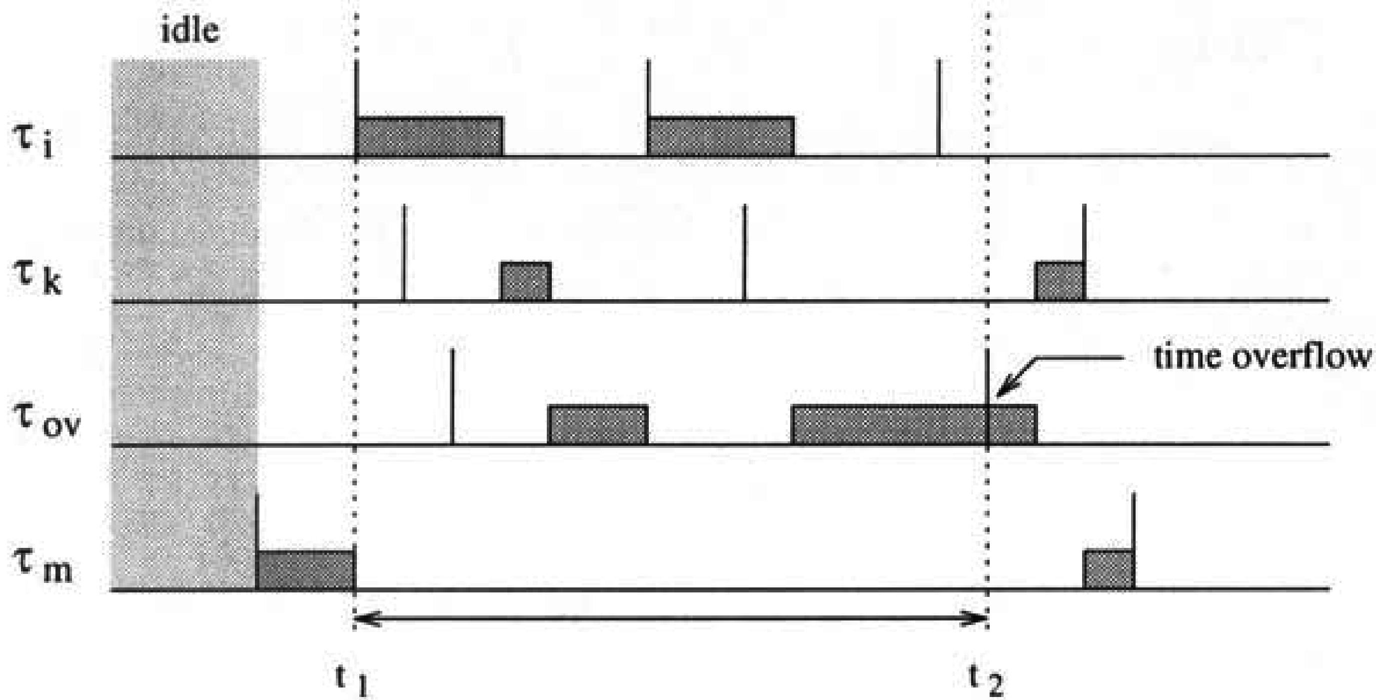
$$\sum_{i=1}^{n} \frac{T}{T_i} C_i = UT$$

- If $U > 1$, then $UT > T$, then the total demand exceeds the available time.

Department of Distributed and Dependable Systems

Schedulability Analysis

- *Proof of if:* Assume that $U < 1$ but the task set is not schedulable. Let t_2 be the instant at which the time-overflow occurs. Let $[t_1, t_2]$ be the longest interval of continuous utilization before the overflow such that only instance with deadline less than or equal to t_2 are executed in $[t_1, t_2]$



Schedulability Analysis

- t_1 must be the release time of some periodic instance. Let $C_p(t_1, t_2)$ be the total computation time demanded by periodic tasks in $[t_1, t_2]$, which can be computed as

$$C_p(t_1, t_2) = \sum_{r_k \geq t_1, d_k \leq t_2} C_k = \sum_{i=1}^{n} \left\lfloor \frac{t_2 - t_1}{T_i} \right\rfloor C_i$$

Department of Distributed and Dependable Systems

Schedulability Analysis

- Now, observe that

$$C_p(t_1, t_2) = \sum_{i=1}^{n} \left\lfloor \frac{t_2 - t_1}{T_i} \right\rfloor C_i \leq \sum_{i=1}^{n} \frac{t_2 - t_1}{T_i} C_i = (t_2 - t_1)U$$

- Since a deadline is missed at t_2, $C_p(t_1, t_2)$ must be greater than the available processor time $(t_2 - t_1)$; thus, we must have

$$(t_2 - t_1) < C_p(t_1, t_2) \leq (t_2 - t_1)U$$

- That is, $U > 1$, which is a contradiction.

Processor Demand Analysis

- When doing the schedulability analysis:
  - $D = T$ – use the processor utilization analysis $U \leq 1$
  - $D < T$ – use processor demand analysis (will follow)

Processor Demand Analysis

- **Theorem:** If $\mathcal{D} = \{d_{i,k} | d_{i,k} = kT_i + D_i, d_{i,k} \leq \min(B_p, H), 1 \leq i \leq n, k \geq 0\}$, then a set of periodic tasks with deadlines less than periods is schedulable by EDF if and only if

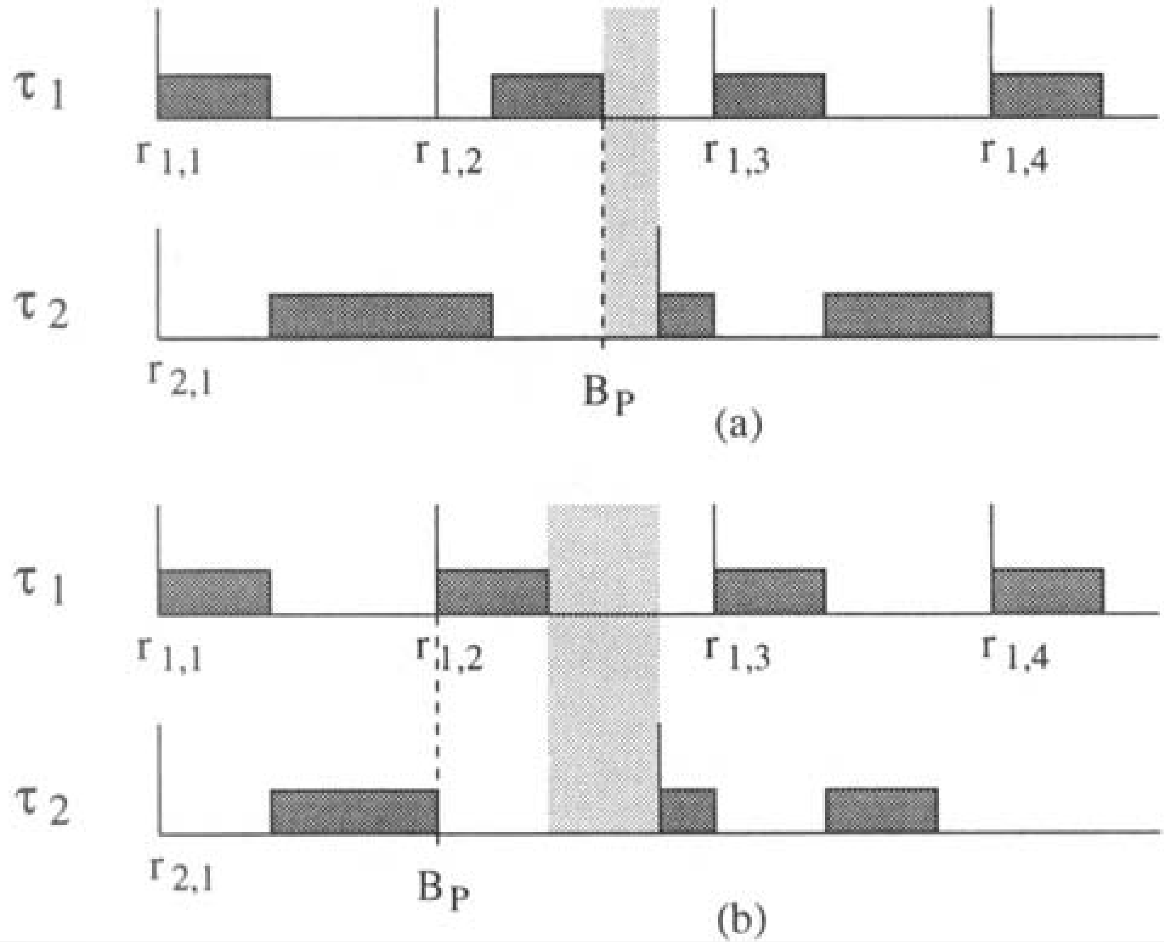
$$\forall L \in \mathcal{D}: L \geq \sum_{i=1}^{n} \left(\left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1 \right) C_i$$

- $H = lcm(T_1, \dots, T_n)$
- B_p signifies a busy period – i.e., the smallest interval $[0, L]$ in which the total processing time $W(L)$ requested in $[0, L]$ is completely executed. The quantity $W(L)$ can be computed as

$$W(L) = \sum_{i=1}^{n} \left\lceil \frac{L}{T_i} \right\rceil C_i$$

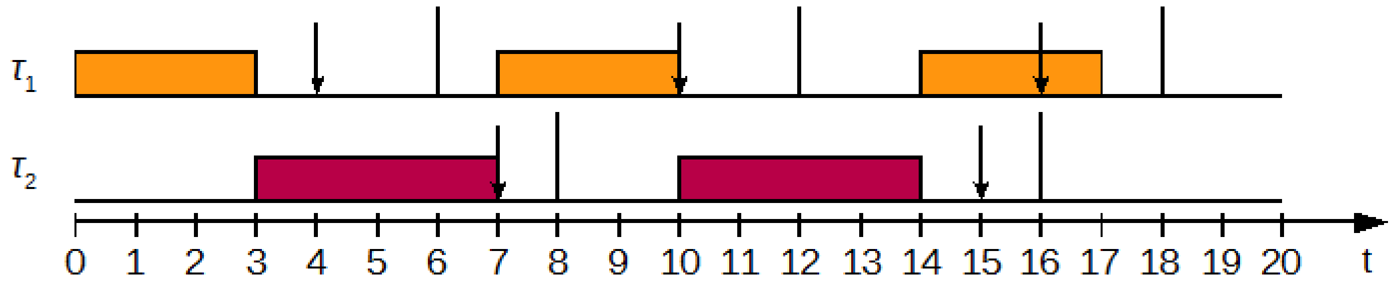
Busy Period

- Coincides either with the beginning of an idle time or with the release of an periodic instance



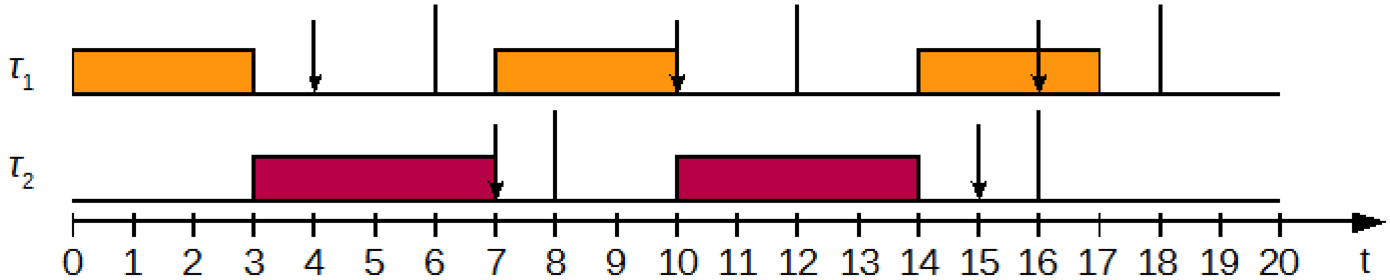
Processor Demand Analysis – Example

Task	T	D	C
τ_1	6	4	3
τ_2	8	7	4



Processor Demand Analysis – Example

Task	T	D	C
τ_1	6	4	3
τ_2	8	7	4



- $C_p(0,7) = \left(\left\lfloor\frac{7-4}{6}\right\rfloor + 1\right)3 + \left(\left\lfloor\frac{7-7}{8}\right\rfloor + 1\right)4 = 3 + 4 = 7 \leq 7$
- $C_p(0,10) = \left(\left\lfloor\frac{10-4}{6}\right\rfloor + 1\right)3 + \left(\left\lfloor\frac{10-7}{8}\right\rfloor + 1\right)4 = 6 + 4 = 10 \leq 10$
- $C_p(0,10) = \left(\left\lfloor\frac{15-4}{6}\right\rfloor + 1\right)3 + \left(\left\lfloor\frac{15-7}{8}\right\rfloor + 1\right)4 = 6 + 8 = 14 \leq 15$
- $C_p(0,10) = \left(\left\lfloor\frac{16-4}{6}\right\rfloor + 1\right)3 + \left(\left\lfloor\frac{16-7}{8}\right\rfloor + 1\right)4 = 9 + 8 = 17 \nleq 16$

deadline miss!

RM vs. EDF

- Let's compare RM vs. EDF in the following:
  - Processor utilization
  - Implementation complexity
  - Runtime overhead
  - Jitter

RM vs. EDF: Processor Utilization

- EDF utilizes processor better than RM

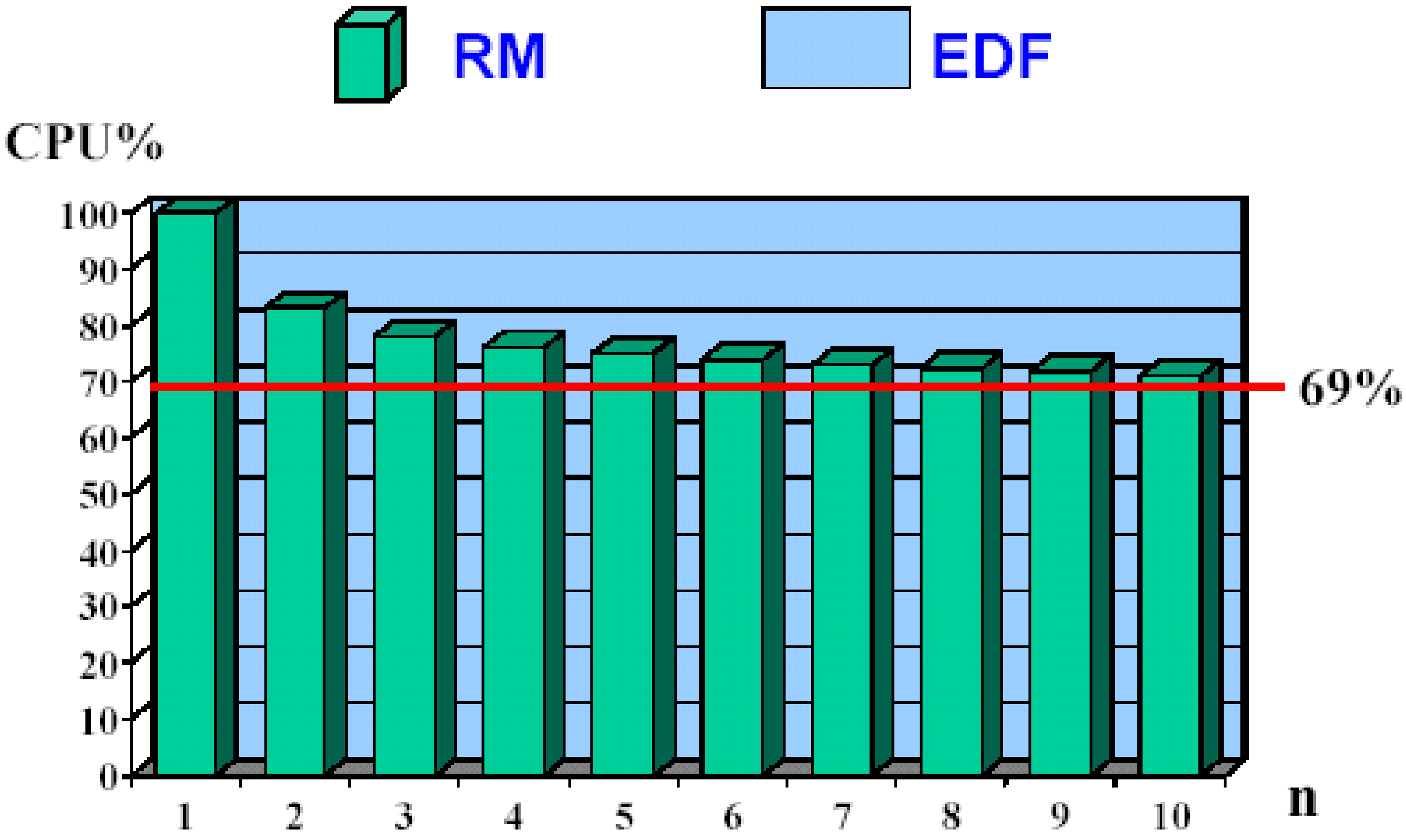
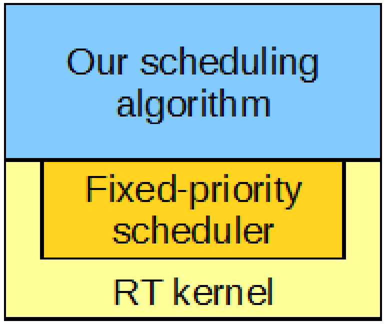


Figure from Issovic, D.:Real-time systems, basic course

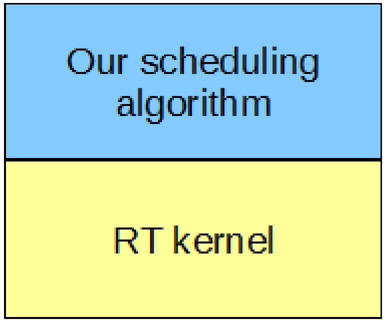
RM vs. EDF: Implementation Complexity

- Case 1
  - on top of existing fixed priority scheduler



  - RM – straightforward
  - EDF – needs re-mapping of priorities at runtime

- Case 2
  - implementation from scratch

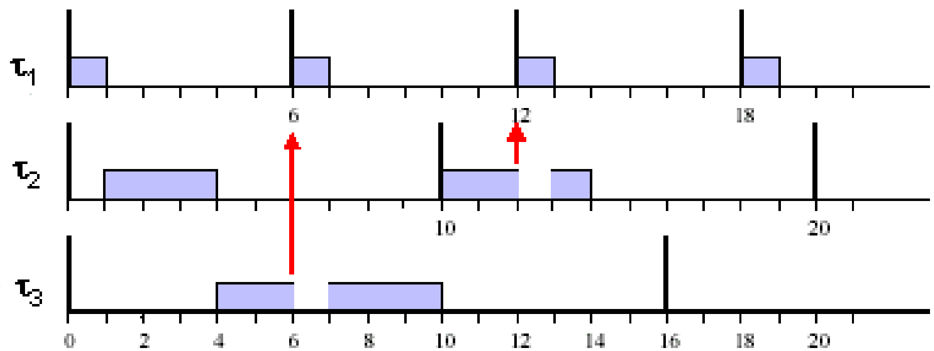


  - Same complexity

Department of Distributed and Dependable Systems

RM vs. EDF: Runtime Overhead

- EDF has higher overhead for task release since absolute deadline must be updated for each instance. RM has higher context-switch overhead due to more preemptions.
- RM Example:



  - If we increase the execution time of τ_3, we get 3 preemptions instead of 2:

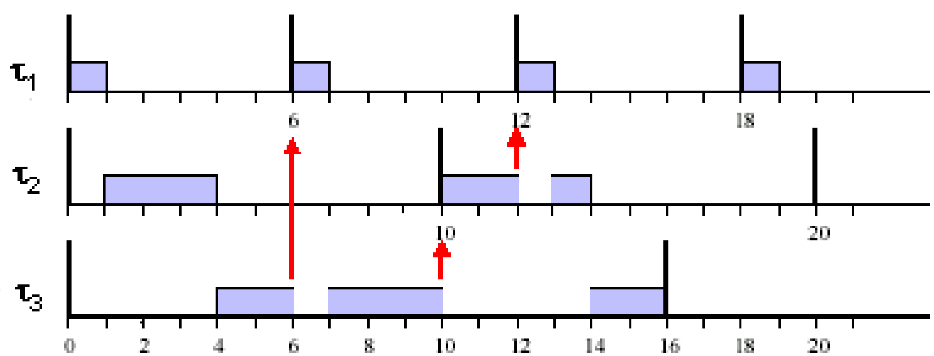
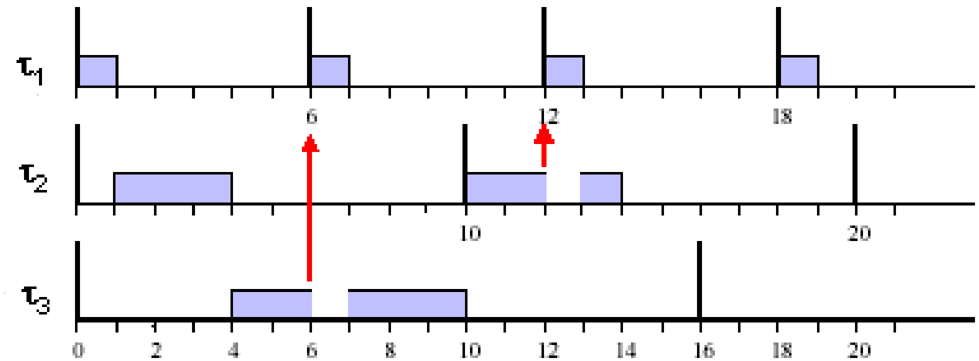


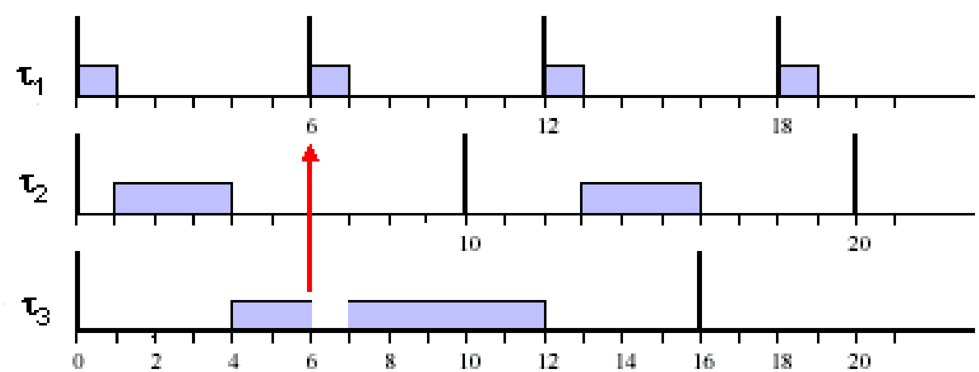
Figure from Issovic, D.:Real-time systems, basic course

RM vs. EDF: Runtime Overhead

- EDF Example:



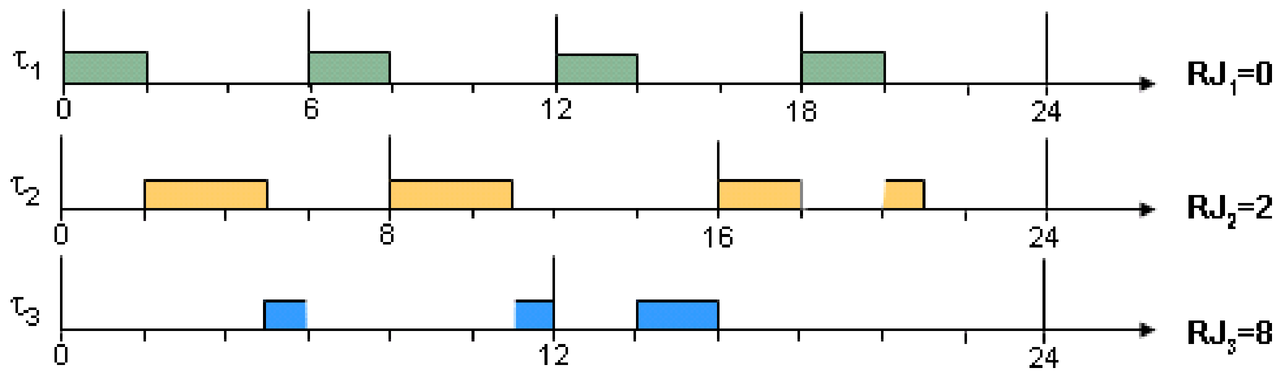
  - If we increase the execution time of τ_3, we get only 1 preemption!



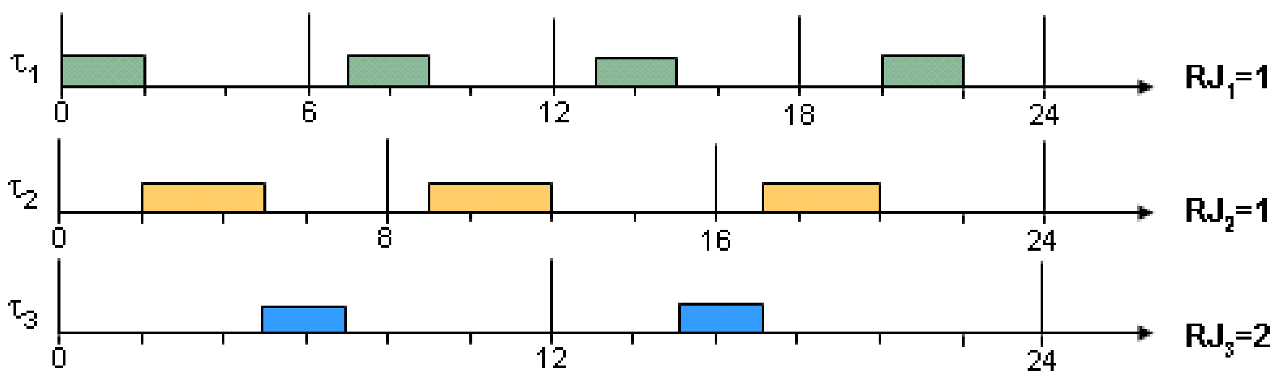
Department of Distributed and Dependable Systems

RM vs. EDF: Effects of Jitter

- **Release jitter under RM:** No release jitter for τ_1 but τ_3 experiences very high jitter.



- **Release jitter under EDF:** For a little increase of release jitter for τ_1 we get large decrease of release jitter for τ_3.



Department of Distributed and Dependable Systems

Figure from Issovic, D.:Real-time systems, basic course

RM vs. EDF: Conclusions

- **RM and EDF have same implementation complexity** – a small additional overhead is needed in EDF to update the absolute deadlines of instances.
- **RM is supported by commercial RTOSs** – a big advantage of RM is that it can be easily implemented on top of fixed priority kernels.
- **Runtime overhead is smaller in EDF** – smaller number of context switches.
- **EDF utilizes the processor better than RM** – EDF achieves full processor utilization, 100%, whereas RM only guarantees 69%
- **EDF is simpler to analyze if $D = T$, RM is simpler for $D < T$**
- EDF is **fair** in reducing jitter, whereas RM only reduces the jitter of the **highest** priority tasks
- **EDF** is more efficient than RM for handling **aperiodic** tasks

Periodic Task Schedulability – Overview

	$D_i = T_i$	$D_i \leq T_i$
Static Priority	**RM** Processor utilization approach $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$	**DM** Response time approach $\forall i: R_i = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \leq D_i$
Dynamic Priority	**EDF** Processor utilization approach $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$	**EDF*** Processor demand approach $\forall L > 0: L \geq \sum_{i=1}^{n} \left(\left\lfloor \frac{L - D_i}{T_i} \right\rfloor + 1 \right) C_i$

Department of Distributed and Dependable Systems