

Schema della lezione

Server periodici

Server procrastinabili

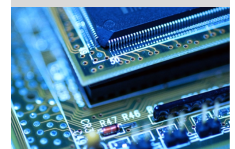
Server sporadici

CBS

Job aperiod. hard R.T.

SERT'20

R8.1



Schema della lezione

Server periodici

Server procrastinabili

Server sporadici

CBS

Job aperiod. hard R.T.

SERT'20

R8.2

# Lezione R8

## Algoritmi a conservazione di banda

Sistemi embedded e real-time

22 ottobre 2020

Marco Cesati

Dipartimento di Ingegneria Civile e Ingegneria Informatica  
Università degli Studi di Roma Tor Vergata

### Di cosa parliamo in questa lezione?

In questa lezione si discutono alcuni algoritmi a conservazione di banda utilizzati per integrare la gestione dei job aperiodici con gli schedulatori priority-driven

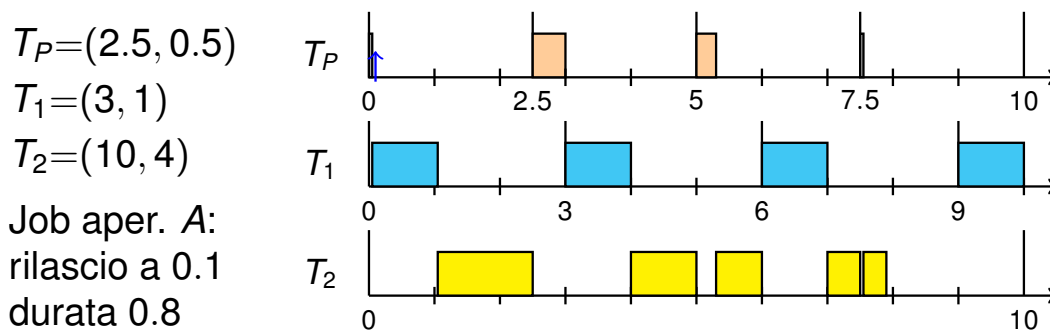
- 1 I server periodici
- 2 Il server procrastinabile
- 3 Il server sporadico
- 4 Il server CBS
- 5 Job aperiodici hard RT

## Schedulazione di job aperiodici con polling

L'algoritmo di *schedulazione con polling* è basato su un task periodico (*server di polling* o *poller*) con fase 0, periodo  $p_s$ , tempo d'esecuzione  $e_s$ , e priorità massima

Il *server di polling* controlla la *coda di job aperiodici*: se è vuota, si auto-sospende fino al prossimo periodo, altrimenti esegue il job in cima alla coda per max  $e_s$  unità di tempo

- Se i parametri del *poller* sono corretti, i job aperiodici non influiscono sulla schedulabilità dei task periodici
- Se il job aperiodico arriva subito dopo l'inizio del periodo del *poller*, non sarà eseguito fino al periodo successivo (i tempi di risposta non sono minimizzati)



la grande limitazione del poller è che vede solo ad inizio periodo, non nel mezzo.

## Server periodici

I *server periodici* sono una classe di task periodici aventi:

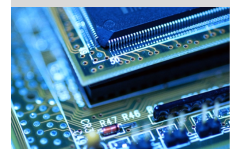
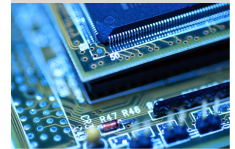
- Periodo  $p_s$ , **budget**  $e_s$ , e dimensione  $u_s = e_s/p_s$
- **Regola di consumo**: come il budget viene consumato
- **Regola di rifornimento**: come il budget viene ripristinato

Si dice che il server periodico è:

- **impegnato** quando ha lavoro da svolgere
- **idle** quando non ha lavoro da svolgere
- **eleggibile**, **pronto** o **schedulabile**: impegnato e con budget positivo

Esempio: il poller è assimilabile ad un server periodico

- impegnato quando la coda di job aperiodici è non vuota
- regola di consumo: sottrae il tempo impiegato ad eseguire un job aperiodico dal budget; **azzerà il budget se la coda è vuota**
- regola di rifornimento: il budget è impostato a  $e_s$  all'inizio di ogni periodo



## Algoritmi a conservazione di banda

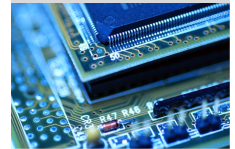
Il problema del server di polling è che il budget è perso non appena la coda di job aperiodici si svuota

Gli algoritmi basati su server periodici che non hanno questo problema sono definiti a *conservazione di banda*

Idea: preservare il budget quando il server periodico è idle per migliorare i tempi di risposta dei job aperiodici

Esistono molti tipi di algoritmi a *conservazione di banda*

- Server procrastinabile
- Server sporadico
- Server a utilizzazione costante
- Server a banda totale
- Algoritmo WFQ
- Algoritmo CBS



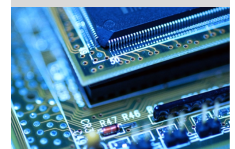
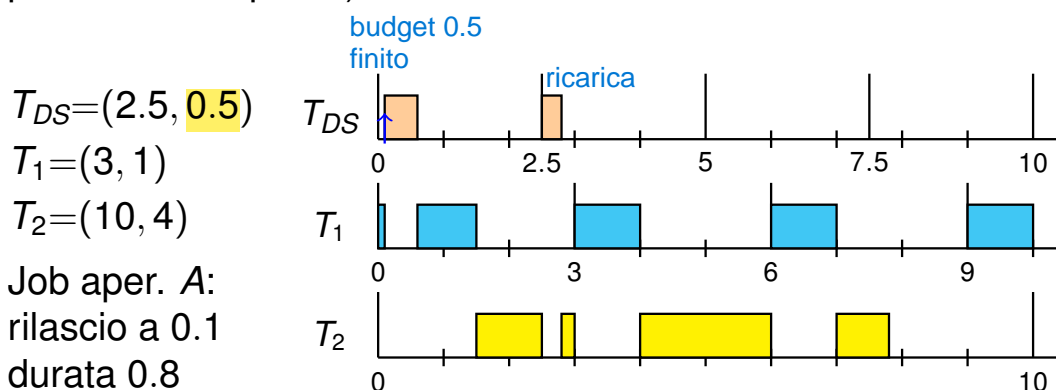
## Server procrastinabile

Il *server procrastinabile* (o *deferrable server*) è il più semplice algoritmo a conservazione di banda

È caratterizzato da un periodo  $p_s$ , da un budget massimo  $e_s$ , e dalle seguenti regole:

- **Regola di consumo:** il budget è decrementato di uno per ogni unità di tempo in cui il server è in esecuzione
- **Regola di rifornimento:** il budget è impostato al valore  $e_s$  agli istanti  $k \cdot p_s$ , per  $k = 0, 1, 2, \dots$

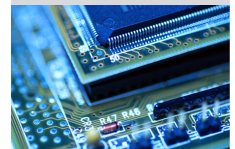
Nota: il budget non si accumula (quello non speso alla fine del periodo viene perso)



## Schedulazione a priorità fissa con server procrastinabile

Algoritmi  
a conservazione  
di banda

Marco Cesati



Schema della lezione

Server periodici

Server procrastinabili

Server sporadici

CBS

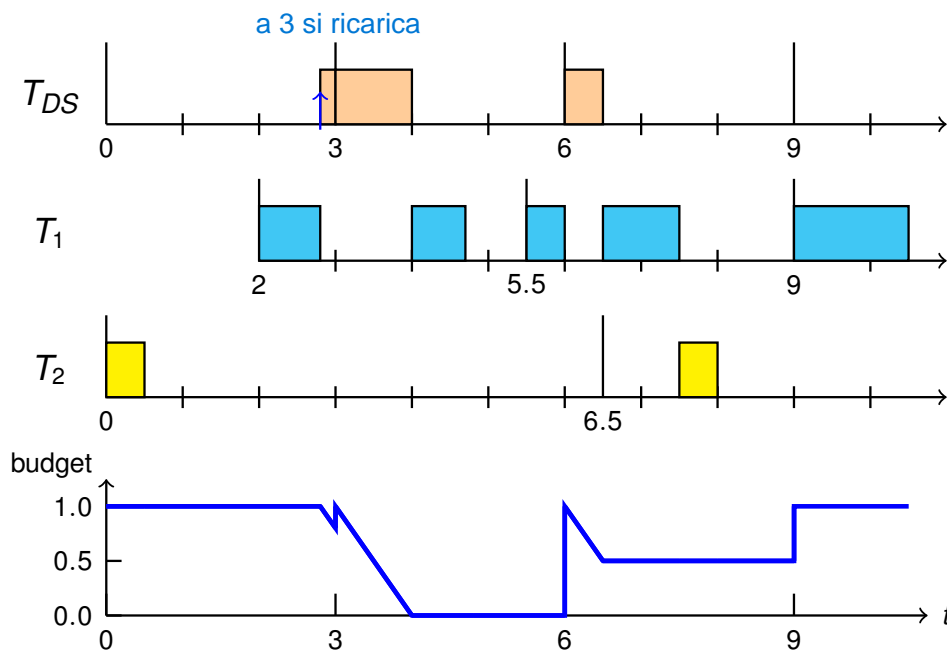
Job aperiod. hard R.T.

SERT'20

R8.7

Sistema:  $T_{DS}=(3, 1)$ ,  $T_1=(2.0, 3.5, 1.5, 3.5)$ ,  $T_2=(6.5, 0.5)$

Job aperiodico A: arrivo a 2.8, durata 1.7

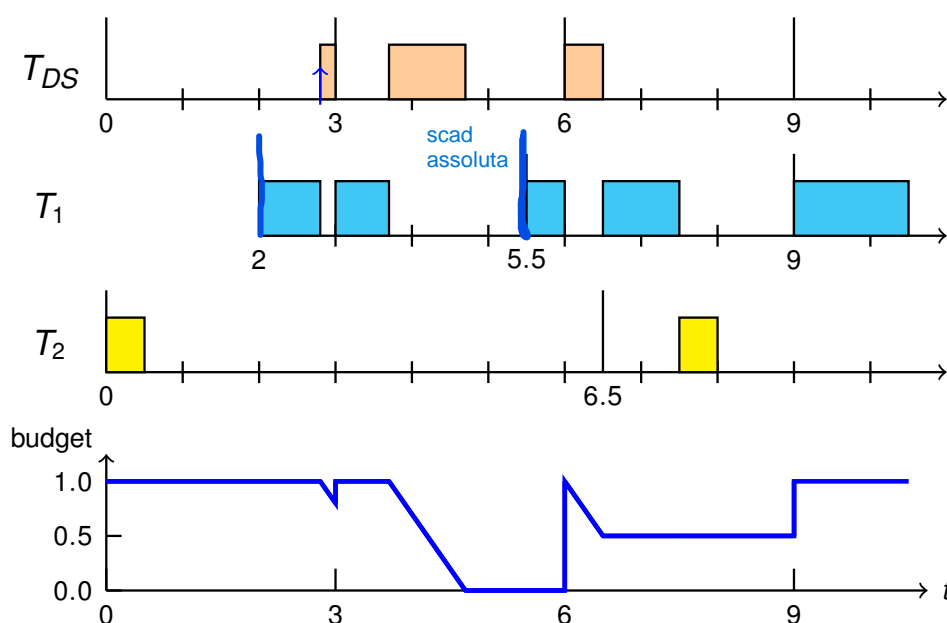


## Schedulazione EDF con server procrastinabile

fase | periodo | es | scad rel

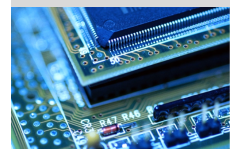
Sistema:  $T_{DS}=(3, 1)$ ,  $T_1=(2.0, 3.5, 1.5, 3.5)$ ,  $T_2=(6.5, 0.5)$

Job aperiodico A: arrivo a 2.8, durata 1.7



Algoritmi  
a conservazione  
di banda

Marco Cesati



Schema della lezione

Server periodici

Server procrastinabili

Server sporadici

CBS

Job aperiod. hard R.T.

SERT'20

R8.8

## Schedulabilità per priorità fissa con server procrastinabile

È possibile applicare il test o le condizioni di schedulabilità per sistemi a **priorità fissa** con server procrastinabile? **Sì, ma...**

Il server procrastinabile non è identico agli altri task periodici:

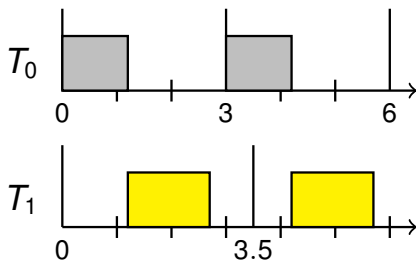
- Se il server è eleggibile e nessun task a priorità maggiore è in esecuzione, viene **subito attivato** dallo scheduler
- Un server con budget  $> 0$  può diventare eleggibile in qualunque istante (dipende dagli arrivi dei job aperiodici)

No poller

$$T_0 = (3, 1.2), T_1 = (3.5, 1.5)$$

$$w_1(t) = 1.5 + \lceil t/3 \rceil 1.2$$

$$w_1(1.5) = 2.7 = w_1(2.7) \leq 3.5$$

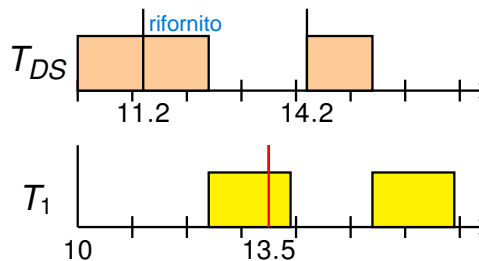


Sì poller

$$T_{DS} = (3, 1.2), T_1 = (3.5, 1.5)$$

$$r_{1,c} = 10, r_A = 10, e_A > 3$$

$$\text{budget}(10) = 1.2, \text{fase} = 2.2$$



T1 manca scadenza, il problema è che T(DS) può partire in ogni istante.

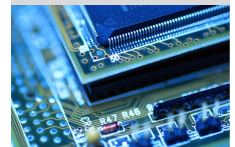
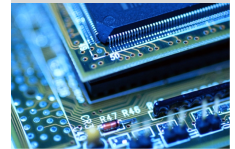
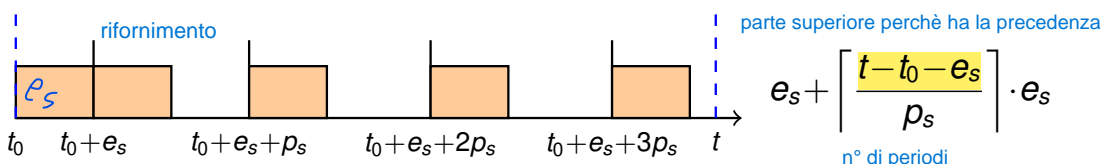
## Istanti critici per sistemi con server procrastinabile

### Lemma (Lehoczky, Sha, Strosnider, 1987, 2000)

In un sistema di task periodici indipendenti e interrompibili a priorità fissa con  $D_i \leq p_i$ , e con un server procrastinabile  $(p_s, e_s)$  con priorità massima, un **istante critico** di un task  $T_i$  si verifica all'istante  $t_0$  se

- a  $t_0$  è rilasciato un job di tutti i task  $T_1, \dots, T_i$
- a  $t_0$  il budget del server è  $e_s$
- a  $t_0$  è rilasciato almeno un job aperiodico che impegna il server da  $t_0$  in avanti
- l'inizio del successivo periodo del server è a  $t_0 + e_s$

Nelle ipotesi del lemma, quanto tempo di processore occupa al massimo il server procrastinabile nell'intervallo  $(t_0, t]$ ?



## Test di schedulabilità con server procrastinabile

Con priorità fissate ed un server procrastinabile di massima priorità, la **funzione di tempo richiesto** è:

quanto esegue il server

$$w_i(t) = e_i + b_i + e_s + \left\lceil \frac{t - e_s}{p_s} \right\rceil e_s + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k \quad \text{per } 0 < t \leq p_i$$

Il test controlla se  $w_i(t) \leq t$  per i valori di  $t \leq D_i$  tali che  $t = h \cdot p_k$ , oppure  $t = e_s + h \cdot p_s$ , oppure  $t = D_i$  ( $h = 0, 1, \dots$ )

Analogamente per il **test di schedulabilità generale**:

$$w_{i,j}(t) = j e_i + b_i + e_s + \left\lceil \frac{t - e_s}{p_s} \right\rceil e_s + \sum_{k=1}^{i-1} \left\lceil \frac{t}{p_k} \right\rceil e_k$$

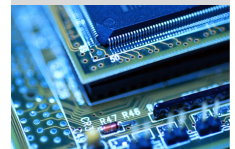
per  $(j-1)p_i < t \leq (j-1)p_i + D_i$

Esempio:  $T_{DS} = (3, 1.2)$ ,  $T_1 = (3.5, 1.5)$

$$w_1(t) = 2.7 + \lceil (t - 1.2)/3 \rceil 1.2$$

$$w_1(1.5) = 3.9 = w_1(3.9) > 3.5 \Rightarrow T_1 \text{ non schedulabile!}$$

Se il server non ha priorità massima, il test fornisce una condizione solo **sufficiente** (può dare falsi negativi)



## Condizione di schedulabilità **RM** con server procrastinabile

### Teorema (Lehoczky, Sha, Strosnider, 1987, 2000)

Un server procrastinabile  $(p_s, e_s)$  ed  $n$  task periodici indipendenti e interrompibili con  $p_i = D_i$  tali che

$$p_s < p_1 < \dots < p_n < 2p_s \quad \text{e} \quad p_n > p_s + e_s$$

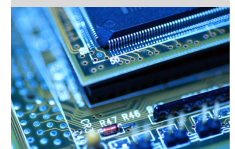
sono schedulabili con RM se l'utilizzazione totale dei task periodici e del server è minore o uguale a

$$U_{RM/DS}(n) = \frac{e_s}{p_s} + n \left[ \left( \frac{e_s + 2p_s}{p_s + 2e_s} \right)^{1/n} - 1 \right]$$

è molto simile a Liu Layland nella forma, infatti prima c'era il "2" sotto l'esponenziale.

- Se  $e_s = 0$ ,  $U_{RM/DS}(n) = U_{RM}(n)$  come se il server non esistesse, ritorno a Liu Layland

- $\lim_{n \rightarrow \infty} U_{RM/DS}(n) = \frac{e_s}{p_s} + \ln \left( \frac{e_s + 2p_s}{p_s + 2e_s} \right)$





## Condizione di schedulabilità RM con server procrastinabile (2)

*Se  $p_s, p_1, \dots, p_n$  non verificano le condizioni del teorema?*

Applichiamo la condizione di schedulabilità task per task:

- Il server non ha alcuna influenza sui task aventi periodo minore di  $p_s$  sono più importanti del server
- Il server è schedulabile se lo è il corrispondente task periodico
- Per ogni task  $T_i$  con  $p_i > p_s$ , il server si comporta come un task periodico, tranne che può eseguire per un tempo  $e_s$  in più (tempo di blocco aggiuntivo):

$$\sum_{k=1}^i \frac{e_k}{p_k} + \frac{e_s}{p_s} + \frac{e_s + b_i}{p_i} \leq U_{RM}(i+1)$$

task stesso    + server    + blocco aggiuntivo (potrebbe eseguire due volte di fila)

Esempio:  $T_{DS}=(3, 1.2)$ ,  $T_1=(3.5, 1.5)$

$$\frac{1.5}{3.5} + \frac{1.2}{3} + \frac{1.2}{3.5} > 1 > U_{RM}(2) \Rightarrow T_1 \text{ forse non schedulabile!}$$

è sufficiente ma non necessario.

in questo esempio cade la condizione " $P_n > E_s + P_s$ "

## Condizione di schedulabilità EDF con server procrastinabile

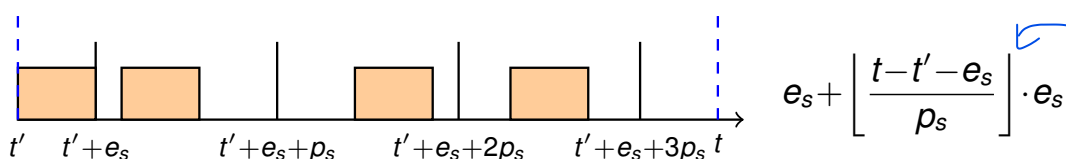
### Teorema (Ghazalie, Baker 1995)

Un task periodico  $T_i$  in un sistema di  $n$  task indipendenti e interrompibili è schedulabile con EDF insieme ad un server procrastinabile ( $p_s, e_s$ ) se

$$\sum_{k=1}^n \frac{e_k}{\min(D_k, p_k)} + \frac{e_s}{p_s} \left( 1 + \frac{p_s - e_s}{D_i} \right) \leq 1$$

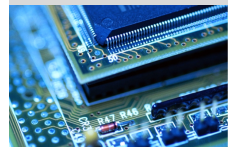
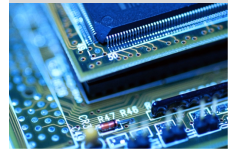
densità tot (<1)

**Dim.** (sketch per  $D_k \geq p_k$ ) Un job di  $T_i$  rilasciato a  $r_i$  manca la scadenza a  $t$ ;  $t' < t$  è l'ultimo istante in cui il processore è idle o esegue un job con scadenza  $> t \Rightarrow r_i \geq t' \Rightarrow 1/(t - t') \leq 1/D_i$



$$t - t' < \sum_{k=1}^n \frac{e_k}{p_k} (t - t') + \frac{e_s}{p_s} (t - t' + p_s - e_s)$$

$t - t'$  non mi è bastato per eseguire tutto il lavoro: eseguire TUTTI i job periodici + tempo preso dal server nell'intervallo tra parentesi.



tempo rubato dal server in questo lasso di tempo. "es" perchè a  $t'$  ha budget massimo, poi il resto è quante altre volte esegue. Inferiore perchè se " $t$ " cade in mezzo, scarto quello che viene dopo.

## Server sporadici

Un server procrastinabile può ritardare i task di priorità minore più di un task periodico con identici parametri

I *server sporadici* sono una classe di server periodici completamente assimilabili come schedulabilità a task periodici con medesimi parametri

Un sistema con task periodici e *server sporadici* può essere analizzato tramite le condizioni ed il test di schedulabilità *generale* dei sistemi per task periodici

Esistono diversi tipi di *server sporadici*: la differenza è tutta nelle due regole di consumo e di rifornimento del budget

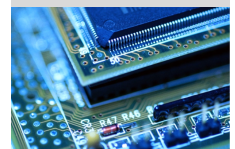
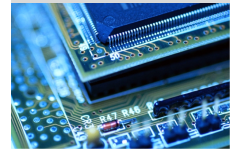
Regole più sofisticate:

- preservano il budget più a lungo o lo riforniscono più velocemente
- sono più difficili e costose da implementare

## Definizioni per server sporadici in sistemi a priorità fissa

- Sistema  $\mathcal{T}$  di task periodici a priorità fissa
- Server sporadico  $T_s = (p_s, e_s)$  con priorità  $\pi_s$
- $\mathcal{T}_H$ : insieme di task di  $\mathcal{T}$  con **priorità maggiore di  $\pi_s$**
- *Intervallo totalmente occupato* di un insieme di task:
  - (1) prima dell'intervallo tutti i job sono stati completati,
  - (2) all'inizio viene rilasciato almeno un job, e
  - (3) la fine dell'intervallo è il primo istante in cui tutti i job rilasciati entro l'intervallo sono completati
- $t_r$ : ultimo istante in cui è stato aumentato il budget
- $t_f$ : primo istante dopo  $t_r$  in cui il server è **in esecuzione**
- $t_e$ : istante che determina il momento del prossimo rifornimento (generalmente sarà a  $t_e + p_s$ )
- **BEGIN**: per ogni  $t$ , considerare l'ultima sequenza di intervalli totalmente occupati contigui dei task  $\mathcal{T}_H$  iniziata prima di  $t$ ; **BEGIN** è l'istante di inizio del primo intervallo totalmente occupato di questa sequenza
- **END**: l'istante finale della sequenza, se precedente a  $t$ , altrimenti  $\infty$

relativi  
a task  
di priorità  
superiore





## Server sporadico semplice

### CONSUMO:

Riduco il budget in modo proporzionale al tempo se:  
Dall'ultimo rifornimento ho già eseguito, oppure  
Sto eseguendo un task periodico meno prioritario di me  
( $END < t$ , ovvero è finita l'intervallo totalmente occupato di task più importanti)

### Regola di consumo

In ogni istante  $t > t_r$ , il budget è decrementato di uno per ogni unità di tempo se una delle due condizioni C1 e C2 è vera:

**C1** Il server è in esecuzione

**C2** Il server è stato in esecuzione dopo  $t_r$  e inoltre  $END < t$   
Altrimenti (se C1 e C2 sono false) il budget è conservato

### Regola di rifornimento

**R1** Ad ogni rifornimento:  $budget \leftarrow e_s$ ,  $t_r \leftarrow$  istante corrente

**R2** All'istante  $t_f$ : se  $END = t_f$ ,  $t_e \leftarrow \max(t_r, BEGIN)$ ; se  $END < t_f$ ,  $t_e \leftarrow t_f$

**R3** Il prossimo rifornimento sarà a  $t_e + p_s$ , con due eccezioni:

(a) se  $t_e + p_s < t_f$ , il budget sarà rifornito non appena esaurito

(b) il budget sarà rifornito a  $t_b < t_e + p_s$  se esiste un intervallo  $[t_i, t_b)$  in cui nessun task di  $\mathcal{T}$  è eseguibile, ed un task di  $\mathcal{T}$  comincia l'esecuzione a  $t_b$

R2: Sia  $END$  la fine sequenza di intervalli totalmente occupati di task più importanti.

1) Se " $t_f = END$ ", ovvero server esegue subito dopo i task più importanti, allora

$t_e = \max(\text{ultimo istante di aumento budget } "t_r", \text{ inizio sequenza iperperiodi } BEGIN)$

2) se " $t_f > END$ ", allora " $t_e = t_f$ " (non perdo tempo).

## Server sporadico semplice (2)

Significato di **C1**: nessun job del server esegue per un tempo maggiore di  $e_s$  in un periodo  $p_s$  non posso andare oltre il massimo

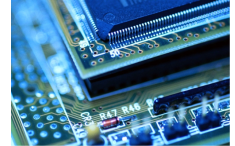
Significato di **C2**: il server conserva il budget se un task di  $\mathcal{T}_H$  è eseguibile oppure il server non ha mai eseguito da  $t_r$ ; altrimenti il budget è sempre consumato

Significato di **R2**:

- se nell'intervallo  $(t_r, t_f)$  sono stati sempre in esecuzione task di  $\mathcal{T}_H$ , il prossimo rifornimento sarà a  $t_r + p_s$
- altrimenti il prossimo rifornimento sarà a  $t_e + p_s$  ove  $t_e$  è l'ultimo istante di  $(t_r, t_f]$  in cui non esegue un task di  $\mathcal{T}_H$

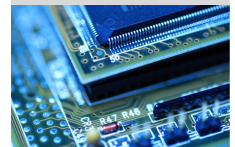
**Significato di R3a**: il job del server ha atteso per più di  $p_s$  unità di tempo prima di iniziare l'esecuzione, quindi il job continua nel prossimo periodo (è richiesto il test di schedulabilità generale)

Significato di **R3b**: il budget è rifornito nell'istante iniziale di ogni intervallo totalmente occupato di  $\mathcal{T}$



1) Se " $t_e + p_s < t_f$ ", ovvero ho fissato il prossimo rifornimento nel passato, allora rifornirò appena lo esaurirò.

2) Se esiste un intervallo in cui non esegue nessun task periodico, rifornisco alla fine di tale intervallo.

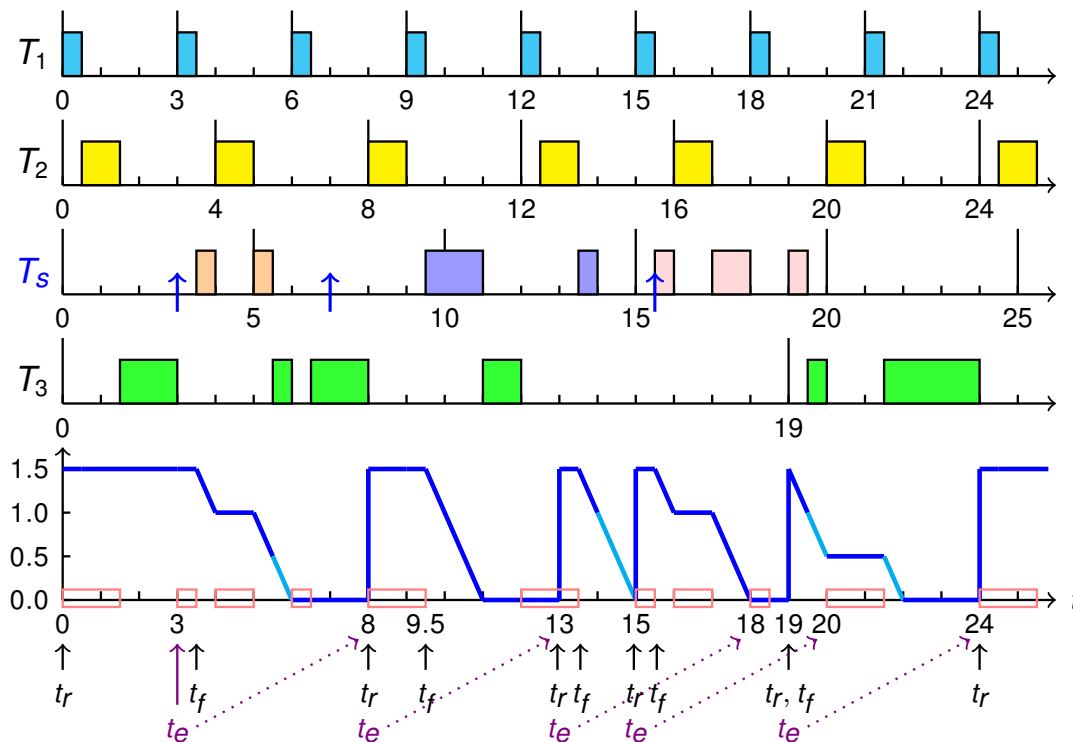


## Schedulazione RM con server sporadico semplice

$T_1$  e  $T_2$  prioritari rispetto  $T_s$

Sistema:  $T_1=(3, 0.5)$ ,  $T_2=(4, 1)$ ,  $T_s=(5, 1.5)$ ,  $T_3=(19, 4.5)$

Aperiodici:  $A_1(r=3, e=1)$ ,  $A_2(r=7, e=2)$ ,  $A_3(r=15.5, e=2)$



I rettangoli rossi sono molto utili, perchè mi rappresentano BEGIN e END per i task più importanti di  $T_s$ .

## Server sporadico/background

Esistono molte varianti di server sporadico, con regole sempre più complesse (e costose da implementare)

Variante più utile e diffusa: *server sporadico/background*

Differenza rispetto al server sporadico semplice: esegue sempre job aperiodici se nessun task periodico è eseguibile  
(nel caso sporadico, se budget è nullo, non potrei!)

### Regola di consumo

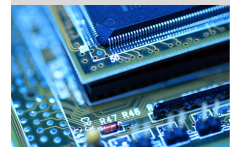
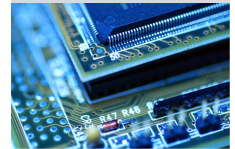
Identica a quella del server sporadico semplice, tranne che se nessun task periodico è eseguibile il budget è uguale a  $e_s$

Differenza è che, nell'intervallo idle: 1) rifornisce all'inizio dell'intervallo, 2) se arriva job per il server, esegue senza consumare budget.

### Regola di rifornimento

Identica a quella del server sporadico semplice, tranne **R3b**: il budget è ripristinato all'inizio di ogni intervallo in cui nessun task periodico è eseguibile;  $t_r$  (e ev.  $t_f$ ) è la fine dell'intervallo

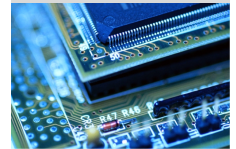
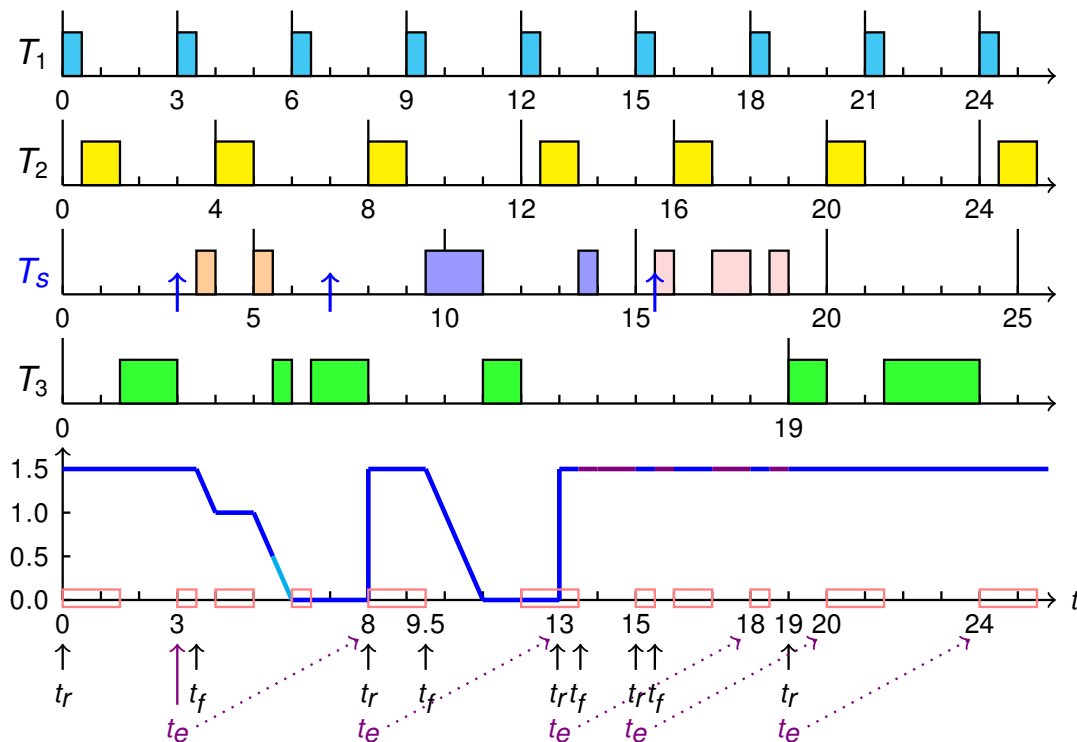
In effetti l'unico caso in cui *non* conviene usare un *server sporadico/background* al posto di uno semplice è quando si utilizzano più **server sporadici** per differenti tipi di job aperiodici



## Schedulazione RM con server sporadico/background

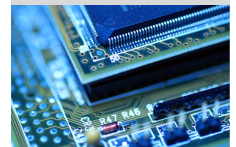
Sistema:  $T_1=(3, 0.5)$ ,  $T_2=(4, 1)$ ,  $T_s=(5, 1.5)$ ,  $T_3=(19, 4.5)$

Aperiodici:  $A_1(r=3, e=1)$ ,  $A_2(r=7, e=2)$ ,  $A_3(r=15.5, e=2)$



## Constant Bandwidth Server

- Inventato da L. Abeni e G. Buttazzo (1998)
- Server per job aperiodici integrabile in uno scheduler a priorità fissa a livello di job
  - Schedulazione di job aperiodici con i vantaggi di EDF rispetto a RM/DM
- Il server è “work conserving”
  - il processore non resta mai inutilizzato se almeno un job è eseguibile
- L'occupazione del processore del server non supera mai la frazione di tempo predefinita (*bandwidth costante*) "impatta" fino a un certo punto.
  - permette di isolare temporalmente i task periodici dal comportamento dei job aperiodici



## Il funzionamento del CBS

Un server CBS è caratterizzato da:

- il periodo  $p_s$ ,
- il budget massimo  $e_s$
- il budget corrente  $c_s$
- la scadenza assoluta corrente  $d_s$

Il rapporto  $u_s = \frac{e_s}{p_s}$  definisce la *bandwidth* del server

Il server CBS è schedulato con EDF insieme agli altri task periodici considerando la scadenza assoluta corrente  $d_s$

Un sistema di task periodici  $T$  ed un server CBS sono schedulabili con EDF se e solo se  $U_T + u_s \leq 1$

ciò perchè il server viene visto come task periodico

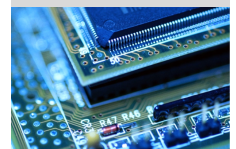
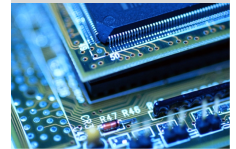
## Il funzionamento del CBS (2)

### Regola di aggiornamento della scadenza

- Inizialmente  $d_s = 0$  (il più importante)
- Non appena il budget corrente  $c_s$  si azzerà,  $d_s \leftarrow d_s + p_s$  (scadenza posticipata  $\Rightarrow$  priorità del server diminuita)
- Se ad un certo istante  $t$ :
  - viene rilasciato un job aperiodico
  - il server non è impegnato (la coda dei job aperiodici è vuota)
  - vale la condizione  $c_s \geq (d_s - t) \cdot u_s$allora  $d_s \leftarrow t + p_s$

### Regole di rifornimento e consumo del budget

- Inizialmente  $c_s \leftarrow e_s$
- $c_s$  viene decrementato proporzionalmente all'esecuzione dei job aperiodici del server
- Se  $c_s$  si azzerà,  $c_s \leftarrow e_s$  (il rifornimento è immediato)

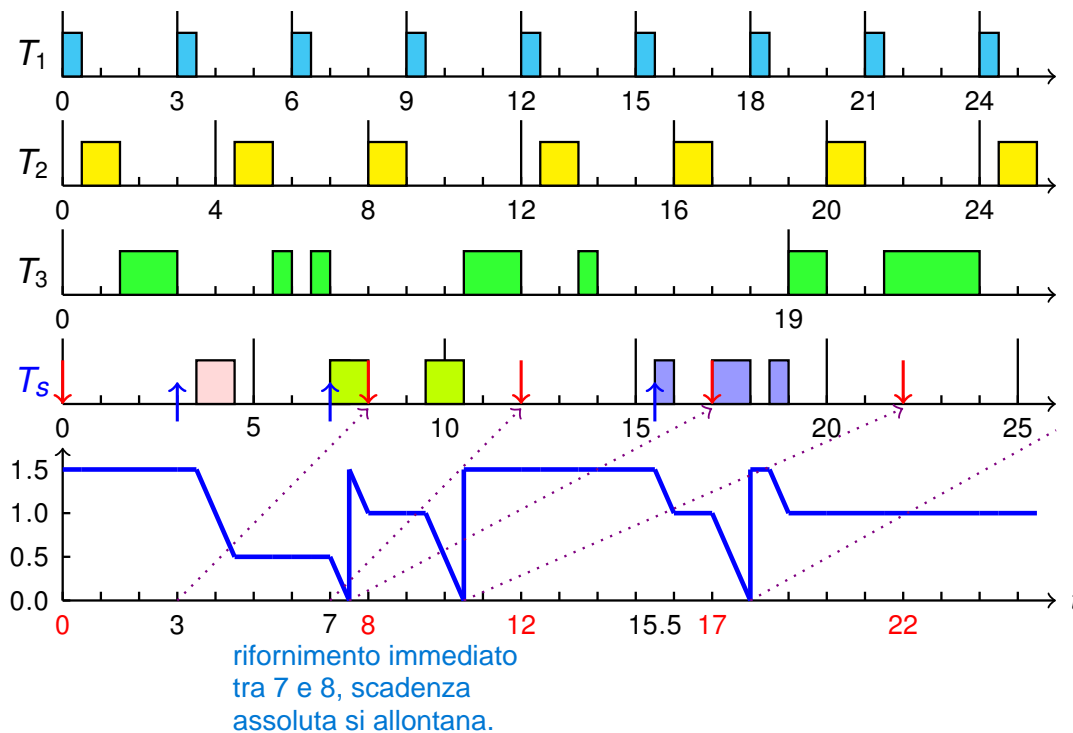


## Schedulazione EDF con server CBS

## Algoritmi a conservazione di banda

## Marco Cesati

Sistema:  $T_s=(5, 1.5)$ ,  $T_1=(3, 0.5)$ ,  $T_2=(4, 1)$ ,  $T_3=(19, 4.5)$   
 Aperiodici:  $A_1(r=3, e=1)$ ,  $A_2(r=7, e=2)$ ,  $A_3(r=15.5, e=2)$



## Schema della lezione

## Server periodici

## Server procrastinabili

## Server sporadici

CBS

Job aperiod. hard R.T.

SERT'20

R8.25

## Schedulabilità EDF di job aperiodici hard real-time

## Algoritmi a conservazione di banda

## Marco Cesati

La *densità di un job aperiodico* avente istante di rilascio  $r$ , massimo tempo di esecuzione  $e$  e scadenza  $d$  è  $\frac{e}{d-r}$

## Teorema

Un sistema di job aperiodici indipendenti e interrompibili è schedabile con EDF se la densità totale di tutti i job attivi (nell'intervallo tra rilascio e scadenza) è in ogni istante  $\leq 1$

**Dim.** (sketch) Un job manca la scadenza a  $t$ ; sia  $t' < t$  l'ultimo istante in cui il processore *non* ha eseguito un job con scadenza  $\leq t \Rightarrow \sum_i e_i > t - t'$  stesso ragionamento: non mi è bastato il tempo

L'intervallo  $(t', t]$  è partizionato in  $(t'=t_1, t_2]$ ,  $(t_2, t_3]$ ,  $\dots$   
 $(t_\ell, t_{\ell+1}=t]$  ove  $t_k$  è l'istante di rilascio o scadenza per qualche  
 job quindi nel singolo intervallo non arriva o scade nessuno.

Sia  $\mathcal{X}_k$  l'insieme di job attivi in  $(t_k, t_{k+1}]$  e sia  $\Delta_k$  la loro densità

$$\sum_i e_i = \sum_{j=1}^{\ell} (t_{j+1} - t_j) \sum_{J_k \in \mathcal{X}_j} \frac{e_k}{d_k - r_k} = \sum_{j=1}^{\ell} \Delta_j (t_{j+1} - t_j) \leq t - t'$$

## Schema della lezione

## Server periodici

## Server procrastinabili

Server sporadici

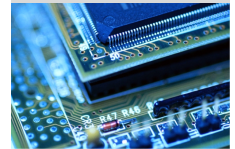
CBS

Job aperiod. hard R.T.

SERT'20

R8.26

## Schedulabilità EDF di job aperiodici hard real-time (2)



[Schema della lezione](#)

[Server periodici](#)

[Server procrastinabili](#)

[Server sporadici](#)

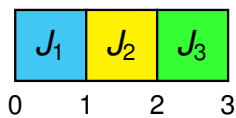
[CBS](#)

[Job aperiod. hard R.T.](#)

Consideriamo i job aperiodici  $J_1:(r=0, e=1, d=2)$ ,  
 $J_2:(r=0.5, e=1, d=2.5)$ , e  $J_3:(r=1, e=1, d=3)$

Intervalli:	$(0, 0.5]$	$(0.5, 1]$	$(1, 2]$	$(2, 2.5]$	$(2.5, 3]$
Job attivi:	$J_1$	$J_1 J_2$	$J_1 J_2 J_3$	$J_2 J_3$	$J_3$
Densità:	0.5	1.0	1.5	1.0	0.5

*Sono schedulabili con EDF? **Sì!***



La condizione del teorema è solo sufficiente!

stesso discorso: se fosse  $\leq 1$  ho la certezza,  
altrimenti dovrei analizzarlo nel dettaglio.