

## DEFINIZIONE DI TASH

UN TASH È UNA SEQUENZA DI ISTRUZIONI CONTINUAMENTE ESEGUITE DAL PROCESSORE FINO AL LORO COMPLETAMENTO

UN PROCESSO È UN PROGRAMMA IN ESECUZIONE, COMPOSTO DA TASH CONCERNENTI (TORN) CHE CONDIVIDONO UNA RISORSA CONCOND

UN TASH HA LE SEGUENTI CARATTERISTICHE:

- ACTIVATION TIME Ti:

INDICA IL TEMPO AL quale IL TASH DIVENTA PRONTO PER L'ESECUZIONE

- START TIME Si:

INDICA IL TEMPO AL quale IL TASH INIZIA LA SUA VERA ESECUZIONE

- FINISHING TIME Sf:

INDICA IL TEMPO AL quale IL TASH TERMINA LA SUA ESECUZIONE

## COMPUTATION TIME $C_i$

WICHS IL TERB DI ESGEZUGEN DCL TASH SIEBEN NECESSARY  
INTEGRATIONE (PREDZURE)

## COMPLETION TIME $K_i$

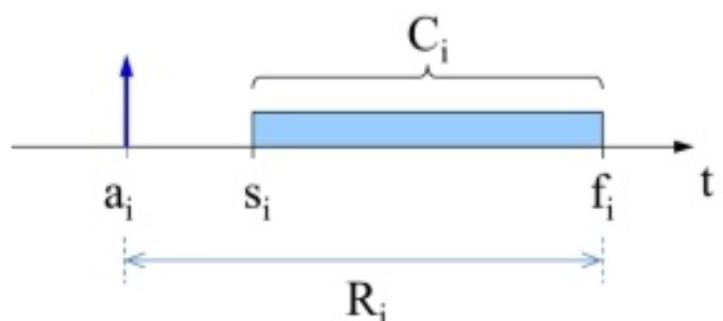
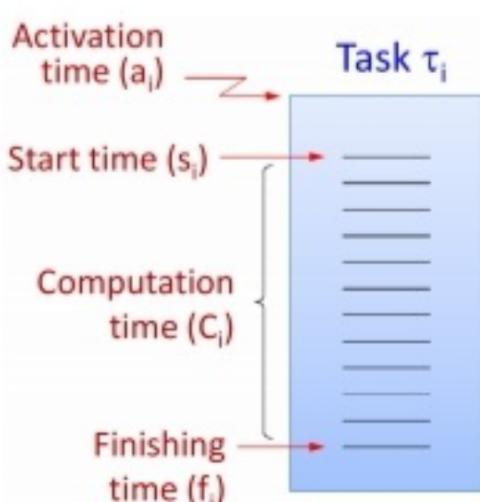
DEFINITS COMT

$$K_i = f_i - s_i$$

## RESPONSE TIME $R_i$

DEFINITS COMT

$$R_i = f_i - a_i$$



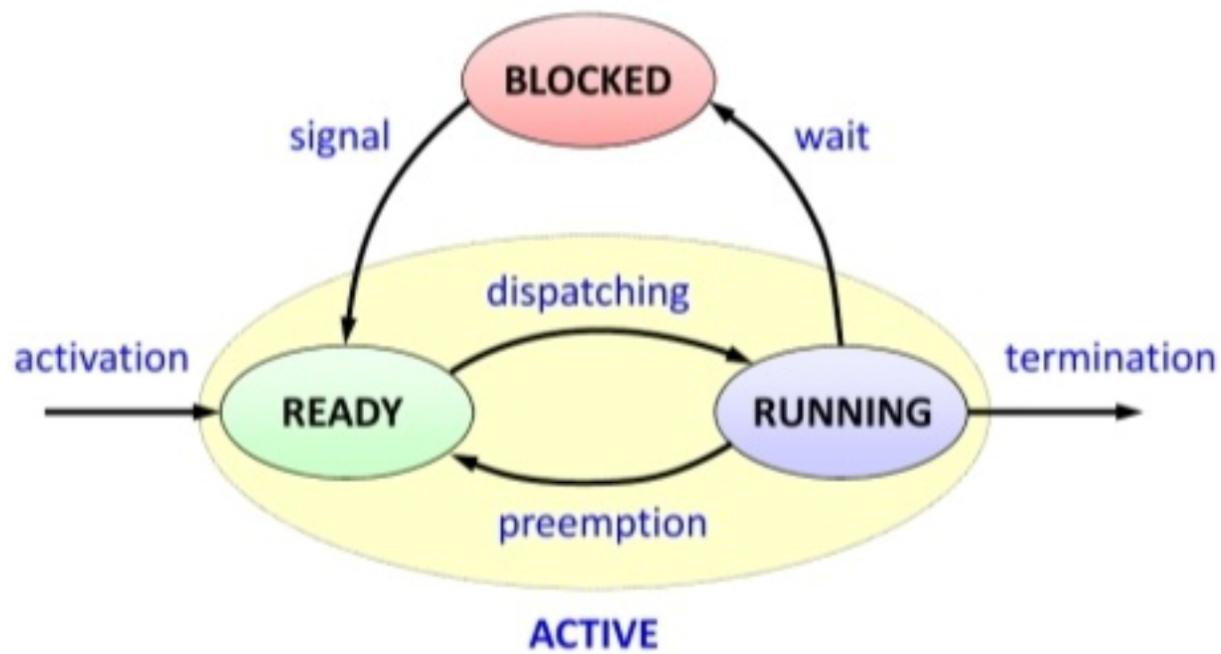
W TASH WEG DEFINITO ATTIVO ST PLO' GSEZUE  
POTENZIALENTG GSEZURO DIC PROSSEURKE.

W TASH ATTIVO G' DEFINITO READY ST G' IN

# NTOSV1 DEL PROFESSORE

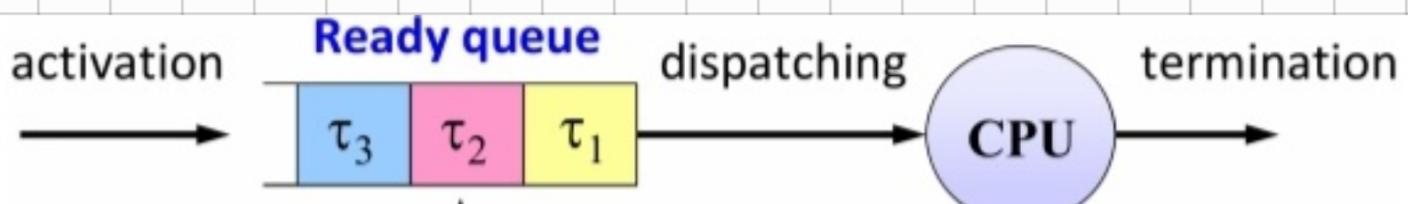
UN TASK VENDE DEFINISCE **RUNNING** SE È  
IN ESECUZIONE

UN TASK VENDE DENSI È **BLOCKED** SE STA  
ATTENDENDO DI ACCEDERE A UN'ALTRA RISORSA



UN TASK **READY** SONO GESTITI NELLA **READY**  
**QUEUE** CHE VENECE OBBLIGATORIAMENTE  
DETERMINARE I **POLITICA DI SCHEDULING**

QUINDI L'OPERAZIONE DI DISPATCHING ASSUME  
IL PROFESSORE AL PRIMO TASK CHE È PRESENTE  
NELLA CUE



TRAVERE IL MECCANISMO DI PREDATORI SI SUSPENDE  
IL RUNNINg TASK CORRENTE A FINEZZA DI UN TASK  
A PROVVISORIO.

QUESTO GARantisce la concorrenza tra i  
TASK nel suo stesso tempo RIDUCE IL TEMPO  
DI RISPOSTA DEI TASK AL PREVISTO INTRODUCE  
DO UN RUNTIME OVERHEAD SUL TEMPO DI  
ESECUZIONE DEI TASK.

## SCHEDULE

UN SCHEDULE È UN ASSEGNAZIONE DI UN TASK  
AI PROCESSORI, E DETERMINA QUANDO CI SPEGNERÀ DÌ  
ESECUZIONE DEI TASK CONSIDERATI

FUNZIONE, SI DEFINISCE UN SCHEDULE PER UN  
 CERS INSIEME DI TASK

$$\Gamma = \{ \gamma_1, \gamma_2, \dots, \gamma_m \}$$

COME UNA FUNZIONE:

$$\mathcal{S}: \mathbb{N}^+ \rightarrow N \text{ con } \mathbb{R}^+ \ni t \mapsto \mathbb{N}^+ \text{ e:}$$

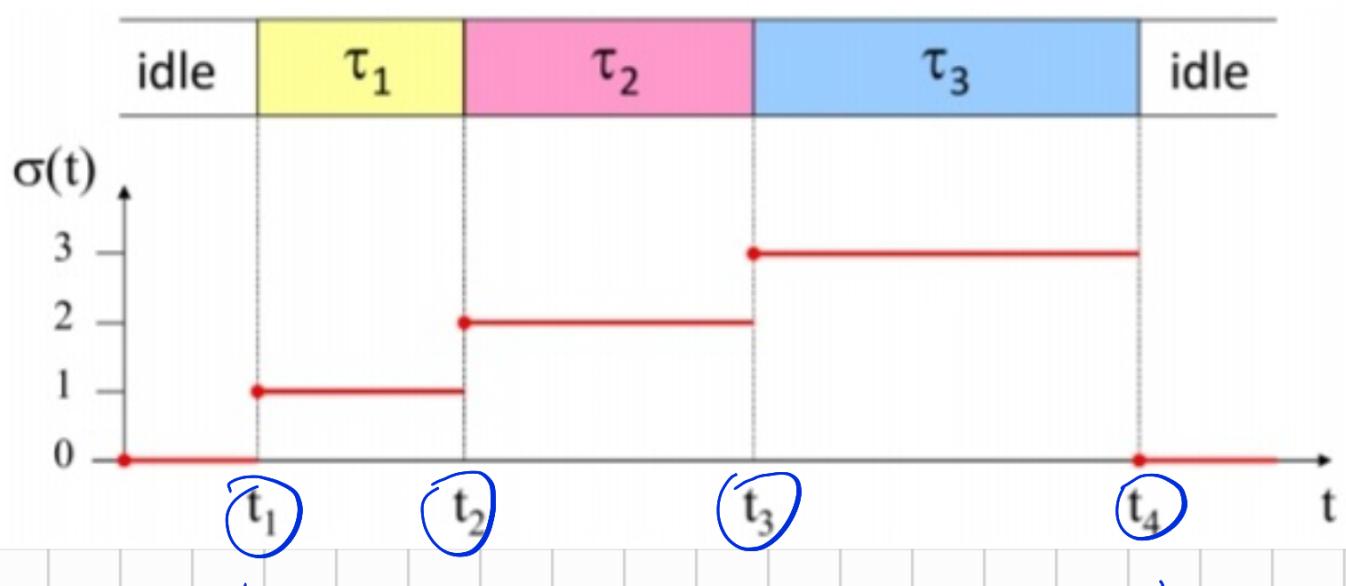
$t_o[t_1, t_2] \in \sigma(t), \sigma(t')$   $\forall t_o[t_1, t_2]$

Dove:

- $\sigma(t) = 0$  St il Processore  $\sigma$  in ISCB al tempo  $t$
- $\sigma(t) = i \in \{1, 2, \dots, n\}$  St il Processore  $i$  esegue il Task  $T_i$  al tempo  $t$

Ciascun intervallo temporale viene definito Time Slice function

Possiamo considerare un'istante di tempo  $t_i$  il Processore esegue un Context switch



qui si fa il context switch!

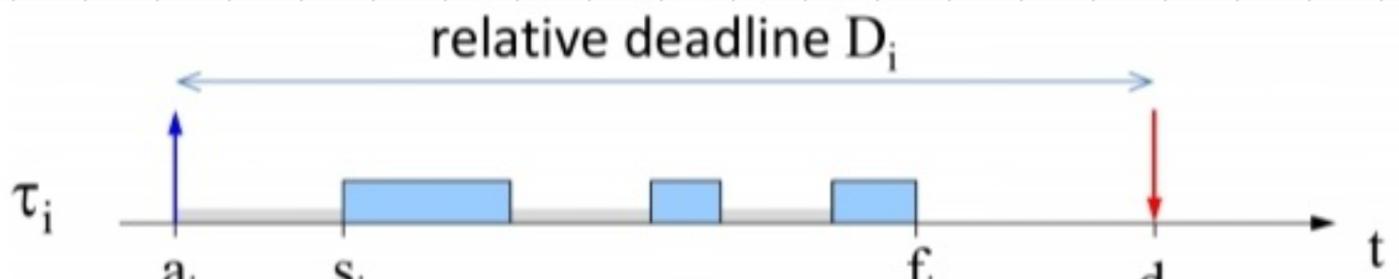
nel caso in cui si esegua parallelismo si può avere  
che le task multipli possono essere  
contemporaneamente attive (running/ready), ma  
solo in task solo running.



## TASK REAL-TIME

le task real-time sono caratterizzate da un  
vincolo temporale sul loro tempo di risposta, questo  
viene definito:

- DEADLINE ASSOLUTA  $d_i$ : viene riferito come il  
tempo entro il quale la task deve essere  
eseguita
- DEADLINE RELATIVA  $D_i$ : definita come  $D_i = d_i - \alpha_i$





UN TASSI REAL-TIME  $i$  È DEFINITO **FEASIBLE** SE  
QUESTO A COMPLETARE LA SUA ESECUZIONE ENTRO IL  
DEADLINE ASSOLUTA, OVVERO

$$f_i \leq d_i \text{ oppure } R_i \leq D_i$$

SI DEFINISCE **LATENCY**  $X_i$  DI UN TASSI  $i$  IL **MINIMO**  
DELAY POSSIBILE CHE  $i$  SUBISCE DOPO IL SUO ARRIVO  
NO G CONSUMA RISORSE A COMPLETARE IL TASSI  
ENTRO IL DEADLINE

VICENE MISURARE AL TEMPO DI ATTIVAZIONE E SI RICHIESTA?

$$X_i = D_i - C_i$$

$\downarrow$        $\downarrow$

deadlines  
relax

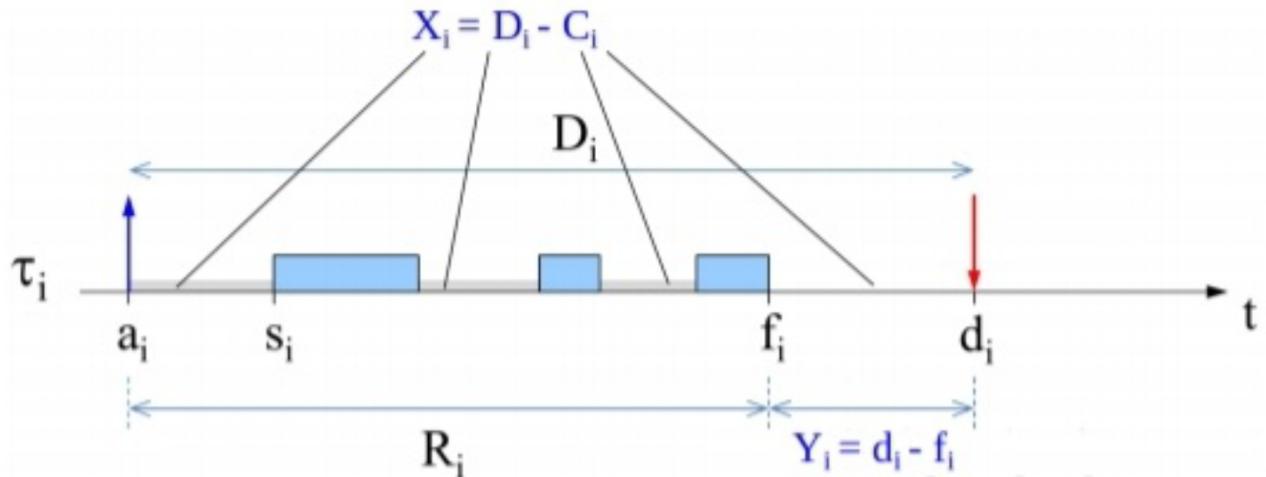
computation  
time

SI DEFINISCE INOLTRE LA **RESIDUAL LATENCY**  $X_i$  DI UN  
TASSI  $i$  LA LATENCY MISURATA DOPO IL COMPLETAMENTO  
DEL TASSI

$$Y_i = d_i - f_i$$

DESONTE ASSOLUTA

FINISHING TIME



SI DEFINISCONO ALTRE DUE PARAMETRI PER I TASSI  
NON-TIME!

### LATENESS $L_i$

INDICA IL DEGLI DI COMPLETAMENTO DEL TASSO  $i$   
RISPETTO ALLA SUA DESONTE ASSOLUTA

$$L_i = f_i - d_i$$

È UNA QUANTITÀ NEGATIVA SE IL TASSO TERMINA PRIMA  
DELLA DESONTE.

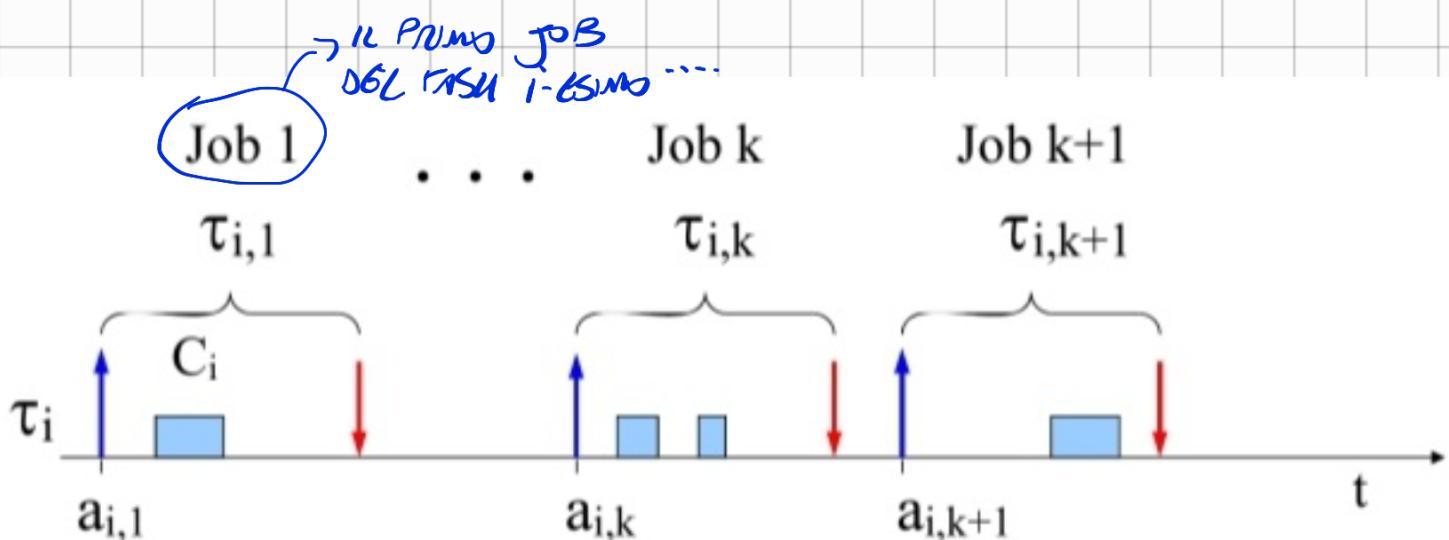
## TARDINESS $E_i$ :

GRIL TEMPO PER IL QUALE IL TASK  $i$  RIMANE ATTIVO DOPPIO SUO DECORRERE:

$$E_i = \max \{ 0; L_i \}$$

SE NON SI HA RISULTATO  
ALLORA  $L_i > 0$   
 $E_i = 0$

INIZIO, CON SEGUENTI CHE VENGONO ESEGUITI PIAVIOLTE  
SU DIFFERENTI INPUT VS UN CERTO NUMERO DI SEQUENZE  
DI ARRIVATI IDENTICI DEFINITI JOBS



## TASK PERIODIC

1 TASK PERIODICO SONO ATTIVATI AUTOMATICAMENTE

2 IL SISTEMA OPERATORICO ASSEGNALE PROBLEMI

1 TASK PERIODICO SONO INVECE ATTIVATI NEL MODO

2 CON CERTI EVENTI

UN TASK PERIODICO È COMPOSTO DA UNA  
SERIE INFINITA DI JOBS CHE VENGONO READERMENTE  
ATTIVATI AD UN RATE COSTANTE CON PERIODO  $T_i$ .

SE  $T_i = D_i$  ALLORA IL TASK VENDE DEFINITO PERMANENTE  
PERIODICO

INOLTRE SI DEFINISCE IL FACTOR DI UTILIZZO DEL  
TASK COME AI PUNTI:

$$U_i = \frac{C_i}{T_i}$$

quanto lavoro  
in CPU nel  
periodo del task

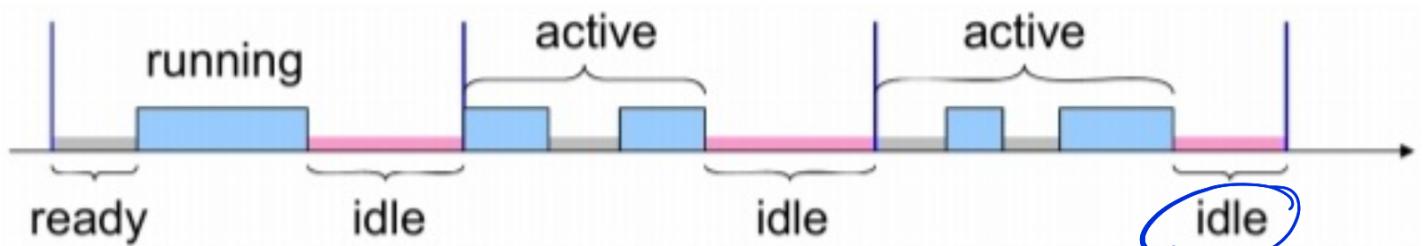
UNA TIPICA IMPLEMENTAZIONE DI UN TASK PERIODICO:

```
wait_for_activation();  
while(condition) {  
    ...  
    wait_for_next_period();  
}
```

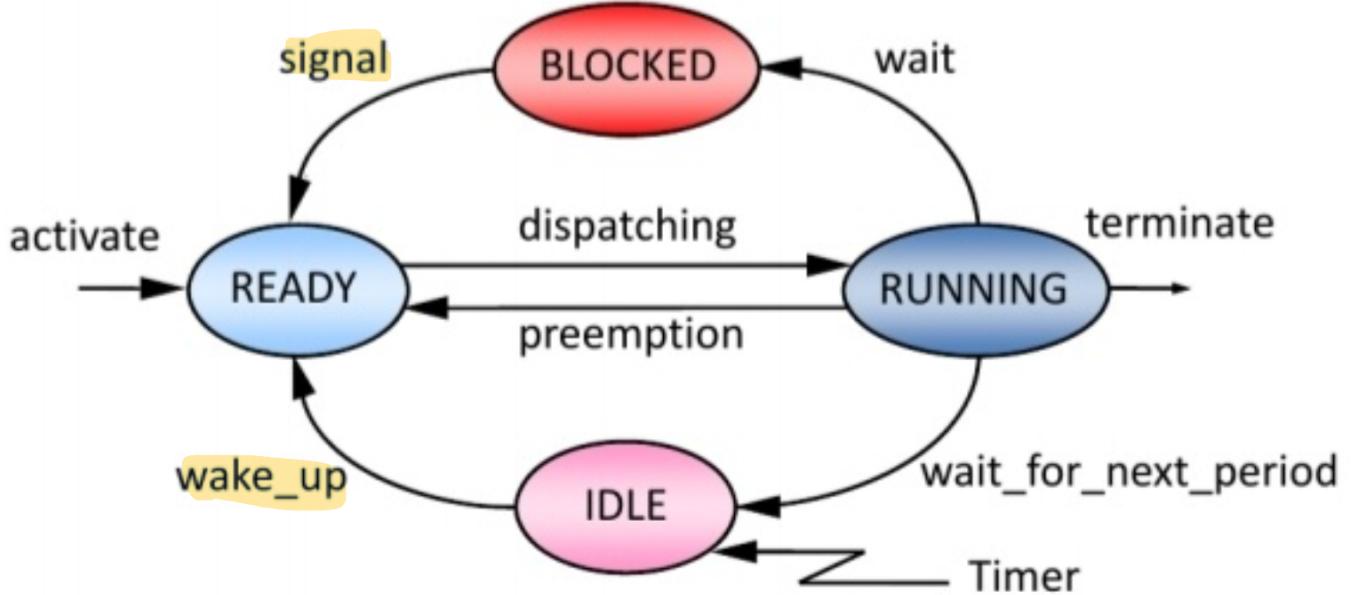
L'UTENTE INVIA ESPORTAMENTE  
UNA CHIUSURA PER INFORMARE L'ESECUZIONE  
DEL TASK DI UNA "TERMINAZIONE"

DOC TASK -> IL TASK SI AUTOSCROLL

È POSSO VEND INVOCARE UN CAVAMENTO  
WNT\_FCNEXT\_PERIOD() FINO ALL'INIZIO DEL PROSSIMO  
PERIODO SI HA ANCORA FONTE G IN IDLE!



IDLE È UNO BLOCKED!  
POSSO POGLIARE COME UNA  
OPERAZIONE



TASK APPROVATO

UN TASK APERIODICO È FORMATO DA UNA SEQUENZA  
INFINITA DI JOBS CHE NON SONO ATTIVATI IN MODO  
REGOLARE

UN TASK SPORADICO È UN TASK APERIODICO DOVE  
I JOBS CONSECUTIVI SONO SEPARATI DA UN TEMPO DI  
INTERARRODO MINIMO  $T_i$ , avendo:

$$\alpha_{i+1} \geq \alpha_i + T_i$$

SÌ DEFINISCE JITTER LA MISURA DI VARIAZIONE DI UN  
EVENTO PERIODICO. SÌ DEFINISCONO:

- ABSOLUTE JITTER:

$$\max_u \{ t_u - \alpha_u \} - \min_u \{ t_u - \alpha_u \}$$

→ DISTANZA TRA IL TEMPO DI ARRIVO  
ED IL TEMPO DI ATTIVAMENTO

- RELATIVE JITTER:

→ DISTANZA TRA COPPIE  
CONSECUTIVE DI JOBS

$$\max_u \left\{ |(t_u - \alpha_u) - (t_{u-1} - \alpha_{u-1})| \right\}$$

Si però invece definiscono JITTER anche per le SCHEDULING TIME, FINISHING TIME, COMPLETION TIME si ha  
 → sostituendo  $t$  con i parametri corrispondenti

RASSUMENDO SI DISTINGUE I VARI PARAMETRI IN DUE  
 macrogruppi:

- PARAMETRI OFFLINE:

Sono SPECIFICI, direttamente dal programma, come ad esempio il PERIODO, il COMPUTATION TIME e la RECEIVING DEADLINE

- PARAMETRI ONLINE

Sono tutti i parametri che dipendono sia dalla SCHEDULING TIME delle effettive esecuzioni del TASK, come ad esempio la CAPACITY, la URGENCY/TIGHTNESS, il JITTER...

FEASIBILITY VS SCHEDULABILITY



SCHEDULING FEASIBLE  
 ⇔  
 TASK SET  
 SCHEDULABLE

CHE SCHEDULE E' DEFINITO **FEASIBLE** SE TUTTI I VINCI SVI  
TUTTI SONO SODDISFAZIONI. IN PAROLE POCHE SONO 3 IPOTESI  
DI VINCI:

- **VINCI DI TEMPO**: (PERIODI, ATTIVAZIONI, DECADIMENTI...)

POTREBBERO ESSERE **ESATTI** SE VENGONO DIRETTAMENTE DEFINITI E  
INCORRISCONO NELLA SPECIFICA DI SYSTEM. ALTRIMENTI SONO DEI **INPUT**

- **VINCI DI PRECEDENZA**:

IMPOSTARE IN FORMA DI LISTA DELL'ESECUZIONE DEI TASSI E SONO OSPERABILI  
GRADUATORI DEL GRADITO RICAVATO DURANTE (DAG)

- **VINCI DI ACCESSO ALLE RISORSE**.

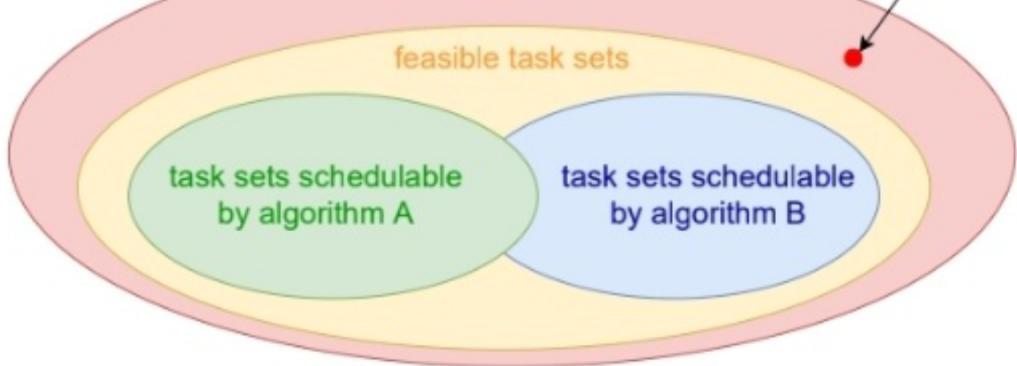
PERMETTE L'IMPLEMENTAZIONE DELLA SINCRONIZZAZIONE NEGLI  
ACCESSI MUTUALMENTE ESCLUSIVI ALLE RISORSE ESCLUSIVE, IN  
MODO DA INSUREARE EVENTUALI CONFLITTI.

PER UN TASSI SET  $\Gamma$  E' DEFINITO **FEASIBLE** SE:

- ESISTE UN ALGORITMO CHE OTTIENE CHE SCHEDULE FEASIBLE  
PER IL TASSI SET  $\Gamma$

MENTRE UN TASSI SET  $\Gamma$  SI DEFINISCE **SCHEDULABLE** DO C'È UN ALGORITMO SE:

- L'ALGORITMO OTTIENE CHE SCHEDULE FEASIBLE PER  $\Gamma$



quindi, purwit le stocanti quanti:

. TASK SET  $\Gamma = \{T_1, \dots, T_m\}$

. INSIEME DI PROCESSORI  $P = \{P_1, \dots, P_m\}$

. INSIEME DI RESORSE  $R = \{R_1, \dots, R_s\}$

. INSIEME DI VINCOLI  $C = \{H_1, \dots, H_u\}$

Si definisce un **Problema di Schedulazione** formato da  
ASSEGNAZIONE DI  $P$  ED  $R$  A  $\Gamma$  che produce uno  
**SCHEDULE FEASIBLE**

questo problema è **NP-COMPLETO**, avendo lao **GRANDE** con  
moltitudine di diversi deterministi casi di risoluzione in  
tempo **POSSIBILE**, ma soluzioni in tempo **ESONERABILE**!

### TASSONOMIA DEI ALGORITMI DI SCHEDULAZIONE

. PREEMPTIVE VS NON PREEMPTIVE

nel caso di preemzione si può interrompere un task a bassa priorità per eseguire un task a più alta priorità

## STATIC vs DYNAMIC

UN ALGORITMO STATICO PRENDE DECISIONI BASATE SU PARAMETRI FISSI CHE VENNO ASSIGNATI AI TUSI PRIMA DELLA LORO ATTIVAZIONE

## OFFLINE vs ONLINE

NEL CASO DI ALGORITMI OFFLINE LE DECISIONI SONO PRESE PRIMA DELL'ATTIVAZIONE DEL TUSI, MENTRE NEL CASO ONLINE LE DECISIONI VENNO PRESE A RUOTE BASEANDO SUL TUSI FIN'ADesso

## OPTIMAL vs HEURISTIC

UN ALGORITMO OTTIMO OTTIENE UNA SCHEDA CHE MINIMIZZA UNA CERTA FUNZIONE DI COSTO DEFINITA DA UNICO CRITERIO DI OTTIMALITÀ, NEL CASO GENERICO SI OBTIENE UNA SCHEDA SUSSA BASE DEI "SUGGERIMENTI" FORNITI DALLA FUNZIONE GENERICA

## GUARANTEED-BASED vs BEST-EFFORT

I PRIMI OBTENGONNO UNA SCHEDA FEASIBILE SE POSSIBILE. I BEST-EFFORT NON HANNO GARANZIA DI OBTENERE UNA SCHEDA FEASIBILE E SONO PIÙ USATI PER TUSI SOFT DEAD-TIME

## TEOREMA DI GRAMM

IL TEOREMA AFFERMA CHE SE UN TUSI-SRT È SCHEDATO IN MODO OTTIMALE SU UN MULTIPROCESSORE CON CERTI

PRESENTI IN POSTI, NUOVA MODALITÀ I PRENDERE IN MODO  
PERÒ DA RISPARMIARE CERTI VINCULI PIÙ VANTAGGIOSI CI  
L'INCERTITÀ DELLO SCHEDELLA  $\Rightarrow$  PERDERSI LE PRESCRIZIONI!

## SCHEDELLA DI TASSI PERIODICO

Dato un insieme di n tassi periodici  $T = \{T_1, \dots, T_n\}$  si ha  
che ciascun tasso  $T_i$  è caratterizzato da:

- TEMPO DI NUOVO INIZIALE  $\phi_i := j_{i,1}$

- WCET  $C_i$

- PERIODO DI ATTIVAZIONE  $T_i$

- DEADLINE RELATIVA  $D_i \leq T_i$

L'OBBIETTIVO È quindi quello di garantire che ciascun job  $T_{i,n}$  di ciascun tasso periodo  $T_i$

- SIA ATTIVATO AL TEMPO  $\alpha_{i,u} := \phi_i + (u-1)T_i$  avrà al  
MULTIPLO DEL PERIODO PARTENDO DALLA FASE INIZIALE

- COMPLETI LA SUA ESECUZIONE ENTRO IL DEADLINE  $d_{i,u} := \alpha_{i,u} + D_i$

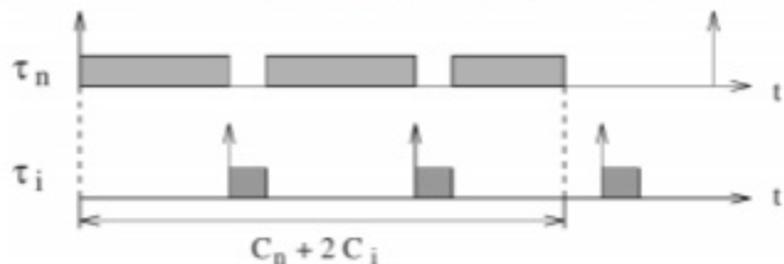
ALTRI PARAMETRI DI INTERESSE PER I TASSI PERIODICI SONO:

- PERIODO:  $H := \text{lcm} \{ T_1, \dots, T_n \}$  (il minimo intero di tempo dopo il quale lo schedula si ripete nello stesso ordine)
- JOB RESPONSE TIME:  $R_{i,u} := f_{i,u} - \alpha_{i,u}$
- TASK RESPONSE TIME:  $R_i := \max_u \{ R_{i,u} \}$  (il tempo di risposta massimo tra tutti i job del task i-simo)

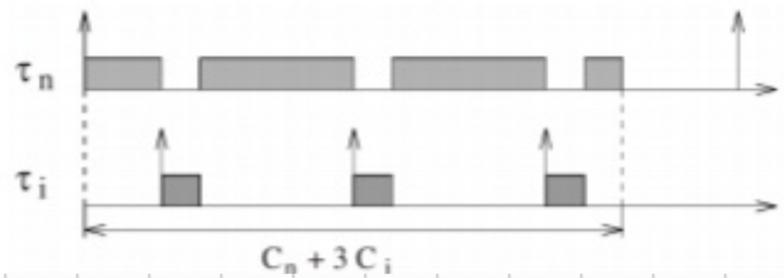
DEFINISMO INOLTRE L'ISTANTE CRITICO DI UN TASK COME IL TEMPO DI ARRIVO CHE PRODUCE IL TASK RESPONSE TIME MASSIMALE

SI VENNE quindi il TASK ARRIVES INSIDE ITS OWN TASK CYCLE POSSIEDONO UNA PRIORITY PONTE.

- Consider the interference of a high priority task  $\tau_i$  with a low priority task  $\tau_n$



- Reducing the phase of  $\tau_i$  increases the response time of  $\tau_n$



PIÙ REDDENDO LA FASE DI PRIMAISI TASK AD ALTA PRIORITY VIENNE RIDUCITO IL TEMPO DI RISPOSTA DI  $\tau_n$

## EXECUTIVE SCHEDULING

SUSSEGUE IL TEMPO IN INTERVALLI (TIME SLOTS) DI DURATA

OVERLAP  $\Delta$  (major cycle). A questo punto i RASH VENIVANO ALLOCATI, STARTAMENTE NEL TIME SLOTS IN modo che LA SOMMA DEI WCET DEI RASH IN OGNI TIME SLOT SEA MASSIMA

L'esecuzione in ciascun time slot è INTUITA da un semplice timer e lo SCHEDULE viene quindi ripetuto dopo un intervallo  $T$  definito major cycle

$$\Delta = \text{mcd} \left\{ T_1, \dots, T_m \right\}$$

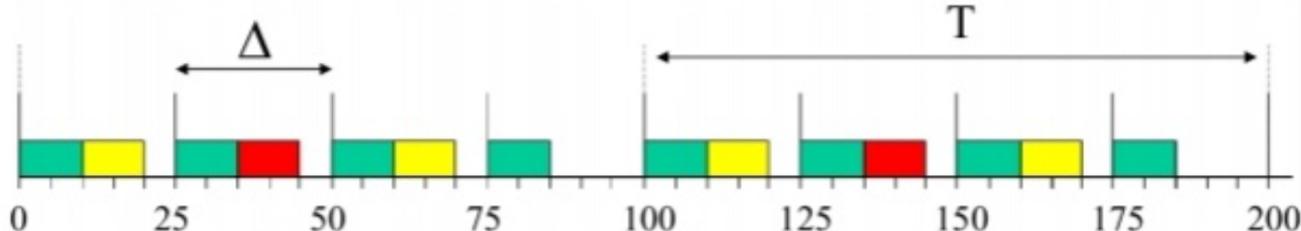
$$T = \text{mcm} \left\{ T_1, \dots, T_m \right\} \rightarrow \text{iperperiodo!}$$

ESEMPIO:

SI HA IL SEGUENTE RASH SET:

task $\tau_i$	WCET $C_i$	period $T_i$
$\tau_A$	10	25
$\tau_B$	10	50
$\tau_C$	10	100

Dove  $T=100$  e  $\Delta=25$ . Si ha il seguente schedule:  
 $\hookrightarrow$  m.c.m       $\hookrightarrow$  M.C.D



- DOVE:
- $T_A$  SI RIPETE IN Ogni TUTTO IL SET ( $T_A = 29$ )
  - $T_B$  SI RIPETE DUE VOLTE PER PERIODO ( $T_B = 50$ )
  - $T_C$  SI RIPETE DUE VOLTE IN OGNI PERIODO ( $T_C = 100$ )
- 

POTREBBERO ESSERE DI SCHEDE UNITE E POSSIBILI DI RENNZARE MA  
CREA PROBLEMI NEL CASO DI RISU AVARIZZI, ANDRO' QUANDO  
UN RISU RUMBO IN ESCALAZIONE PER PIU' TEMPO RISPETTO A  
PUBBLICO PREVISTO.

IN QUESTO CASO SI PUO' NOTARE IN DUE MIGLIORI SITUAZIONI:

1. LASCIARE IN ESECUZIONE IL RISU → SI POSSIBBILITA' PERDERSI  
CREANDO UN EFFETTO DOMINO CON UN NUOVO RISU
2. INTERROMPERE IL RISU → POSSIBBILITA' LASCIARE IL SISTEMA  
IN UNO STATO INCONSENTE.

INCURSI DI NUOVI PROBLEMI POSSONO NEGLI DIFOLLOTTI  
ESPLICATI DALLA SCHEDE UNITE NEL CASO IN CUI SI ABBIANO DELLE  
VARIAZIONI NEI PARAMETRI

---

FATTORE DI UTILIZZO  $U$  E UPPER BOUND

IL FATTORE DI UTILIZZO DEFINISCE QUANTO TIENE VIVO SPazio  
DELL'OPERATORE FINO ALLA TERMINAZIONE DEL RISU.

$$U := \sum_{i=1}^m \frac{c_i}{T_i}$$

SI DEFINISCE L'UPPER BOUND DI U PER UN TASK SET  $\Gamma$   
SOPRA UN CERTO ALGORITMO DI SCHEDULING A UN QUANTUM:

$$U_{UB}(\Gamma, A)$$

E, SE SI VERIFICA LA CONDIZIONE  $U = U_{UB}(\Gamma, A)$  ALLORA SI  
HA CHE  $\Gamma$  UTILIZZA COMPLETAMENTE IL PROCESSORE

SI DEFINISCE QUINDI IL LEAST UPPER BOUND LA QUANTITÀ:

$$U_{LOB}(A) := \min_{\Gamma} U_{UB}(\Gamma, A)$$

L'OBBIETTIVO SARÀ QUINDI RIUSCIRE DI OTTENERE IL MASSIMO DELLA  
CAPACITÀ DI COMPUTAZIONE DEL PROCESSORE PER I TASK PER OTTENERE  
UN UPPER BOUND CHE SFUORI IL PROCESSORE E CREARE QUINDI  
IL MINIMO DI QUESTO UPPER BOUND PER IL TASK SET CONSIDERATO.

QUINDI UN CERTO TASK SET È SCHEDULABILE TRAMITE UN  
ALGORITMO A SE:

$$U \subseteq U_{UB}(A)$$

AUTORENTI di  $\sigma$  SONO VERSICHE

$$U > 1$$

### THEOREM UPPER BOUND

SE IL FATTOR DI UTILIZZO DEL PROCESSORE  $U$  È UNO O  
PIÙ SEI  $\Gamma$  È VERSICHE DI CIO, ALLORA  $\Gamma$  È UNO FEASIBILE

$$> 1$$

Dimo:

PER PROVARE SI HA CHE  $U > 1$  CONSIDERARE L'IPERPERIODO  $H$ ,  
CHE PER DEFINIZIONE È SOMMA  $H > 0$ , E SCARICO:

$$U > 1 \Rightarrow UH > H$$

QUESTO IMPLICA CHE, AVENDO  $U$  PER COME  $\sigma$  DEFINITO:

$$\sum_{i=1}^m \frac{C_i}{T_i} H > H \Rightarrow \sum_{i=1}^m \frac{H}{T_i} C_i > H$$

Dove:

- $\frac{H}{T_i}$  È IL NUMERO DI VOLTE IN CUI IL FATTOR  $T_i$  È USCITO NELL'IPERPERIODO

- $\frac{H}{T_i} C_i$  È IL TEMPO DI COMPUTAZIONE NCESSARIO PER FARE  $T_i$  NELL'IPERPERIODO

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

IL TOTALE DI COMPUTAZIONI DIVISI DA TUTTO IL TASSO SET NELL'INTERVALLO

PUNTI SE SI OLTREPASSA  $H$  SI HA CHE IL TASSO SET NON È  
FEASIBLE!

## NATO MAXIMIZING SCHEDULING

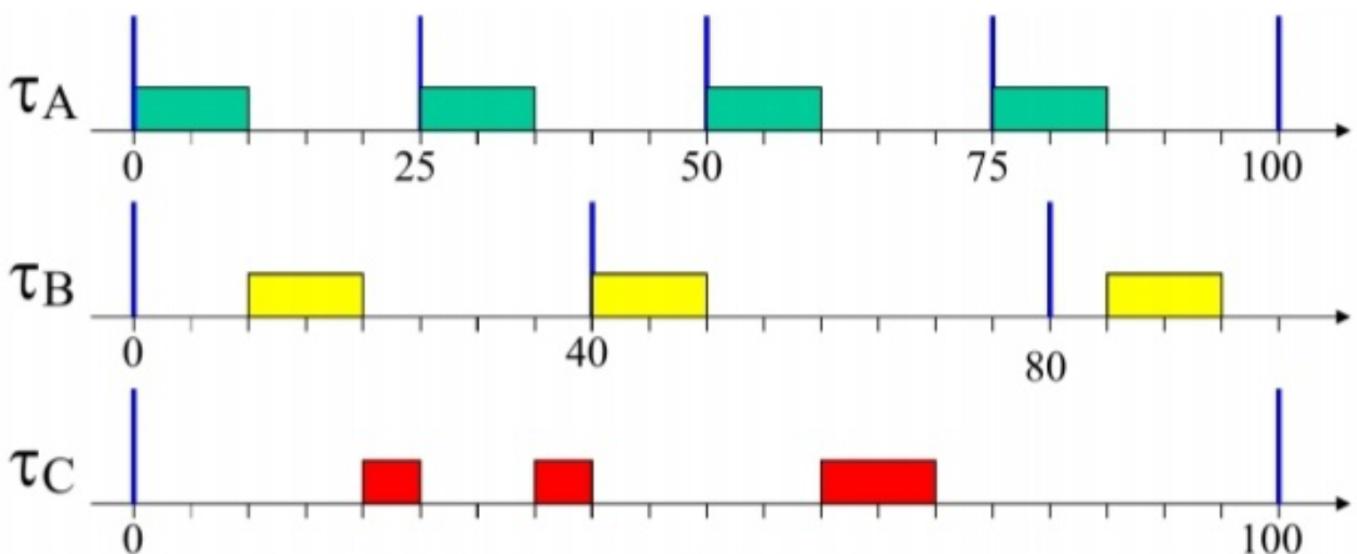
→ SCHEDULING ON THE  
PRIORITY-DRIVEN

QUESTO TIPO DI SCHEDULING ASSEGNA LE PROVVISI IN MANIERA  
ONDE STATICA SUA BASE DEI TASSI PERIODICI.

VENGONO UTILIZZATI PER SCHEDULARE I TASSI PUNZENTI  
PERIODICI, avendo perciò  $D_i = T_i$  E TASSI  $T_i$ . PUNZI, A CAUSÀ  
 DI UN TASSO, VENGONO ASSEGNAI CON PROVVISI FISSATI CON IL  
INVERZAMENTE PROPORZIONALE AL PERIODO DEL TASSO

PIU', AD ESEMPIO, SE HAI CHE

Provv.  $P_A \rightarrow$  Provv.  $P_B \rightarrow$  Provv.  $P_C$



## TEOREMA OTTIMALITÀ DI RM

PER LA SCHEDULUNG DI TSKU PERIODICI CON DEADLINES UMANI AL PERIODICO

RM È OTTIMALE, PER QUANTO RISULTA LA FEASIBILITY, USPETTO A SUOI CRITERI D'ALGORITMO A PROGETTO FISSATO

- SE UNO SCHEDULE A PROGETTO FISSATO È FEASIBILE PER IL TSKU SET  $\Gamma$   $\Rightarrow$  LO SCHEDULE DI RM È FEASIBILE PER  $\Gamma$
- SE LO SCHEDULE DI RM NON È FEASIBILE PER UN TSKU SET  $\Gamma$   $\Rightarrow$  NON ESISTE UNO SCHEDULE A PROGETTO FISSATO PER  $\Gamma$

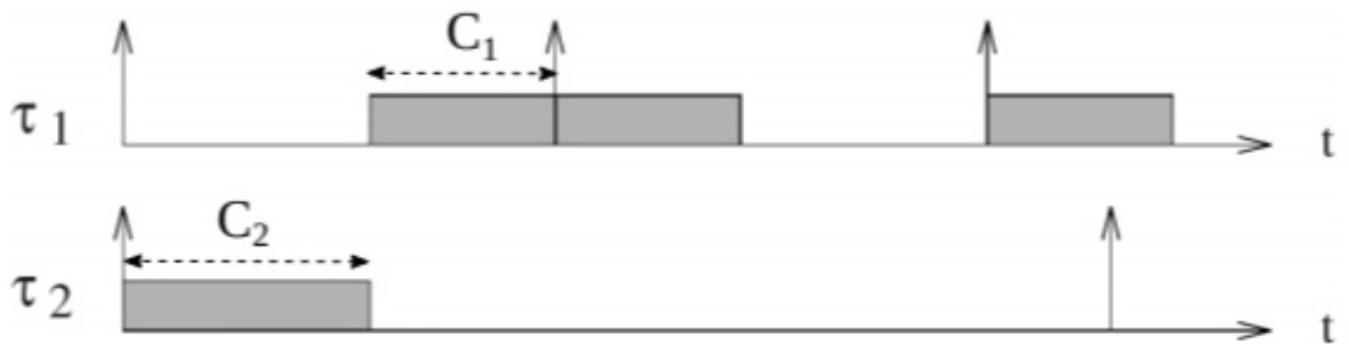
Dove i due segmenti sono EQUIVALENTI ( $a \rightarrow c \rightarrow b \rightarrow \bar{a}$ )

IN CONSIDERANDO CHE UNICO POSSIBILE CONDURRE IL SUO TEMPO DI RISPOSTA PEGGIORARE IL SUO ISSTANTE CRITICO ALLORA È SUFFICIENTE VERIFICARE L'OTTIMALITÀ DELL'ALGORITMO PROGETTO NELL'ISTANTE CRITICO, ANBRO:

SE UNO SCHEDULE A PROGETTO FISSATO È FEASIBILE PER UN TSKU SET  $\Gamma$  NELL'ISTANTE CRITICO, ALLORA LO SCHEDULE RM È FEASIBILE PER  $\Gamma$  NELL'ISTANTE CRITICO

Dim:

CONSIDERAMO UN TSKU-SET  $\Gamma$  COSTITUITO DA DUE TSKU PERIODICI  $T_1$  E  $T_2$  TAKI CHE  $T_1 < T_2$



SE LE PROVVISI SONO ASSEGNAME TRAMITE RM  
ALLORA:

→ È l'opposto di PMA PERIODICO IN QUESTO CASO IL FLASH CON PERIODO MINIMALE MA LA PROVVISI PUÒ AVERE PERIODO DIVERSIBILE ESSERE IL CONTRARIO

$$\text{PROVVISI } T_2 > \text{PROVVISI } T_1$$

⇒ LO SCHEDULÉ È FEASIBLE PER Γ NON ISCRIVENDO  
SE:

$$C_1 + C_2 \leq T_1$$

→ SE I TEMPI DI COMPUTAZIONE DEI DUE TASK RIGURGANO NEL PRIMO PERIODO MURO IL VARIABILE È SOVRACCARICO

FEASIBLE

→ SE L'ISCRIZIONE CRITICA

PRESI SONO SUFFICIENTE DIMOSSRARE CHE SE  $C_1 + C_2 \leq T_1$   
NUOLO SCHEDULÉ RM È FEASIBLE PER Γ;

CONSIDERO IL NUMERO DI PERIODI DI  $T_1$  INTERAMENT CONTENUTI  
IN  $T_2$  COME LI PENSANDO:

→ PARTE INFERIORE  
INTERNA

$$F := \lfloor T_2 / T_1 \rfloor$$

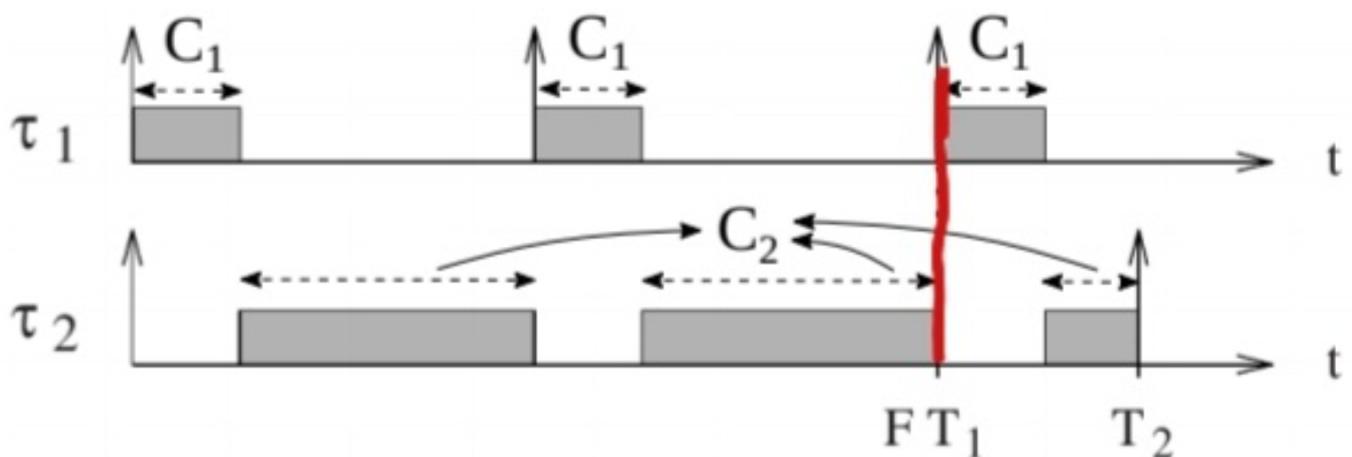
SE LE PROVVISI SONO ASSEGNAME SECONDO RM NUO

la priorità di  $T_1$  > priorità di  $T_2$ . Si distinguono due casi:

① Se tutti i job di  $T_1$  che sono stati riusciti entro l'intervalle  $[0, T_2]$  sono completati prima

che il secondo job di  $T_2$  sia riuscito, allora:

$$C_1 < T_2 - FT_1$$



Allora si trovi se è schedulingabile se

$$(F+1)C_1 + C_2 \leq T_2$$

è bisogni provare dimostrare che!

$$C_1 + C_2 \leq T_1 \Rightarrow (F+1)C_1 + C_2 \leq T_2$$

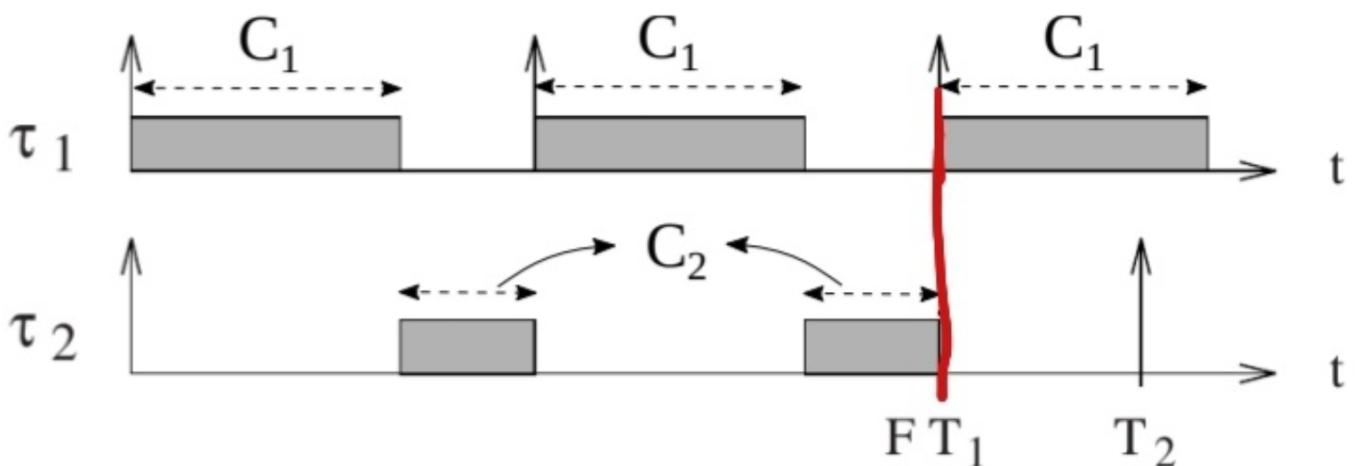
①

2

Allora job di  $T_1$  che sono solo risorse sono  
intervalli  $[0, T_1]$  non sono stati completati

Prossimo del numero del secondo job di  $T_2$ , quando

$$C_1 \geq T_2 - FT_1$$



Allora la task set è schedulabile se

$$FC_1 + C_2 \leq FT_1$$

E bisogna provare dimostrare che

$$C_1 + C_2 \leq T_1 \Rightarrow FC_1 + C_2 \leq FT_1$$

2

1

DIMOSTRAZIONE CHE  $G_1 + G_2 \leq T_1 \Rightarrow (F+1)G_1 + G_2 \leq T_2$

$G_1 + G_2 \leq T_1 \Leftrightarrow FG_1 + FG_2 \leq FT_1$  MA  $F \geq 1 \Rightarrow$

$\Rightarrow FG_1 + G_2 \leq FG_1 + FG_2 \leq FT_1 \Rightarrow$

$\Rightarrow (F+1)G_1 + G_2 \leq FT_1 + G_1 \Rightarrow$

$\Rightarrow (F+1)G_1 + G_2 \leq FT_1 + G_1 < T_2$  MA  $G_1 < T_2 - FT_1$

$\Rightarrow (F+1)G_1 + G_2 \leq T_2$  ✓

2

DIMOSTRAZIONE CHE  $G_1 + G_2 \leq T_1 \Leftrightarrow FG_1 + G_2 \leq FT_1$

$G_1 + G_2 \leq T_1 \Rightarrow FG_1 + FG_2 \leq FT_1$  MA  $F \geq 1 \Rightarrow$

$\Rightarrow FG_1 + G_2 \leq FG_1 + FG_2 \leq FT_1$  MA  $F \geq 1 \Rightarrow$

$\Rightarrow FG_1 + G_2 \leq FT_1$  ✓

TEST SCHEDULABILITY RAI (LSD/LYND)

$O(n)$

$S \in \{0, 1\}^{2^n - 1}$

PER UN CERTO TEST-SCHED

$T_i = D_i$

D)  $m$  TESTI PUNZIONE PERIODICA  $\Rightarrow \Gamma_G^L$  SCHEDULABILE

## DA RM

IL TEST FORNISCE CHE CONSIDERA SUFFICIENZA, PUOI POSSERE ANCHE ESISTERE TASI-SGI SENZA SCHEDEAUZI  
CHE NON SONO SUFFICIENTI A CONDIZIONE, MA SE CI SONO SUFFICIENTI ALLORA SI HA CHE CI SONO SCHEDEAUZI SECONDO RM.

LA DIMOSTRAZIONE SI DÀ IN PASSI:

- ① SI ASSUME CHE I PROVVISI SIANO SECONDO RM,  
MENTRE I PROVVISI SONO INVERAMENTE PROPORZIONALI  
AI PROBLEMI
- ② SI ASSUME CHE TUTTI I TASI SERVANO CONTEMPORANEAMENTE,  
MENTRE SI CONSIDERA IL WORST CASE SCENARIO
- ③ SI NUMERANO TUTTI I TEMPI DI COMPUTAZIONE AL MASSIMO  
IN MODO DA UTILIZZARE IL PIÙ POSSIBILE IL PROCESSORE
- ④ SI CALCOLA L'UPPER BOUND UUB
- ⑤ SI MINIMIZZA UUB INSERENDO NEI PARAMETRI  
IN MODO DA OTTENERE UUB

Dimi:

CONSIDERARE UN FASH SET FORMATO DA DUE FASH PERIODI  $T_1$  &  $T_2$  CON  $T_1 < T_2$

DEFINISMO IL NUMERO DI PERIODI DI  $T_1$  INTRAMENTO CONTENUTI IN  $T_2$  COME LI PUO' DIV.

$$F := \left\lfloor \frac{T_2}{T_1} \right\rfloor$$

SI DISTINGUENO PUNTI DUE CASI:

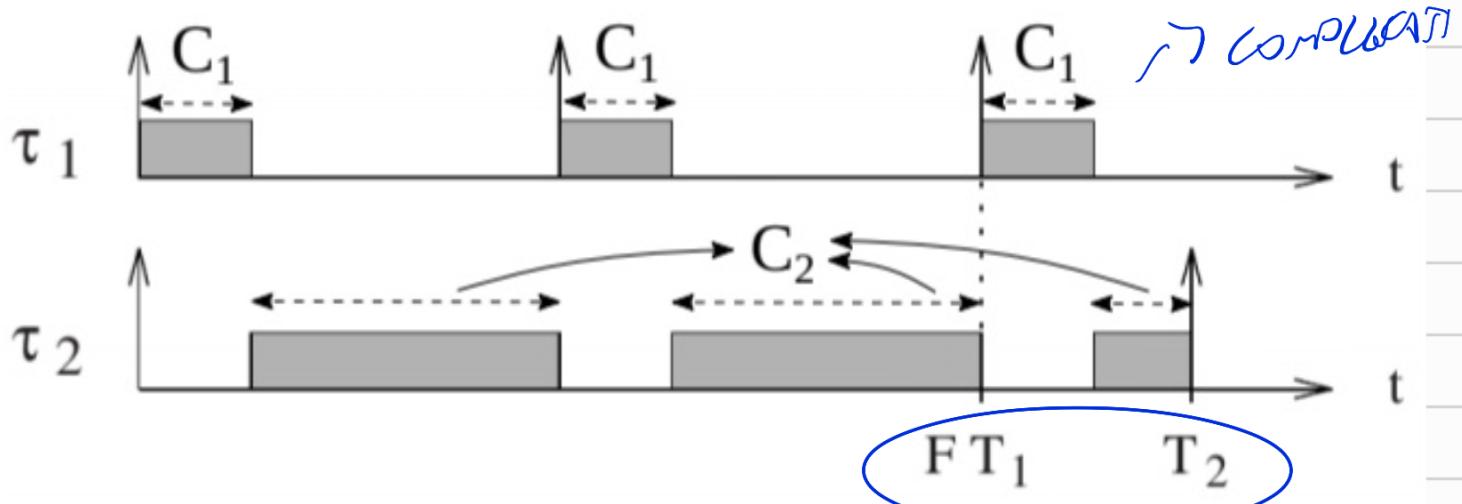
1)

TUTTI I JOBS DI  $T_1$   
TEMPO  $[0, T_2)$

PRESENTI OLTRE I INTERVALLI DI  
SOPRA COMPLETAMENTE PRIMA CHE IL

SECONDO JOB DI  $T_2$  Venga RELEASE:

$$C_1 < T_2 - FT_1$$

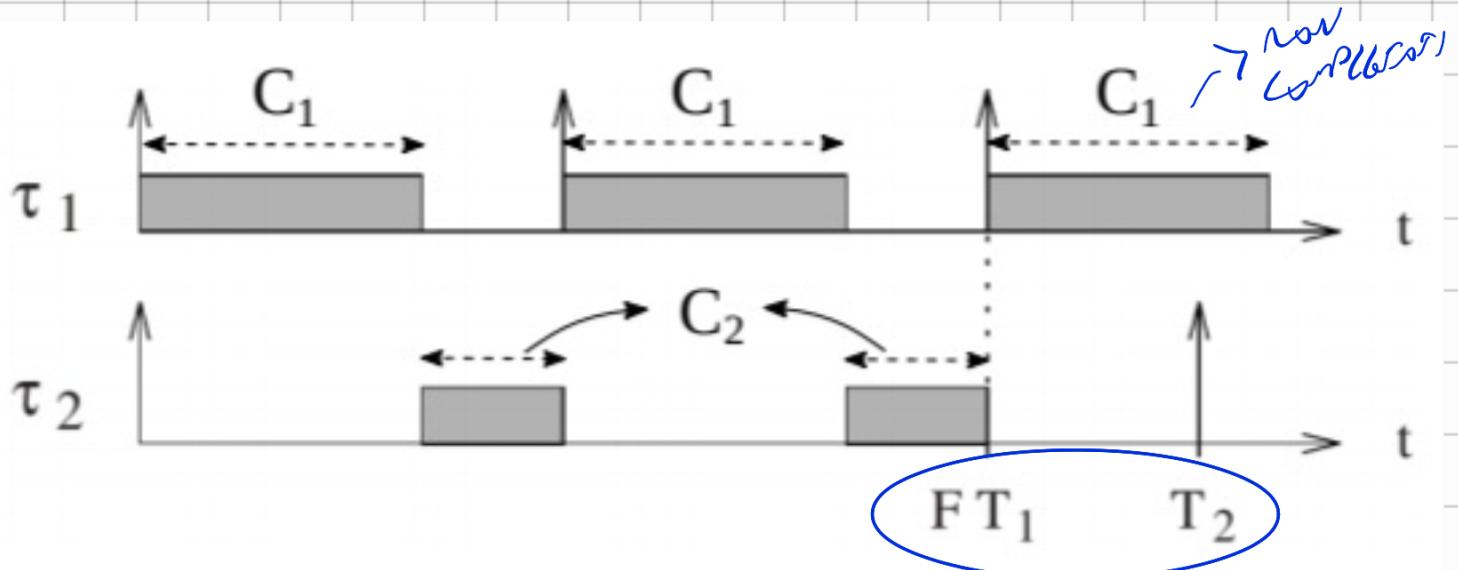


2) PURCHASING JOB DI  $T_1$  RISUSCITO ENTRO L'INTERVALLI

$[0, T_2)$  non  $G^v$  complessi prima di  $T_1$

se il secondo job di  $T_2$  sia risuscitato:

$$C_1 \geq T_2 - FT_1$$



ANALIZZIAMO IL  $C_1$   $\rightarrow C_1 < T_2 - FT_1$

$$C_2^{\max} = T_2 - (F+1)C_1 \quad \begin{array}{l} \text{durata delle complessioni} \\ \text{delle risu. } T_1 \end{array}$$

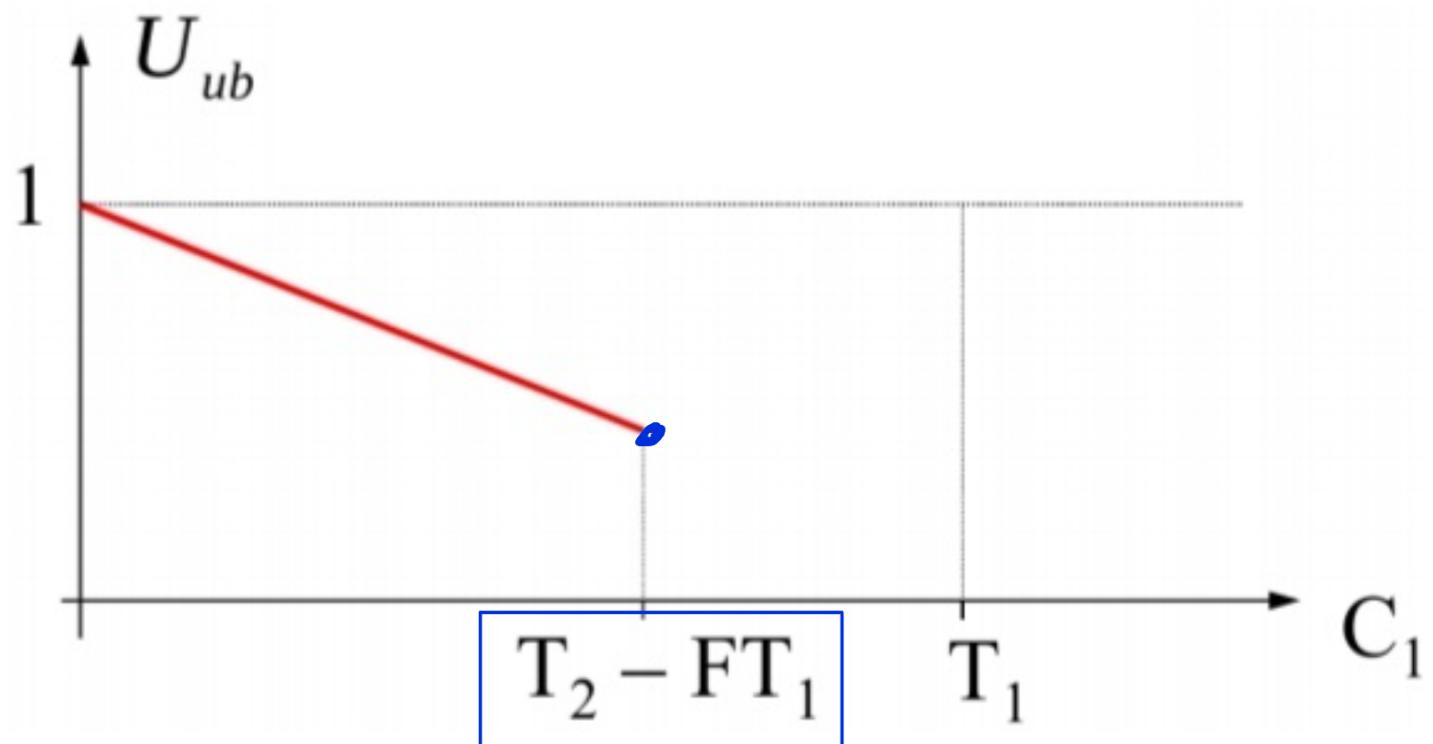
$$U_{UB} := \frac{C_1}{T_1} + \frac{C_2^{\max}}{T_2} = \frac{C_1}{T_1} + 1 - \frac{C_1}{T_2}(F+1)$$

$$= 1 + \frac{C_1}{T_2} \left( \frac{T_2}{T_1} - (F+1) \right) \quad \begin{array}{l} \text{→ } G \text{ less funzione di } C_1 \\ \text{↓} \end{array}$$

$\zeta_0 \rightarrow$  DECRESCE  
POSIÇÃO  $F := \lfloor T_2/T_1 \rfloor$

QUEST 12 VOLUME MÍNIMO DE  $C_{UB}$  SE HOUVE PONTO

$$C_1 = T_2 - FT_1$$



ANALISANDO IC  $\zeta_0 \rightarrow$   $C_1 \geq T_2 - FT_1$

$$\underline{C_2^{\max} = F(T_1 - C_1)}$$

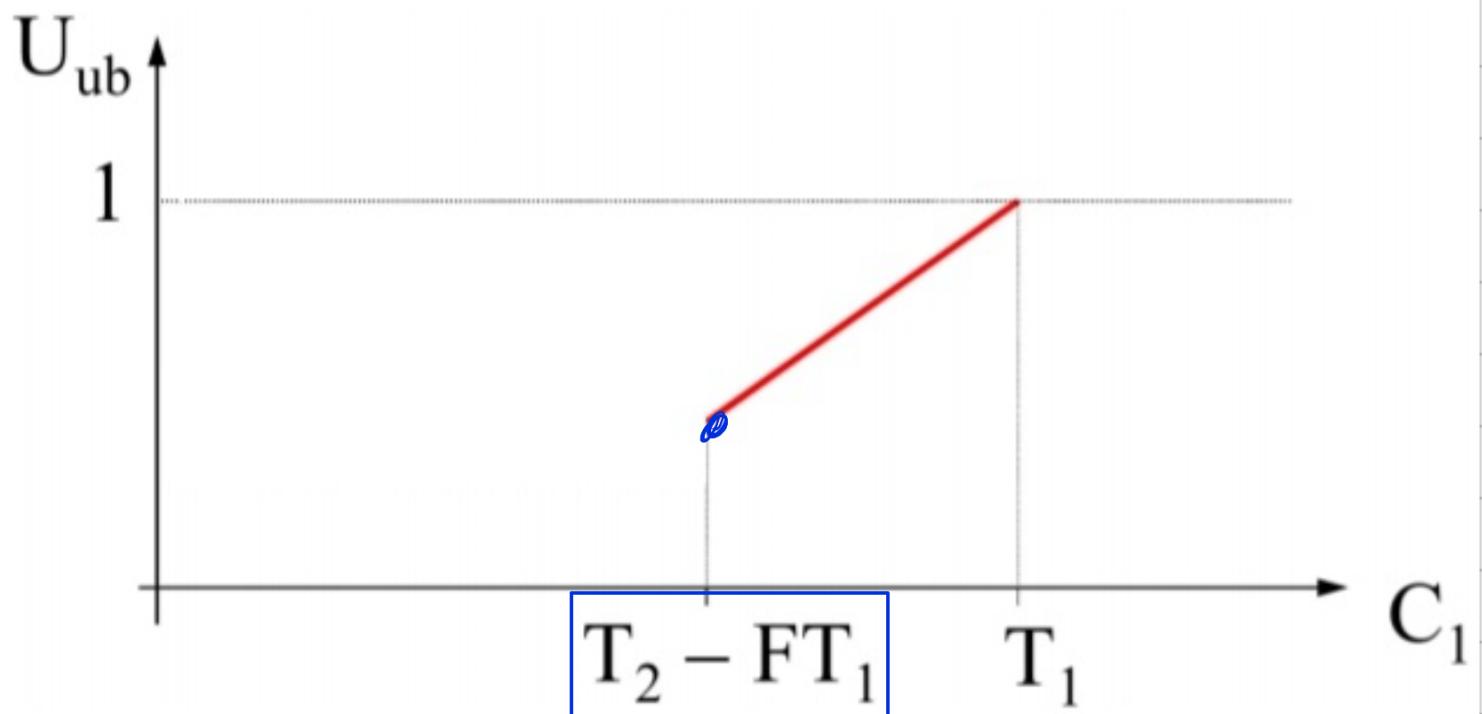
$$\underline{U_{UB} := \frac{C_1}{T_1} + F \frac{(T_1 - C_1)}{T_2} = \frac{FT_1}{T_2} + \frac{C_1}{T_2} \left( \frac{T_2}{T_1} - F \right)}$$

$\zeta_0$  OBSCURUS

QUEST 13  $C_{UB}$  MÍNIMOS COM  $C_1$  GÉMICO VOLUME MÍNIMO N

$U_{UB}$  SI HA PENSATO

$$C_1 = T_2 - FT_1$$



A QUESTO PUNTO SI CALCOLA IL VALORE:

$$U_{UB}^{\min, C_1}$$

DI  $U_{UB}$  RISPETTO A  $C_1$

$$\Rightarrow U_{UB}^{\min, C_1} = U_{UB} \left| \underbrace{C_1 = T_2 - FT_1}_{\substack{\text{PUNTO DI} \\ \text{MINIMO}}} \right.$$

$$= F \frac{T_1}{T_2} + \frac{T_2 - FT_1}{T_2} \left( \frac{T_2}{T_1} - F \right)$$

$$= F \frac{T_1}{T_2} + \left( 1 - F \frac{T_1}{T_2} \right) \left( \frac{T_2}{T_1} - F \right)$$

$$= F \frac{T_1}{T_2} + \left( \frac{T_1}{T_2} \cdot \frac{T_2}{T_1} \right) \left( 1 - F \frac{T_1}{T_2} \right) \left( \frac{T_2}{T_1} - F \right)$$

$$= F \frac{T_1}{T_2} + \frac{T_1}{T_2} \left( \frac{T_2}{T_1} - F \right) \left( \frac{T_2}{T_1} - F \right)$$

$$= \boxed{\frac{T_1}{T_2} \left( F + \left( \frac{T_2}{T_1} - F \right)^2 \right)}$$

DEFINITIONS IN PARTIAL DECOMPOSITION OF  $T_2/T_1$  COME:

$$G_1 = T_2/T_1 - F$$

$\in$  COLUMNS

$$\boxed{U_{QB}^{min, G_1} = \frac{T_1}{T_2} \left( F + \underbrace{\left( \frac{T_2}{T_1} - F \right)^2}_{G} \right)}$$
 CORRECTION

FURTHERMORE  $\propto G$ :

$$\boxed{U_{QB}^{min, G_1} = \frac{F+G^2}{T_2/T_1}} = \frac{F+G^2}{\frac{T_2}{T_1} + F - F} = \frac{F+G^2}{F+G}$$

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

MISSESINGO (in part double) si ha che:

$$0 \leq G < 1 \Rightarrow G(1-G) > 0 \Rightarrow U_{UB}^{m,n, C_1, F}$$

numbers con  $F$

$$\Rightarrow U_{UB}^{m,n, C_1, F} = U_{UB}^{m,n, C_1} \Big|_{F=1} = \frac{T_1}{T_2} \left( 1 + \left( \frac{T_2}{T_1} - 1 \right)^2 \right) =$$

$$= \frac{U^2 - 2U + 2}{U} \quad \text{con } U = \frac{T_2}{T_1}$$

ora si misura rispetto a  $U$ :

$$\frac{dU_{UB}^{m,n, C_1, F}}{dU} = \frac{(2U-2)U - (U^2 - 2U + 2)}{U^2} = \frac{U^2 - 2}{U^2} < 0$$

PER  $\boxed{U = \sqrt{\Sigma}}$

$\hookrightarrow \sqrt{m+n}$  crescente (?)

$$\Rightarrow U_{UB} = U_{UB}^{m,n, C_1, F} \Big|_{U=\sqrt{\Sigma}} = 2(\sqrt{2}-1) \approx 0.83$$

$\hookrightarrow$  i 2 portano 1  
sono 2 fasi

nel caso di fissi con periodo armonico  $\mathcal{N}_1 \in \mathcal{N}_2$

con  $T_1 < T_2$  si ha che:

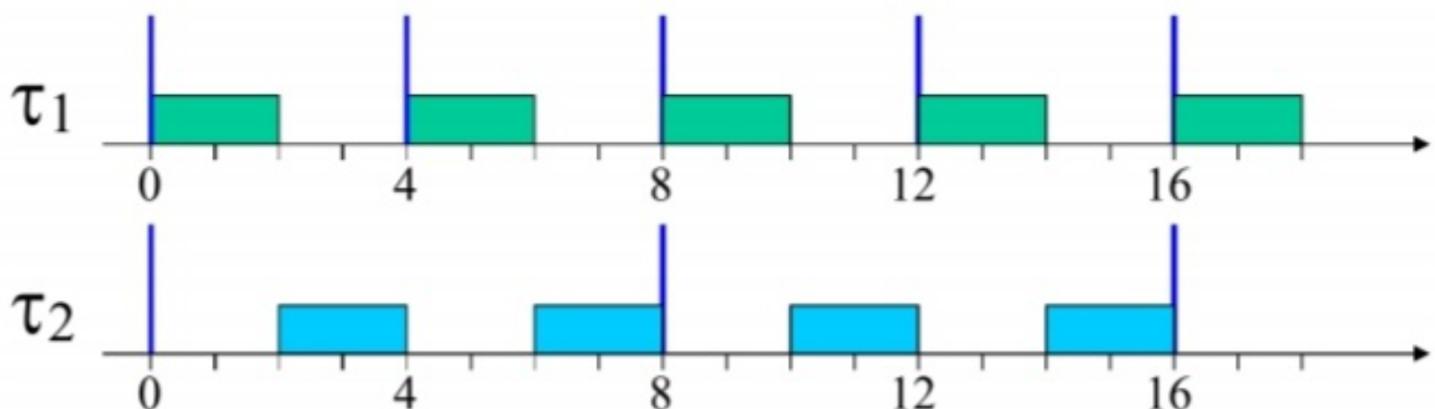
$\frac{T_2}{T_1} \in \mathbb{N}$  per definizione di periodo armonico e, dunque:

Che  $T_1 < T_2$ , allora:

$$F := \left\lfloor \frac{T_2}{T_1} \right\rfloor = \frac{T_2}{T_1} \Rightarrow U_{WB} =$$

$$\frac{T_1}{T_2} \left( F + \left( \frac{T_2}{T_1} - F \right)^2 \right) = 1$$

puossi perciò risalire per periodi armonici se ha le  
medesime efficienze e utilizzo del processore.



Abbiamo visto la dimostrazione del fatto che  
se i due risulti veramente nel caso di **no** risulta:

• le condizioni per le scintillazioni di 2  
risulti con  $T_1 < T_2$  sono:

$$C_1 = T_2 - FT_1 \text{ con } F=1 \Rightarrow T_2 < 2T_1$$

$$\Rightarrow C_1 = T_2 - FT_1 = T_2 - T_1$$

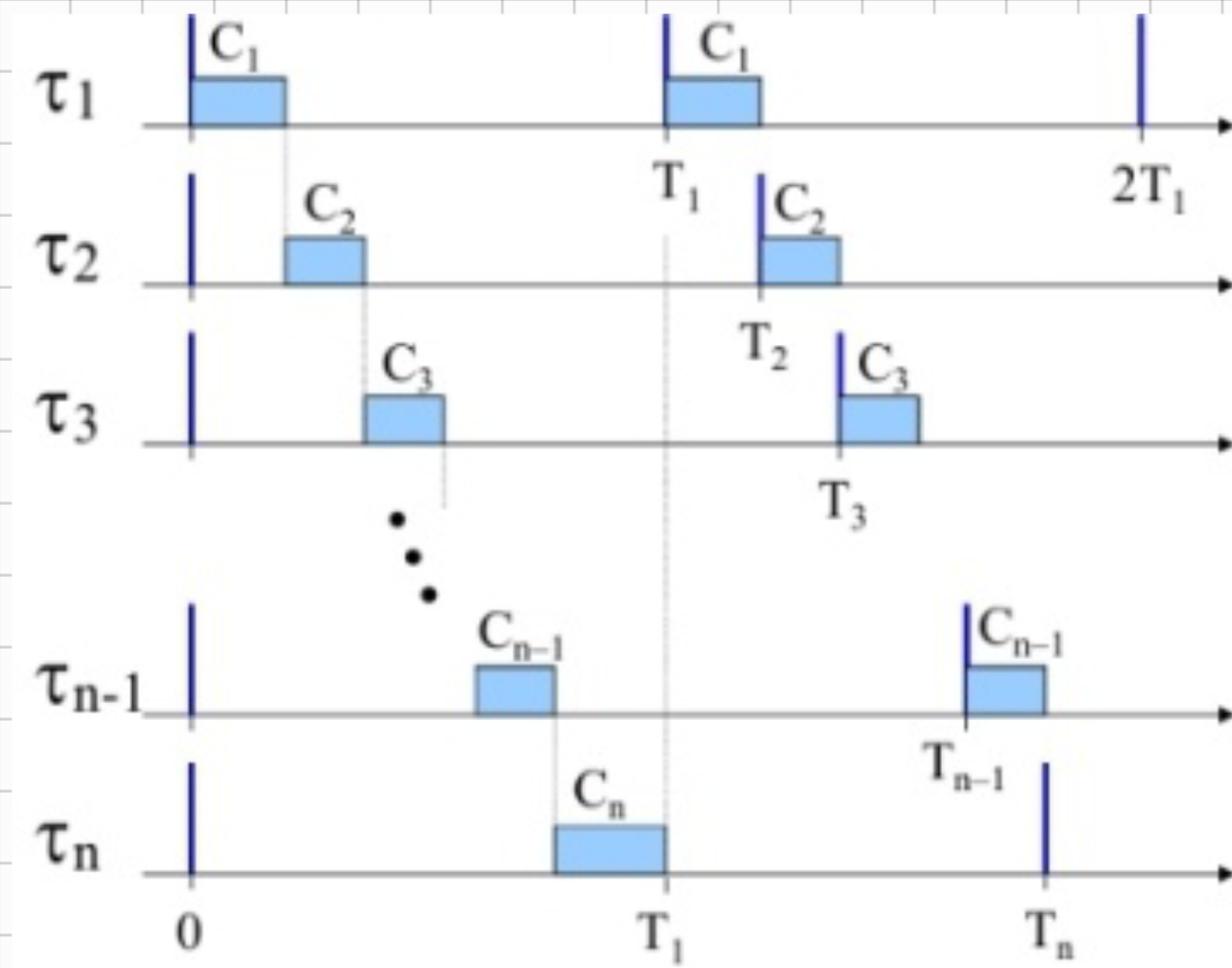
$$C_2 = F(T_1 - C_1) = T_1 - C_1 = T_1 - (T_2 - T_1) = 2T_1 - T_2$$

$$T_1 < T_2 < \dots T_m$$

- NEL CASO DI SCONTI CONSIDERARE UNA LINEA DI KARNO CON  
CONDIZIONI PRECEDENTI PRESE

$$T_m < 2T_1, C_1 = T_2 - T_1, C_2 = T_3 - T_2, \dots$$

$$\Rightarrow C_m = T_1 - \left( \sum_{i=1}^{m-1} C_i \right) = 2T_1 - T_m$$



SI CALCULS L'UPPER BOUND SU U BASEDONC SUCHE  
WORST-CASE CONDITIONS D'SENARIOU BRUT:

$$U_{UB} = \sum_{i=1}^n \frac{G_i}{T_i} = \frac{T_2 - T_1}{T_1} + \frac{T_3 - T_2}{T_2} + \dots + \frac{T_n - T_{n-1}}{T_{n-1}} + \\ + \frac{\sum_{i=1}^{n-1} T_i - T_n}{T_n} = \\ = \frac{T_2}{T_1} + \frac{T_3}{T_2} + \dots + \frac{T_n}{T_{n-1}} + \frac{\sum_{i=1}^{n-1} T_i}{T_n}$$

DEFINISIOMA RUMS

$$R_i = \frac{T_{i+1}}{T_i}$$

$$H_i = 1, \dots, n-1$$

$$\text{CON} \prod_{i=1}^{n-1} R_i = \frac{T_n}{T_1}$$

RUMS

$U_{UB} = \sum_{i=1}^n R_i + \frac{2}{R_1 R_2 \dots R_{n-1}} - n$	G SIMPLIFIÉE
---	--------------

RESPONSE A R<sub>i</sub>:

$$\frac{\partial U_{UB}}{\partial R_i} = 1 - \frac{2}{R_i^2} \cdot \underbrace{\frac{1}{R_1 R_2 \dots R_{i-1} R_{i+1} \dots R_n}}_{=} =$$

$$= 1 - \frac{Z}{R_i P} \text{ con } P = \prod_{i=1}^{n-1} R_i$$

usum  $\frac{\partial U_{UB}}{\partial R_i} = 0$  per  $R_i P = Z$

us  $R_i P = Z \quad \forall i = 1, \dots, n-1$  se  $R_i \geq z^{\frac{1}{n}}$

$$\Rightarrow P = \left(z^{\frac{1}{n}}\right)^{n-1}$$

calcularmos infint  $U_{UB}$ :

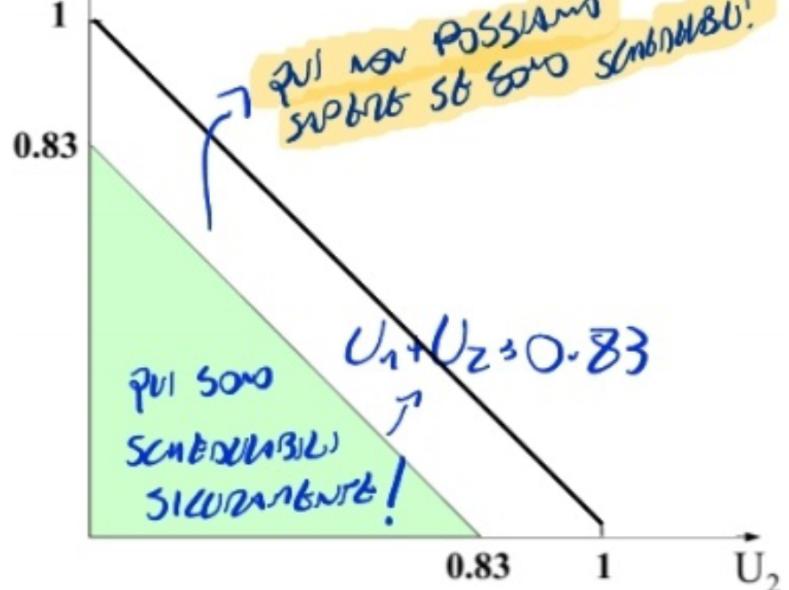
$$U_{UB} = \sum_{i=1}^n R_i + \frac{Z}{P} - \frac{1}{n} \left| R = z^{\frac{1}{n}}, P = \left(z^{\frac{1}{n}}\right)^{n-1} \right.$$

$$= \dots \\ = n \left( z^{\frac{1}{n}} - 1 \right)$$

$n$	$U_{lub}$
1	1.000



1	1.000
2	0.828
3	0.780
4	0.757
5	0.743
10	0.718
20	0.705
50	0.698
100	0.696
1000	0.693



↳ all'ombra del rischio  
dove il rischio di investimento

## RM HYPERBOLIC BOUNDS

SG  $\prod_{i=1}^m (U_i + 1) \leq 2$  per un rischio SG di rischio

PUNTO PERSICO MIGLIOR SCONTO

Secondo RM

SI HA CHE LE WORST CASE CONDITIONS PER UN SCONTO SI DEDUCONO DA UN RISCHIO CON

$$T_1 < T_2 < \dots < T_m$$

Suo cnb:

$$C = \left( \sum_{i=1}^{m-1} S_i \right)$$

$$C_m \leq T_1 - \sum_{i=1}^m C_i$$

$$= \boxed{2T_1 - T_m}$$

Dim:

SATISFIES ONE OF WORST CASE SCHEDULABILITY CONDITION SO IT'S PLANNED!

$$\boxed{\sum_{i=1}^m C_i \leq T_1}$$

$$\Rightarrow \sum_{i=1}^{m-1} C_i + C_m \leq T_1 \Rightarrow C_m \leq T_1 - \sum_{i=1}^{m-1} C_i$$

$$\Rightarrow C_m \leq 2T_1 - T_m \rightarrow \text{DIVISION PER } T_m$$

∴ CONSIDERS ONE  $\sum_{i=1}^{m-1} C_i = T_m - T_1$  AS WORST CASE SCHEDULABILITY CONDITION  $\Rightarrow$

$$\Rightarrow U_m \leq \frac{2T_1}{T_m} - 1 \Rightarrow U_{m+1} \leq \frac{2T_1}{T_m} = \frac{2}{\overline{R_i}} = \frac{2}{\prod_{i=1}^m R_i}$$

$$\prod_{i=1}^{n-1} R_i$$

$$= \frac{2}{\prod_{i=1}^{n-1} (U_i + 1)} \Rightarrow$$

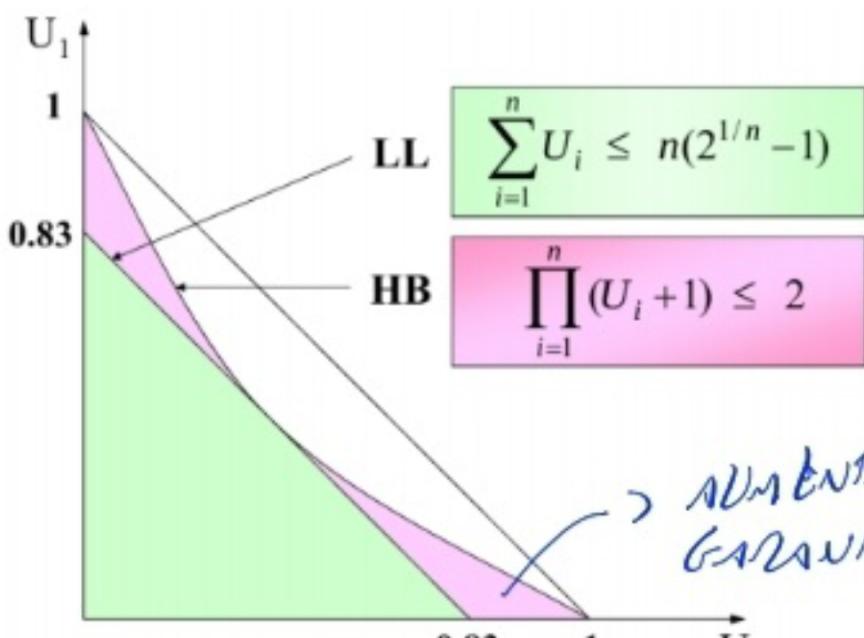
$$\boxed{\prod_{i=1}^n (U_i + 1) \leq 2}$$

□

POSSI IL VINCERE I PREMII UGLI MASSIMIZZARE STAZIO,  
AVENDO SE IL VINCERE NON VINCERE SOLO:

L'UNA SCELTA NON FEASIBILE DI RM POSSIBILE  
CON PUBBLICHEZZZO DEL PROCESSORE

IN PRACTICE SE NON VINCERE SOLO SISSIMO SPARIRE PER  
COSÌ CHE GLI SYSTEMI CON CORDI FISSI-SOT NON  
Sono POSSIBILI SCENDENDO RM!



MENTRE LA ZONA CHE GARantisce la SCELTA POSSIBILE

## ALGORITMO DEDICATO MONOTONICO

→ ONDULATI  
STATIC-PRIORITY-DRIVEN

In PRATICA G' UNA ESTENSIONE DI DM AI TASI PERIODICI  
CON DECADIMENTI VINCOLATI, INIZIO:

→ PER RISORSI  $D_i = T_i$



$$D_i \leq T_i$$

In questo algoritmo le priorità sono assegnate in maniera  
INVERSALEMENTE PROPORTZIONALE alle DECADIMENTI NEGATIVI

## OTTIMIZZARE DJ DM

DM È OTTIMALE PER QUANTO RIGUARDA LA FLESSIBILITÀ  
 MA NON CON ALGORITMO A PRIORITÀ FISSATE:

- SE IL SCHEDULING A PROTOTIPO FISSATO È FLESSIBILE PER UN TASI-SET  $\Gamma$ , ALLORA LO SCHEDULING PROPOSTO DA DM SARÀ FLESSIBILE PER  $\Gamma$
- SE LO SCHEDULING PROPOSTO DA DM NON È FLESSIBILE PER UN CERTO TASI-SET  $\Gamma$  ALLORA NESSUNO SCHEDULING A PROTOTIPO FISSATO È FLESSIBILE PER  $\Gamma$

## PERIODI DI MIGRAZIONE

SE PROVAMO A VERIFICARE LA SCHEDABILITÀ DELL'ALGORITMO  
TRAMITE L'UTILIZZO DEI TEST **LL** E **AB** DOBBIANO PER  
FORZA SOSTITUIRE I PERIODI CON LE DENSITÀ POICHE' DM,  
PER DEFINIZIONE, NON È CRI:

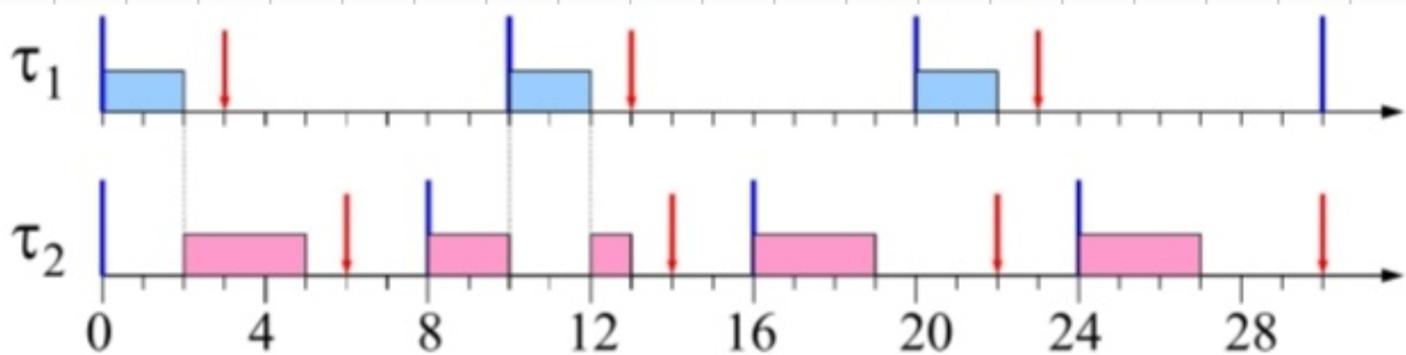
$$D_i \leq T_i$$

FACENDO QUESTA SOSTITUZIONE PERIODI SI INCONTRA NEL PROBLEMA CHE  
ANCHE LE DENSITÀ SONO MINORI DELL'UNO AI PERIODI  
E QUESTO CAUSA UNA FORTE CONSERVATIVITÀ CHE RISOLVE I  
DUE TEST PLESSIMISSIONE.

In particolare, per capire che cosa succede i test non  
vengono verificati, si può comunque schedularlo il task set!

## ESEMPIO:

Il seguente task-set risulta schedulabile:



Anche se **LL** fallisce:

$$C_1 + C_2 = 2 + 3 = 7 > 1 \quad \times$$

$D_1$   $D_2$  3 6 6

E HB FALSEE:

$$\cdot \left( \frac{C_1 + 1}{D_1} \right) \left( \frac{C_2 + 1}{D_2} \right) = \frac{5}{2} > 2 \quad \times$$

IN QUESTO CASO OCCORRE SCRIVERE LA **RESPONSE TIME ANALYSIS**. PER UNA TSI  $T_i$ :

1) SI CALCOLA L'**INTERFERENZA**  $I_i$  DOVUTA ALLA PRESENZA DI TSI A PROVVISORIA MIGRAZIONE NELL'INTERVALLO  $[0, R_i]$ .

$$I_i = \sum_{T_u | D_u \in D_i} z_{i,u} C_u$$

$\hookrightarrow$  PER TUTTI I TSI  $u$   
A PROVVISORIA MIGRAZIONE

Dove  $z_{i,u}$  è il numero di uscita di  $T_u$  in  $[0, R_i]$

$$\Rightarrow I_i = \sum_{u=1}^{i-1} \left\lceil \frac{R_i}{T_u} \right\rceil C_u$$

$\hookrightarrow$  ASSUMENDO CHE I TSI SONO ORDINATI PER VERSO DEL PRIMO CHE HA PROVVISORIA PIÙ ALTA.

2) A QUESTO PUNTO SI CALCOLA IL TEMPO DI DESCRIZIONE

$$R_i = C_i + T_i \cdot C_i + \sum_{u=1}^{i-1} \left\lceil \frac{R_i}{T_u} \right\rceil C_u$$

TUTTOVA ESSENDO CHE ABBIANO IL TERMINE  $R_i$  IN COMUNE  
 I VARI OGNI' GRUPPI, POSSONO PIÙ ESSERE DISPOSTI IN  
 MOLTIOS GRUPPI MA SENZA L'UTILIZZO DI UN **ALGORITMO**  
**ITERATIVO**:

3) INFINE SI DVE VERIFICARE CHE

$$R_i \leq D_i$$

L'ALGORITMO ITERATIVO È QUINDI SVOLGEO IN DUE STEP:

- **STEP 0**: SI OTTENNE IL VALORE INIZIALE DEL TERMINE  $R_i$   
 DISPOSTA (entro il tempo) IN POSIZIONE MINIMA POSSIBILE  
 I RISULTATI ARRIVANO ALLO STESSO INSTANTE  
CASO PEGGIORIO

$$R_i^{(0)} = \sum_{h=1}^i C_h$$

- **STEP J**:  $R_i^{(J)} = C_i + \sum_{u=1}^{i-1} \left\lceil \frac{R_i^{(J-1)}}{T_u} \right\rceil C_u$

€ SJ CONTINUES AD ITERARE FINO A FINIRE O' VERSO UNA  
SOLUZIONE CONSIDERATE:

$$R_i^{(j)} > R_i^{(j-1)} \text{ AND } R_i^{(j)} \leq D_i$$

**input** : A set  $\Gamma$  of  $n$  periodic tasks  $\tau_1, \dots, \tau_n$  with constrained deadlines  
**output** : TRUE if the task set  $\Gamma$  is schedulable by DM, FALSE otherwise

```

1 foreach task  $\tau_i \in \Gamma$  do
2    $I_i = \sum_{k=1}^{i-1} C_k$ 
3   do
4      $I_i = R_i + C_i$ 
5     if  $R_i > D_i$  then
6       return FALSE
7     end
8      $I_i = \sum_{k=1}^{i-1} \left\lceil \frac{R_i}{T_k} \right\rceil C_k$ 
9     while  $I_i + C_i > R_i$ ;
10  end
11 return TRUE

```

LA RISPOSTA ALLA ANALYSIS PRESENTE LA SEGUENTE  
COMPLESSITÀ:

- COMPLESSITÀ POLINOMIALE NEL NUMERO DI TASK  $n$
- COMPLESSITÀ POLINOMIALE NEL MASSIMO NUMERO DI STAZIONI  
PER OGNA TASK  $N$

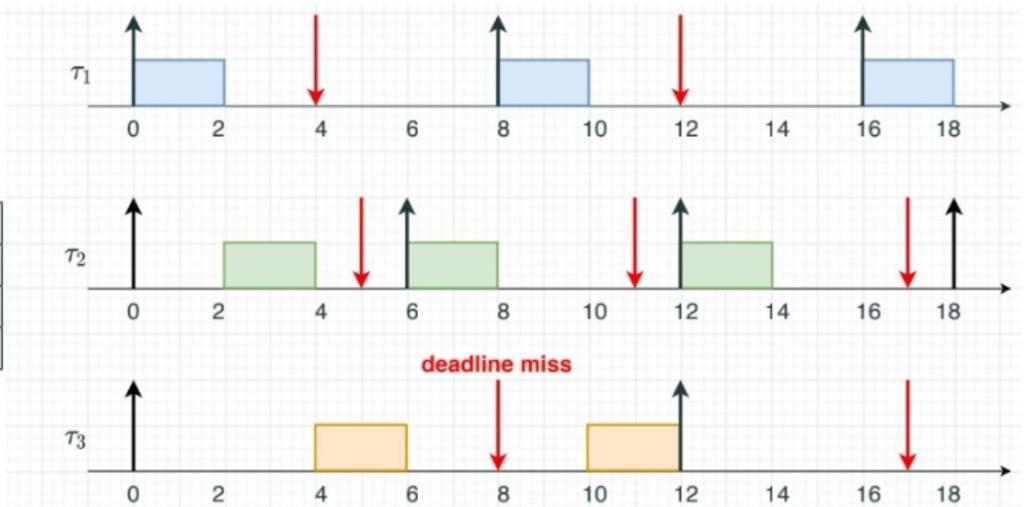
QUESTO PRESENTA UNA COMPLESSITÀ PSEUDO-POLINOMIALE



$O(m \cdot N)$

## ESEMPLO SCENARIO DI INFESIBILE

task $\tau_i$	$C_i$	$T_i$	$D_i$
$\tau_1$	2	8	4
$\tau_2$	2	6	5
$\tau_3$	4	12	8



## APPROXIMAZIONE DI RESPONSE TIME ANALYSIS:

$$1) . R_1^{(0)} = C_1 = 2 < D_1 \quad \left\{ \Rightarrow R_1 \leq 2 \right.$$

$$. R_1^{(1)} = C_1 = 2 < D_1 \quad \left\{ \right.$$

$$2) . R_2^{(0)} = C_1 + C_2 = 5 < D_2 \quad \left\{ = R_2 \leq 5 \right.$$

$$. R_2^{(1)} = C_2 + \lceil R_2^{(0)} / f_1 \rceil C_1 = C_2 + C_1 = 5 < D_2 \quad \left\{ \right.$$

$$3) . R_3^{(0)} = C_1 + C_2 + C_3 = 8 = D_3$$

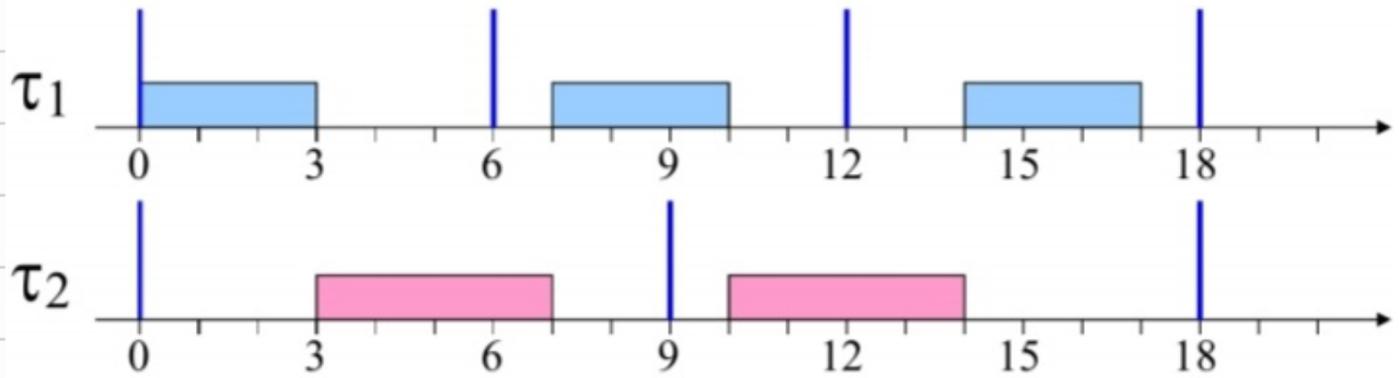
$$R_3^{(1)} = C_3 + \lceil R_3^{(0)}/T_1 \rceil C_1 + \lceil R_3^{(0)}/T_2 \rceil C_2 =$$

$$= C_3 + C_1 + 2C_2 = 10 > D_3 \Rightarrow \text{INFEASIBLE!}$$

## ALGORITMO GARDNER-DENNING - FIRST (EDF)

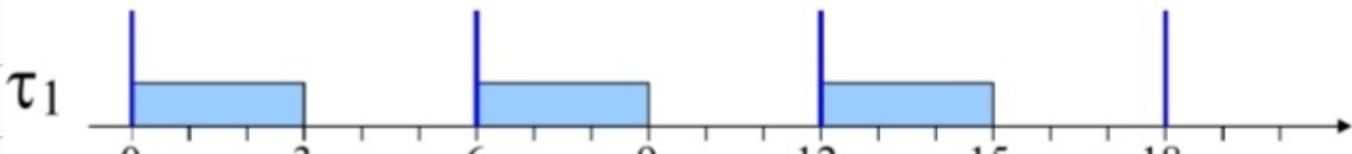
→ ONDE A  
PRIORITÀ  
DINAMICA!

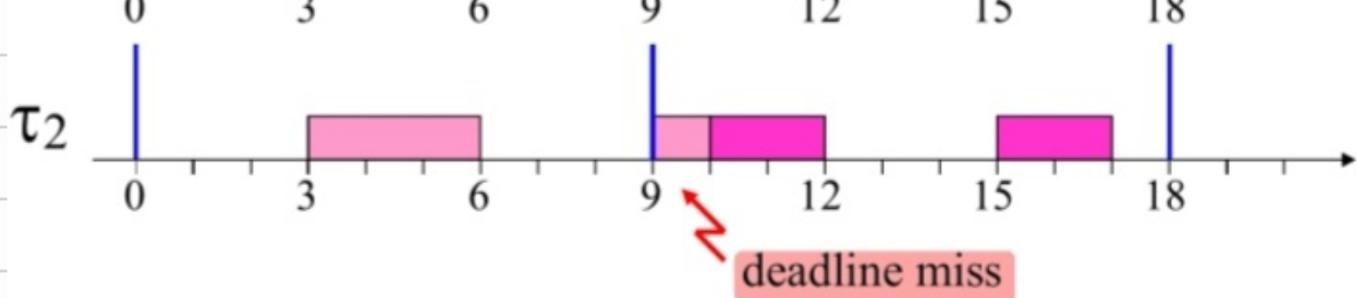
È un algoritmo di scheduling di task per unità di tempo (DTI) in cui un task ha una priorità dinamica inversamente proporzionale alla sua deadline assoluta



In questo caso vediamo che ogni volta si va a modificare la priorità dei task da eseguire sulla base di "quanto è vicino la deadline assoluta".

Possiamo anche vedere come, per questo esempio, si ha che lo scheduale EDF è feasible, mentre lo scheduale RM non è feasible proprio perché l'utile dei task unici lì è AB non BASTA.





## TEST DI GARANZIA PER EDF (TEOREMA)

IL TEOREMA AFFERMA CHE UN INSIEME DI TASSI PURAMENTE PERIODICI È SCHEDULABILE SECONDO GDF  $\Leftrightarrow U \leq 1$

IL TEST HA GRADO COMPLESSITÀ POLINOMIALE  $O(n)$  DIPENDENTE DA  
NUMERO  $n$  DI TASSI

IN PARIOLA, ESSENDO UNA CONDIZIONE NECESSARIA E SUFFICIENTE, SI HA CHE:

• NECESSITÀ: SE UN INSIEME DI TASSI PURAMENTE PERIODICI È SCHEDULABILE SECONDO GDF, ALLORA  $U \leq 1$

• SUFFICIENZA: SE  $U \leq 1 \Leftrightarrow$  ALLORA UN INSIEME DI TASSI PURAMENTE PERIODICI È SCHEDULABILE SECONDO GDF

DIM NECESSITÀ:

SE  $U > 1 \Rightarrow UT > T$  con  $T = T_1 \cdot T_2 \cdot \dots \cdot T_m > 0$ .

ALLORA SI HA CHE

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

PER DEFINIZIONE DI  $U$

$$\Rightarrow \sum_{i=1}^n \frac{1}{T_i} C_i > T \text{ Dato } \frac{1}{T_i} \text{ sono numeri di Russo} \text{ per il tasso nel periodo } [t_1, t_2]$$

$\Rightarrow$  La richiesta totale nel periodo  $[t_1, T]$  è maggiore rispetto a  $T$   $\Rightarrow$  il tasso set non è pensabile!

### D.m. sufficienza:

Assumiamo, per assurdo, che il tasso-set non è scrutabile per GDF ma che  $\forall i$  definibile!

- $t_1$  := primo istante di tempo nel quale una deadline viene rispettata
- $[t_1, t_2]$  := l'intervallo di tempo misurato da quando inizio dell'elenco processi prima dell'istante  $t_2$  durante il quale sorgono i jobs  $i_1, i_2, \dots, i_n$  con tempi di arrivo  $d_{i,u} \geq t_1$  e deadlines assoluti di istante  $t_2$  sono eseguiti
- $C_p(t_1, t_2)$  := la richiesta totale del processore durante l'intervallo  $[t_1, t_2]$

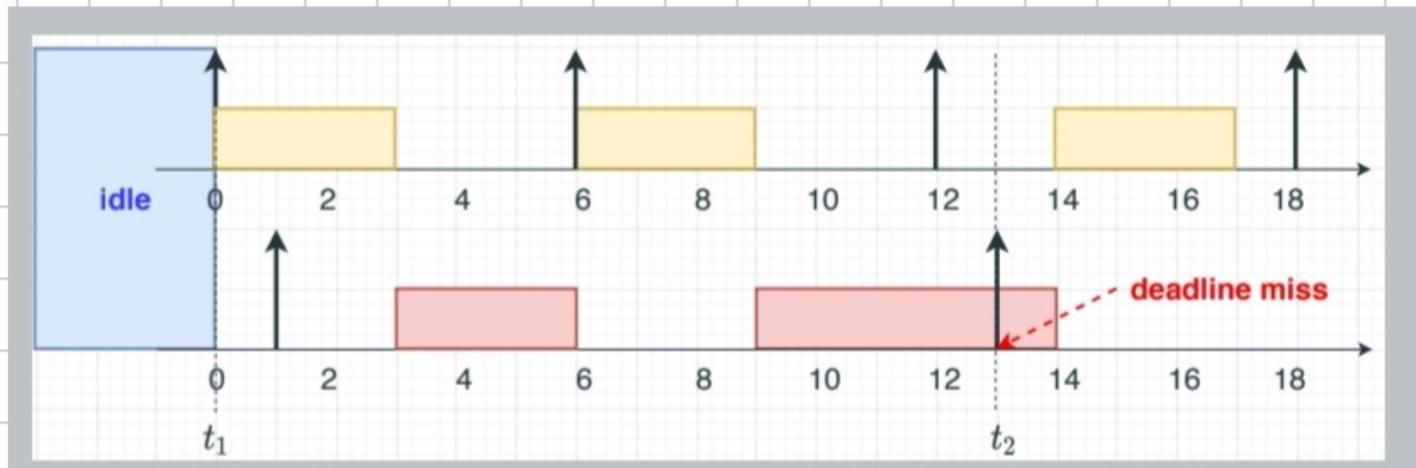
$$\Rightarrow C_p(t_1, t_2) = \sum_{\forall i, u | d_{i,u} \geq t_1 \wedge d_{i,u} \leq t_2} C_i = \sum_{i=1}^n \left\lfloor \frac{t_2 - t_1}{T_i} \right\rfloor C_i \leq \sum_{i=1}^n \frac{t_2 - t_1}{T_i} C_i$$

$\rightarrow \leq 1$  per hp

$$= (t_2 - t_1) U$$

ma  $C_p(t_1, t_2) > t_2 - t_1$  considerando che le deadline viene mancata al tempo  $t_2$

$$\Rightarrow t_2 - t_1 \leq p(t_1, t_2) \leq (t_2 - t_1) \cup \dots \cup \square \text{ assurdo!}$$



## OTTIMIZZAZIONE DI EDF

EDF RISULTA OTTIMALE, PER QUESTI RAGIONI LA FEASIBILITY, MA TUTTI CON ACCORDI DI SCHEDULING SE:

- UNO SCHEDULE È FEASIBLE PER UN TASK-SET  $\Gamma$ , ALLORA LO SCHEDULE DI EDF È FEASIBLE PER  $\Gamma$
- LO SCHEDULE DI EDF NON È FEASIBLE PER UN TASK-SET  $\Gamma$  ALLORA NESSUN SCHEDULE È FEASIBLE PER  $\Gamma$

QUESTO RISULTATO È INDIPENDENTE DALLA PENSIERA

Dim:

CONSIDERIAMO UN SCHEDULE FEASIBLE  $\sigma$  PER UN TASK-SET  $\Gamma$  E LO DIVIDIAMO IN TIME SLICES DI UNA UNITÀ DI TEMPO. DEFINIAMO:

- $\delta(t)$ : il task in esecuzione durante il tempo  $[t, t+1]$
- $E(t)$ : l'indice del task con la deadline assoluta minore al tempo  $t$
- $t_E$ : il tempo  $\geq t$  al quale  $\gamma_{\sigma(t)}$  è eseguito per prima  
 $\hookrightarrow$  task con la deadline minore!
- $d_{\max}$ :  $\max_{i \in \{1, \dots, n\}} \{d_i\}$  ovvero la deadline assoluta maggiore

S) trasformo quindi lo schedule  $\sigma$  in uno schedule EDF:

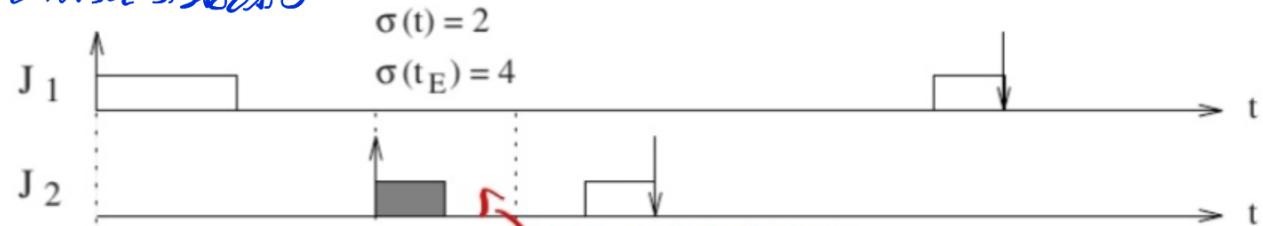
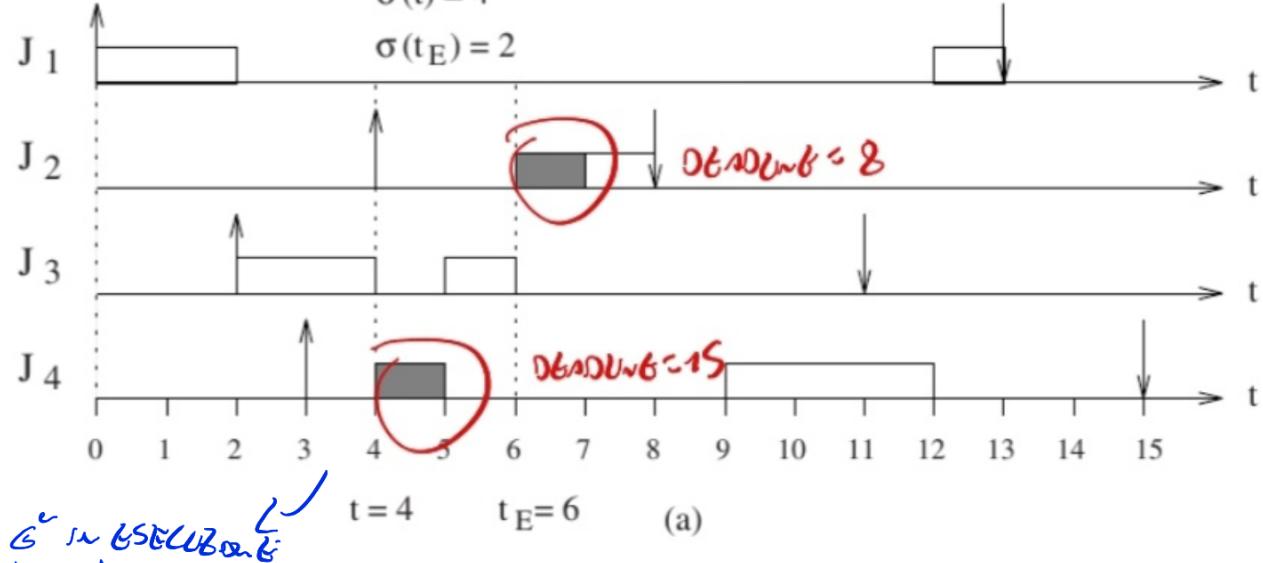
```

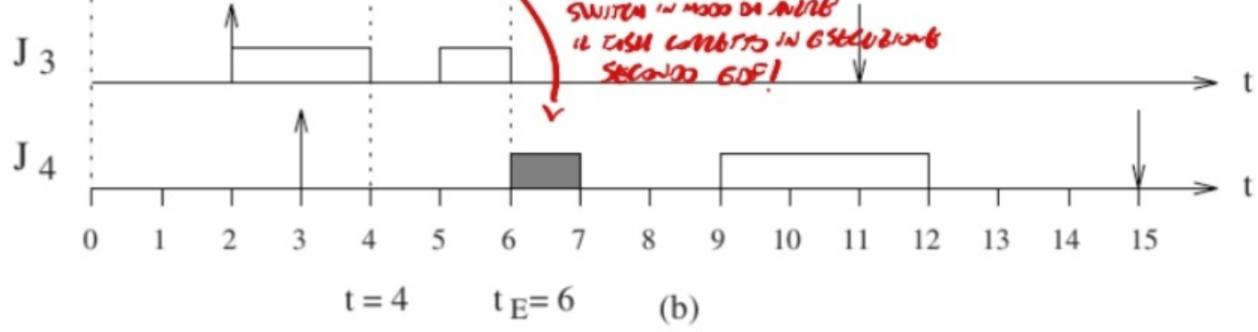
input : A feasible schedule  $\sigma$  for task set  $\Gamma$ 
output : An EDF schedule  $\sigma_{EDF}$  for task set  $\Gamma$ 

1 foreach  $t \in \{0, 1, \dots, d_{\max} - 1\}$  do
2   if  $\sigma(t) \neq \sigma_{EDF}(t)$  then
3      $\sigma(t_E) = \sigma(t)$  //  $[t_E, t_E + 1)$  allocated to the task executed during  $[t, t + 1]$ 
4      $\sigma(t) = E(t)$  //  $[t, t + 1)$  allocated to the task with min absolute deadline at  $t$ 
5   end
6 end
7 return TRUE
    
```

INIZIO I DUE TASK IN NUOVO  
 DI INIZIO IN ESECUZIONE IL TASK CON LA  
 DEADLINE PIÙ VICINA

PUNTO S) VADO A SCAMBIARE I TASK SET IN MODO DA RENDERE  
 VERA LA CONDIZIONE CHE VENGANO DATI PRIORITY AL TASK CON LA DEADLINE  
 PIÙ VICINA:





IN PARTICOLARE SI HA CHE LA TRASPOSIZIONE DEI DUE TASI PRESERVA CARA PUR LA SCHEDULABILITÀ ANTEO  $t_E$  È FEASIBILE PUR POICHÉ:

- SE UN TIME SLICE DEL TASI  $T_i$  È ANTICIPATO ALLORA LA FEASIBILITY DI  $T_i$  È PRESERVATA
- SE UN TIME SLICE DEL TASI  $T_i$  È POSTICIPATO AL TEMPO  $t_E$ , ALLORA  $t_E + 1 \leq d_E :=$  LA PRIMA DEADLINE ASSOLUTA PER  $T_i$  POICHÉ È FEASIBILE  
 $\Rightarrow t_E + 1 \leq d_E \leq d_i$  TASI  $T_i$  PER DEFINIZIONE DI  $d_E \Rightarrow$  IL TIME SLICE POSTICIPATO AL TEMPO  $t_E$  È COMUNQUE SCHEDULABILE. □

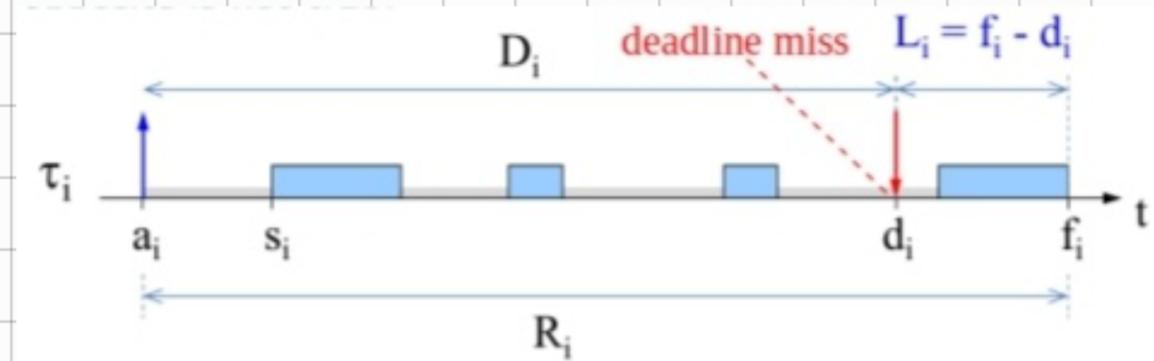
**OTTIMIZZAZIONE DI EDF MINIMIZZANDO LA MAX LATENESS**

DATO UN INSERTE DI  $m$  TASI INDEPENDENTI, CIASCU ALGORITMO CHE ESEGUE I TASI IN ORDINE DI DECRESSENZA DI DEADLINES ASSOLUTE È OTTIME NISPETTO AL MINIMIZZARE LA MASSIMA LATENESS.

$$L_{\max} := \max_{i \in \{1, \dots, m\}} \{L_i\}$$

PUNTO) SE UN CERTO ALGORITMO MINIMIZZA  $L_{\max}$ , ALLORA È OTTIMALE PER

PUNTO NCESSO LA FEASIBILITY, MA L'OPPOSTO NON VALE



D.m:

Consideriamo lo stesso programma utilizzato per la dimostrazione di ottimale di GDF, ovvero la trasposizione dei task.

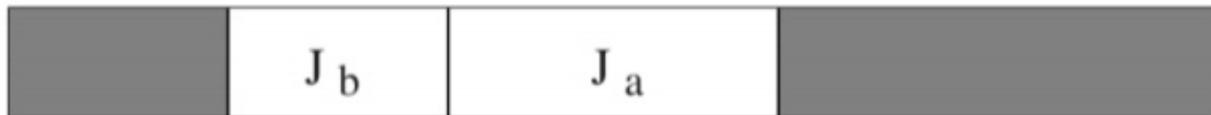
Quindi la trasposizione tra due succ  $\sum_a \in \sum_b$  non fa aumentare  $L_{\max}$ .

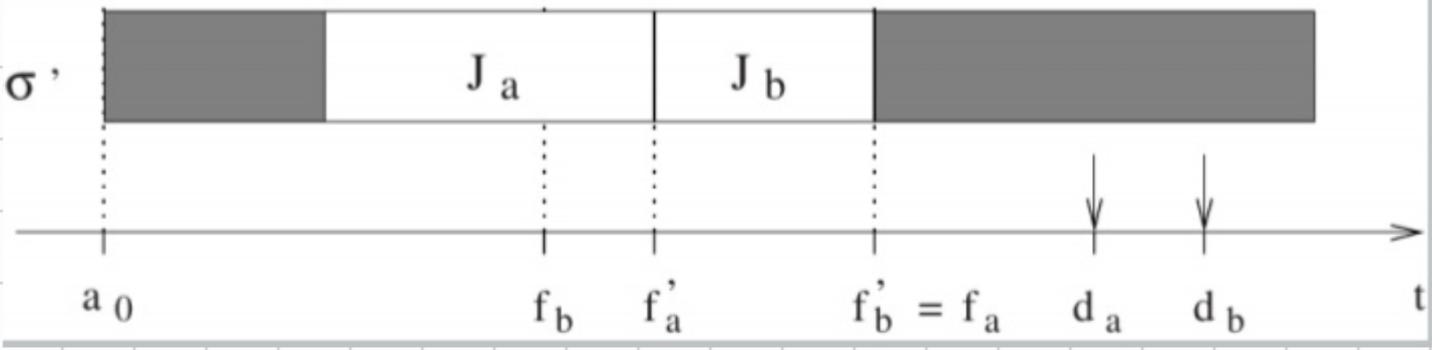
- Se  $\sum_a$  anticipato ( $f'_a < f_a$ ) e  $\sum_b$  posticipato ( $f'_b > f_b$ )
- Se  $L'_a > L'_b \Rightarrow L'_{\max} = L'_a = f'_a - d_a < f_a - d_a = L_{\max}$  → perché  $f'_a < f_a$
- Se  $L'_a \leq L'_b \Rightarrow L'_{\max} = L'_b = f'_b - d_b = f_a - d_b < f_a - d_a = L_{\max}$  → perché  $d_a < d_b$

punto con un numero finito di trasposizioni si ha che il più grande trasformato in GDF è, dove che  $L_{\max}$  non più aumentare, nello GDF ottimo

□

σ





## CALCULO DI RICHIESTA DEL PROCESSORE

UN INSIEME DI M MASSI PERIODICO  $\{T_1, \dots, T_n\}$  SONO CON  $D_i \leq T_i \forall i$   
 È SODDISFAZIONE SECONDO EDF  $\Leftrightarrow$  IN OGNI INTERVALLO DI TEMPO  $\{t_1, t_2\}$  LA  
 RICHIESTA DEL PROCESSORE  $g(t_1, t_2)$  NON SUPERVA IL TEMPO DISPONIBILE:

$$g(t_1, t_2) \leq t_2 - t_1 \text{ con } t_1 < t_2$$

Dove la richiesta del processore all'intervallo  $[t_1, t_2]$  è il tempo  
 di calcolo richiesto dai jobs arrivati in  $[t_1, t_2]$  con DENSITÀ  
 ASSOLUTA  $\leq t_2$

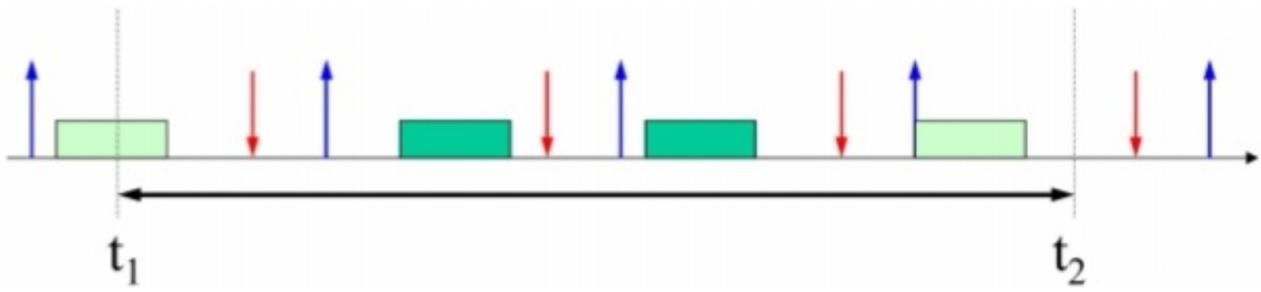
$$g(t_1, t_2) = \sum_{i=1}^m \eta_i(t_1, t_2) C_i$$

Dove  $\eta_i(t_1, t_2)$  è il numero di jobs di  $T_i$  che contribuiscono  
 alla richiesta del processore in  $[t_1, t_2]$

$$\eta_i(t_1, t_2) := |\{T_i, u | \alpha_i, u \in [t_1, t_2] \wedge d_i, u \leq t_2\}| = \max \{0, n_2^i - n_1^i\}$$

$$n_2^i := |\{T_i, u | \alpha_i, u \in [\phi_i, t_2] \wedge d_i, u \leq t_2\}| = \lfloor (t_2 + t_1 - D_i - \phi_i) / T_i \rfloor$$

$$n_1^i := |\{T_i, u | \alpha_i, u \in [\phi_i, t_1]\}| = \lceil t_1 - \phi_i / T_i \rceil$$



Si considera il worst case scenario dove **TUTTI i TASK SONO ATTIVATI AL TEMPO 0** ovvero  $\phi_i = 0 \forall \varphi_i$ :

$$df(t) := g(0, t) = \sum_{i=1}^m \eta_i(0, t) c_i = \sum_{i=1}^m \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor c_i$$

quindi un insieme sincronizzabile di m task periodici  $\{T_1, \dots, T_m\}$  con  $D_i \leq T_i \forall T_i$  è schedutabile secondo EDF  $\Leftrightarrow df(t) \leq t \forall t$

al fine di controllare la complessità delle procedure si ha che:

- ① Essendo un insieme di task periodici sincroni si verifica il criterio scattivo per  $t \leq H$  dove  $H$  è l'iperperiodo
- ②  $df(t)$  è una funzione **gradino** che aumenta quando  $t$  cresce di un periodo assoluto  $\Rightarrow$  se  $df(t) < t$  per  $t = d_i$  allora  $df(t) < t \mid d_i < t < d_{i+1}$   
perciò si verifica il criterio solo per i modi di  $t$  che sono coincidenti con le densità assolute.
- ③ Si verifica il criterio almeno fino a quando  $d_{max} := \max_i \{d_i\} \leq H$

④ Si cerca  $dbf(t)$  const:

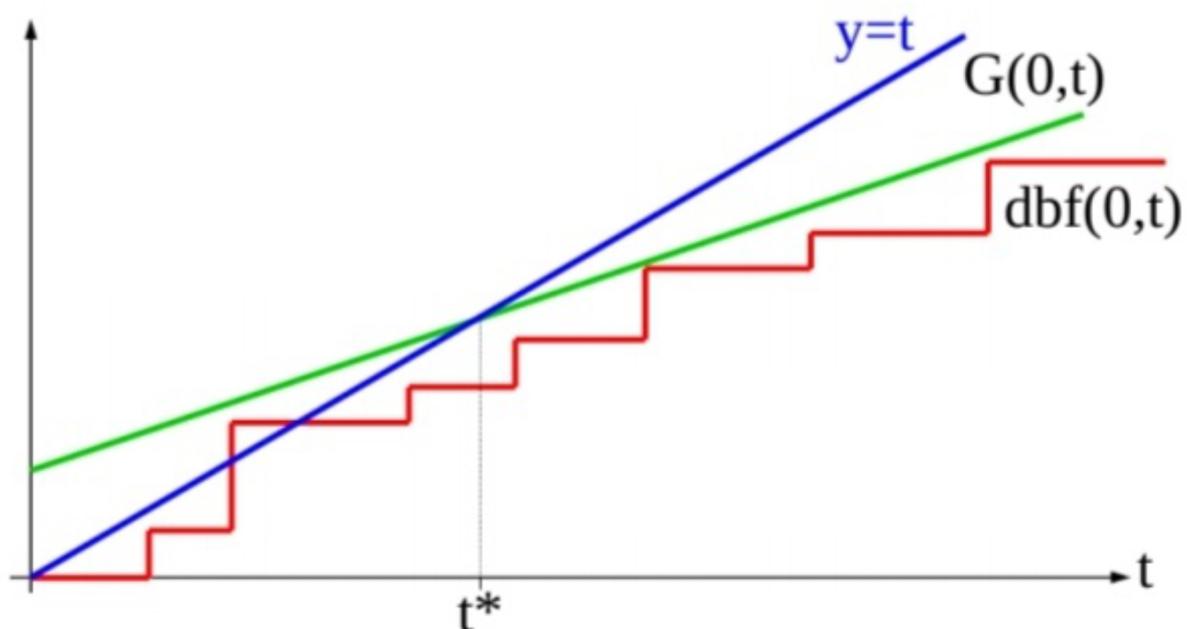
$$dbf(t) = \sum_{i=1}^m \left\lfloor \frac{t+T_i-D_i}{T_i} \right\rfloor C_i \leq \sum_{i=1}^m \frac{t+T_i-D_i}{T_i} C_i = \sum_{i=1}^m (T_i - D_i) U_i + tU$$

$\Rightarrow G(0, t) := \sum_{i=1}^m (T_i - D_i) U_i + tU$  è una funzione crescente con  
pendente  $U$

$\Rightarrow$  Se ci sono  $\exists t^* | G(0, t^*) = t^*$  dove:

$$t^* = \sum_{i=1}^m (T_i - D_i) U_i / (1-U)$$

$\Rightarrow dbf(t) \leq G(0, t) \leq t \quad \forall t \geq t^* \Rightarrow$  verifica il criterio solo per  $t \geq t^*$



**RECAP**: const si costruisce la complessità:

① si verifica il criterio solo per  $t \leq H$

② si verifica il criterio solo per valori di  $t$  uguali a  $H$

## DEDICATI ASSOLUTI

- ③ SI VERIFICA IL CITERNO NELLO PIÙ A GRADO  $d_{\max} \{ d_i \} \leq H$
- ④ SI VERIFICA IL CITERNO SOLO PER  $t \leq t^*$

quando il test della richiesta del processore  $G$ :

un insieme di  $n$  task periodico sincronizzati con  $D_i \leq T_i + T_i$   
è schedutabile secondo EDF  $\Leftrightarrow U < 1$  e  $\Delta_{\text{bf}}(t) \leq t$   
 $\Delta_{\text{bf}}(t)$  dove  $D = \{ d_i | d_i \leq \min \{ d_{\max}, t^* \} \}$  e  $t^* = \sum_{i=1}^n (T_i - D_i) U_i / (1 - U)$

## RM vs EDF

### SCHEDULING RM:

- RISULTA MENO EFFICIENTE ( $U < 1$ )

- È PIÙ FACILE DA IMPLEMENTARE

- Ha un comportamento più prevedibile quando ci si trova in situazioni di overloading, ma tuttavia se ho che i task a basso prioreti sono spesso bloccati

### SCHEDULING EDF:

- È più efficiente perché  $U_{WB} = 1$
  - offre un numero di prelazioni minore e quindi un minor overhead
  - è più responsive nella gestione dei task appena arriva perché distribuisce meglio l'uso del processore.
- 

### RASSIATO SCHEDULING TASK PERIODICI

NE ABBIAMO VISTI TRE TIPI:

- OFFLINE (TIMEOUT SCHEDULING)
- ONLINE STATIC PRIORITY (RM, SJN)
- ONLINE DYNAMIC PRIORITY (GDF)

E ABBIAMO VISTO TUTTI I REQUISITI DI ANALISI DELLA SCHEDULABILITÀ?

1)

### UTILIZATION BASED ANALYSIS

- LL BOUND PER RM  $\rightarrow U \leq n(2^{m-1})$  SUFFICIENTE
- HB BOUND PER RM  $\rightarrow \prod_{i=1}^n (U_i + 1) \leq 2$  SUFFICIENTE  
per risolvere algoritmo

• LL BOUND PER TASK-SET APPROXIMATIVO  $\rightarrow 1/c_1$  NECESSARIA E SUFFICIENTE

- EDF Bound  $\rightarrow$   $U \leq 1$  NECESSARY & SUFFICIENT

PRESGNTING TURS UNA COMPLESSITA' POLYNOMIALI NEL NUMERO  $n$

O) TASK  $\rightarrow$   $O(n)$

2)

### RESPONSE TIME ANALYSIS

- $R_i \leq D_i \quad \forall i \in \{1, \dots, n\}$  DOVE  $R_i = C_i + \sum_{u=1}^{i-1} \lceil R_u / T_u \rceil C_u$

NECESSARY & SUFFICIENT

CUT HA UNA COMPLESSITA' PSEUDO-POLYNOMIAL

3)

### PROCESSOR DEMAND ANALYSIS

- $df(t) \leq t + t_0 D$  NECESSARY & SUFFICIENT

CUT HA UNA COMPLESSITA' PSEUDO-POLYNOMIAL

	$D_i = T_i \quad \forall i \in \{1, \dots, n\}$	$\exists i \in \{1, \dots, n\} \mid D_i < T_i$
RM	LL bound for harmonic task sets LL bound, HB bound response time analysis	response time analysis
EDF	EDF bound	processor demand approach



columns OBI TAKS



columns OBI TAKS

PUNTO DE PENDIENTE

SUSTITUIRLO EN LA ECUACION