# Priority-driven Scheduling of periodic tasks: EDF

## OUTLINE

- Dynamic priority algorithms for scheduling periodic tasks
  - Optimality of EDF
  - Schedulability test

**Ref: [Liu]**

- Ch. 6 (pg. 124 – 129)



# Schedulable utilization $U_{EDF} = 1$ if $D_i = p_i$

- Theorem: A system of independent preemptable periodic tasks with relative deadlines equal to their periods is feasible iff its total utilization $U$ is less than or equal to 1.
- Proof:
  - necessary condition (only-if): if the task set is feasible, then $U \leq 1$;

    this is the same as:

    if $U > 1$, then the task set is not feasible,

    which is obvious (we have already proved it);
  - sufficient condition (if): if $U \leq 1$, then the task set is feasible;

    this is the same as:

    if the task set is not feasible, then $U > 1$,

    we show that if EDF fails to find a feasible schedule, then the total utilization must exceed 1.

# $U_{EDF} = 1$ when $D_i = p_i$ (2)

- sufficient condition (if): if $U \le 1$, then the task set is feasible;

  which is the same as:

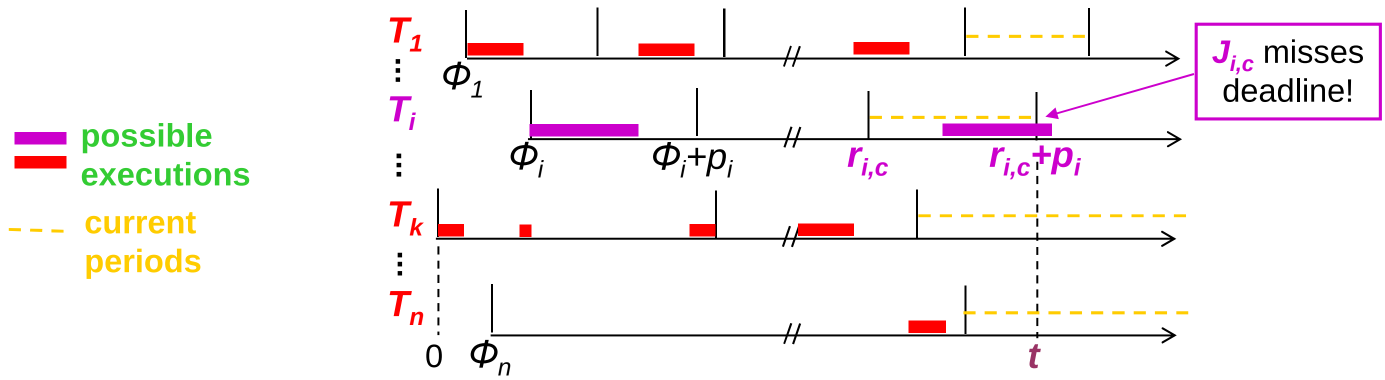
- if EDF fails to find a feasible schedule, then the total utilization must exceed 1.

  - assumptions:
    - at time $t$, job $J_{i,c}$ of task $T_i$ (released at $r_{i,c}$) misses its deadline;
    - the processor never idles prior to $t$;
    - current period of a task $T_k$ is the period that starts before $t$ and ends at or after $t$; (current job: executed in the current period, with $r_{kc} < t$ and $d_{kc} \ge t$)
  - there are 2 cases to consider:
    - case 1: current period of every task begins at or after $r_{i,c}$ ;
    - case 2: current period of some task may start before $r_{i,c}$ .



# $U_{EDF} = 1$ when $D_i = p_i$ (3)

- case 1: current period of every task begins at or after $r_{i,c}$ (no current job with deadline after $t$ is given any processor time before $t$ )



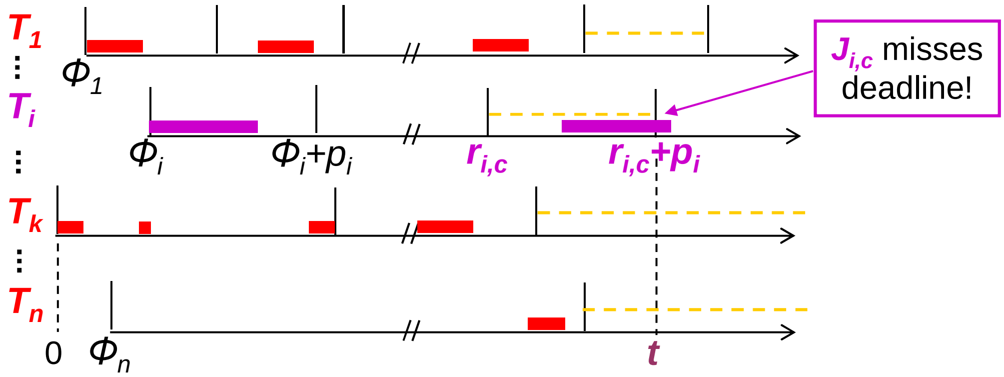

- the number of full periods of task $T_i$ up to time $t$ is: $(t - \Phi_i)/p_i$ (this ratio is an integer number, since $(t - \Phi_i)$ is a multiple of $p_i$ )
  - total processor time required by $T_i$ up to time $t$ is: $\boxed{(t - \Phi_i)e_i / p_i}$
- the number of full periods of any task $T_k \ne T_i$ up to time $t$ is: $\lfloor (t - \Phi_k)/p_k \rfloor$
  - total processor time required by $T_k$ up to time $t$ is: $\boxed{\lfloor (t - \Phi_k)/p_k \rfloor e_k}$

# $U_{EDF} = 1$ when $D_i = p_i$ (4)

- case 1(cont.):

possible executions

current periods




- total processor time required to complete all jobs (of $T_i$ and all $T_{k \neq i}$) with deadline at or before $t$ is: $(t - \Phi_i)e_i / p_i + \Sigma_{k \neq i} \lfloor (t - \Phi_k)/p_k \rfloor e_k$

- if $J_{i,c}$ misses its deadline at $t$, then the total processor time required by the task set before $t$ exceeds the total available time $t$

$$
\begin{aligned}
t &< \frac{(t-\phi_i)e_i}{p_i} + \sum_{k \neq i} \left\lfloor \frac{t-\phi_k}{p_k} \right\rfloor e_k \\
&\leq t \cdot \frac{e_i}{p_i} + t \cdot \sum_{k \neq i} e_k / p_k \\
&= t \cdot U \\
&\Rightarrow U > 1
\end{aligned}
$$

- this implies $U > 1$



# $U_{EDF} = 1$ when $D_i = p_i$ (5)

- case 2: some current periods start before $r_{i,c}$

possible executions

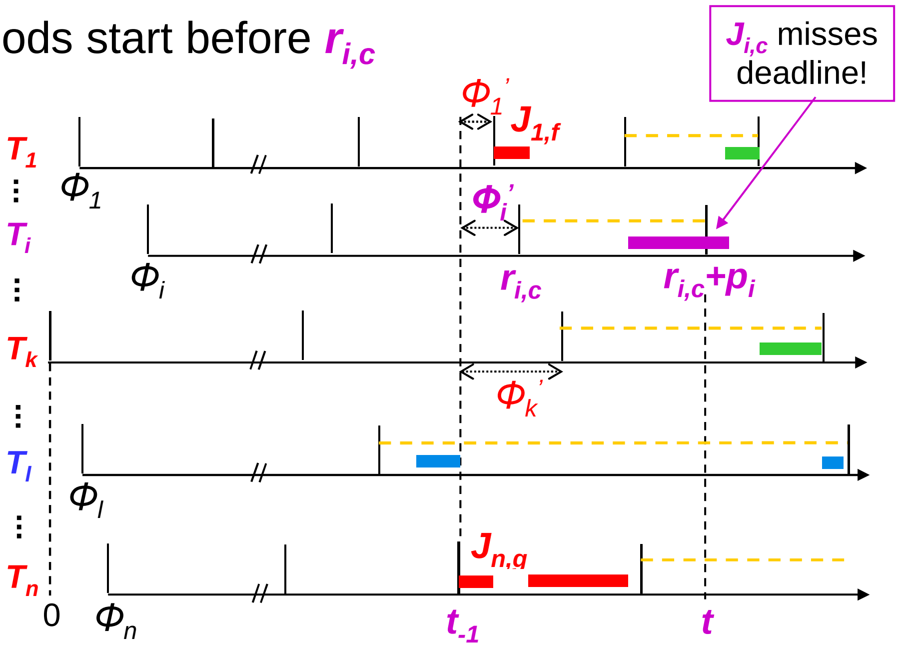
current periods

**T: Set of all tasks.**

**T': tasks with $r'_c < r_{i,c}$ (current job starting before $r_{i,c}$) and with deadline $d'_c > t$ )**
**($T_l$ in the figure belongs to T')**

**T* = T - T': tasks with $r^*_c \geq r_{i,c}$ ($T_i$ belongs to T*), or with $r^*_c < r_{i,c}$ and $d'_c = t$.**

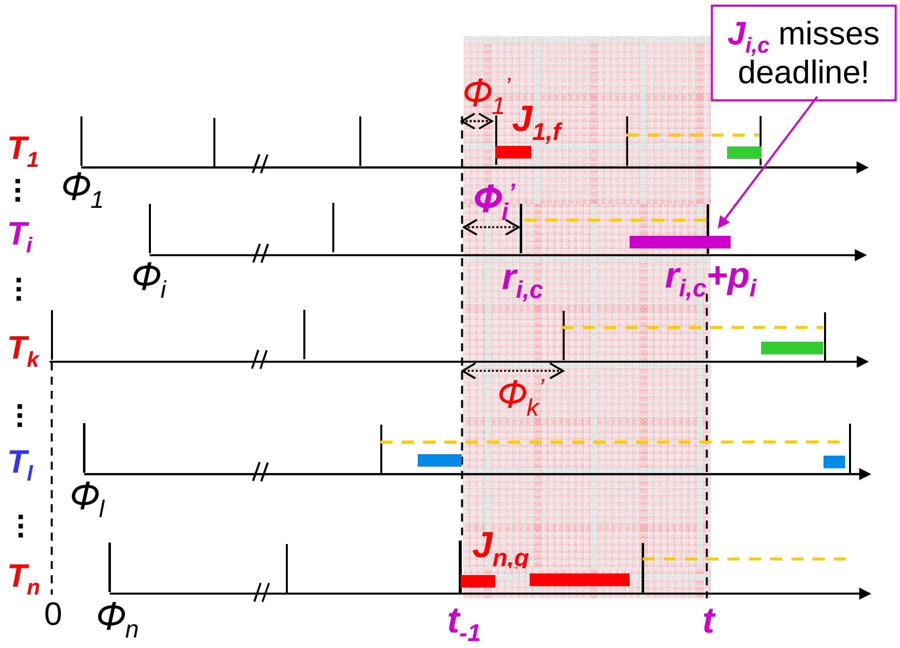
**T = T' + T***




- **$t_{-1}$ : last point in time before $t$ when some current job in T' is executing ($t_{-1} \leq r_{i,c}$).**
- **current jobs in T' may have executed before $t_{-1}$ ; but no job in T' is given any processor time in [$t_{-1}$ , $t$].**
- **the only jobs executed in [$t_{-1}$ , $t$] belong to T*, e.g. $J_{1,f}$ and $J_{n,g}$ in the figure (not in the current period), or jobs with current period ending at $t$ (e.g. $J_{i,c}$).**

# $U_{EDF} = 1$ when $D_i = p_i$ (6)

- case 2 (cont):




- let $\Phi_k'$ be the release time of the first job of task $T_k$ in **T*** starting from $t_{-1}$

**IN THE SEGMENT $[t_{-1}, t]$ :**

- processor time required by $J_{i,c}$ and other jobs of $T_i$ is:
  $\boxed{(t - t_{-1} - \Phi_i')e_i / p_i}$
  (= $me_i$ with $m$ integer ≥ 1);
- none of the current jobs with deadline after $t$ is given any processor time;
- the only jobs executed are: jobs of $T_i$ released at or after $t_{-1}$ (including $J_{i,c}$) and other jobs (of T" = T*- $T_i$) with deadline before $t$ (not in the current period) or at $t$ ;
- total processor time required by all these other jobs (of tasks in T") is:
  $\boxed{\Sigma_{k \in T"} \lfloor (t - t_{-1} - \Phi_k')/p_k \rfloor e_k}$

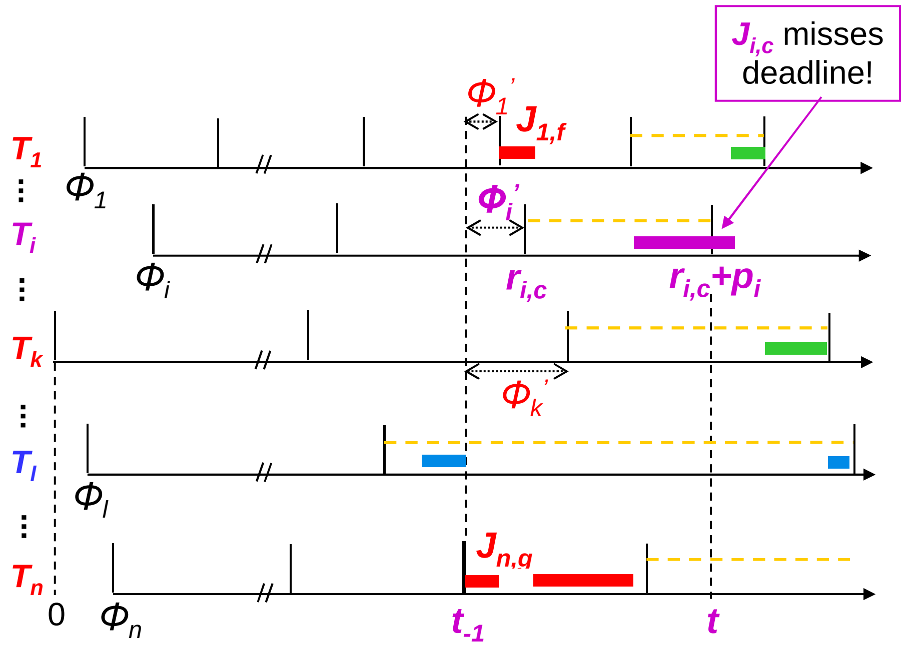


# $U_{EDF} = 1$ when $D_i = p_i$ (7)

- case 2 (cont):

possible executions

current periods




- if $J_{i,c}$ misses its deadline at $t$, then total processor time required by $J_{i,c}$ and all other jobs (of T") in $[t_{-1}, t]$ exceeds the total available time $t - t_{-1}$:

$$t - t_{-1} < (t - t_{-1} - \Phi_i')e_i / p_i + \Sigma_{k \in T"} \lfloor (t - t_{-1} - \Phi_k')/p_k \rfloor e_k$$

$$t - t_{-1} < (t - t_{-1})e_i / p_i + (t - t_{-1})\Sigma_{k \in T"} e_k / p_k$$

$$t - t_{-1} < (t - t_{-1})(e_i / p_i + \Sigma_{k \in T"} e_k / p_k + \Sigma_{j \in T'} e_j / p_j)$$

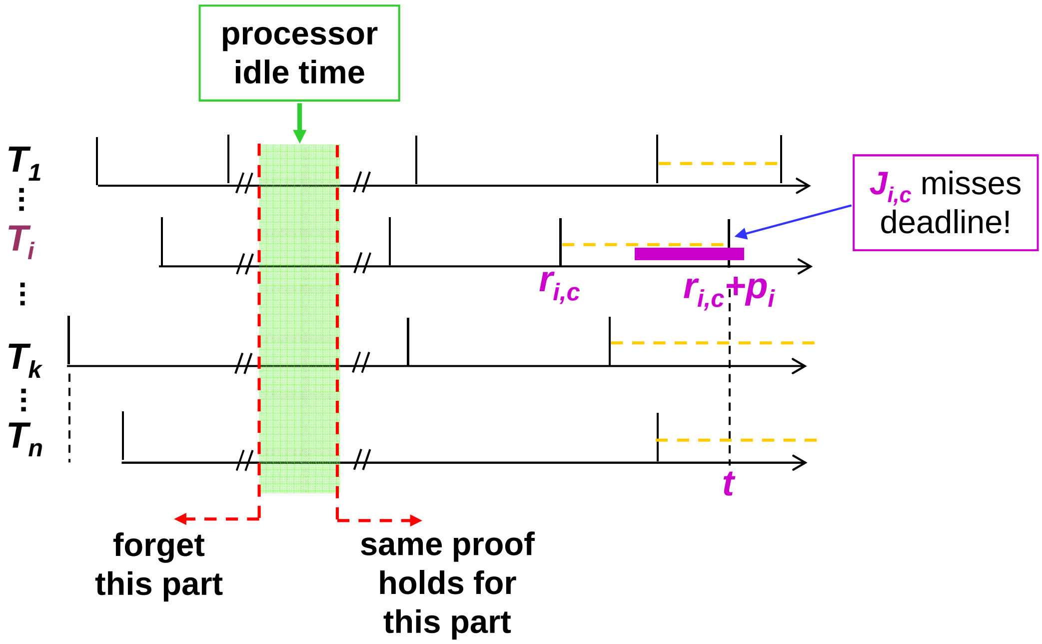
$$t - t_{-1} < (t - t_{-1})U$$

T = T*+T' = $T_i$ +T"+T'

$\boxed{U > 1}$

# $U_{EDF} = 1$ when $D_i = p_i$ (8)

- What about the assumption that the processor never idles?






# EDF: $D_i \geq p_i$

- From the above theorem, it follows that:

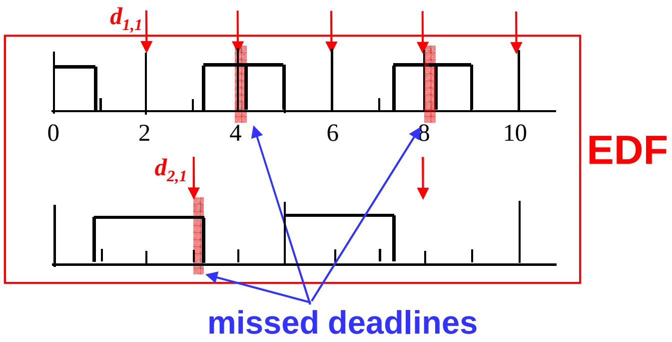
1. A system of independent preemptable periodic tasks with relative deadlines longer than their periods is feasible iff their total utilization is less than or equal to 1.
2. The schedulable utilization $\mathbf{U_{EDF}}$ of the EDF algorithm for independent preemptable periodic tasks with relative deadlines equal to or longer than their periods is 1.

# EDF: $D_i < p_i$ (for some tasks)

- When the relative deadlines of some tasks are less than their period, the system may not be feasible, even when its total utilization is less than 1.

  - ex.: $T_1$=(2,0.9); $T_2$=(5,2.3,3) $U$ = 0.9/2+2.3/5 = 0.91 < 1

  - nevertheless the system is not feasible:






# EDF: DENSITY

- To take into account the possibility that $\boldsymbol{D_i < p_i}$ for some tasks, we define the **DENSITY** of a periodic task $T_i$ as the ratio between its execution time $e_i$ and the minimum between its period $p_i$ and its relative deadline $D_i$ :

$$\delta_i = e_i / \min\{p_i , D_i\}$$

- the density $\Delta$ of a system of $n$ periodic tasks is the sum of the densities of all its tasks:

$$\Delta = \Sigma_{i=1 \text{ to } n}\, e_i / \min\{p_i , D_i\}$$

- since $e_i / \min\{p_i, D_i\} \geq e_i / p_i$ for any $i$, the density of a system of $n$ periodic tasks is never less than its total utilization:

$$\Delta \geq U$$

- if the density of a system is >1 the system may not be feasible

  - ex.: $T_1$=(2,0.9); $T_2$=(5,2.3,3); $\Delta$ = 0.9/2+2.3/3 = 7.3/6 > 1

# EDF: $D_i \leq p_i$

- When $D_i \leq p_i$ for all tasks of a task set, its schedulability may be verified with the following:
- Theorem: a system $T$ of independent, preemptable, periodic tasks, with relative deadlines not longer than their periods, can be feasibly scheduled in one processor if its total density is less than or equal to 1.
- Proof:
- Let's consider the set $T = \{T_1, T_2, \dots T_n\}$ with $D_k \leq p_k$ for any $k$.
- the density of a task $T_k$ is: $\delta_k = e_k / \min\{D_k, p_k\}$
- the total density of the task set $T$ is: $\Delta = \Sigma_{k=1 \text{ to } n} \delta_k$
- we will prove that if the task set $T$ is not schedulable, then $\Delta > 1$.



# EDF: $D_i \leq p_i$ (2)

- let $t$ be the time instant in which a job $J_{i,c}$ of task $T_i$ , released at $r_{i,c}$, misses its deadline: $t = r_{i,c} + D_i$ and let's put the time origin $t = 0$ at the last instant before $t$ in which the processor was idle or executing jobs with deadline after $t$ (we may ignore what happened before $t = 0$).
- then in $[0, t]$ the processor is always executing jobs released at or after 0 and having deadlines $\leq t$ ; if $J_{i,c}$ misses its deadline at $t$, the total time requested by these jobs is greater than $t$ :

$$t < \mathrm{TR}_i + \mathrm{TR}_o$$

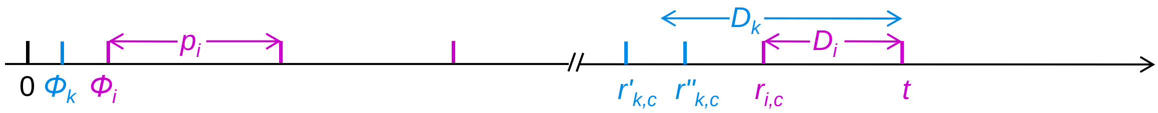
where:

$\mathrm{TR}_i$ = total time requested by all jobs of $T_i$ released before $t$

$\mathrm{TR}_o$ = total time requested by all other jobs of $T$ with deadline $\leq t$

- let's compute these two amounts of time:

# EDF: $D_i \leq p_i$ (3)




- let's compute the two amounts of time $TR_i$ and $TR_o$ :
  - the first job of $T_i$ is released at $\Phi_i$; the last job of $T_i$ before $t$ is released at $r_{i,c} = t - D_i$; the number of full periods of $T_i$ in the time interval $[\Phi_i , r_{i,c}]$ is: $(r_{i,c} - \Phi_i)/p_i$ (this ratio is an integer number); the number of jobs of $T_i$ released in $[\Phi_i , r_{i,c}]$ is:

    $(r_{i,c} - \Phi_i)/p_i + 1 = (t - D_i - \Phi_i)/p_i + 1$

  ➢ $TR_i = [(t - D_i - \Phi_i)/p_i + 1]e_i$
  - the first job of $T_k \in T_o$ is released at $\Phi_k$; the last job of $T_k$ released (at $r_{k,c}$) before $t$ is executed before $t$ only if its deadline $d_{k,c}$ is not later than $t$ : $d_{k,c} = r_{k,c} + D_k \leq t$ i.e. only if $r_{k,c} \leq t - D_k$
  - then the jobs of $T_k$ executed before $t$ are those released in the time interval $[\Phi_k , t - D_k]$; their number is: $\lfloor (t - D_k - \Phi_k)/p_k \rfloor + 1$

  ➢ $TR_o = \Sigma_{k\in o,\, k\neq i} [\lfloor (t - D_k - \Phi_k)/p_k \rfloor + 1]e_k$



# EDF: $D_i \leq p_i$ (4)

- if $J_{i,c}$ misses its deadline at $t$ then the processor time requested by all jobs of $T$ up to the time $t$ is greater than $t$ :

  $t < [(t - D_i - \Phi_i)/p_i + 1]e_i + \Sigma_{k\in o,\, k\neq i} [\lfloor (t - D_k - \Phi_k)/p_k \rfloor + 1]e_k$

  $\leq [(t - D_i)/p_i + 1]e_i + \Sigma_{k\in o,\, k\neq i} [(t - D_k)/p_k + 1]e_k$
  - consider the first term: $[(t - D_i)/p_i + 1]e_i$
    - if $D_i = p_i$ , the term reduces to: $te_i /p_i = t\delta_i$
    - if $D_i < p_i$ , the term is $< [(t - D_i)/D_i + 1]e_i = te_i /D_i = t\delta_i$

    in any case the first term is $\leq t\delta_i$: $[(t - D_i)/p_i + 1]e_i \leq t\delta_i$
  - similarly for the second term:

    $\Sigma_{k\in o,\, k\neq i} [(t - D_k)/p_k + 1]e_k \leq \Sigma_{k\in o,\, k\neq i}\, t\delta_k = t\Sigma_{k\in o,\, k\neq i}\, \delta_k$
- then the inequality:

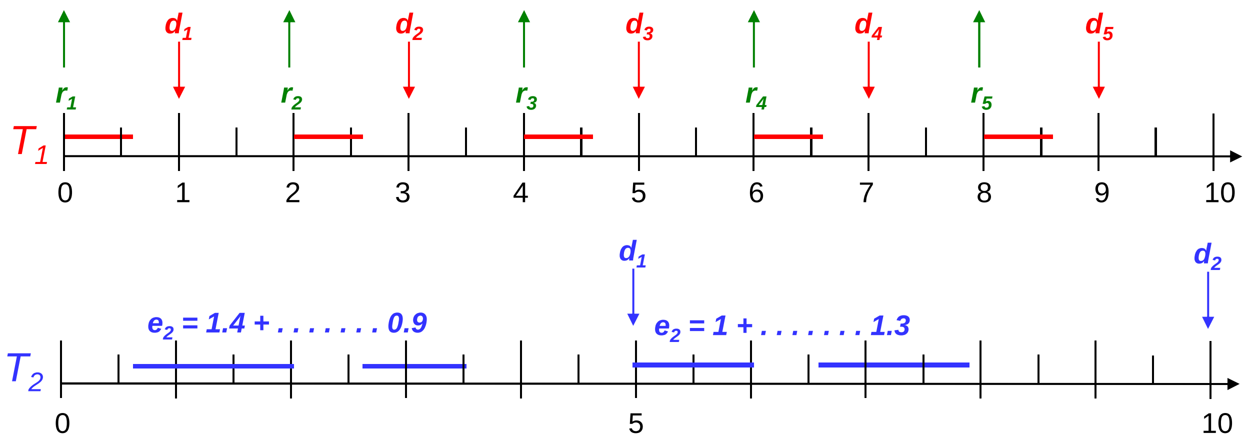
  $t < [(t - D_i - \Phi_i)/p_i + 1]e_i + \Sigma_{k\in o,\, k\neq i} [\lfloor (t - D_k - \Phi_k)/p_k \rfloor + 1]e_k$

  becomes: $t < t\delta_i + t\Sigma_{k\in o,\, k\neq i}\, \delta_k = t\Sigma_{k=1 \text{ to } n}\, \delta_k = t\Delta$
- $t < t\Delta$ implies $\Delta > 1$.

➢ the theorem gives only a sufficient condition for the schedulability of task sets with $D_i \leq p_i$ : a task set may be schedulable even when $\Delta > 1$

# Feasible system with Δ > 1

- Example: $T_1$ = (2,0.6,1); $T_2$ = (5,2.3)
  - Density Δ = 0.6/1 + 2.3/5 = 0.6 + 0.46 =1.06 > 1
  - EDF schedule:




  - The system is feasible with Δ > 1



# EDF: arbitrary deadlines

- From the previous two theorems, follows the following general:
- Theorem: a system *T* of independent, preemptable, periodic tasks, with arbitrary relative deadlines and periods, can be feasibly scheduled in one processor if its density is less than or equal to 1 (if $\Delta \leq 1$).
- Proof:
  - case 1. $D_k \geq p_k$ for all *k*:
    in this case $\delta_k = e_k / min\{D_k, p_k\} = e_k / p_k = u_k$ *for all k*; then $U = \Delta$; $\Delta \leq 1$ implies $U \leq 1$, which, according to the first of the last two theorems, is a necessary and sufficient condition for schedulability;
  - case 2. $D_k < p_k$ for some *k*:
    according to the last theorem, $\Delta \leq 1$ is a sufficient condition for schedulability.

# Schedulability test

SCHEDULABILITY TEST:

- For a system $T = \{T_1, T_2, \dots T_n\}$ of independent periodic tasks, we are given:
  - $p_i$, $e_i$, $D_i$ (period, execution time and relative deadline) of every $T_i$
  - a priority driven algorithm to schedule $T$ preemptively on one processor
- Determine whether all the deadlines of every task are always met

- If the schedulability test is efficient, it can be used as an on-line acceptance test.



# Schedulability test for EDF

- When the scheduling algorithm is EDF, the 2 theorems presented previously (about optimality and density) provide the theoretical basis for a simple general schedulability test:
- Check the inequality:

$$\boxed{\Sigma_{k=1 \text{ to } n}\, e_k \,/\, \min(D_k, p_k) \leq 1}$$

  - if it is satisfied, the system is schedulable
  - if it is not satisfied then:
    - if $D_k \geq p_k$ for all $k$, then the inequality reduces to $U \leq 1$ which is both a necessary and sufficient condition; if the inequality is not satisfied ⇨ the system is not schedulable;
    - if $D_k < p_k$ for some $k$, then the inequality is only a sufficient condition, ⇨ the system may not be schedulable.

# Robustness of schedulability test

- Robustness against variations of execution times:
  - a system which passes the schedulability test remains feasible even if some jobs have a shorter execution time (recall that the execution of independent preemptable jobs on one processor is predictable): no anomalous behaviour may take place.
- Robustness against variations of interrelease times:
  - the theorem in slide 2 has been proved taking into account the total demand of processor time by the jobs of each task, regardless of the actual values of their periods: if the interrelease time of some jobs is longer, then the total demand of processor time becomes smaller; as a consequence:
  - a system which passes the schedulability test remains feasible even if some jobs have a longer interrelease time.



# Schedulability test as a design aid

- The simple schedulability test:

$$\boxed{\Sigma_{k=1 \text{ to } n}\, e_k \,/\, \min\,(D_k\,,\,p_k)\ \leq 1}$$

  may be used during system design to guide the choices of periods of tasks related to their execution times:
  - adding a new task to the systems without causing the other tasks to miss any deadline may require to adjust task periods in such a way that the test is satisfied;
  - if the other task periods cannot be modified, the test inequality provides the minimum value for the period of the new task that guarantees feasibility of the system.
- The same simple schedulability test:

$$\Sigma_{k=1 \text{ to } n}\, e_k \,/\, \min\,(D_k\,,\,p_k)\ \leq 1$$

  may be used as an online acceptance test for adding new tasks.

# EDF: time demand analysis

- The simple sufficient condition for EDF schedulability:

  $\boxed{\Delta \leq 1}$

  may be conveniently used during the design phase to validate a set of periodic tasks with arbitrary deadlines.

- When $D_k \leq p_k$ for any $k = 1, .. n$, a more accurate schedulability test may be performed, based on the evaluation of the following 2 functions:
  - **W(*t*)**: **time demand**: processor time demanded by all jobs released before *t*, i.e. in the time interval $[0, t)$;
  - **V(*t*)**: **demanded time**: processor time that must have been provided to meet the demand of all jobs having deadlines before or at *t*, i.e. in the time interval $[0, t]$.



# Processor time demand functions

- the number of jobs of a task $T_k$ released in the time interval $[0, t)$ is $\lceil (t - \Phi_k)/p_k \rceil$; the total processor time demanded by all these jobs is $\lceil (t - \Phi_k)/p_k \rceil e_k$; then, for the whole task set:

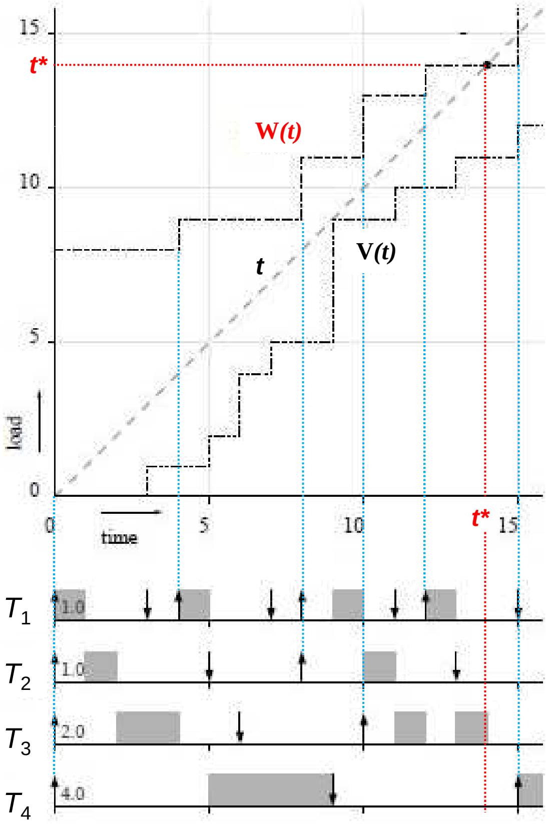
$$W(t) = \Sigma_{k=1 \text{ to } n} \lceil (t - \Phi_k)/p_k \rceil e_k$$

- to evaluate V(*t*):
  - the number of full periods of $T_k$ in $[0, t]$ is $\lfloor (t - \Phi_k)/p_k \rfloor$; the number of jobs of $T_k$ released in $[0, t]$ is $\lfloor (t - \Phi_k)/p_k \rfloor + 1$;
  - since $D_k \leq p_k$, all of these jobs except the last one (let's call $r_{k,c}$ its release time) must certainly have been completed before *t*, while the last one must have been executed before t only if its deadline $d_{k,c} = r_{k,c} + D_k$ is not later then *t*, i.e. if $r_{k,c} + D_k \leq t$, or $r_{k,c} \leq t - D_k$
  - then the jobs of $T_k$ that must have been completed before *t* are those released in $[\Phi_k, t - D_k]$; their number is $\lfloor (t - D_k - \Phi_k)/p_k \rfloor + 1$;
  - the processor time that must have been used by all these jobs is

$$V(t) = \Sigma_{k=1 \text{ to } n} (\lfloor (t - D_k - \Phi_k)/p_k \rfloor + 1)e_k$$

# Example: time demand W(*t*)

ex: EDF schedule of $T_1 = (4,1,3)$; $T_2 = (8,1,5)$; $T_3 = (10,2,6)$; $T_4 = (15,4,9)$




**W(*t*)** is a step function that increases of a quantity $e_k$ whenever a job of $T_k$ is released: i.e. at $t = ip_k$, for $k$ = 1,2,3,4 and $i$ = 0,1,2,3,…

The 45° line represents the available processor time.

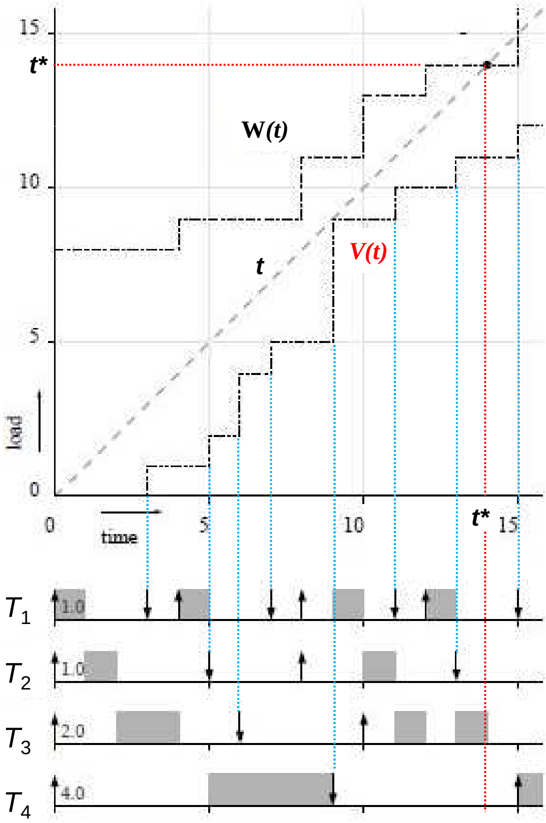
The difference between W($t$) and this line represents the amount of processor time necessary to complete all jobs that have been released up to $t$. While all of these jobs have not completed, W($t$) lies above the 45° line.

When W($t$) touches the 45° line, $t^* = W(t^*)$: the available processor time has been sufficient to satisfy the demand of all jobs released up to $t^*$ (the processor starts an idle period, unless a new job is released at $t^*$).



# Example: demanded time V(*t*)

ex: EDF schedule of $T_1 = (4,1,3)$; $T_2 = (8,1,5)$; $T_3 = (10,2,6)$; $T_4 = (15,4,9)$




**V(*t*)** is a step function that increases of a quantity $e_k$ at each instant in which a job of $T_k$ has its deadline:
i.e. at $t = id_k$, for $k$ = 1,2,3,4 and $i$ = 1,2,3,…

As long as V($t$) lies below the 45° line, $V(t) \leq t$, then all deadlines up to $t$ have been met.

If V($t$) crosses the 45° line at $t = t^\wedge$, $V(t^\wedge) > t^\wedge$, the available processor time has not been sufficient to satisfy the demand of all jobs with deadlines before or at $t^\wedge$ and the deadline at $t^\wedge$ is missed.

(In the figure no deadline is missed.)

# Demanded time function V(*t*)

$$V(t) = \Sigma_{k=1 \text{ to } n} (\lfloor (t - D_k - \Phi_k)/p_k \rfloor + 1)e_k$$

- A set of tasks with $D_k \leq p_k$ (for any $k$) is feasible if and only if the time demanded by its jobs never exceeds the available time, i.e. iff

  $\boxed{V(t) \leq t}$ (for any $t$)

- In case all tasks have zero phase ($\Phi_k = 0$, for any $k$), the demanded time function becomes:

  $$\mathbf{V(t)} = \Sigma_{k=1 \text{ to } n} (\lfloor (t - D_k)/p_k \rfloor + 1)e_k = \mathbf{\Sigma_{k=1 \text{ to } n} \lfloor (t - D_k + p_k)/p_k \rfloor e_k}$$

  - If $V(t) \leq t$ throughout the first hyperperiod, then the time demand of all jobs released in the hyperperiod never exceeds the available processor time; since the schedule repeats itself identical in each hyperperiod, the task set is feasible.
  - Since V(*t*) increases at each deadline $d_k$ and remains constant until the next deadline $d_{k+1}$, the condition $V(t) \leq t$ may be verified only for values of *t* equal to jobs' deadlines in the first hyperperiod:

    $$\boxed{\Sigma_{k=1 \text{ to } n} \lfloor (t - D_k + p_k)/p_k \rfloor e_k \leq t \quad \text{for } t = d_k \ (d_k \leq H)}$$



# Demanded time V(*t*) and utilization U

- For tasks sets with $D_k = p_k$ (for all $k$), the condition $V(t) \leq t$ becomes:

  $$V(t) = \Sigma_{k=1 \text{ to } n} \lfloor (t - D_k + p_k)/p_k \rfloor e_k = \Sigma_{k=1 \text{ to } n} \lfloor t/p_k \rfloor e_k \leq t$$

  which is equivalent to the condition $U \leq 1$.
- Proof:
  - part 1: if $U \leq 1$, then $V(t) \leq t$ for all $t > 0$.

    if $U \leq 1$, then, for all $t$, we have:

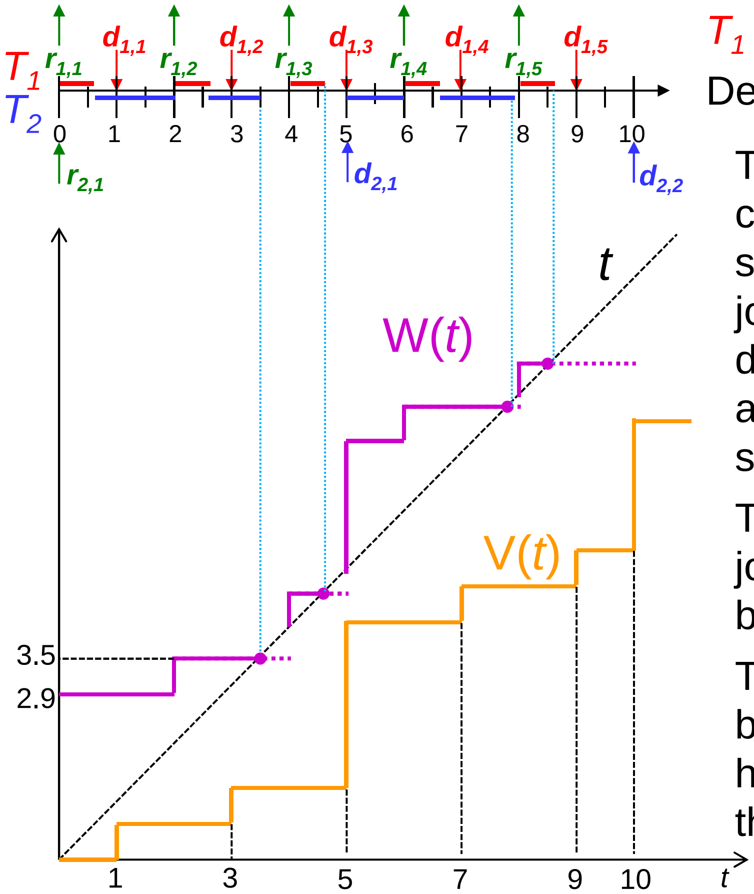
    $t \geq Ut = \Sigma_{k=1 \text{ to } n} (t/p_k)e_k \geq \Sigma_{k=1 \text{ to } n} \lfloor t/p_k \rfloor e_k = V(t)$
  - part 2 if $V(t) \leq t$ for all $t > 0$, then $U \leq 1$.

    we show that if $U > 1$, then $V(t) > t$ for at least one value of $t$.

    if $U > 1$, then for $t^* = lcm(p_1, \ldots, p_n)$ we have:

    $t^* < Ut^* = \Sigma_{k=1 \text{ to } n} (t^*/p_k)e_k = \Sigma_{k=1 \text{ to } n} \lfloor t^*/p_k \rfloor e_k = V(t^*)$

# Example: W($t$) and V($t$)




$T_1$ = (2,0.6,1); $T_2$ = (5,2.3)

Density Δ = 1.06 > 1 - EDF schedule

The time demand function **W($t$)** crosses the 45° line at $t$ = 3.5 and the system becomes idle; when a new job ($J_{1,3}$) is released at $t$ = 4, its time demand (of $e_1$ = 0.6 time units) starts at $t$ = 4; thus the new time demand steps up to 4 + 0.6: W(4) = 4.6

The same happens each time a new job is released after the system has become idle: at $t$ = 5, $t$ = 8, $t$ = 10.

The demanded time function **V($t$)** lies below the 45° line during the whole hyperperiod of the task set: therefore the task set is feasible.