SAPIENZA
UNIVERSITÀ DI ROMA

Algoritmi di scheduling
- Parte 3

Automazione

21/10/2015

Vincenzo Suraci

STRUTTURA DEL NUCLEO TEMATICO

- SERVIZIO IN BACKGROUND
- SERVER
- ESERCITAZIONE

ALGORITMO CON SERVIZIO IN BACKGROUND

SERVIZIO IN BACKGROUND

Dato un problema di scheduling (equivalente) di n di task periodici HARD REAL TIME:

- REQUISITI DI SISTEMA: (n, T_1, T_2, …, T_n)
- VINCOLI DI SISTEMA: (C_1, C_2, …, C_n)

e di m task aperiodici SOFT REAL TIME.

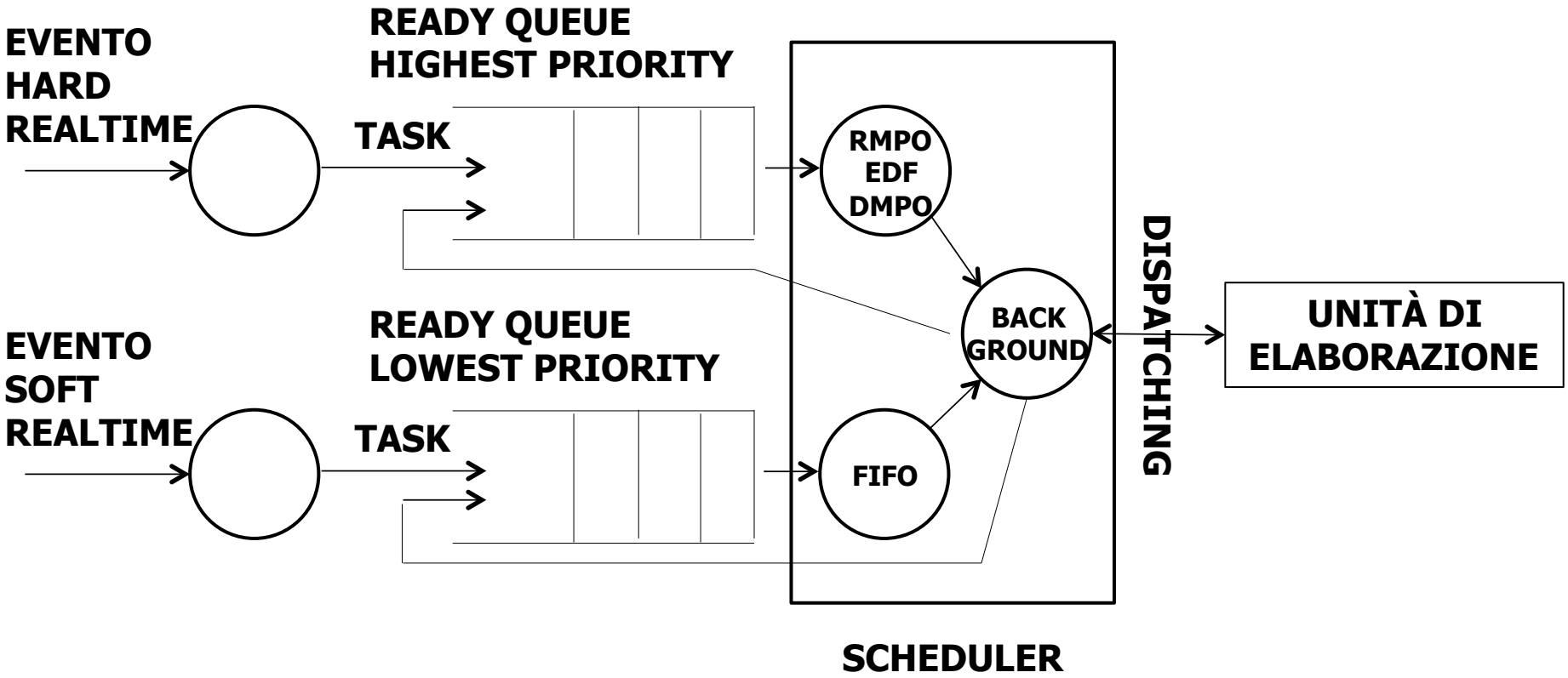
Lo scheduling di task misti tramite **SERVIZIO IN BACKGROUND** esegue i task aperiodici solo negli istanti di tempo in cui l'unità di elaborazione è libera e non ci sono task periodici hard real time da eseguire.

Lo scheduling dei task periodici è pertanto realizzabile con uno qualsiasi degli algoritmi visti in precedenza (RMPO, EDF, DMPO).

Lo scheduling dei task aperiodici è in generale realizzato attraverso una metodologia FIFO, abilitata ogni qualvolta il processore è libero da task hard real time.

SERVIZIO IN BACKGROUND

Lo schema funzionale di uno scheduling che attua SERVIZIO IN BACKGROUND è:



ESEMPIO

PROBLEMA

Dato il problema di scheduling di task misti con 2 task periodici HARD REAL TIME:

- REQUISITI DI SISTEMA: (2, T_1 = 8 t.u. , T_2 = 16 t.u.)
- VINCOLI DI SISTEMA: (C_1 = 2 t.u. , C_2 = 4 t.u.)

1 task aperiodico HARD REAL TIME

- REQUISITI DI SISTEMA: (1, T_3^{min} = 32 t.u.)
- VINCOLI DI SISTEMA: (C_3 = 8 t.u.)

1 task aperiodico SOFT REAL TIME, la cui prima occorrenza è così caratterizzata:

- $a_4(1)$ = 9 t.u.
- $C_4(1)$ = 9 t.u.
- $D_4(1)$ = 55 t.u.

Mostrare la schedulabilità con SERVIZIO IN BACKGROUND, dove:

- ALGORITMO TASK HARD REAL TIME = RMPO
- ALGORITMO TASK SOFT REAL TIME = FIFO

ESEMPIO cont'd

SVOLGIMENTO

Innanzitutto consideriamo il problema di scheduling di task equivalenti:

- REQUISITI DI SISTEMA: (3, T_1 = 8 t.u. , T_2 = 16 t.u. , T_3 = 32 t.u.)
- VINCOLI DI SISTEMA: (C_1 = 2 t.u. , C_2 = 4 t.u. , C_3 = 8 t.u.)

Quindi verifichiamo che NON sia inammissibile:

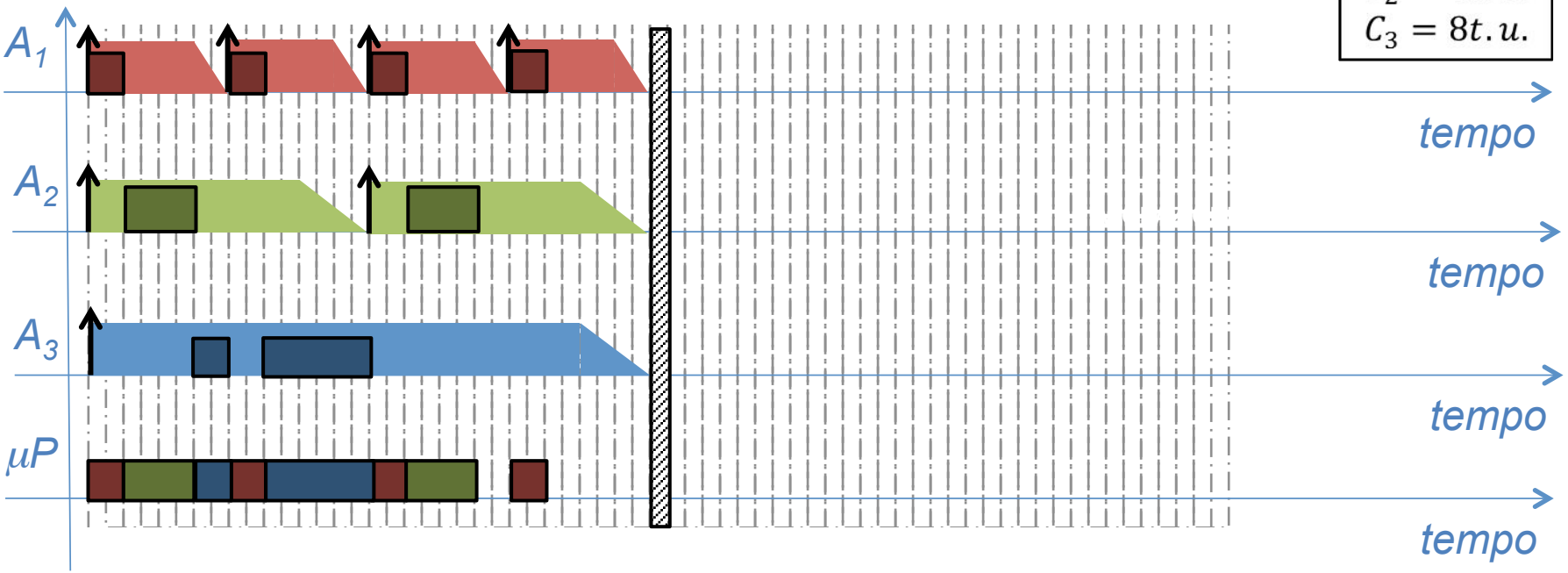
$$U = \sum_{i=1}^{3} \frac{C_i}{T_i} = \frac{2}{8} + \frac{4}{16} + \frac{8}{32} = \frac{8+8+8}{32} = \frac{24}{32} = 0{,}75 < 1$$

Notiamo inoltre che i tre periodi sono legati tra loro da relazioni armoniche, pertanto la condizione U = 0,75 < 1 diventa anche sufficiente a garantire la schedulabilità del problema equivalente.

ESEMPIO cont'd

Disegnamo la soluzione del problema di scheduling con RMPO:

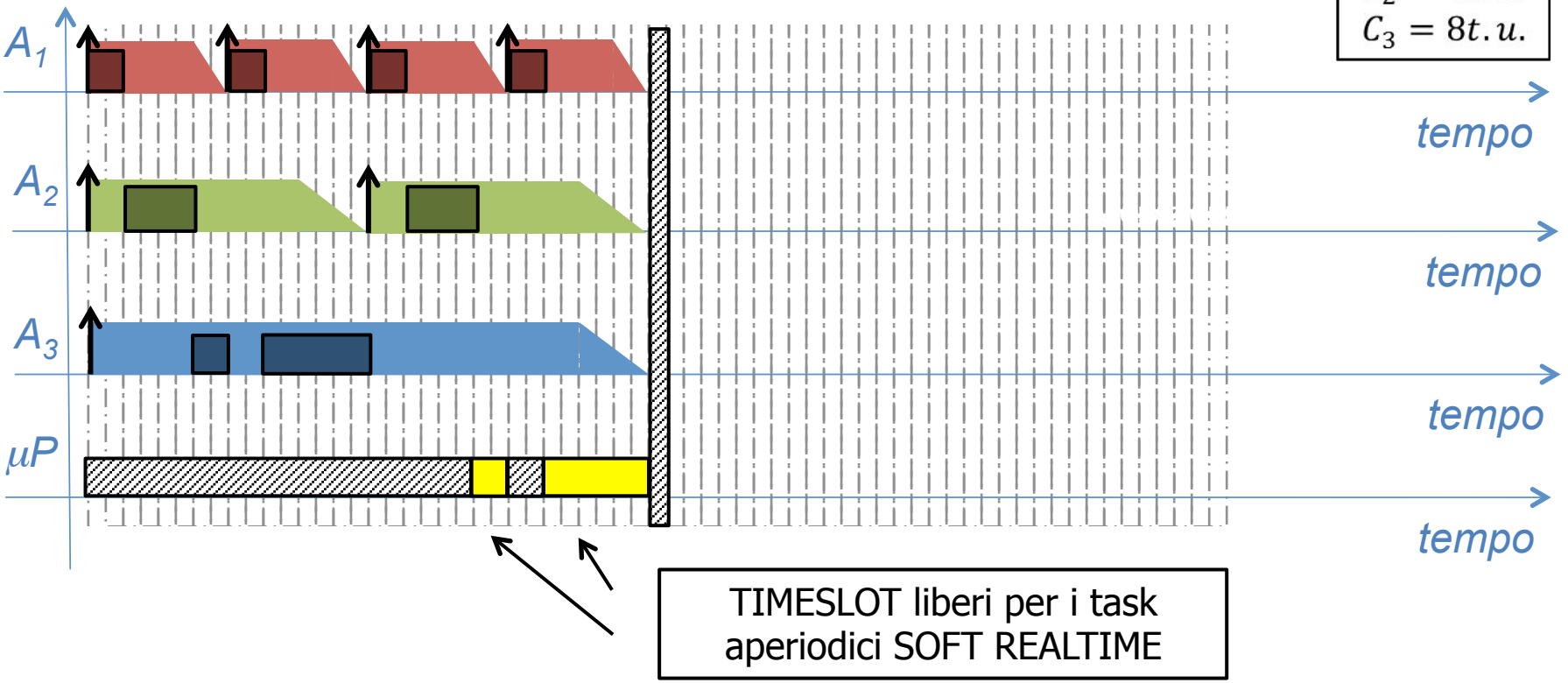
$C_1 = 2t.u.$
$C_2 = 4t.u.$
$C_3 = 8t.u.$



ESEMPIO cont'd

Evidenziamo i TIMESLOT in cui il processore è libero per i task aperiodici SOFT REALTIME

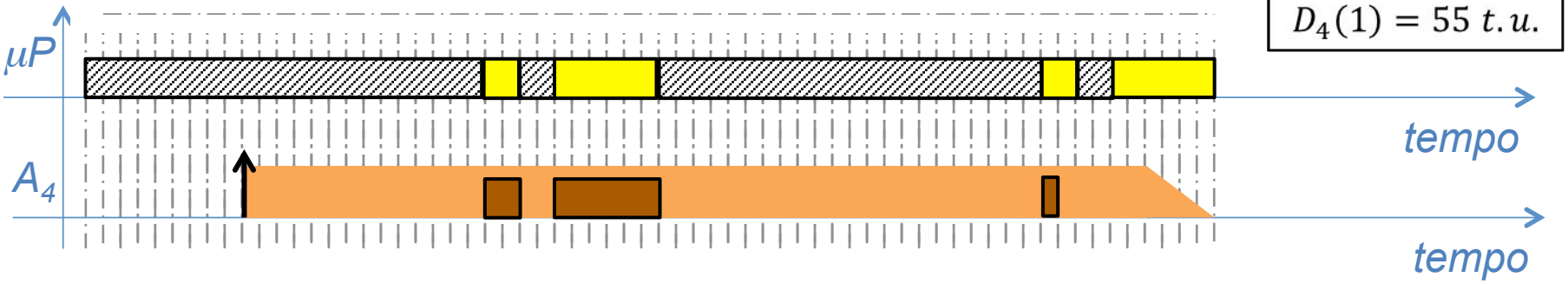
$C_1 = 2t.u.$
$C_2 = 4t.u.$
$C_3 = 8t.u.$



ESEMPIO cont'd

Nei TIMESLOT in cui il processore è libero si può eseguire l'occorrenza del task A_4.

$$a_4(1) = 9\ t.u.$$
$$C_4(1) = 9\ t.u.$$
$$D_4(1) = 55\ t.u.$$



SERVIZIO IN BACKGROUND

Con questa tipologia di scheduling i task APERIODICI vengono eseguiti con la MINOR PRIORITÀ possibile.

È evidente pertanto che in condizioni in cui il fattore di utilizzo dei task periodici è prossimo all'unità, il **tempo di esecuzione** dei task aperiodici è estremamente **limitato** e il **response time molto elevato**.

Algoritmi di scheduling più flessibili sono quelli ottenibili tramite il ricorso ad un processo server.

ALGORITMI TRAMITE SERVER

PROCESSO SERVER

DEFINIZIONE

Si definisce PROCESSO SERVER un task periodico caratterizzato da:

- Periodo di attivazione T_{srv};
- Computation time C_{srv};

SERVER

Dato un problema di scheduling (equivalente) di n di task periodici HARD REAL TIME:

- REQUISITI DI SISTEMA: (n, T_1, T_2, …, T_n);
- VINCOLI DI SISTEMA: (C_1, C_2, …, C_n);

dati m task aperiodici SOFT REAL TIME

dato un processo server

Lo scheduling di task misti tramite **SERVER** esegue i task aperiodici solo negli istanti di tempo in cui il processo server è in esecuzione.

ALGORITMO POLLING SERVER

POLLING SERVER

Il servizio POLLING SERVER è caratterizzato da un PROCESSO SERVER il cui computation time C_{srv} dipende dai task aperiodici in attesa di essere eseguiti.

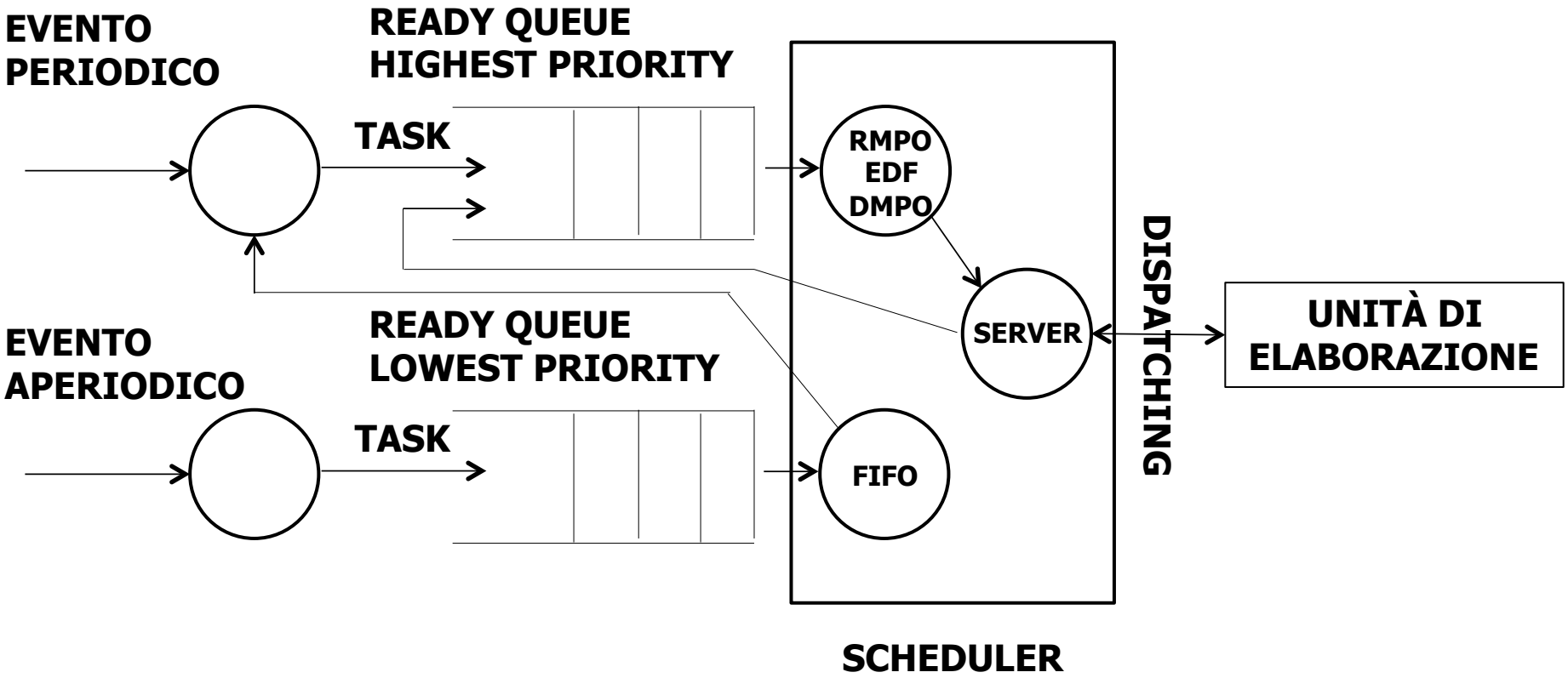
Quando un nuovo task del processo server è attivato, il suo computation time è calcolato **sommando i computation time** dei task aperiodici in attesa di essere eseguiti.

Se la somma supera la **capacità massima del server** (C_{srv}^{MAX}) il computation time viene posto pari alla capacità massima del server ($C_{srv} = C_{srv}^{MAX}$).

Se non sono presenti task aperiodici in coda, quando un nuovo task del processo server è attivato, il suo computation time è posto pari a zero ($C_{srv} = 0$).

POLLING SERVER

Lo schema funzionale di uno scheduling di task misti con server è:



ESEMPIO

PROBLEMA

Dato il problema di scheduling di task misti con 2 task periodici HARD REAL TIME

- REQUISITI DI SISTEMA: (2, T_1 = 8 t.u. , T_2 = 16 t.u.);
- VINCOLI DI SISTEMA: (C_1 = 4 t.u. , C_2 = 2 t.u.);

ed un 1 task aperiodico SOFT REAL TIME, la cui prima occorrenza è così caratterizzata:

- $a_3(1)$ = 14 t.u.
- $C_3(1)$ = 5 t.u.
- $d_3(1)$ = 47 t.u.

Mostrare la schedulabilità con POLLING SERVER, dove:

- ALGORITMO TASK HARD REAL TIME = RMPO;
- PROCESSO SERVER con $T_{srv} = 12$ t. u. e $C_{srv}^{MAX} = 3\ t.u.$;

ESEMPIO cont'd

SVOLGIMENTO

Verifichiamo che il problema dato con TASK PERIODICI NON sia inammissibile:

$$U = \sum_{i=1}^{2} \frac{C_i}{T_i} = \frac{4}{8} + \frac{2}{16} = \frac{8+2}{16} = \frac{10}{16} = 0{,}625 < 1$$

Essendo U ≤ 1 il problema NON È INAMMISSIBILE.

Notiamo che essendo U = 0,625 < 0,693 < U_{lsm}(RMPO) è verificata la CONDIZIONE SUFFICIENTE di schedulabiltà del problema.

Notiamo inoltre che i due periodi sono legati tra loro da relazioni armoniche, pertanto la condizione U = 0,75 < 1 diventa anche sufficiente a garantire la schedulabilità del problema equivalente.

ESEMPIO cont'd

L'aggiunta del PROCESSO SERVER implica un diverso fattore di utilizzazione:

$$U = \frac{C_{srv}^{MAX}}{T_{srv}} + \sum_{i=1}^{2} \frac{C_i}{T_i} = \frac{3}{12} + 0{,}625 = 0{,}875 < 1$$

Essendo U ≤ 1 il problema NON È INAMMISSIBILE.

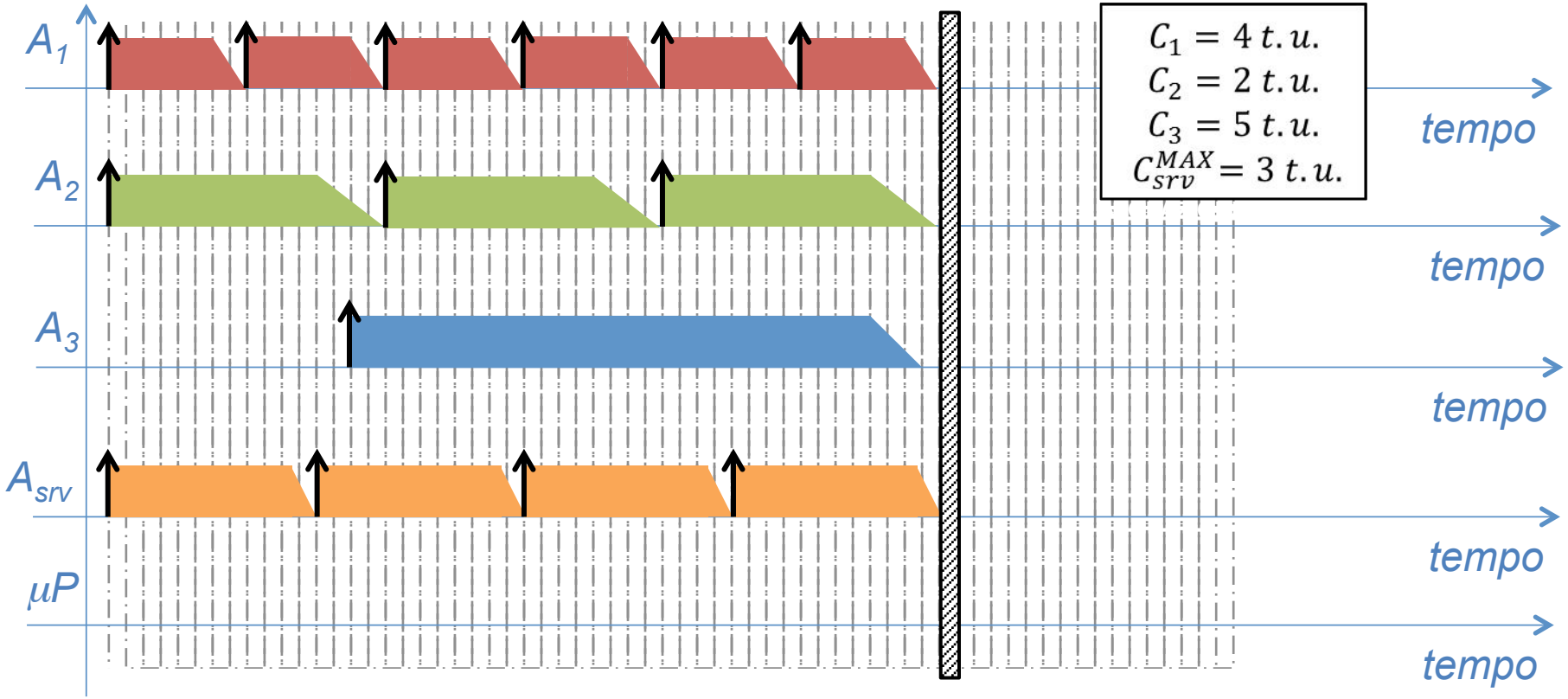
Notiamo che:

$$U_{lsm}(RMPO) = 3\left(2^{1/3} - 1\right) \approx 0{,}78$$

Pertanto NON è verificata la condizione sufficiente di schedulabilità del problema quando si considera il PROCESSO SERVER al massimo della sua capacità.

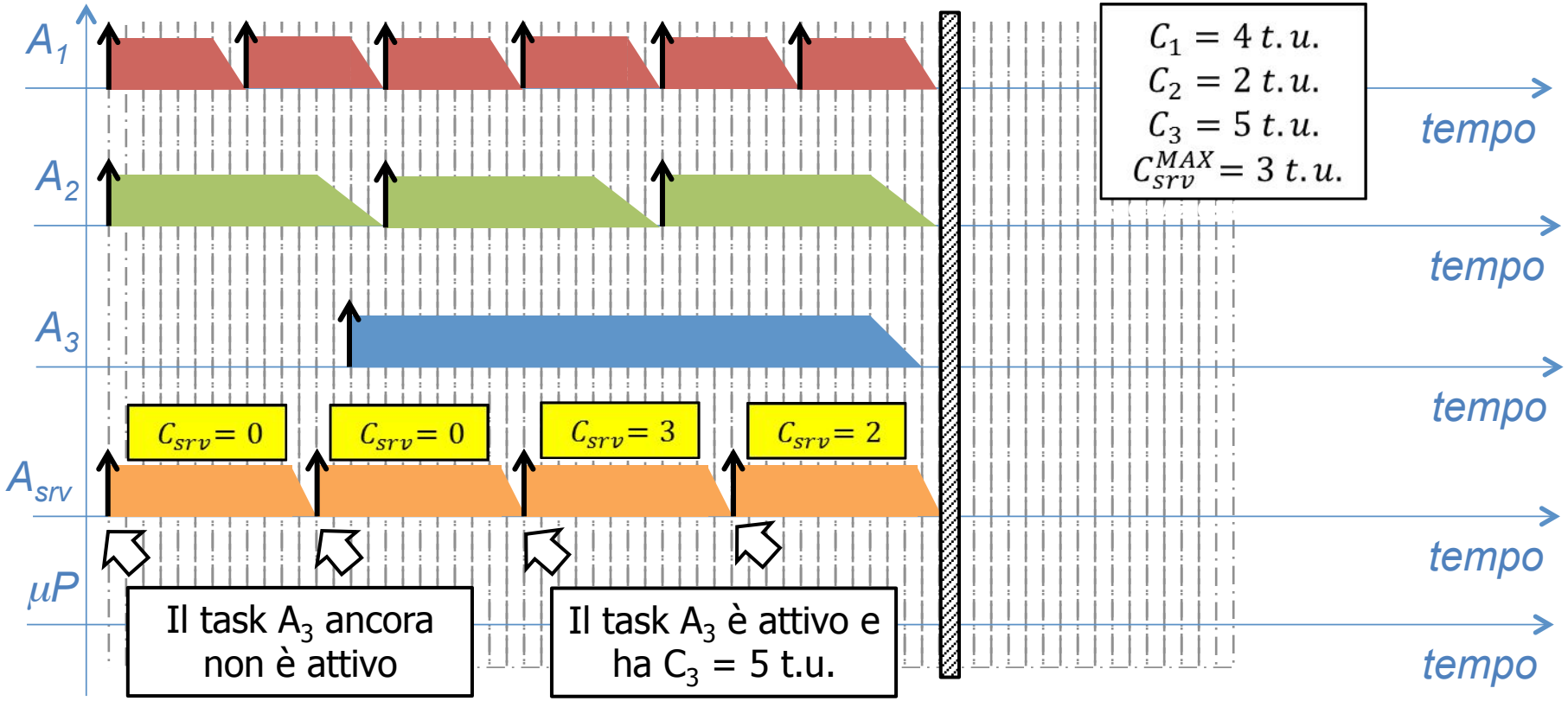
ESEMPIO cont'd

Disegniamo la soluzione del problema di scheduling con RMPO, ricordando che le priorità dei task sono ordinate in base all'inverso del periodo di attivazione: A_1, A_{srv} e A_3.



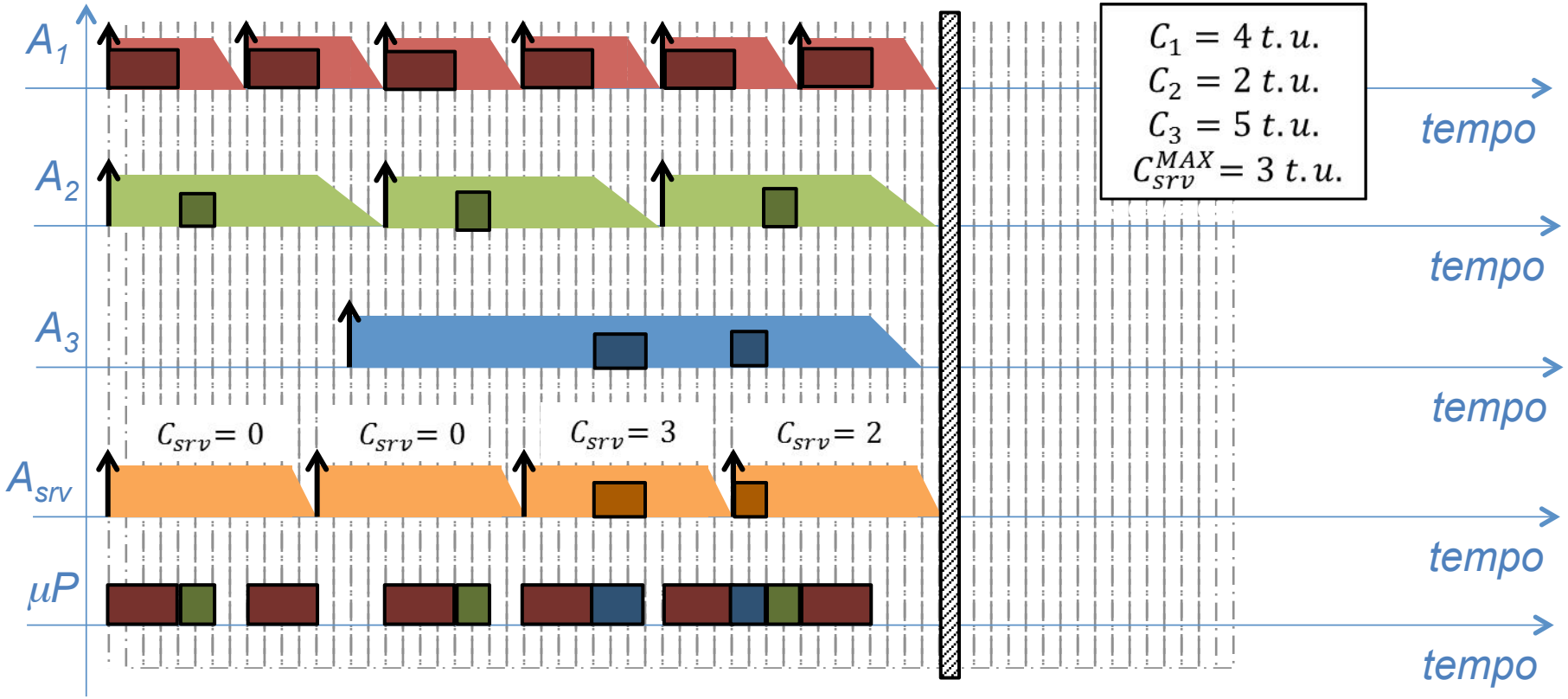
ESEMPIO cont'd

Possiamo associare alle occorrenze del PROCESSO SERVER i computation time, ricordando che non possiamo superare 3 t.u. per occorrenza.



ESEMPIO cont'd

Note priorità e computation time, si completa lo scheduling applicando le regole dell'algoritmo RMPO.



POLLING SERVER

OSSERVAZIONE

È importante notare che variando opportunamente la CAPACITÀ MASSIMA del server (C_{srv}^{MAX}) si può aumentare o diminuire a piacere la priorità del PROCESSO SERVER rispetto agli altri task.

Pertanto il POLLING SERVER permette di privilegiare, quando necessario, la prontezza di risposta dei processi aperiodici.

ALGORITMO DEFERRING SERVER

DEFERRABLE SERVER

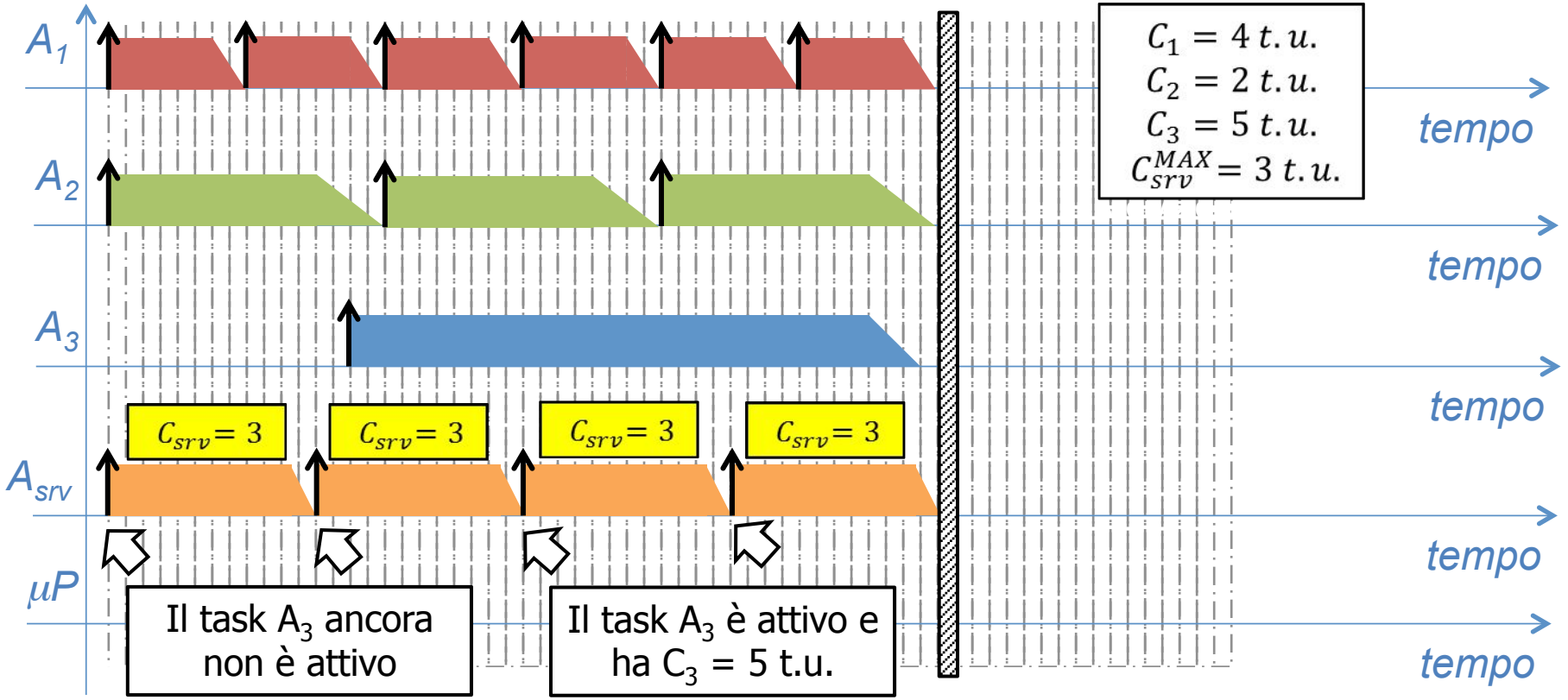
Il servizio DEFERRABLE SERVER è caratterizzato da un PROCESSO SERVER il cui computation time C_{srv} NON DIPENDE dai task aperiodici in attesa di essere eseguiti.

Il computation time viene posto pari alla capacità massima del server ($C_{srv} = C_{srv}^{MAX}$).

Il DEFERRABLE SERVER usa in maniera poco efficiente le risorse ma diminuisce il tempo di risposta dei processi aperiodici.

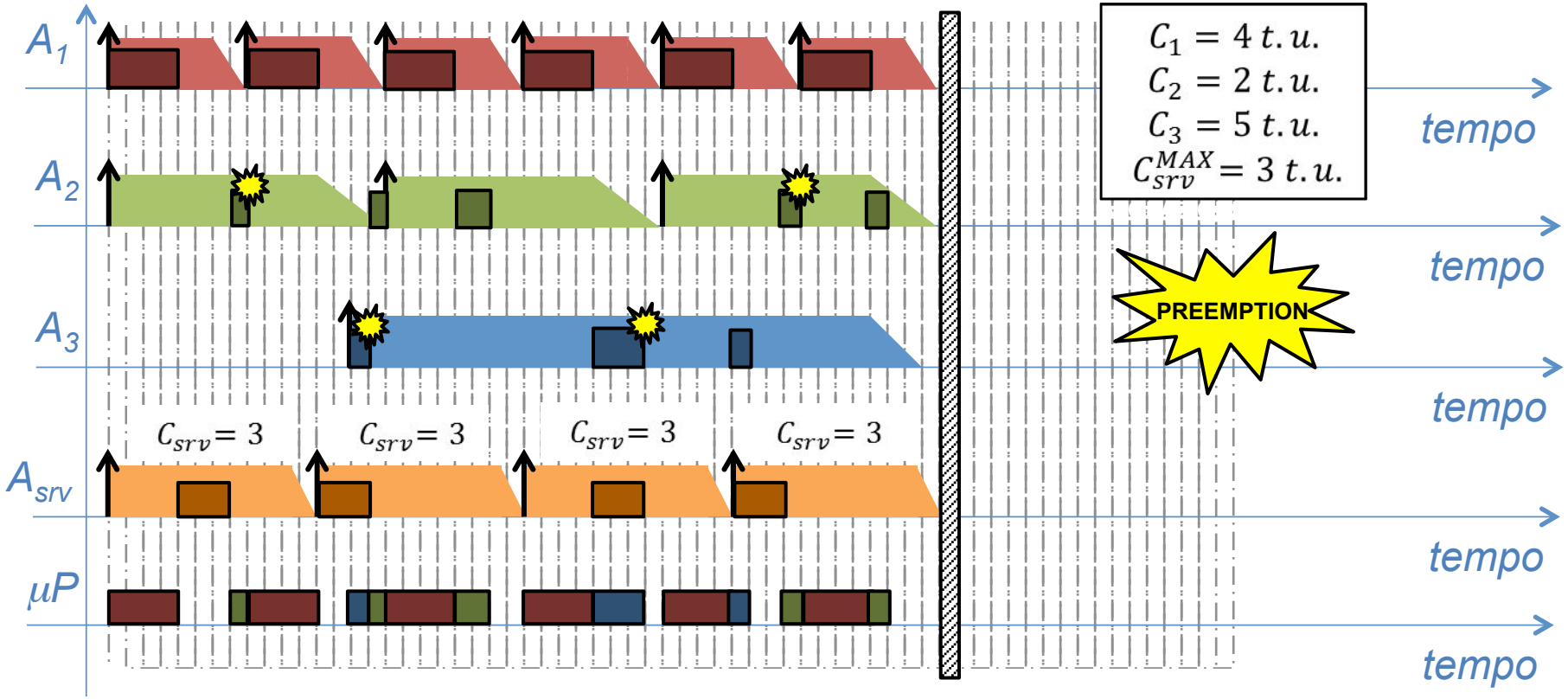
DEFERRABLE SERVER - ESEMPIO

Riprendendo l'esempio dato, vediamo come cambiano i computation time:



DEFERRABLE SERVER - ESEMPIO

Note priorità e computation time, si completa lo scheduling applicando le regole dell'algoritmo RMPO.



ESERCITAZIONE

ESERCIZIO 1

PROBLEMA

In un sistema di AUTOMAZIONE devono essere GARANTITI tre task:

1. A **livello di campo** è necessario garantire il processo di **lettura dei dati da sensore** che impiega 8 t.u. per essere elaborato e che deve essere ripetuto ogni 16 t.u.
2. A **livello di coordinamento** è necessario garantire un **task meccanico** che deve essere ripetuto ogni 48 t.u. e la cui elaborazione necessita 12 t.u.
3. A **livello di campo** è necessario garantire il processo di **elaborazione della legge di controllo** e di attuazione della relativa azione di intervento; tale processo impiega 6 t.u. per essere elaborato e deve essere ripetuto ogni 24 t.u.

Ipotizzando di volere un sistema di controllo HARD REAL TIME e di avere a disposizione di un sistema MONOPROCESSORE, calcolare il fattore di utilizzazione e tracciare il grafico dello scheduling dei task mediante **algoritmo EDF**.

ESERCIZIO 1 cont'd

SVOLGIMENTO

Il problema può essere tradotto in un problema di scheduling di task periodici dove:

- REQUISITI DI SISTEMA: ($n = 3$, $T_1 = 16$, $T_2 = 48$, $T_3 = 24$)
- VINCOLI DI SISTEMA: ($C_1 = 8$, $C_2 = 12$, $C_3 = 6$)

Come prima cosa è fondamentale verificare la condizione necessaria di schedulabilità:

$$U = \sum_{i=1}^{3} \frac{C_i}{T_i} = \frac{8}{16} + \frac{12}{48} + \frac{6}{24} = \frac{24 + 12 + 12}{48} = \frac{48}{48} = 1$$

Dato che U = 1 il problema NON È INAMMISSIBILE.

Inoltre, dato che utilizziamo l'algoritmo EDF, la condizione è anche sufficiente. Quindi il problema è SCHEDULABILE.

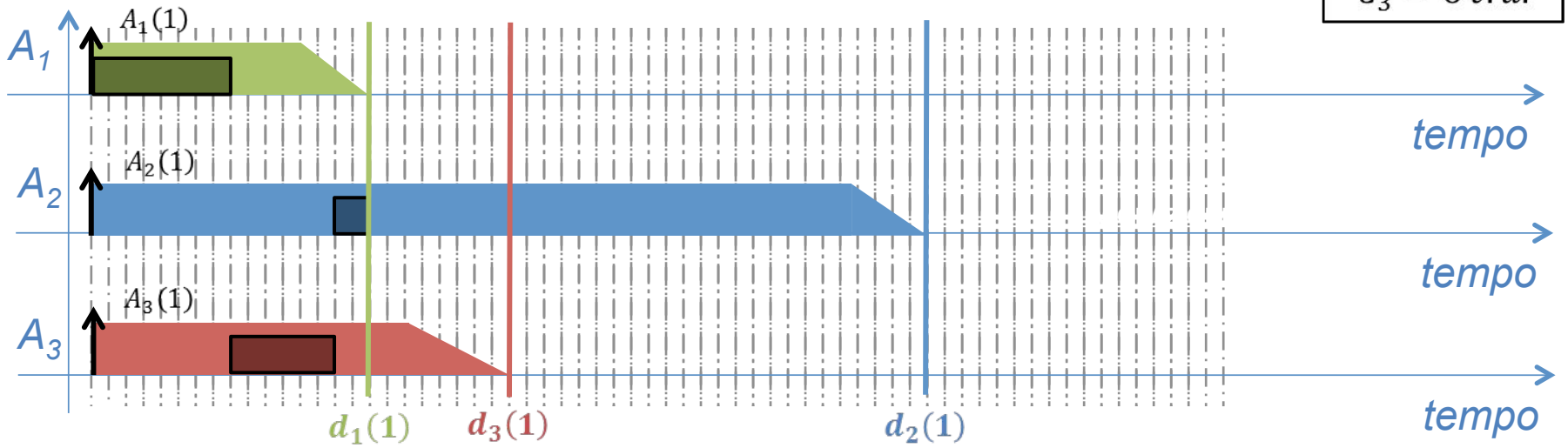
ESERCIZIO 1 cont'd

All'inizio si presentano nella READY QUEUE tre task attivi: $A_1(1)$, $A_2(1)$, $A_3(1)$. Le priorità dei tre task dipendono dal reciproco delle loro deadline assolute, pertanto:

PRIORITÀ

$$A_1(1) = 1/d_1(1) = 1/16 = 0{,}0625$$
$$A_3(1) = 1/d_3(1) = 1/24 = 0{,}041\overline{6}$$
$$A_2(1) = 1/d_2(1) = 1/48 = 0{,}0208\overline{3}$$

$C_1 = 8\ t.u.$
$C_2 = 12\ t.u.$
$C_3 = 6\ t.u.$



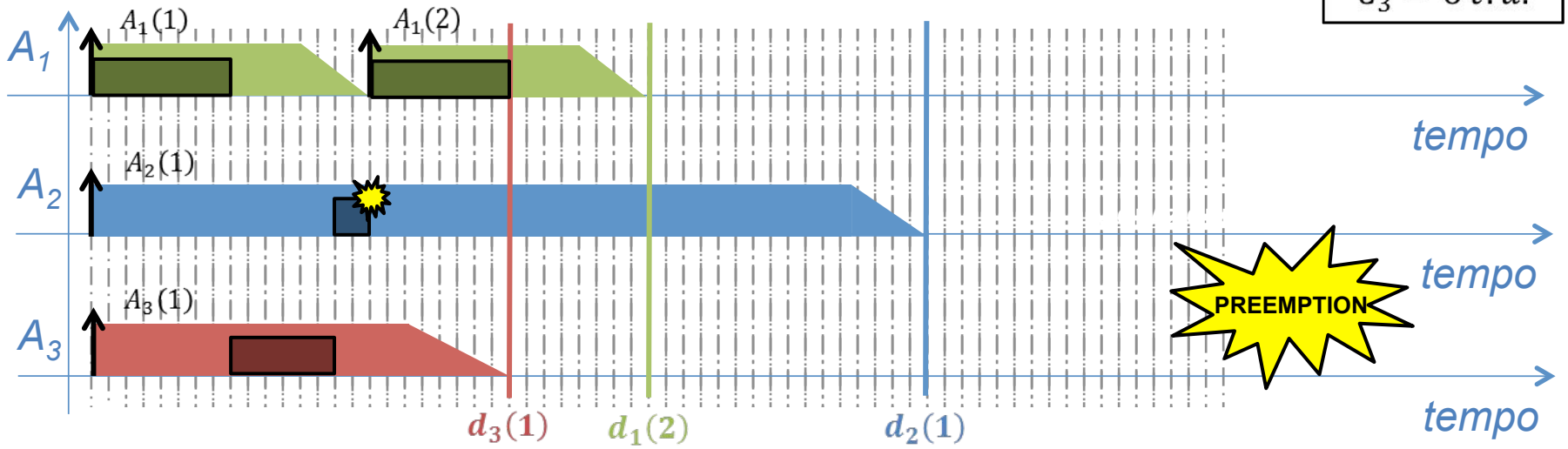
ESERCIZIO 1 cont'd

Con l'arrivo nella READY QUEUE del task $A_1(2)$, la priorità dei task attivi deve essere ricalcolata, ottenendo la seguente classifica di priorità:

PRIORITÀ

$$A_3(1) = 1/d_3(1) = 1/24 = 0{,}041\overline{6}$$
$$A_1(2) = 1/d_1(2) = 1/32 = 0{,}03125$$
$$A_2(1) = 1/d_2(1) = 1/48 = 0{,}0208\overline{3}$$

$C_1 = 8\ t.u.$
$C_2 = 12\ t.u.$
$C_3 = 6\ t.u.$



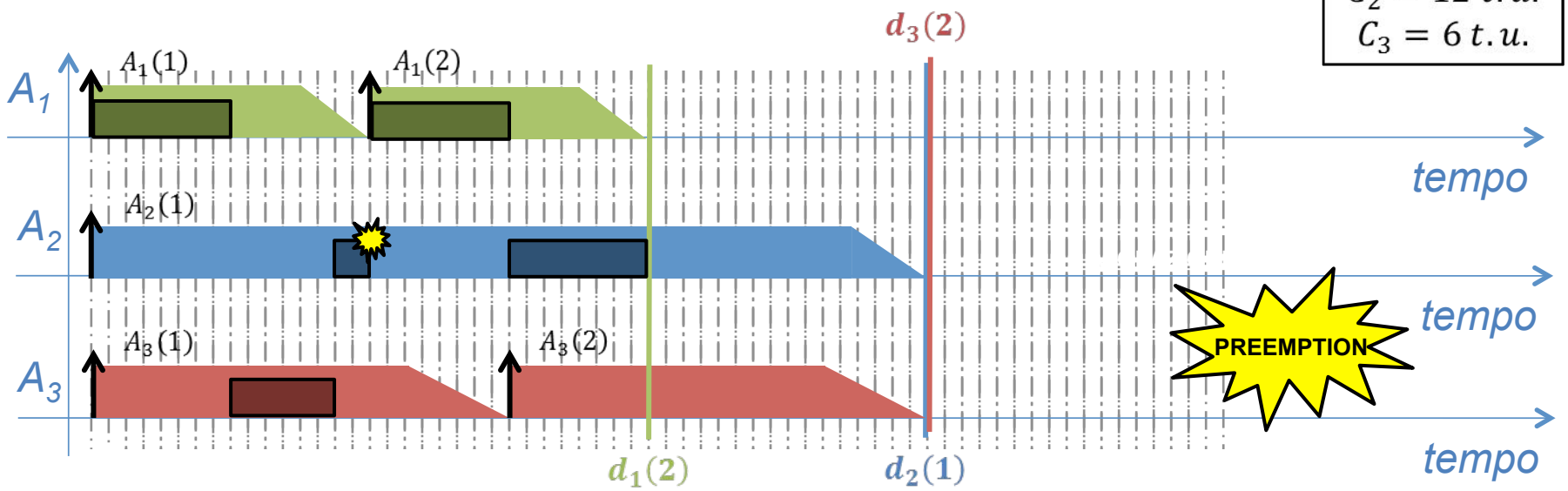
ESERCIZIO 1 cont'd

Con l'arrivo nella READY QUEUE del task $A_3(2)$, la priorità dei task attivi deve essere ricalcolata, ottenendo la seguente classifica di priorità:

PRIORITÀ

$$A_1(2) = 1/d_1(2) = 1/32 = 0{,}03125$$
$$A_2(1) = 1/d_2(1) = 1/48 = 0{,}0208\overline{3}$$
$$A_3(2) = 1/d_3(2) = 1/48 = 0{,}0208\overline{3}$$

$C_1 = 8\ t.u.$
$C_2 = 12\ t.u.$
$C_3 = 6\ t.u.$



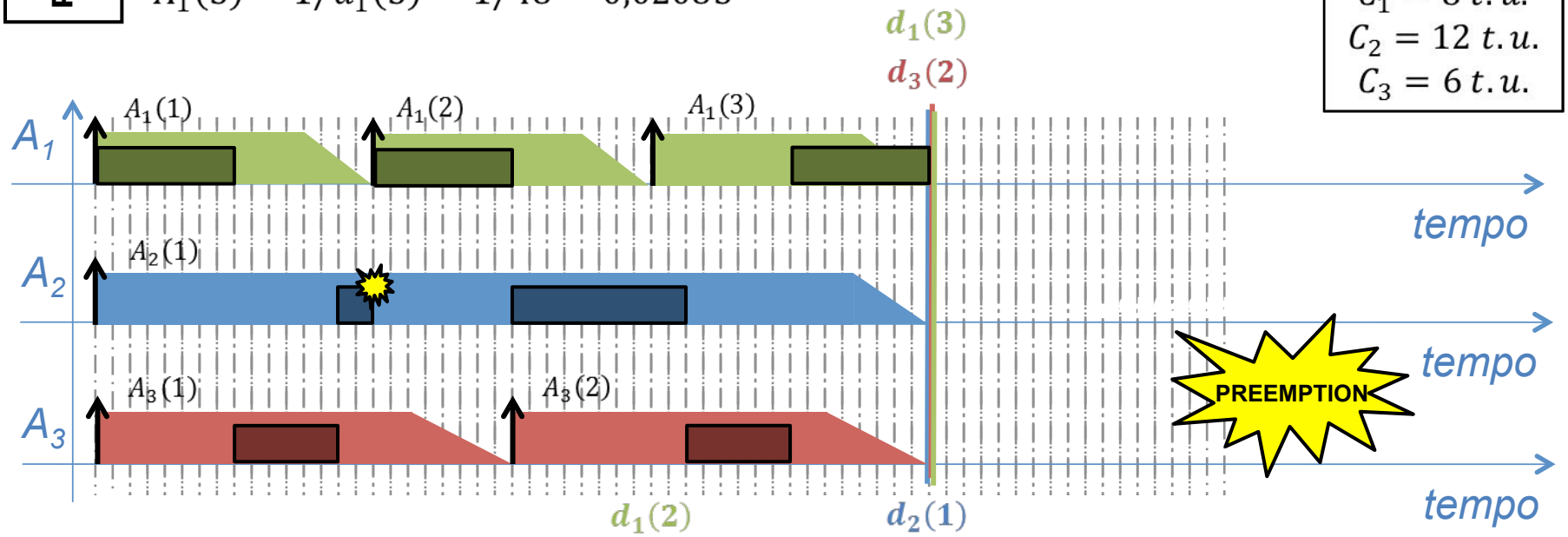
ESERCIZIO 1 cont'd

Con l'arrivo nella READY QUEUE del task $A_3(2)$, la priorità dei task attivi deve essere ricalcolata, ottenendo la seguente classifica di priorità:

PRIORITÀ

$$A_2(1) = 1/d_2(1) = 1/48 = 0{,}0208\overline{3}$$
$$A_3(2) = 1/d_3(2) = 1/48 = 0{,}0208\overline{3}$$
$$A_1(3) = 1/d_1(3) = 1/48 = 0{,}0208\overline{3}$$

$C_1 = 8\ t.u.$
$C_2 = 12\ t.u.$
$C_3 = 6\ t.u.$



ESERCIZIO 2

PROBLEMA

Una ditta agroalimentare deve progettare un sistema di AUTOMAZIONE di un impianto di packaging il cui sistema di controllo è basato su un singolo PROCESSORE che deve GARANTIRE l'esecuzione di tre task a livello di coordinamento:

1. **L'imballatrice** viene attivata ogni 16 t.u. e impiega 3 t.u. a completare il task
2. **L'impacchettatrice** viene attivata ogni 4 t.u. e impiega 1 t.u. a completare il task
3. **L'etichettatrice** viene attivata ogni 8 t.u. e impiega 2 t.u. a completare il task

Ipotizzando di avere implementato nel processore un algoritmo **TIMELINE SCHEDULING**, studiarne la schedulabilità.

ESERCIZIO 2 cont'd

SVOLGIMENTO

Il problema può essere tradotto in un problema di scheduling di task periodici dove:

- REQUISITI DI SISTEMA: (n = 3, T_1 = 16, T_2 = 4, T_3 = 8)
- VINCOLI DI SISTEMA: (C_1 = 3, C_2 = 1, C_3 = 2)

Come prima cosa è fondamentale verificare la condizione necessaria di schedulabilità:

$$U = \sum_{i=1}^{3} \frac{C_i}{T_i} = \frac{3}{16} + \frac{1}{4} + \frac{2}{8} = \frac{3+4+4}{16} = \frac{11}{16} = 0{,}6875 < 1$$

Dato che U < 1 il problema NON È INAMMISSIBILE.

ESERCIZIO 2 cont'd

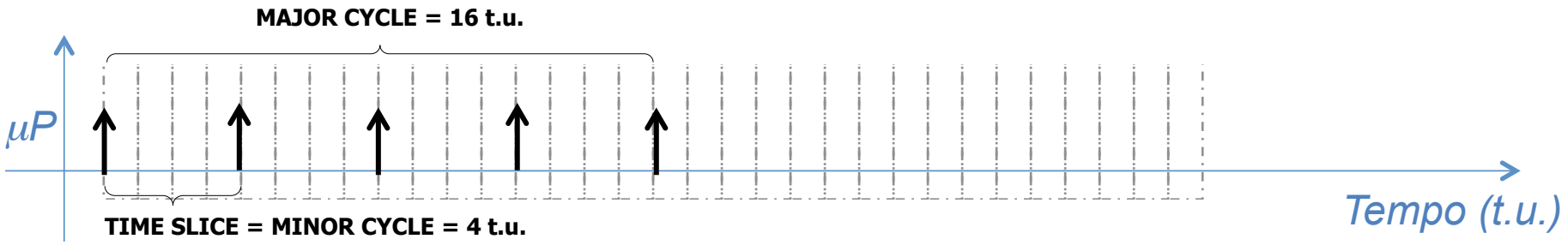
Calcoliamo il MINOR CYCLE e il MAJOR CYCLE:

$$minor\ cycle = MCD(16,4,8) = 4\ \text{t. u}$$

$$major\ cycle = mcm(16,4,8) = 16\ \text{t. u.}$$

Notiamo che l'ipotesi dell'algoritmo TIMELINE SCHEDULING è verificata, infatti tutti i computation time dei task periodici ($C_1 = 3$, $C_2 = 1$, $C_3 = 2$) sono inferiori al minor cycle.

Dividiamo l'asse temporale del μPROCESSORE in TIMESLICE ognuna della durata di t.u. pari al MINOR CYCLE.

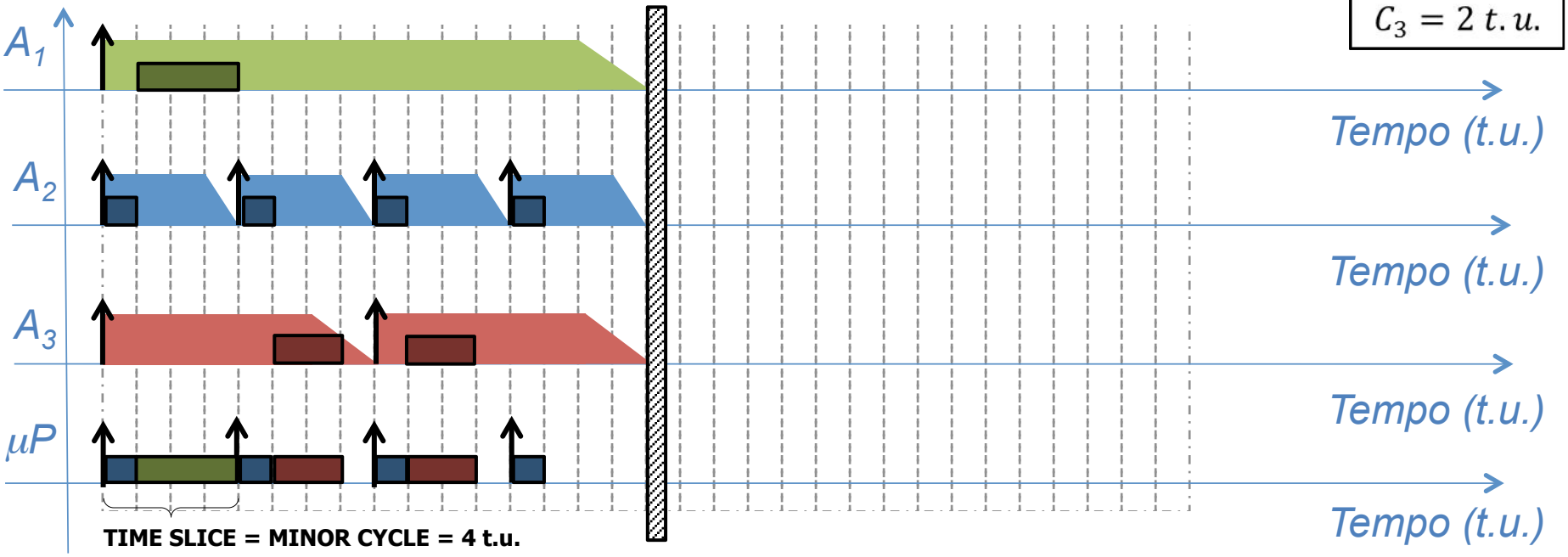


ESERCIZIO 2 cont'd

Tracciamo il diagramma temporale dei 3 task periodici ed identifichiamo una possibile soluzione, notando che:

1. Il task A_1 dovrà ripetersi MAJOR CYCLE / T_1 = 1 volta in un MAJOR CYCLE
2. Il task A_2 dovrà ripetersi MAJOR CYCLE / T_2 = 4 volte in un MAJOR CYCLE
3. Il task A_3 dovrà ripetersi MAJOR CYCLE / T_3 = 2 volte in un MAJOR CYCLE

$$C_1 = 3\ t.u.$$
$$C_2 = 1\ t.u.$$
$$C_3 = 2\ t.u.$$



BIBLIOGRAFIA

Sezione 2.4 (pagg. 61-64)

TITOLO

Sistemi di automazione industriale

Architetture e controllo

AUTORI

Claudio Bonivento

Luca Gentili

Andrea Paoli

EDITORE

McGraw-Hill