GESTIONE DEL SOVRACCARICO (OVERLOAD)

CAUSE: 1) ATTIVAZIONI ECCESSIVE

2) TEMPO DI ESECUZIONE ECCESSIVO (C)

Dobbiamo definire che cos'è il CARICO

↳ Teoria delle code:

$$\rho = \lambda C$$

- ρ: carico
- λ: frequenza media con cui arrivano le richieste
- C: tempo di computazione richiesto dalla singola richiesta

es.

$$\Rightarrow \rho = \lambda_1 C_1 + \lambda_2 C_2 \rightarrow \rho = \sum_i \lambda_i C_i = \rho = \sum_i \frac{C_i}{T_i} = U$$

↳ $\rho = U$ se il problema è totalmente periodico

Se $\rho > 1$ → siamo in overload

Cenno di un problema aperiodico:

$$\beta = \max_{t_1, t_2 \in [t_a, t_b]} \frac{g(t_1, t_2)}{t_2 - t_1}$$

$g(t_1, t_2)$ — computazione richiesta da t_1 a t_2

$t_2 - t_1$ — tempo computazionale residuo dei processi da terminare entro t_2 calcolando da t_1

istanti che andremo con l'obiettivo di massimizzare (rispetto a t_1 , t_2) $\frac{g(t_1, t_2)}{t_2 - t_1}$

Oss

Il numero di controlli da fare andando t_1 e t_2 è asintoticamente $O((t_a - t_b)^2)$

es. prendo t_1 e lo metto a t_b, t_2 a $t_b + 1$, ma ne rimangono da $t_b + 2$ a t_a e poi li sposto di nuovo

Possiamo fare t_1 = TEMPO CORRENTE = a_i — ossia solo quando si attiva un processo

$t_2 = \forall d_j$ attiva

per ogni deadline attiva

Mo Tu We Th Fr Sa Su

No. SISTEMI INF.

Date 30 · 10 · 17

In particolare se:

$t_i = a_i$

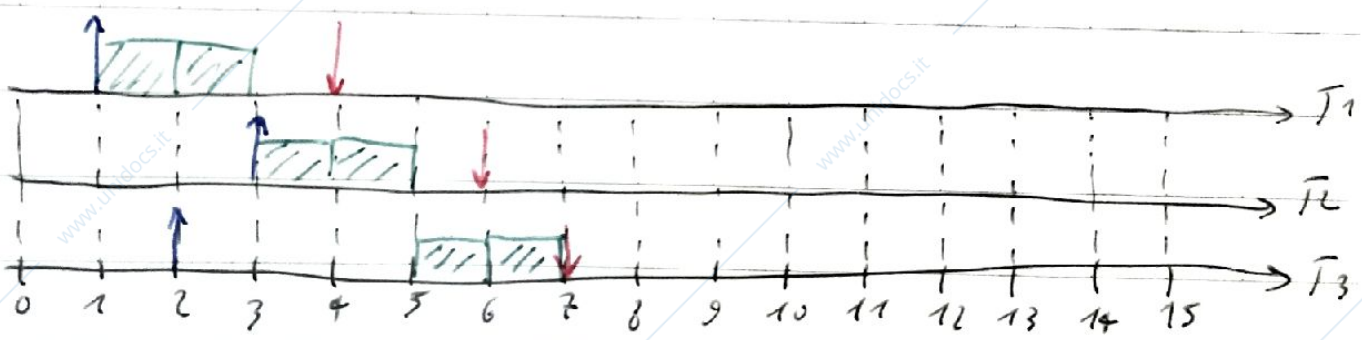
dobbiamo considerare tutte le solite finestre

$[d_j, t]$ $\forall d_j$ ATTIVA

es.

	J_1	J_2	J_3
a_i	1	3	2
c_i	2	2	2
d_i	4	6	7

Supponiamo di usare EDF



- Istante 1: $t_1 = 1$ $t_2 = 4$

$$\frac{g(t_1,t_2)}{t_2 - t_1} = \frac{2}{3} \quad \leftarrow C_1$$

da 1 a 4 devo fare solo J1

$\beta_1 = \frac{2}{3}$

- Istante 2: $t_1 = 2$ $t_2 \in \{4, 7\}$

1° caso ($t_2 = 4$)

$$\frac{g(2,4)}{2} = \frac{1}{2}$$

Solo J1 va fatto entro 4

C_1 rimasta da fare

2° caso ($t_2 = 7$): C_1 residuo

$$\frac{g(2,7)}{7-2} = \frac{1+2}{5} = \frac{3}{5} \quad \leftarrow C_3$$

MAX

$\beta_2 = \frac{3}{5}$

J_3 da fare entro 7

4

No. SISTEMI INF.

Date 30 · 10 · 19

- Istante 3: $t_1 = 3$ $t_2 \in \{6, 7\}$

$t_1 = 3$ $t_2 = 6$

$\frac{g(6,3)}{6-3} = \frac{2}{3}$ — C_2

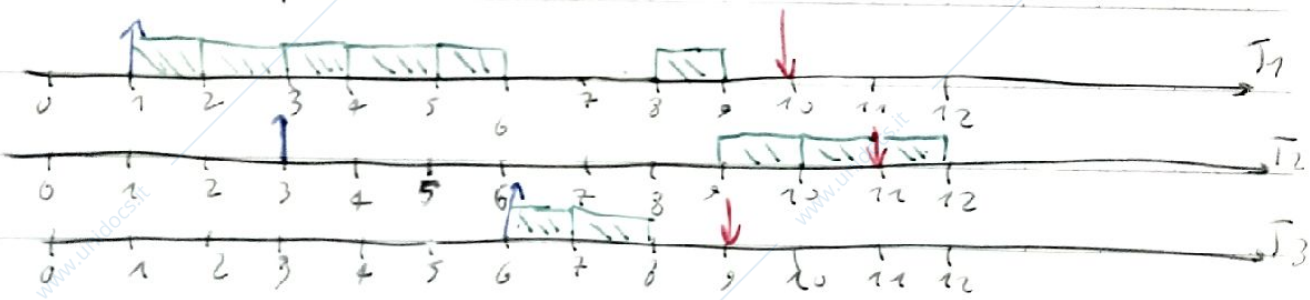
$t_1 = 3$ $t_2 = 7$

$\frac{g(7,3)}{7-3} = \frac{4}{4} = 1$ — $C_2 + C_3$

$p_3 = 1$ ← max

Es.

	T_1	T_2	T_3
a_i	1	3	6
c_i	6	3	2
d_i	10	11	9



- Istante 1: $t_1 = 1$ $t_2 = 10$ $\frac{g(t_2 - t_1)}{9} = \frac{6}{9} = \frac{2}{3}$

$p_1 = \frac{2}{3}$

- Istante 3: $t_1 = 3$ $t_2 \in \{10, 11\}$

$\frac{g(10,3)}{7} = \frac{4}{7}$ $\frac{g(11,3)}{8} = \frac{4+3}{8} = \frac{7}{8}$ ← max

$p_2 = \frac{7}{8}$

- Istante 5: $t_1 = 6$ $t_2 \in \{9, 10, 11\}$

$\frac{g(9,6)}{3} = \frac{2}{3}$ — C_3

Ci resida

$$\frac{1}{4}(10,6) = \frac{(2)+(2)}{4} = \frac{3}{4}$$

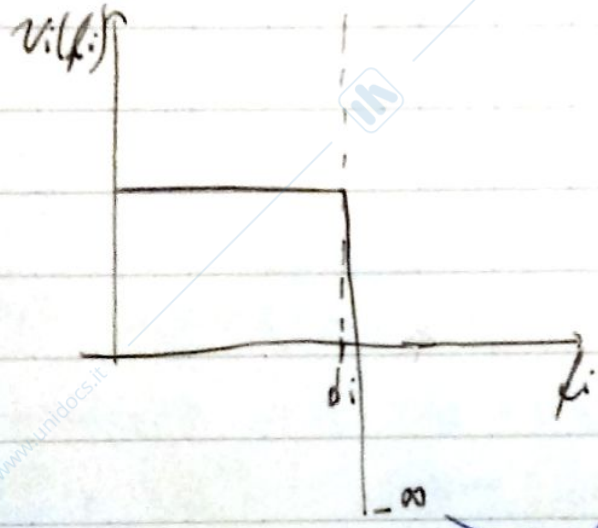
C3

Ci resi. C3 C2

$$\frac{1}{5}(14,6) = \frac{(1)+(2)+(3)}{5} = \frac{6}{5} > 1 \text{ overload}$$

Gestione dell'overload come problema di ottimizzazione

Si da ad ogni processo una funzione di valore



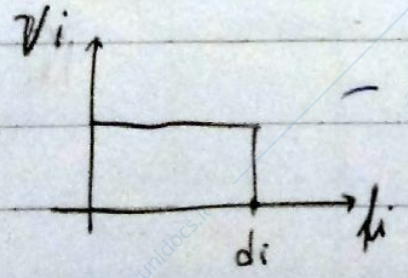
$V_i(f_i)$

↓

valore dato dal processo i-esimo quando è terminato a f_i

perdo tutto il valore accumulato se non lo eseguo in tempo

processo critico



- questi processi si dicono FIRM
- se non terminiamo in tempo il valore scende a 0

Gamma

Dunque se abbiamo i processi:

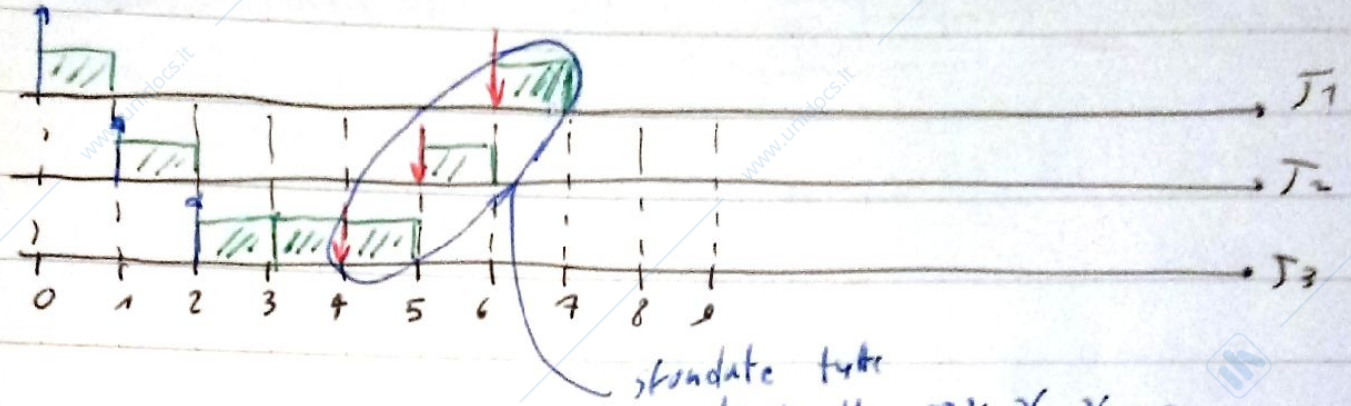
$1 \rightarrow f_1 \rightarrow v_1(f_1)$
$2 \rightarrow f_2 \rightarrow v_2(f_2)$
$3 \rightarrow f_3 \rightarrow v_3(f_3)$
$4 \rightarrow f_4 \rightarrow v_4(f_4)$
...

$\rightarrow S \rightarrow \sum v_i(f_i) = \Gamma(S)$

	J_1	J_2	J_3
a_i	0	1	2
c_i	2	2	3
d_i	6	5	4
v_i	1	1	1

→ tipo FIRM: terminati entro di $v_i = v_i$

non terminati entro di $v_i = 0$



sfondate tutte le deadline → $v_1, v_2, v_3 = 0$

$\Gamma_{EDF} = 0$

Oss. Si possono trovare degli esempi con tutti i processi ben definiti (ossia che posti singolarmente non sfondano la loro deadline) dove $\Gamma_{EDF} \geq 1$ e $\frac{\Gamma_{EDF}}{\Gamma^*} = \frac{1}{n}$ che per $n \to +\infty$ tende comunque asintoticamente a 0

Nell'esempio precedente posso prendere almeno $\Gamma^* = 2$, abolendo J_3

$$\alpha_A = \frac{\Gamma_A(P)}{\Gamma^*(P)}$$ (problema)

α_A: FATTORE COMPETITIVO → PERFORMANCE DELL'ALGORITMO

CLAIROVEGGENTE: ossia l'algoritmo ottimo perché sa tutto

Voglio minimizzare $\alpha_A \in [0,1]$

α_A ← nostro algoritmo → $\min_P \frac{T_A(P)}{T^*}$

quantità che almeno prendiamo ← ossia il caso peggiore

Es.

Se $\alpha_A = \frac{1}{2}$

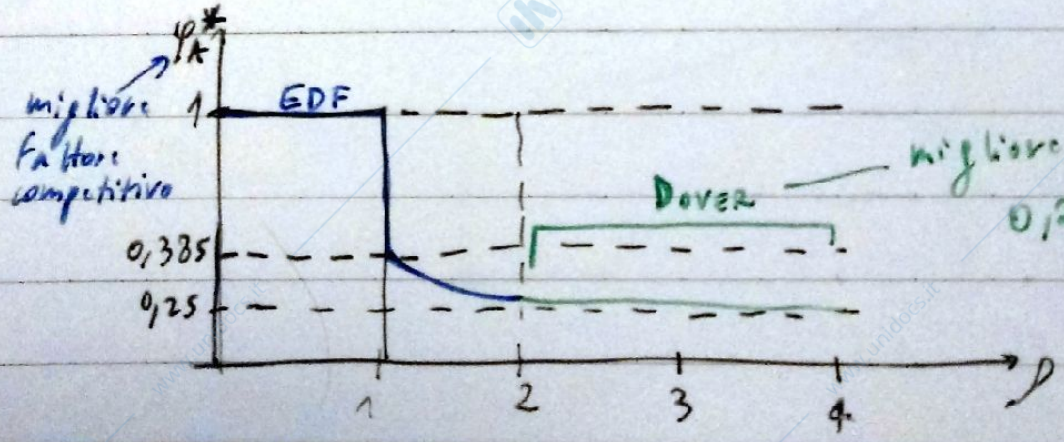
↳ il nostro algoritmo ci conviene di prendere almeno $\frac{1}{2}$ della soluzione ottima

EDF nel caso peggiore ci da 0

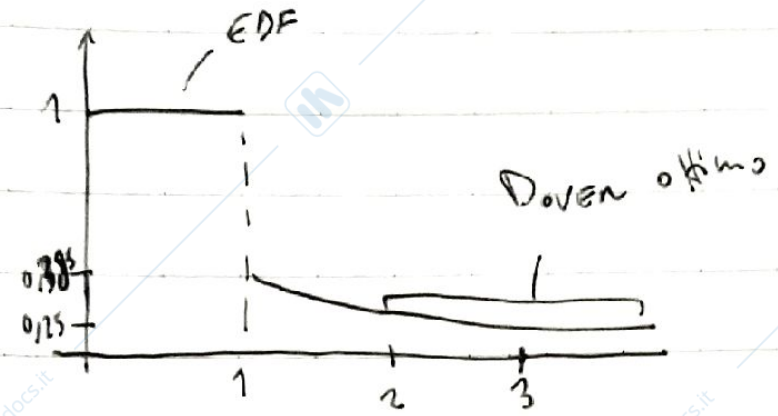
Se il problema è schedulabile EDF è ottimo

se il problema non è schedulabile EDF non è quello ottimo

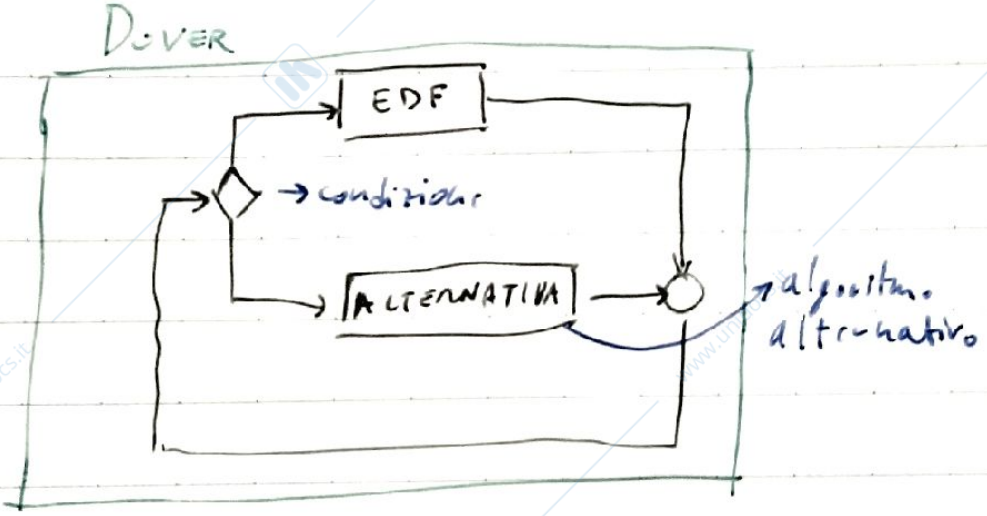
Oss. più il carico cresce più è difficile avere un buon fattore competitivo.



DOVER

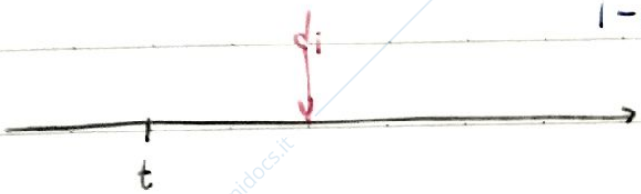


→ Come funziona DOVER?



Dobbiamo definire LST: (t)

↳ latest starting time del processo i-esimo a t



↑ di - ci (ultimo istante nel quale il processo se inizia in t riesce a finire in tempo, altrimenti no)

ad ogni t

1) J_{CURR} ← Processo scelto da EDF all'istante t

2) Poi verifichiamo se esiste un processo J_z t.c. $LST_z(t) = t$ $(z \neq CURR)$

↓

vuol dire che o lo eseguo in t o altrimenti non lo terminerò mai

TEST → $V_z > (1+\sqrt{k})(V_{CURR} + V_{PREV})$

- V_z: valore di z — verifichiamo che J_z sia abbastanza importante
- $(1+\sqrt{k})$: ★
- V_{CURR}: valore di J_{CURR}
- V_{PREV}: Somma dei valori dei processi parzialmente eseguiti in precedenza (escluso J_{CURR} e J_z)

TEST = VERO ⇒ eseguire J_z invece che J_{CURR}

FALSO ⇒ Scartare (per sempre) J_z

★ $k = \frac{\bar{V}_{MAX}}{\bar{V}_{MIN}}$

$\bar{V}_{MAX} = MAX\left\{\frac{V_i}{C_i}\right\}$ dei processi attivi

$\bar{V}_{MIN} = MIN\left\{\frac{V_i}{C_i}\right\}$ non residuale

$\bar{V}$ — si dice densità di valore

da tabella

3

Mo Tu We Th Fr Sa Su

No. SISTEMI INF.

Date 31 · 10 · 19

Oss. Se il test è vero eseguiremo J_z anche nell'istante successivo, a meno che esista J_w in LST a $t+1$.

- Test leggermente diverso → $V_w > (1+\sqrt{k})\,V_z$
- quando J_z (o J_w) è terminato stiamo EDF

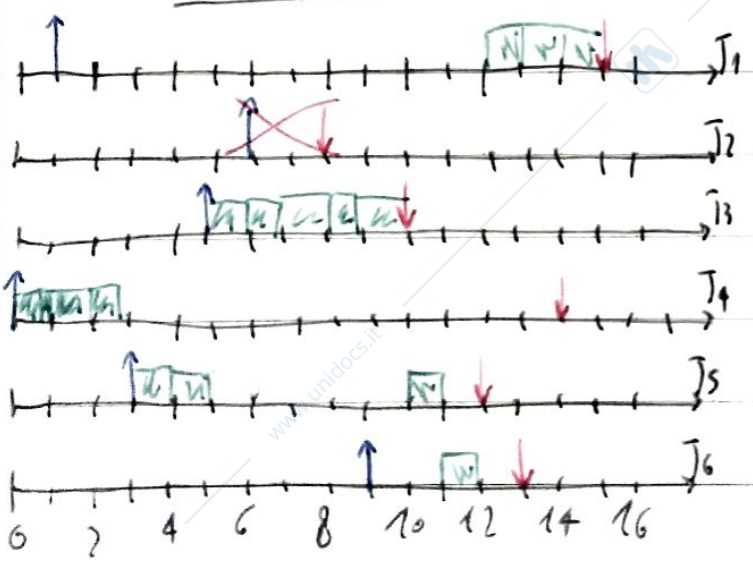
$J_{curr} = J_2$ $LST_3(6) = 6$ *

Es.

	T_1	T_2	T_3	T_4	T_5	T_6
a_i	1	6	5	0	3	9
C_i	3	1	5	3	3	1
d_i	15	8	10	14	12	13
V_i	1	1	17	1	1	3

non ci interessa che $LST_3(5) = 5$ poiché $J_{curr} = J_3$

LST



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	/	12	12	12	12	12	12	12	12	12	12	12	12	/	/	/
2	/	/	/	/	/	/	7	7								
3	/	/	/	/	/	5	6	7	8	9	/	/	/	/	/	
4	11	12	13	/	/	/	/	/	/	/	/	/	/	/	/	
5	/	/	/	9	10	11	11	11	11	11	11	/	/	/	/	
6	/	/	/	/	/	/	/	/	/	12	12	12	/	/	/	

$V_3 > (1+\sqrt{k})(V_2 + V_{PRUV})$?

T_1, T_2, T_3, T_5 attivi

↳ $k = \frac{\bar{V}_{MAX}}{\bar{V}_{MIN}} = \frac{MAX\{\frac{1}{3}, 1, \frac{17}{5}, \frac{1}{3}\}}{MIN\ \ //} = \frac{\frac{17}{5}}{\frac{1}{3}} = \frac{51}{5}$

↳ $V_{PRUV} = 1$

↳ C_5 residuo

$V_3 > \left(1 + \sqrt{\frac{51}{5}}\right)(1+1)$ ✓ → eseguiamo T_3

V_3: 17 — $\sqrt{\frac{51}{5}}$: 3.19

istante 7: $T_2 = T_3$ $T_W = T_2$ → $L_{ST_2}(7) = 7$

TEST: $V_2 > (1+\sqrt{k})V_3$?

$1 > (1+\sqrt{\frac{51}{5}})17$ → FALSO !!!!!

↓

Continuiamo a eseguire T_3

→ Rimuoviamo T_2

Non ha senso eseguito perché perderebbe comunque il suo valore (non finendo in tempo)