Real-Time Scheduling

Priority-Driven Scheduling

Fixed-Priority

Fixed-Priority Algorithms

Recall that we consider a set of n tasks $\mathcal{T} = \{T_1, \ldots, T_n\}$

Any fixed-priority algorithm schedules tasks of $\mathcal{T}$ according to fixed (distinct) priorities *assigned to tasks.*

We write $T_i \sqsupset T_j$ whenever T_i has a higher priority than T_j.

To simplify our reasoning, assume that

all tasks are in phase, i.e. $\varphi_k = 0$ for all T_k.

We will remove this assumption at the end.

Fixed-Priority Algorithms – Reminder

Recall that Fixed-Priority Algorithms do not have to be optimal.
Consider $\mathcal{T} = \{T_1, T_2\}$ where $T_1 = (2, 1)$ and $T_2 = (5, 2.5)$

$U^{\mathcal{T}} = 1$ and thus $\mathcal{T}$ is schedulable by EDF

If $T_1 \sqsupset T_2$, then $J_{2,1}$ misses its deadline
If $T_2 \sqsupset T_1$, then $J_{1,1}$ misses its deadline

We consider the following algorithms:

- **RM** = assigns priorities to tasks based on their periods
  the priority is inversely proportional to the period p_i
- **DM** = assigns priorities to tasks based on their relative deadlines
  the priority is inversely proportional to the relative deadline D_i

(In all cases, ties are broken arbitrarily.)

We consider the following questions:

- Are the algorithms optimal?
- How to efficiently (or even online) test for schedulability?

Maximum Response Time

Which job of a task T_i has the maximum response time?

As all tasks are in phase, the first job of T_i is released together with (first) jobs of all tasks that have higher priority than T_i.

This means, that $J_{i,1}$ is the most preempted of jobs in T_i.

It follows, that $J_{i,1}$ has the maximum response time.
Note that this relies heavily on the assumption that tasks are in phase!

Thus in order to decide whether $\mathcal{T}$ is schedulable, it suffices to test for schedulability of the first jobs of all tasks.

Optimality of RM for Simply Periodic Tasks

Definition 16

A set $\{T_1, \ldots, T_n\}$ is **simply periodic** if for every pair T_i, T_ℓ satisfying $p_i > p_\ell$ we have that p_i is an integer multiple of p_ℓ

Example 17

The helicopter control system from the first lecture.

Theorem 18

A set $\mathcal{T}$ of n simply periodic, independent, preemptable tasks with $D_i = p_i$ is schedulable on one processor according to RM **iff** $U^{\mathcal{T}} \leq 1$.

i.e. on simply periodic tasks RM is as good as EDF

Note: Theorem 18 is true in general, no "in phase" assumption is needed.

Proof of Theorem 18

By Theorem 14, every schedulable set $\mathcal{T}$ satisfies $U^{\mathcal{T}} \leq 1$.

We prove that if $\mathcal{T}$ is not schedulable *according to RM*, then $U^{\mathcal{T}} > 1$.

Assume that a job $J_{i,1}$ of T_i misses its deadline at $D_i = p_i$. W.l.o.g., we assume that $T_1 \sqsupset \cdots \sqsupset T_n$ according to RM.

Let us compute the total execution time of $J_{i,1}$ and all jobs that preempt it:

$$E = e_i + \sum_{\ell=i+1}^{n} \left\lceil \frac{p_i}{p_\ell} \right\rceil e_\ell = \sum_{\ell=i}^{n} \frac{p_i}{p_\ell} e_\ell = p_i \sum_{\ell=i}^{n} u_\ell \leq p_i \sum_{\ell=1}^{n} u_\ell = p_i U^{\mathcal{T}}$$

Now $E > p_i$ because otherwise $J_{i,1}$ meets its deadline. Thus

$$p_i < E \leq p_i U^{\mathcal{T}}$$

and we obtain $U^{\mathcal{T}} > 1$.

Optimality of DM (RM) among Fixed-Priority Algs.

Theorem 19
A set of independent, preemptable periodic tasks with $D_i \leq p_i$ that are in phase (i.e., $\varphi_i = 0$ for all $i = 1, \ldots, n$) can be feasibly scheduled on one processor according to DM if it can be feasibly scheduled by some fixed-priority algorithm.

Proof.
Assume a fixed-priority feasible schedule with $T_1 \sqsupset \cdots \sqsupset T_n$.

Consider the least i such that the relative deadline D_i of T_i is larger than the relative deadline D_{i+1} of T_{i+1}.

Swap the priorities of T_i and T_{i+1}.

The resulting schedule is still feasible.

DM is obtained by using finitely many swaps. □

Note: If the assumptions of the above theorem hold and all relative deadlines are equal to periods, then RM is optimal among all fixed-priority algorithms.

Note: no "in phase" assumption is needed here.

Fixed-Priority Algorithms: Schedulability

We consider two schedulability tests:

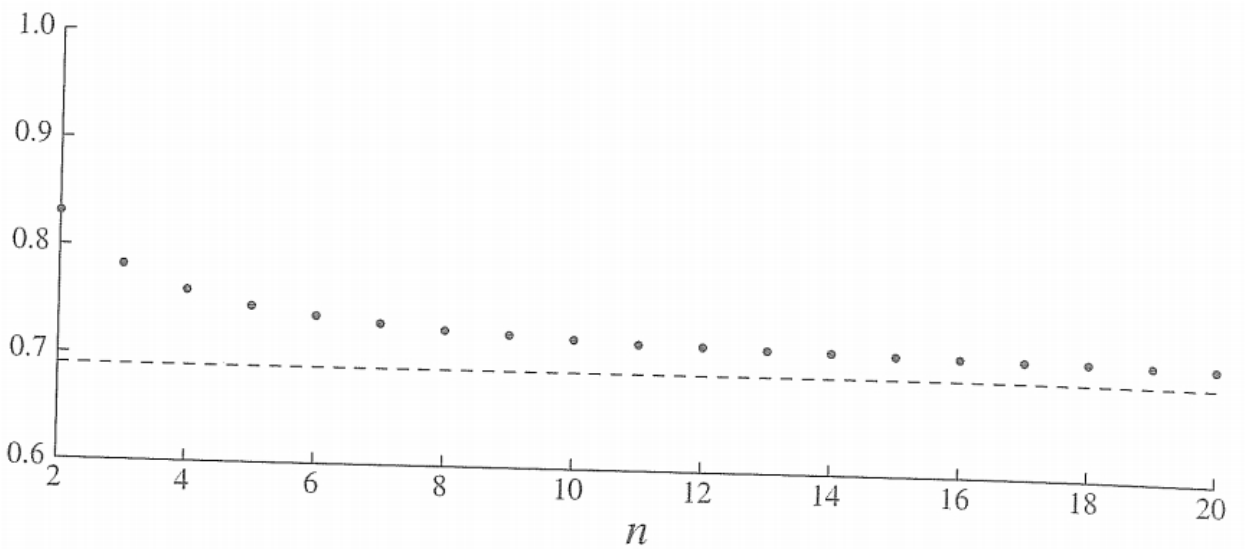
- Schedulable utilization U_{RM} of the RM algorithm.
- Time-demand analysis based on response times.

Schedulable Utilization for RM

Theorem 20

Let us fix $n \in \mathbb{N}$ and consider only independent, preemptable periodic tasks with $D_i = p_i$.

- *If $\mathcal{T}$ is a set of n tasks satisfying $U^{\mathcal{T}} \leq n(2^{1/n} - 1)$, then $U^{\mathcal{T}}$ is schedulable according to the RM algorithm.*
- *For every $U > n(2^{1/n} - 1)$ there is a set $\mathcal{T}$ of n tasks satisfying $U^{\mathcal{T}} \leq U$ that is not schedulable by RM.*



Note: Theorem 20 holds in general, no "in phase" assumption is needed.

Schedulable Utilization for RM

It follows that the maximum schedulable utilization U_{RM} over independent, preemptable periodic tasks satisfies

$$U_{RM} = \inf_n n(2^{1/n} - 1) = \lim_{n \to \infty} n(2^{1/n} - 1) = \ln 2 \approx 0.693$$

Note that $U^{\mathcal{T}} \leq n(2^{1/n} - 1)$ is a sufficient but not necessary condition for schedulability of $\mathcal{T}$ using the RM algorithm (an example will be given later)

We say that a set of tasks $\mathcal{T}$ is *RM-schedulable* if it is schedulable according to RM.

We say that $\mathcal{T}$ is *RM-infeasible* if it is not RM-schedulable.

Proof – Special Case

To simplify, we restrict to two tasks and always assume $p_2 \leq 2p_1$.
(the latter condition is w.l.o.g., proof omitted)

Outline: Given p_1, p_2, e_1, denote by max_e_2 the maximum execution time so that $\mathcal{T} = \{(p_1, e_1), (p_2, max\_e_2)\}$ is RM-schedulable.

We define $U_{e_1}^{p_1,p_2}$ to be $U^{\mathcal{T}}$ where $\mathcal{T} = \{(p_1, e_1), (p_2, max\_e_2)\}$.
We say that $\mathcal{T}$ fully utilizes the processor, any increase in an execution time causes RM-infeasibility.

Now **we find the (global) minimum** $minU$ **of** $U_{e_1}^{p_1,p_2}$**.**
Note that this suffices to obtain the desired result:

- Given a set of tasks $\mathcal{T} = \{(p_1, e_1), (p_2, e_2)\}$ satisfying $U^{\mathcal{T}} \leq minU$ we get $U^{\mathcal{T}} \leq minU \leq U_{e_1}^{p_1,p_2}$, and thus the execution time e_2 cannot be larger than max_e_2. Thus, $\mathcal{T}$ is RM-schedulable.
- Given $U > minU$, there must be p_1, p_2, e_1 satisfying $minU \leq U_{e_1}^{p_1,p_2} < U$ where $U_{e_1}^{p_1,p_2} = U^{\mathcal{T}}$ for a set of tasks $\mathcal{T} = \{(p_1, e_1), (p_2, max\_e_2)\}$.

  However, now increasing e_1 by a sufficiently small $\varepsilon > 0$ makes the set RM-infeasible without making utilization larger than U.

Proof – Special Case (Cont.)

Consider two cases depending on e_1:

1. $e_1 < p_2 - p_1$:

Maximum RM-feasible max_e_2 (with p_1, p_2, e_1 fixed) is $p_2 - 2e_1$.
Which gives the utilization

$$U_{e_1}^{p_1,p_2} = \frac{e_1}{p_1} + \frac{max\_e_2}{p_2} = \frac{e_1}{p_1} + \frac{p_2 - 2e_1}{p_2} = \frac{e_1}{p_1} + \frac{p_2}{p_2} - \frac{2e_1}{p_2} = 1 + \frac{e_1}{p_2}\left(\frac{p_2}{p_1} - 2\right)$$

As $\frac{p_2}{p_1} - 2 \leq 0$, the utilization $U_{e_1}^{p_1,p_2}$ is minimized by maximizing e_1.

2. $e_1 \geq p_2 - p_1$:

Maximum RM-feasible max_e_2 (with p_1, p_2, e_1 fixed) is $p_1 - e_1$. Which gives the utilization

$$U_{e_1}^{p_1,p_2} = \frac{e_1}{p_1} + \frac{max\_e_2}{p_2} = \frac{e_1}{p_1} + \frac{p_1 - e_1}{p_2} = \frac{e_1}{p_1} + \frac{p_1}{p_2} - \frac{e_1}{p_2} = \frac{p_1}{p_2} + \frac{e_1}{p_2}\left(\frac{p_2}{p_1} - 1\right)$$

As $\frac{p_2}{p_1} - 1 \geq 0$, the utilization $U_{e_1}^{p_1,p_2}$ is minimized by minimizing e_1.

The minimum of $U_{e_1}^{p_1,p_2}$ is attained at $e_1 = p_2 - p_1$.
(Both expressions defining $U_{e_1}^{p_1,p_2}$ give the same value for $e_1 = p_2 - p_1$.)

Proof – Special Case (Cont.)

Substitute $e_1 = p_2 - p_1$ into the expression for $U_{e_1}^{p_1,p_2}$:

$$U_{p_2-p_1}^{p_1,p_2} = \frac{p_1}{p_2} + \frac{p_2 - p_1}{p_2}\left(\frac{p_2}{p_1} - 1\right) = \frac{p_1}{p_2} + \left(1 - \frac{p_1}{p_2}\right)\left(\frac{p_2}{p_1} - 1\right)$$

$$= \frac{p_1}{p_2} + \frac{p_1}{p_2}\left(\frac{p_2}{p_1} - 1\right)\left(\frac{p_2}{p_1} - 1\right) = \frac{p_1}{p_2}\left(1 + \left(\frac{p_2}{p_1} - 1\right)^2\right)$$

Denoting $G = \frac{p_2}{p_1} - 1$ we obtain

$$U_{p_2-p_1}^{p_1,p_2} = \frac{p_1}{p_2}(1 + G^2) = \frac{1 + G^2}{p_2/p_1} = \frac{1 + G^2}{1 + G}$$

Differentiating w.r.t. G we get

$$\frac{G^2 + 2G - 1}{(1 + G)^2}$$

which attains minimum at $G = -1 \pm \sqrt{2}$. Here only $G = -1 + \sqrt{2} > 0$ is acceptable since the other root is negative.

Proof – Special Case (Cont.)

Thus the minimum value of $U_{e_1}^{p_1,p_2}$ is

$$\frac{1+(\sqrt{2}-1)^2}{1+(\sqrt{2}-1)} = \frac{4-2\sqrt{2}}{\sqrt{2}} = 2(\sqrt{2}-1)$$

It is attained at periods satisfying

$$G = \frac{p_2}{p_1} - 1 = \sqrt{2}-1 \qquad \text{i.e. satisfying } p_2 = \sqrt{2}p_1.$$

The execution time e_1 which at full utilization of the processor (due to max_e_2) gives the minimum utilization is

$$e_1 = p_2 - p_1 = (\sqrt{2}-1)p_1$$

and the corresponding $max\_e_2 = p_1 - e_1 = p_1 - (p_2 - p_1) = 2p_1 - p_2$.

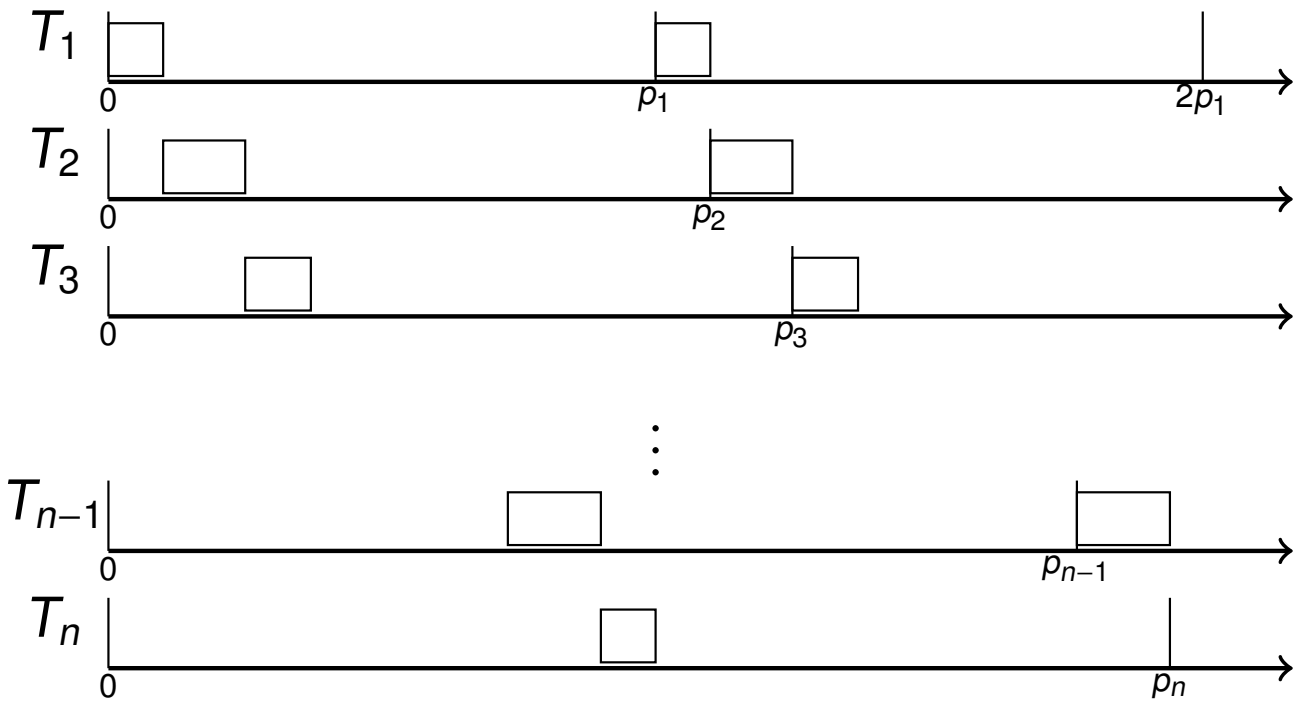
Scaling to $p_1 = 1$, we obtain a completely determined example

$$p_1 = 1 \quad p_2 = \sqrt{2} \approx 1.41 \quad e_1 = \sqrt{2}-1 \approx 0.41 \quad max\_e_2 = 2-\sqrt{2} \approx 0.59$$

that fully utilizes the processor (no execution time can be increased) but has the minimum utilization $2(\sqrt{2}-1)$.

Proof Idea of Theorem 20

Fix periods $p_1 < \cdots < p_n$ so that (w.l.o.g.) $p_n \leq 2p_1$. Then the following set of tasks has the smallest utilization among all task sets that fully utilize the processor (i.e., any increase in any execution time makes the set unschedulable).



$$e_k = p_{k+1} - p_k \qquad \text{for } k = 1, \ldots, n-1$$

$$e_n = p_n - 2\sum_{k=1}^{n-1} e_k = 2p_1 - p_n$$

Time-Demand Analysis

Consider a set of n tasks $\mathcal{T} = \{T_1, \ldots, T_n\}$.
Recall that we consider only independent, preemptable, in phase (i.e. $\varphi_i = 0$ for all i) tasks without resource contentions.

Assume that $D_i \leq p_i$ for every i, and consider an arbitrary fixed-priority algorithm. W.l.o.g. assume $T_1 \sqsupset \cdots \sqsupset T_n$.

Idea: For every task T_i and every time instant $t \geq 0$, compute the total execution time $w_i(t)$ (the time demand) of the first job $J_{i,1}$ and of all higher-priority jobs released up to time t.

If $w_i(t) \leq t$ for some time $t \leq D_i$, then $J_{i,1}$ is schedulable, and hence all jobs of T_i are schedulable.

Time-Demand Analysis

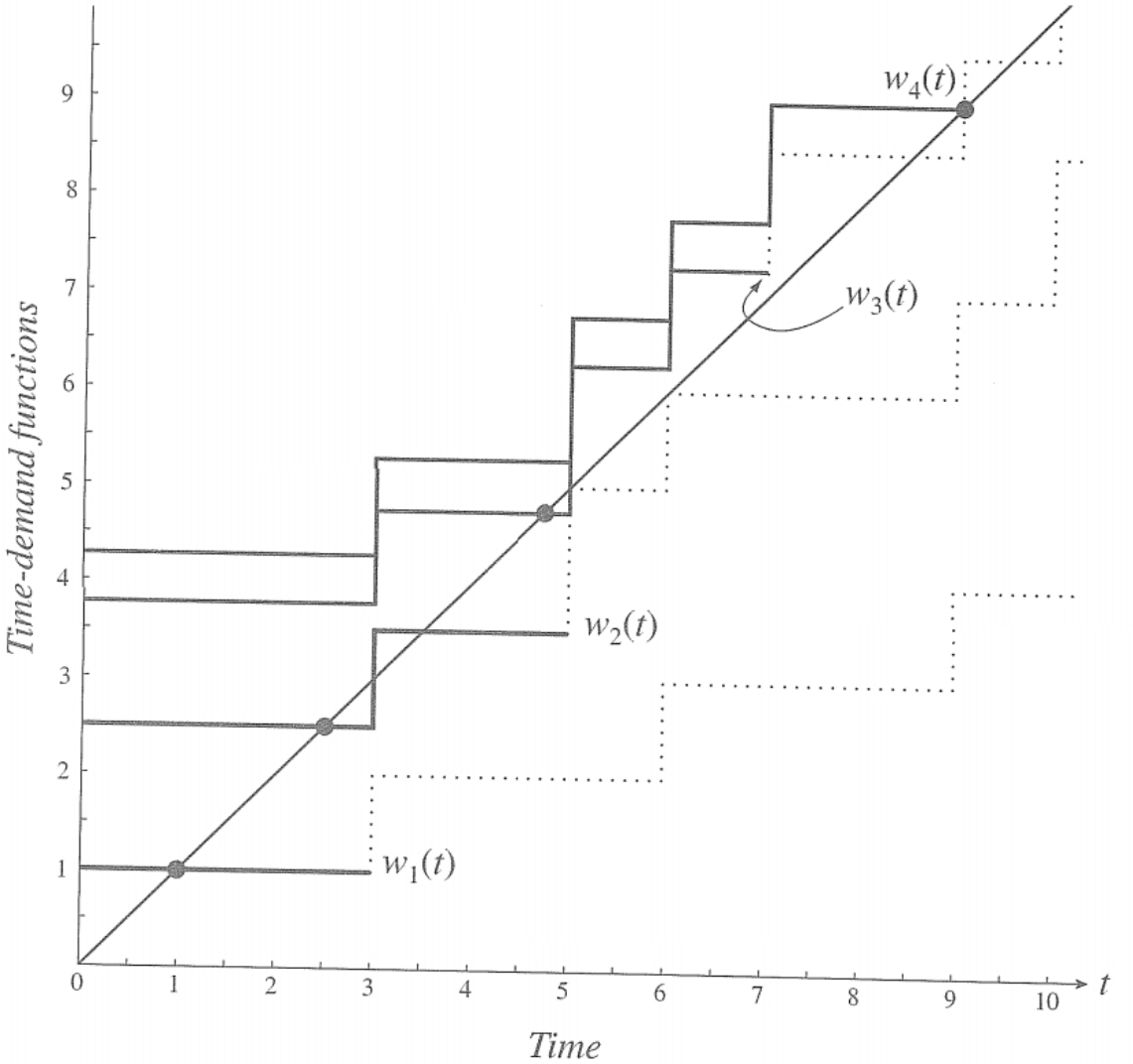
- Consider one task T_i at a time, starting with highest priority and working to lowest priority.
- Focus on the first job $J_{i,1}$ of T_i.

  If $J_{i,1}$ makes it, all jobs of T_i will make it due to $\varphi_i = 0$.
- At time t for $t \geq 0$, the processor time demand $w_i(t)$ for this job and all higher-priority jobs released in $[0, t]$ is bounded by

$$w_i(t) = e_i + \sum_{\ell=1}^{i-1} \left\lceil \frac{t}{p_\ell} \right\rceil e_\ell \qquad \text{for } 0 < t \leq p_i$$

  (Note that the smallest t for which $w_i(t) \leq t$ is the response time of $J_{i,1}$, and hence the maximum response time of jobs in T_i).
- If $w_i(t) \leq t$ for some $t \leq D_i$, the job $J_{i,1}$ meets its deadline D_i.
- If $w_i(t) > t$ for all $0 < t \leq D_i$, then the first job of the task cannot complete by its deadline.

Time-Demand Analysis – Example



Example: $T_1 = (3, 1)$, $T_2 = (5, 1.5)$, $T_3 = (7, 1.25)$, $T_4 = (9, 0.5)$

This is schedulable by RM even though

$U^{\{T_1,\ldots,T_4\}} = 0.85 > 0.757 = U_{RM}(4)$

Time-Demand Analysis

- The time-demand function $w_i(t)$ is a staircase function
  - Steps in the time-demand for a task occur at multiples of the period for higher-priority tasks
  - The value of $w_i(t) - t$ linearly decreases from a step until the next step
- If our interest is the schedulability of a task, it suffices to check if $w_i(t) \leq t$ at the time instants when a higher-priority job is released and at D_i
- Our schedulability test becomes:
  - Compute $w_i(t)$
  - Check whether $w_i(t) \leq t$ for some t equal either to D_i, or to $j \cdot p_k$ where $k = 1, 2, \ldots, i$ and $j = 1, 2, \ldots, \lfloor D_i/p_k \rfloor$

Time-Demand Analysis – Comments

- Time-demand analysis schedulability test is more complex than the schedulable utilization test but more general:
  - Works for *any* fixed-priority scheduling algorithm, provided the tasks have short response time ($D_i \leq p_i$)
    Can be extended to tasks with arbitrary deadlines
- Still more efficient than exhaustive simulation.
- Assuming that the tasks are in phase the time demand analysis is complete.

We have considered the time demand analysis for tasks in phase. In particular, we used the fact that the first job has the maximum response time.

This is not true if the jobs are not in phase, we need to identify the so called *critical instant*, the time instant in which the system is most loaded, and has its worst response time.

Critical Instant – Formally

Definition 21

A **critical instant** t_{crit} of a task T_i is a time instant in which a job $J_{i,k}$ in T_i is released so that $J_{i,k}$ either does not meet its deadline, or has the maximum response time of all jobs in T_i.

Theorem 22

In a fixed-priority system where every job completes before the next job in the same task is released, a critical instant of a task T_i occurs when one of its jobs $J_{i,k}$ is released at the same time with a job from every higher-priority task.

Note that the situation described in the theorem does not have to occur if tasks are not in phase!

Critical Instant and Schedulability Tests

We use critical instants to get upper bounds on schedulability as follows:

- Set phases of all tasks to zero, which gives a new set of tasks $\mathcal{T}' = \{T'_1, \ldots, T'_n\}$

  By Theorem 22, the response time of the first job $J'_{i,1}$ of T'_1 in $\mathcal{T}'$ is at least as large as the response time of every job of T_i in $\mathcal{T}$.

- Decide schedulability of $\mathcal{T}'$, e.g. using the timed-demand analysis.
  - If $\mathcal{T}'$ if schedulable, then also $\mathcal{T}$ is schedulable.
  - If $\mathcal{T}'$ is not schedulable, then $\mathcal{T}$ does not have to be schedulable.

    But may be schedulable, which make the time-demand analysis incomplete in general for tasks not in phase.

Dynamic vs Fixed Priority

- EDF
  - pros:
    - optimal
    - very simple and complete test for schedulability
  - cons:
    - difficult to predict which job misses its deadline
    - strictly following EDF in case of overloads assigns higher priority to jobs that missed their deadlines
    - larger scheduling overhead
- DM (RM)
  - pros:
    - easier to predict which job misses its deadline (in particular, tasks are not blocked by lower priority tasks)
    - easy implementation with little scheduling overhead
    - (optimal in some cases often occurring in practice)
  - cons:
    - not optimal
    - incomplete and more involved tests for schedulability