Sisteme Incorporate

Cursul 9

Algoritmi de planificare

Cuprins

- Sisteme de timp real

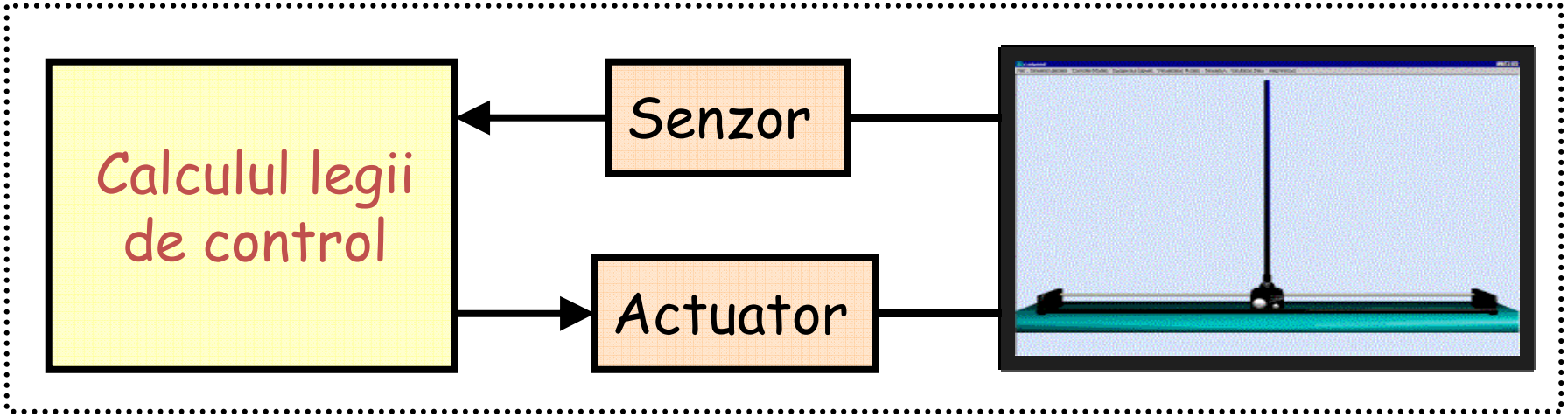
- Algoritmi de planificare de timp real
  - Cu prioritati fixe (RMS)
  - Cu prioritati dinamice (EDF)

Sisteme de timp real

- Definitie
  - Sunt sisteme a caror corectitudine depinde de aspectele lor functionale cat si de cele temporale
- Masurarea performantei
  - Punctualitatea lor in indeplinirea constrangerilor de timp (deadlines)
  - Viteza/performanta medie sunt metrici secundare
- Proprietate de baza
  - Previzibilitatea raspunsului la constragerile de timp

Exemplu sistem de timp real

- Sistem de control digital
  - Executa periodic urmatoarele sarcini:
  - masoara starea sistemului
  - actioneaza sistemul in conformitate cu statusul curent



Care este provocarea?

- Specificarea constrangerilor de timp pentru sistemele de timp real

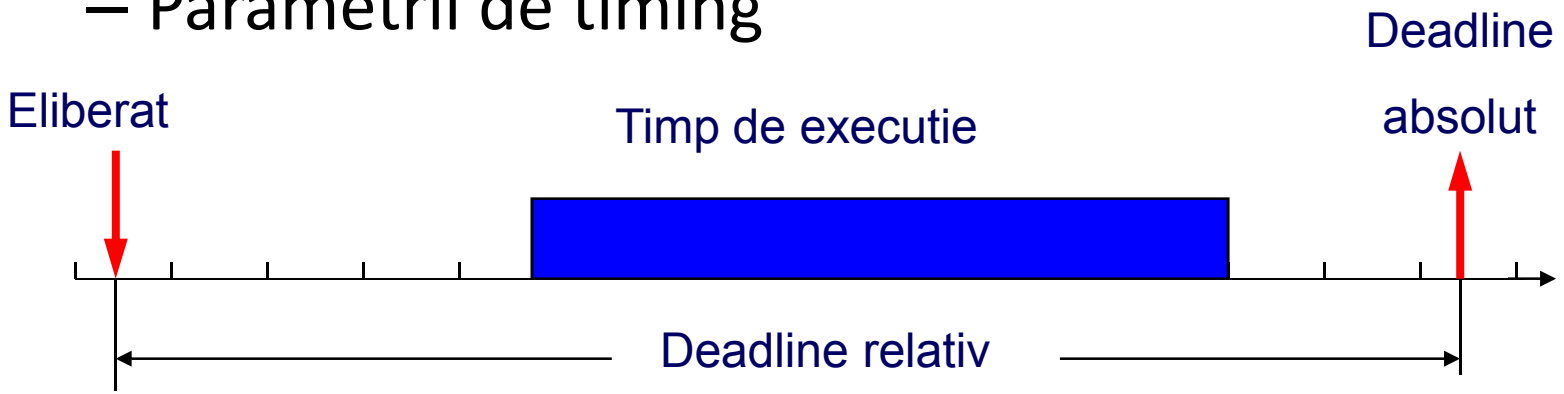
- Obtinerea unui raspuns previzibil din partea sistemului la satisfacerea tuturor constrangerilor de timp real
  - poate sa implice si existenta altor sisteme de timp real care afecteaza constrangerile sistemului tinta

Real-Time Workload

- Unitatea de munca (job)
  - Un calcul, o citire din fisier, o transmisie a unui mesaj etc

  Atribute:

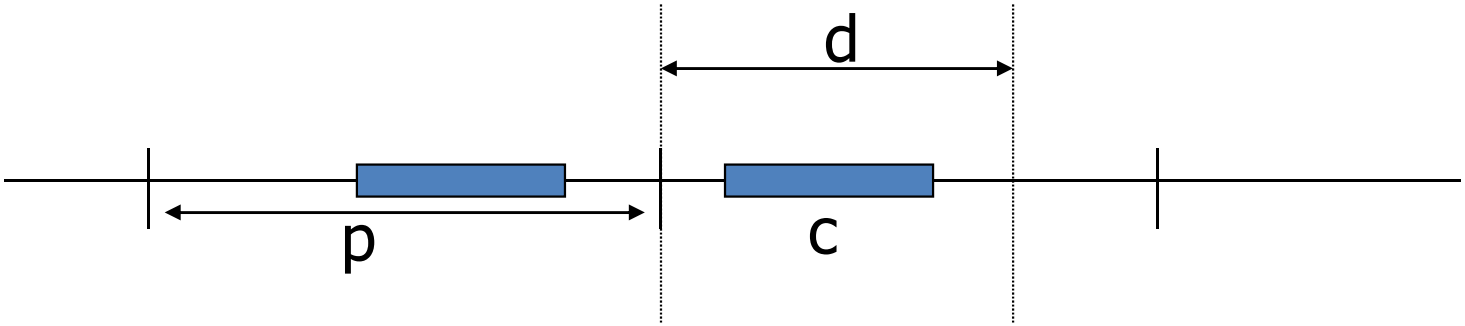
  - Resursele necesare pentru a progresa
  - Parametrii de timing



Proces Periodic

- Activat la intervale regulate de timp
  - Ex. Clipeste cursorul odata la 1 secunda
- P = (s, c, p, d)
  - s = momentul de start
  - c = worst case execution time (WCET)
  - p = perioada
  - d = deadline
  - c <= d <= p

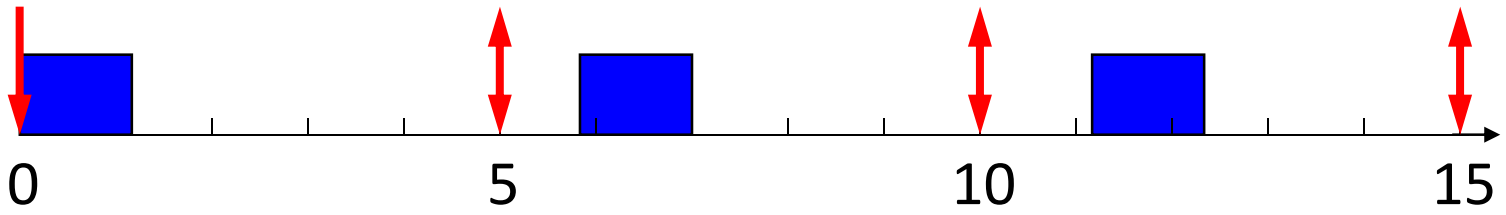
Proces Periodic



- Perioada: intervalul dintre doua activari succesive ale aceluiasi proces.
- Timp de initializare: timpul la care procesul devine initializat.
- Deadline: timpul la care procesul trebuie sa-si inceteze executia.
- In majoritatea cazurilor, d = p

Proces Periodic

- Task periodic(*p,e*)
  - Executia lui se repeta
  - Perioada *p* (0 < *p*)
  - Timp de executie *e*
    - maximul timpului de executie (0 < *e* < *p*)
  - Utilizarea: U = e/p



Worst Case Execution Time

- Pentru un task T care ruleaza pe un procesor P, WCET este timpul maxim de executie al lui T pe P pentru toate intrarile posibile ale lui T
- Analiza WCET: analiza statica pentru a estima WCET
  - Analiza cailor de executie a programului
  - Modelare micro-arhitecturala

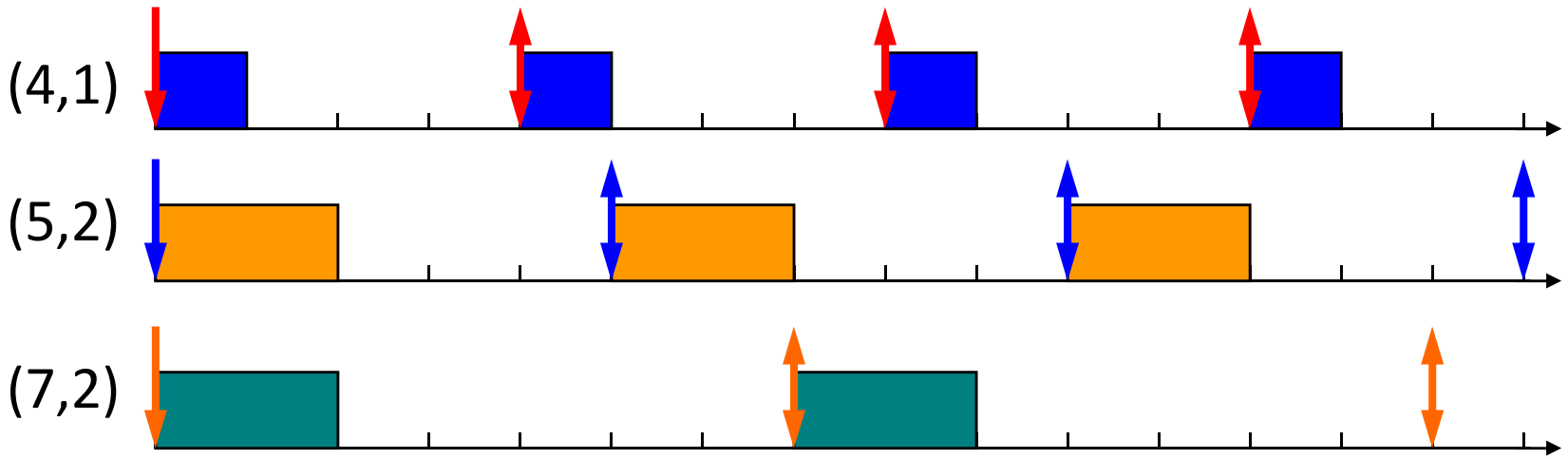
Hard vs. Soft Deadline

- Hard deadline
  - Consecinte foarte grave, chiar dezastruoase pot avea loc daca termenul nu este respectat
  - Pre-validarea este esentiala: este posibil ca toate deadline-urlie sa fie satisfacute, chiar si in cel mai rau caz?

- Soft deadline
  - In cazul ideal, deadline-ul trebuie respectat la performata maxima. Performanta scade in cazul ratarii termenelor limita.
  - Abordarea cea mai buna: garantii statistice

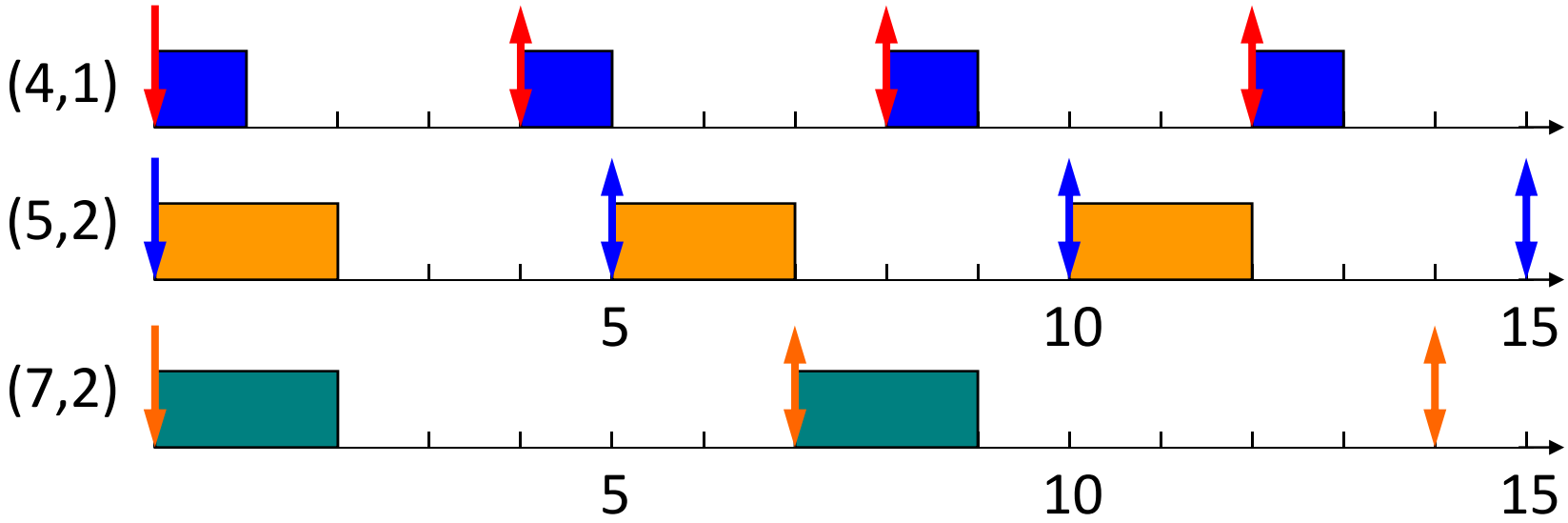
Planificabilitate

- Proprietatea care indica daca un sistem de timp real (un set de task-uri de timp real) poate sa respecte termenele limita.



Planificarea de timp real

- Determina ordinea executiei pentru task-urile de timp real
- Planificare cu prioritati
  - Statice
  - Dinamice



Rate monotonic scheduling (RMS)

- RMS (Liu si Layland 1973)
  - utilizat pe scara larga
- RMS este un algoritm optim cu prioritati fixe
  - Daca exista o planificare care satisface toate termenele limita cu prioritati fixe, atunci RMS va produce o planificare fezabila

Modelul de analiza RM

- Toate procesele ruleaza pe un singur CPU
- Costul si timpul de schimbare a contextului este nul
- Fara dependente de date intre procese (cozi, stive, semafoare blocante/non-blocante)
- Deadline la sfarsitul perioadei (p = d)
- Ruleaza procesul cu cea mai mare prioritate

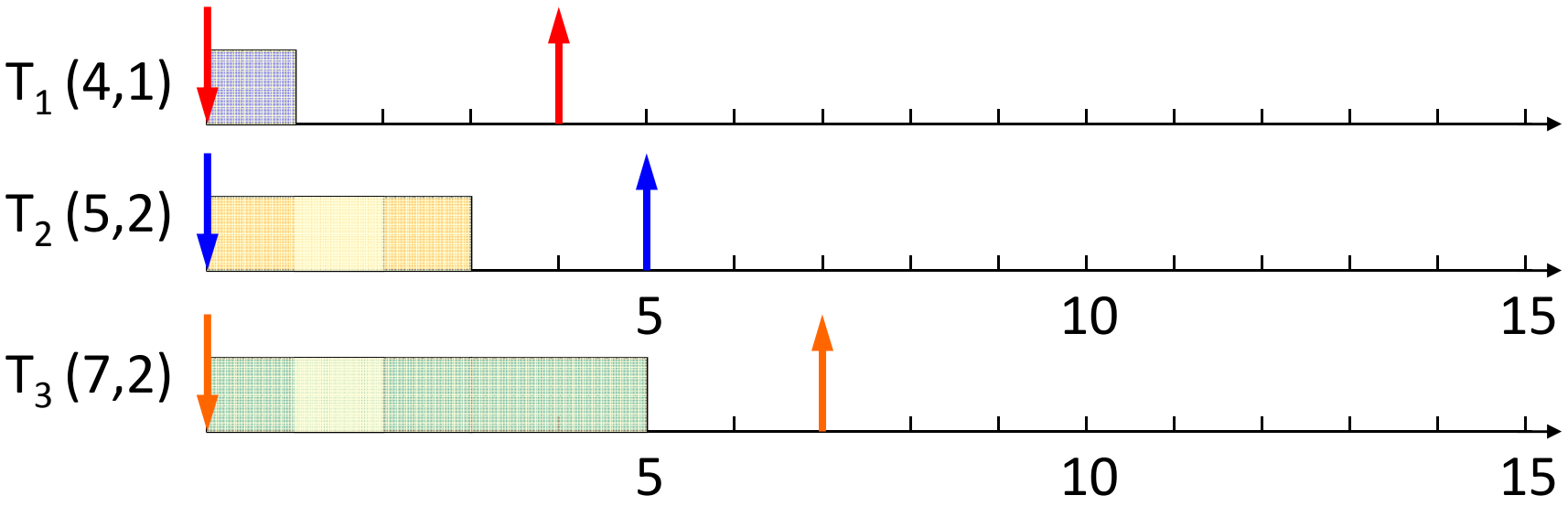
Prioritati RMS

- Planificare optima cu prioritati fixe:
  - Procesul cu perioada cea mai scurta primeste prioritatea cea mai mare;
  - Prioritatea este invers proportionala cu perioada

- Intuitie: Procesele care au nevoie de atentie frecvent (perioada mica) trebuie sa primeasca prioritate mai mare.

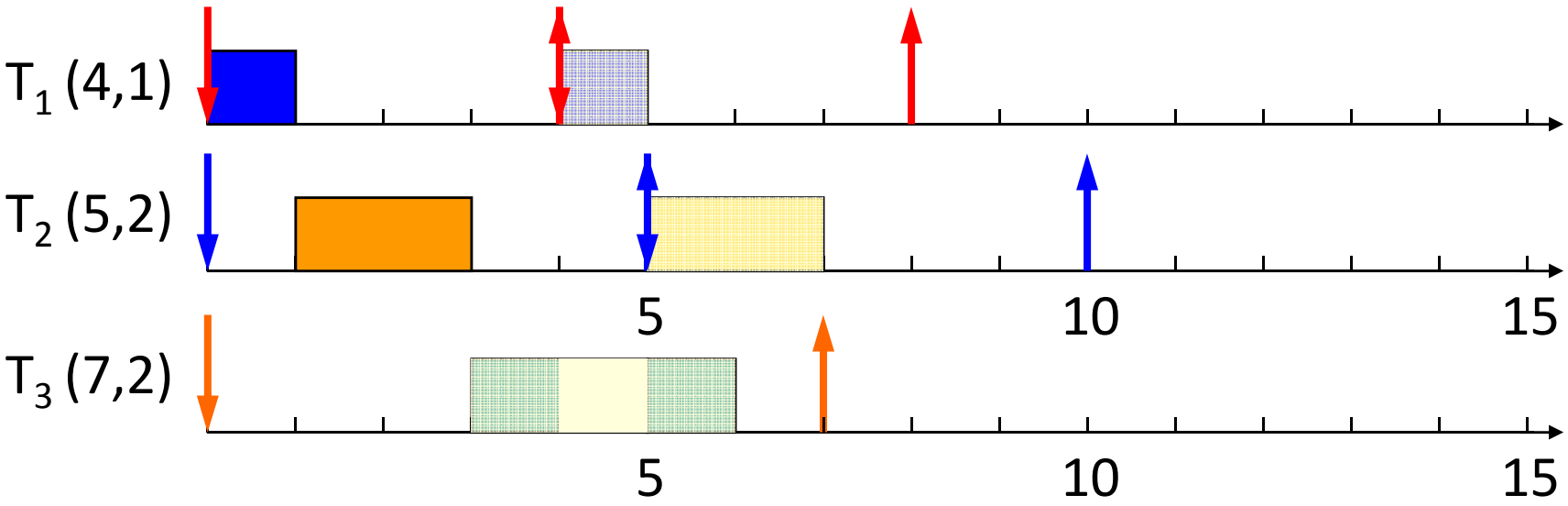
RM (Rate Monotonic)

- Task-ul cu perioada cea mai mica primeste prioritatea cea mai mare
- Executa sarcina cu prioritatea cea mai mare



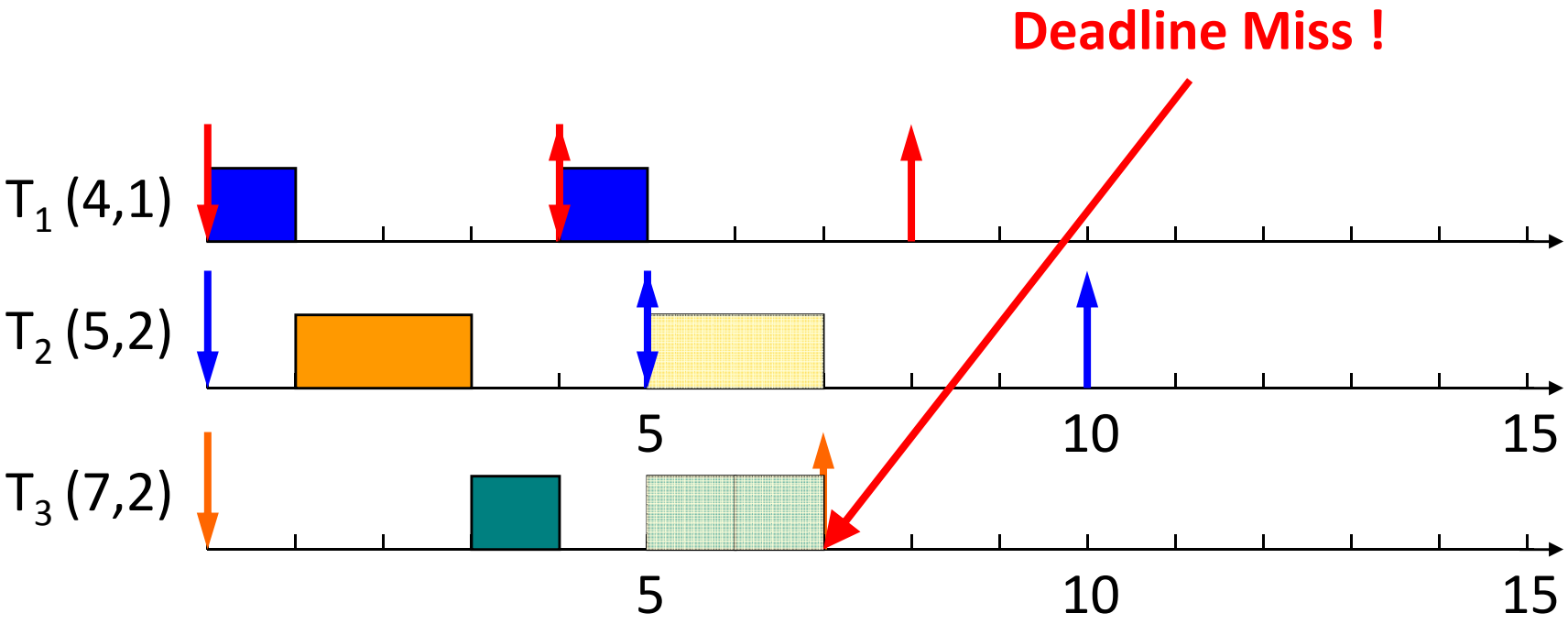
RM (Rate Monotonic)

- Executa sarcina cu prioritatea cea mai mare



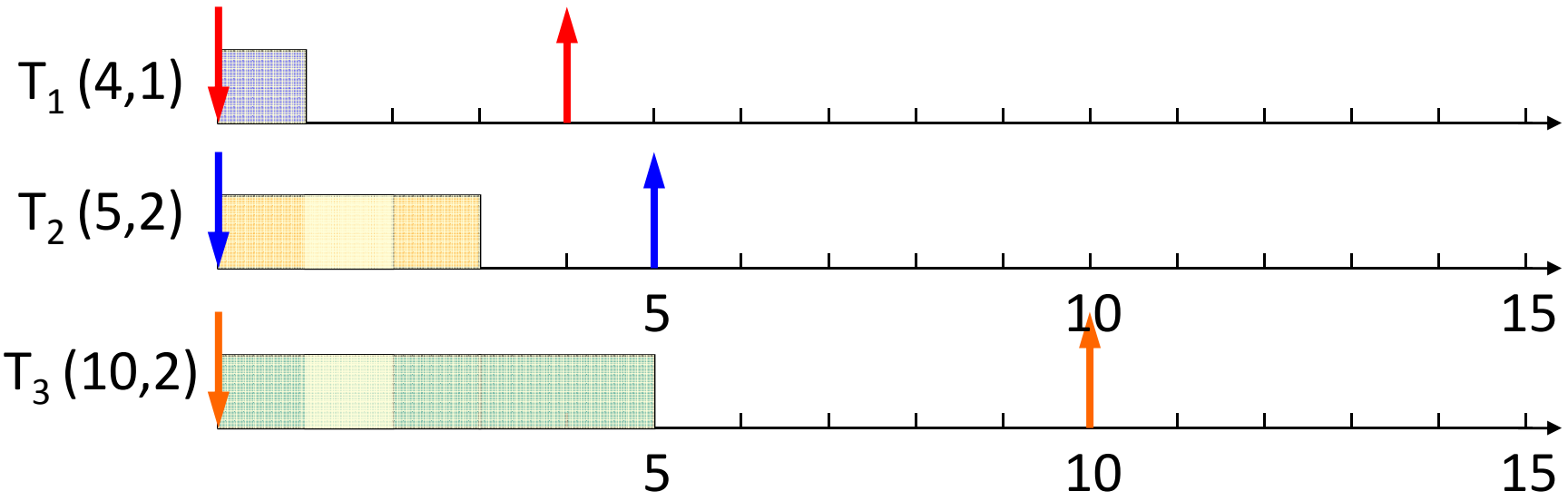
RM (Rate Monotonic)

- Executa sarcina cu prioritatea cea mai mare



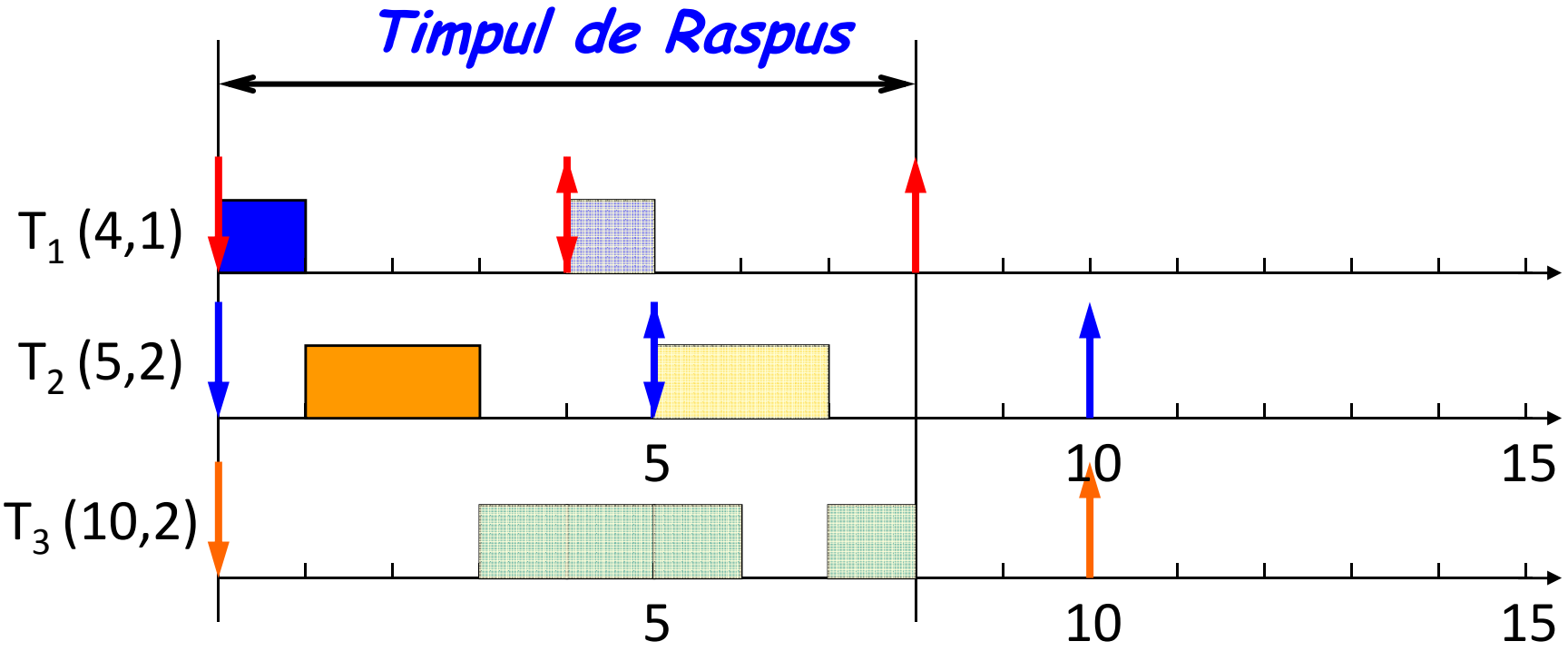
Timpul de raspuns

- Durata de la release pana la terminarea procesului



Timpul de raspuns

- Durata de la release pana la terminarea procesului

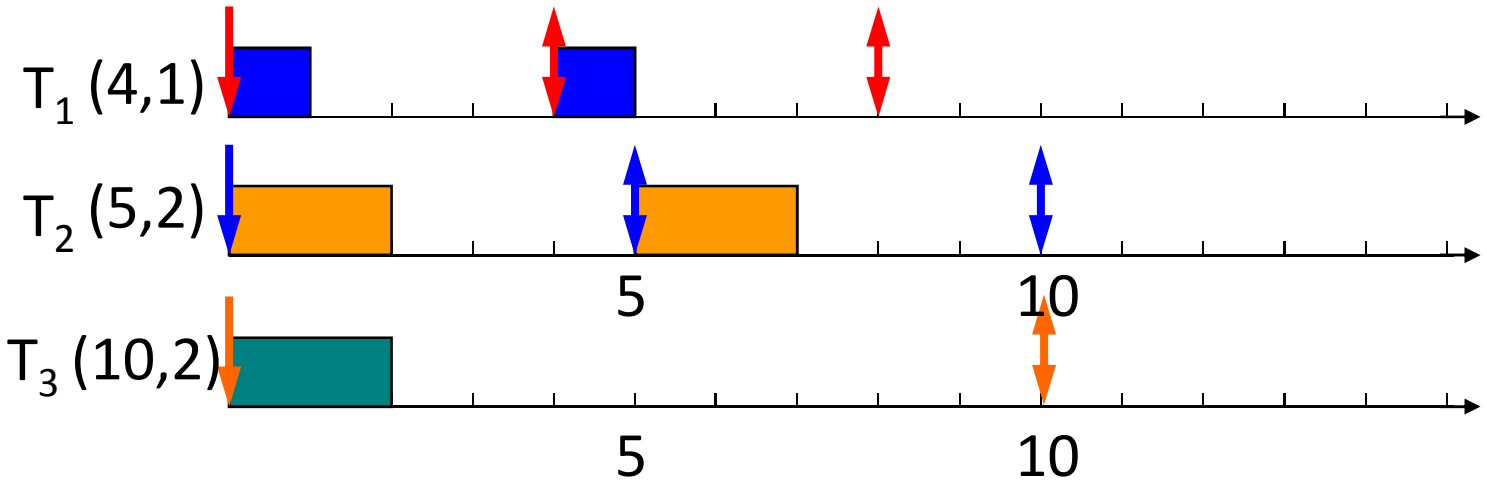


Evaluarea timpului de raspus

- Response Time (r_i) [Audsley et al., 1993]

$$r_i = e_i + \sum_{T_k \in HP(T_i)} \left\lceil \frac{r_i}{p_k} \right\rceil \cdot e_k$$

- HP(T_i) : multimea task-urilor cu prioritate mai mare decat T_i



RM – Analiza planificabilitatii

- Un sistem de timp real este planificabil cu RM

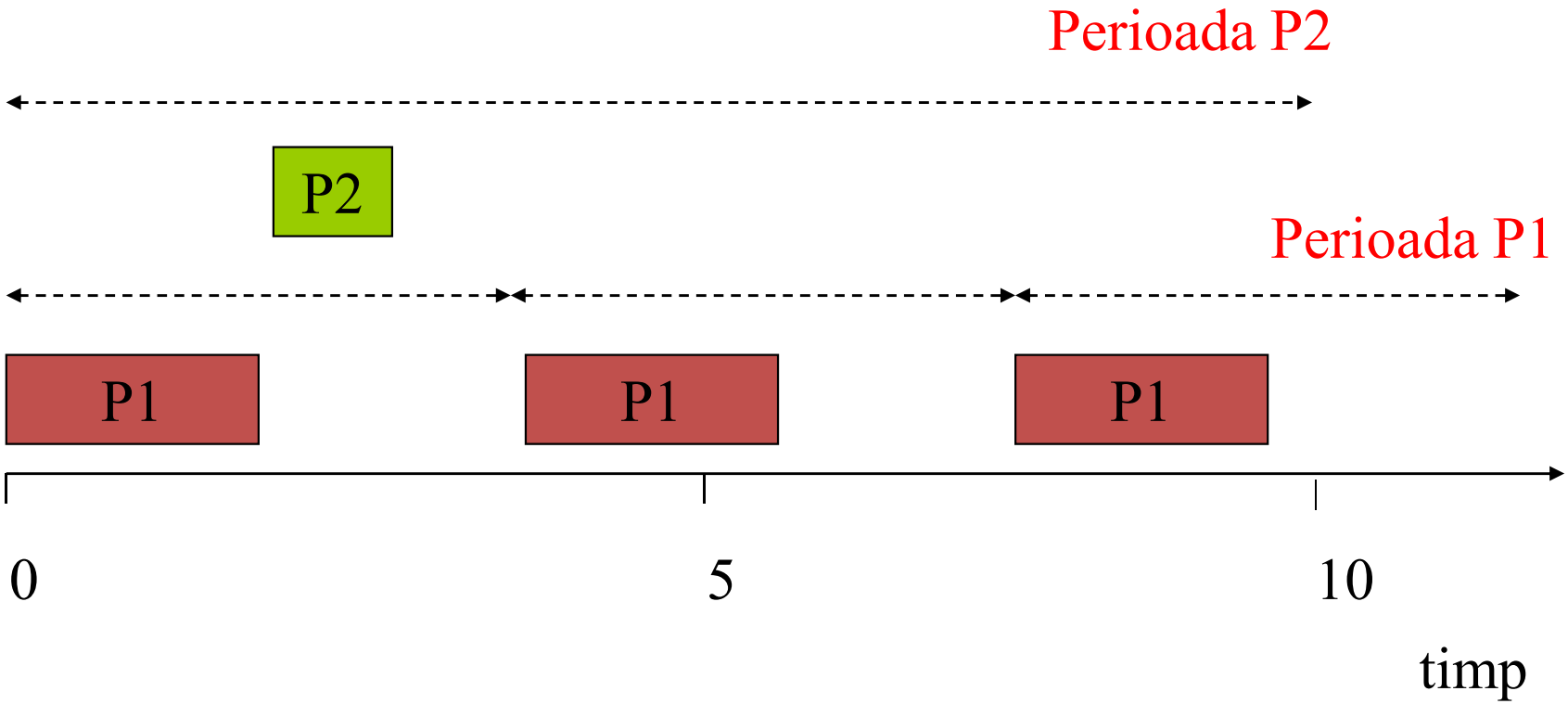
  Daca si numai daca $r_i \leq p_i$ pentru toate task-urile $T_i(p_i,e_i)$

  Joseph & Pandya,

  “Finding response times in a real-time system”,

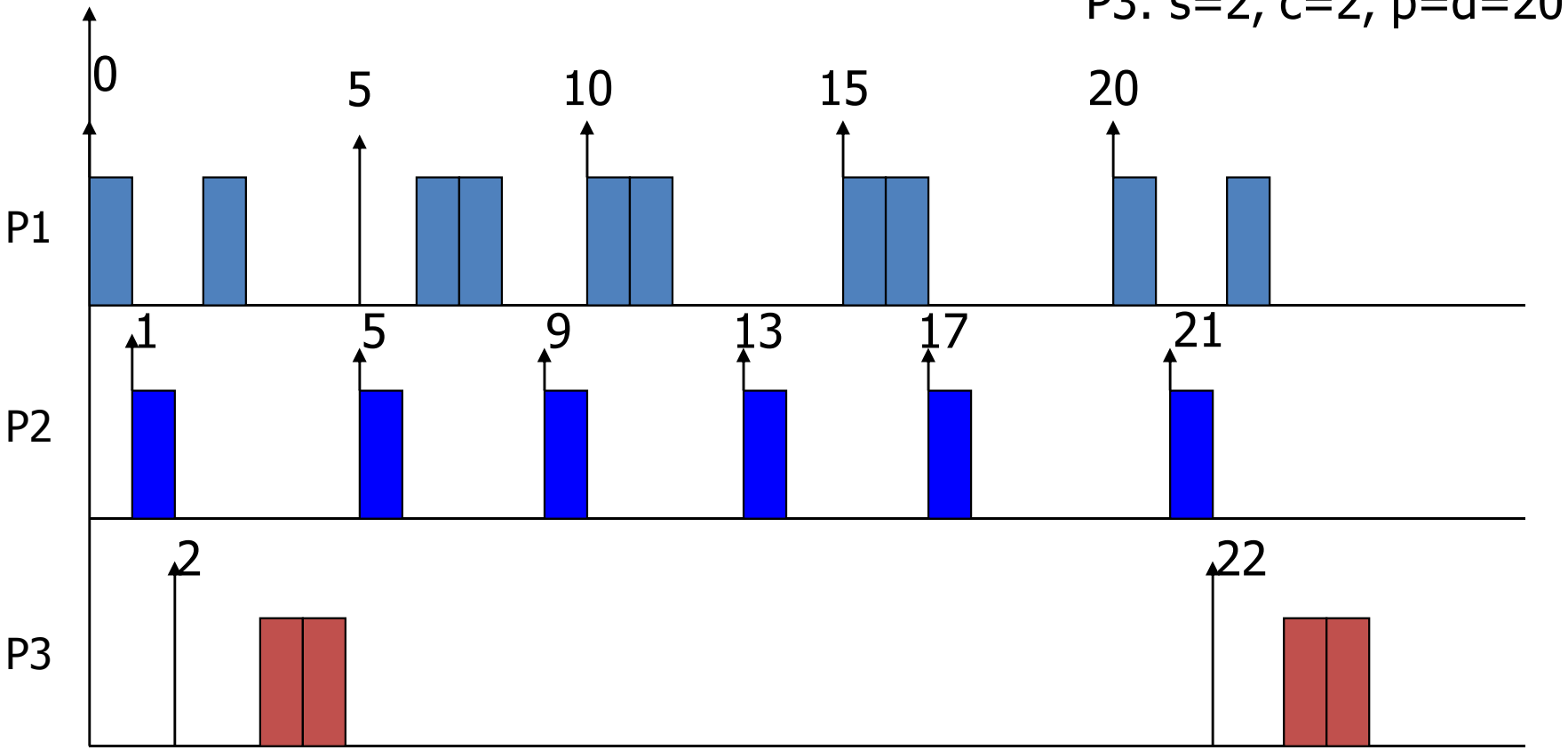
  The Computer Journal, 1986.

Exemplu RMS



Exemplu RMS

P1: s=0, c=2, p=d=5
P2: s=1, c=1, p=d=4
P3: s=2, c=2, p=d=20



RMS – Utilizarea procesorului

- Utilizarea pentru n procese este

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i}$$

- U<=1 este o conditie necesara dar nu si suficienta pentru fezabilitate, indiferent de politica de planificare
- Planificarea cu prioritati fixe este fezabila daca

$$U \leq n\left(2^{\frac{1}{n}} - 1\right)$$

- Pe masura ce numarul sarcinilor tinde la infinit, utilizarea maxima se apropie de 69%

$$\lim_{n->\infty} n\left(2^{\frac{1}{n}} - 1\right) = \ln 2 \approx 0.69$$

RMS – Utilizarea procesorului

- Pentru exemplul anterior:

P1: s=0, c=2, p=d=5
P2: s=1, c=1, p=d=4
P3: s=2, c=2, p=d=20

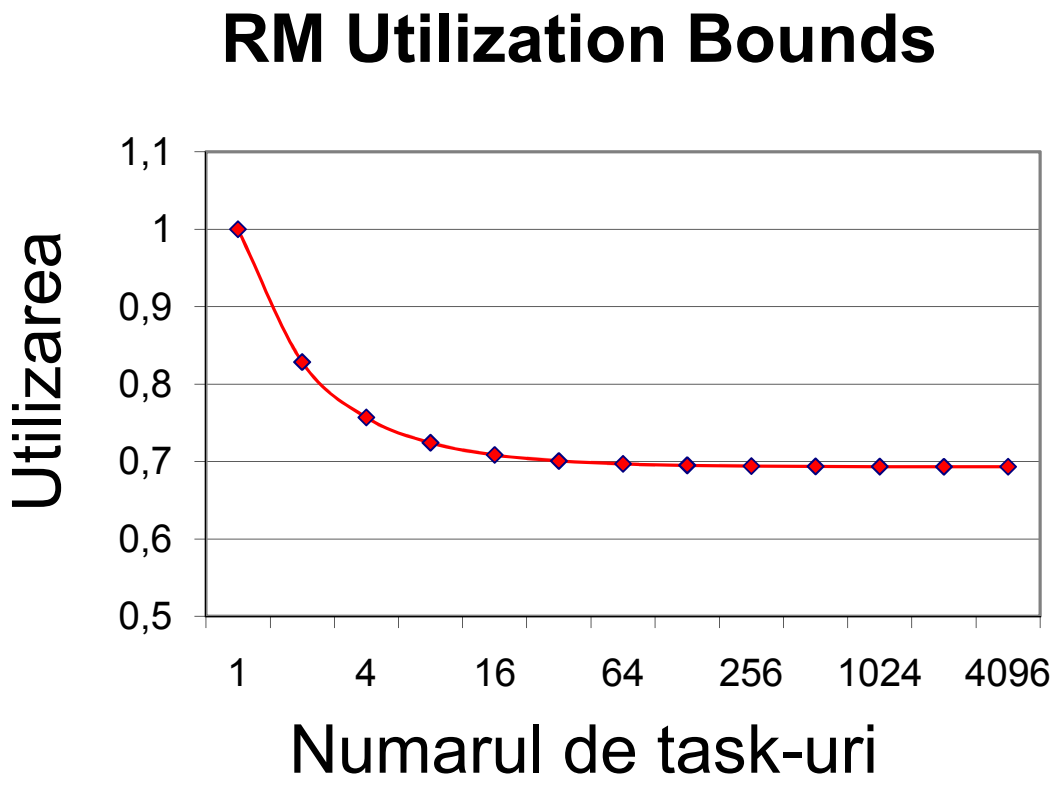
$$U = \frac{2}{5} + \frac{1}{4} + \frac{2}{20} = 0.75 \qquad U \leq 3\left(2^{\frac{1}{3}} - 1\right) = 0.779..$$

Sistemul este planificabil.

RM – Limitarea utilizarii

- Planificarea cu prioritati fixe este fezabila daca

$$\sum U_i \leq n\,(2^{1/n}-1)$$



Analiza rate-monotonic

- Timpul de raspuns: timpul necesar pentru terminarea procesului.
- Momentul critic: starea pentru care planificatorul da timpii de raspuns cei mai mari.
- Instanta critica se petrece atunci cand toate procesele cu prioritate mare sunt gata de executie.

Momentul critic pentru un proces

- Deadline al unui task = timpul urmatoarei cereri
- Overflow se pretrece la momentul t daca t este deadline-ul unei cereri nesatisfacute
- Un algoritm de planificarea este fezabil daca procesele pot fi planificate a. i. nu se pretrece nici un overflow
- Timpul de raspuns la o cerere a unui proces este intervalul de timp dintre cerere si raspunsul dat procesului respectiv

Momentul critic pentru un proces

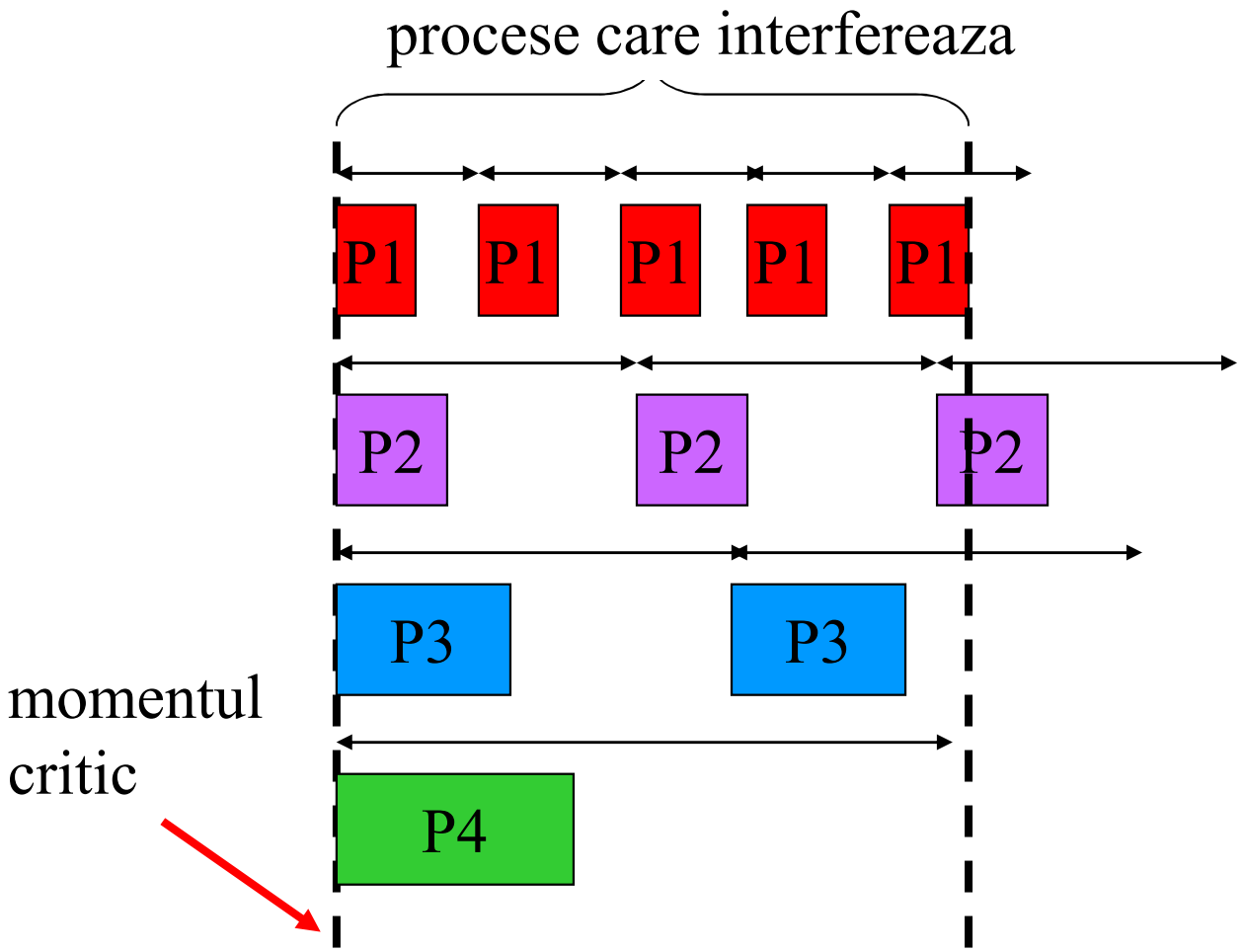
- Momentul Critic = momentul de timp pentru care o cerere catre procesul respectiv produce un timp de raspuns maxim.

- Interval critic = intervalul de timp de la momentul critic pana la raspunsul procesului la cererea primita.

Cand se intampla momentul critic?

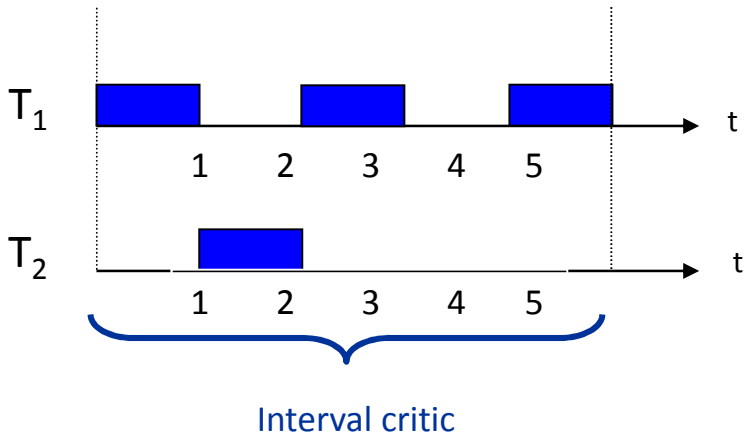
- *Teorema: Pentru orice proces, momentul critic se produce atunci cand procesul respectiv primeste o cerere simultana cu alte cereri catre toate procesele cu prioritati mai mari.*
- Putem determina asa daca o schema de atribuire de prioritati va produce o planificare fezabila.
  - Daca cererile adresate tuturor proceselor in momentele lor critice sunt satisfacute inainte de deadline, atunci schema de planificare folosita este fezabila.

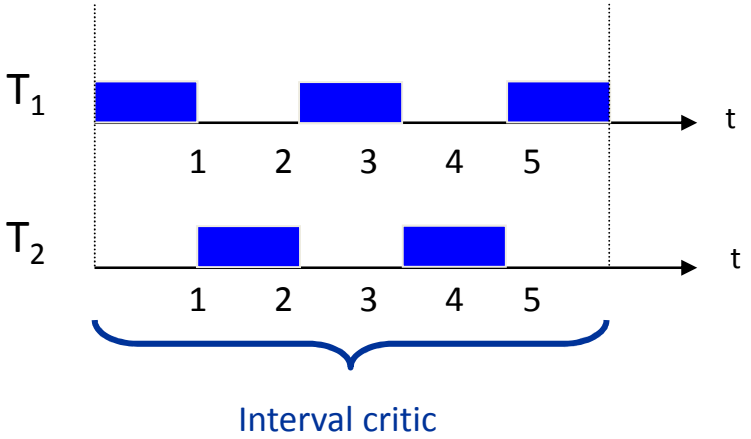
Momentul Critic



Exemplu

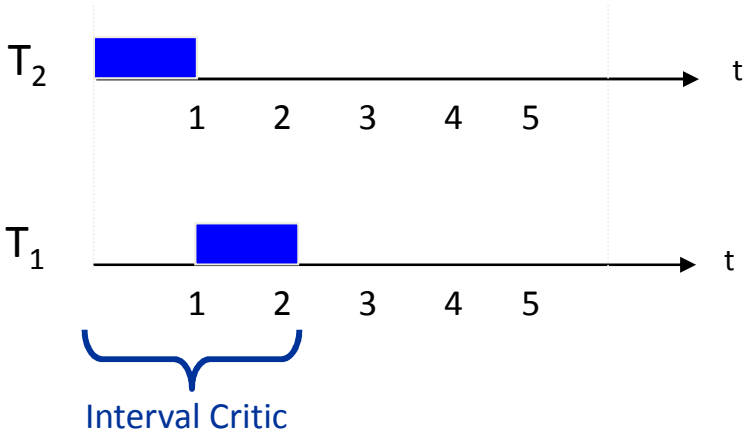
- Doua procese τ_1 & τ_2 cu T_1=2, T_2=5, & C_1=1, C_2=1
- τ_1 are prioritate mai mare decat τ_2
  - Schema de atribuire de prioritati este fezabila
  - Pot sa maresc C_2 la 2 si inca pot face planificare





Exemplu (cont.)

- τ_2 are prioritate mai mare decat τ_1
  - Atribuirea de prioritati este inca fezabila
  - Nu pot sa maresc peste C_1=1, C_2=1



Testarea planificarii

Conditie suficienta dar nu si necesara

$$c_1 \times \left\lceil \frac{p_i}{p_1} \right\rceil + c_2 \times \left\lceil \frac{p_i}{p_2} \right\rceil + \ldots\ldots\ldots + c_i \times \left\lceil \frac{p_i}{p_i} \right\rceil \leq p_i$$

Planificarea earliest-deadline-first (EDF)

- EDF: algoritm de planificare cu atribuire dinamica de prioritati.
- Procesul care este cel mai aproape de termenul limita are prioritatea cea mai mare.
- Necesita recalcularea prioritatilor proceselor la fiecare intrerupere de timer.
- EDF poate utiliza si 100% din CPU.

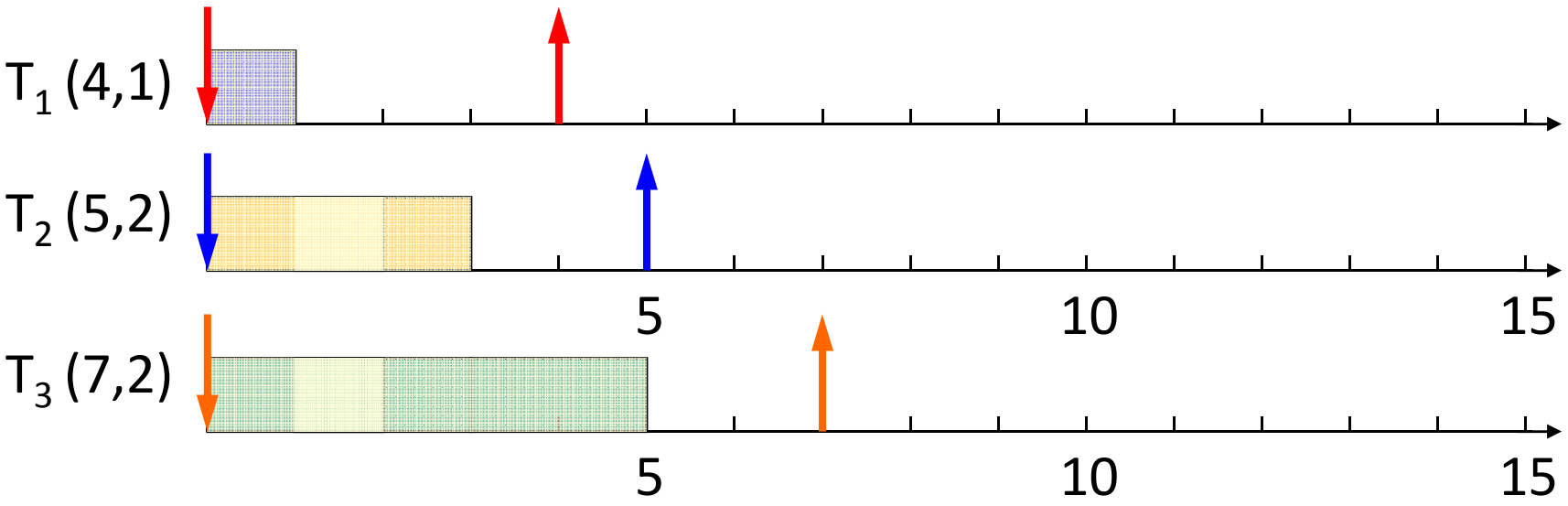
Implementarea EDF

- La fiecare intrerupere de ceas:
  - Calculeaza timpul pana la deadline;
  - Alege procesul cu deadline-ul cel mai apropiat.

- De obicei este considerat un algoritm prea costisitor pentru a putea fi utilizat practic.

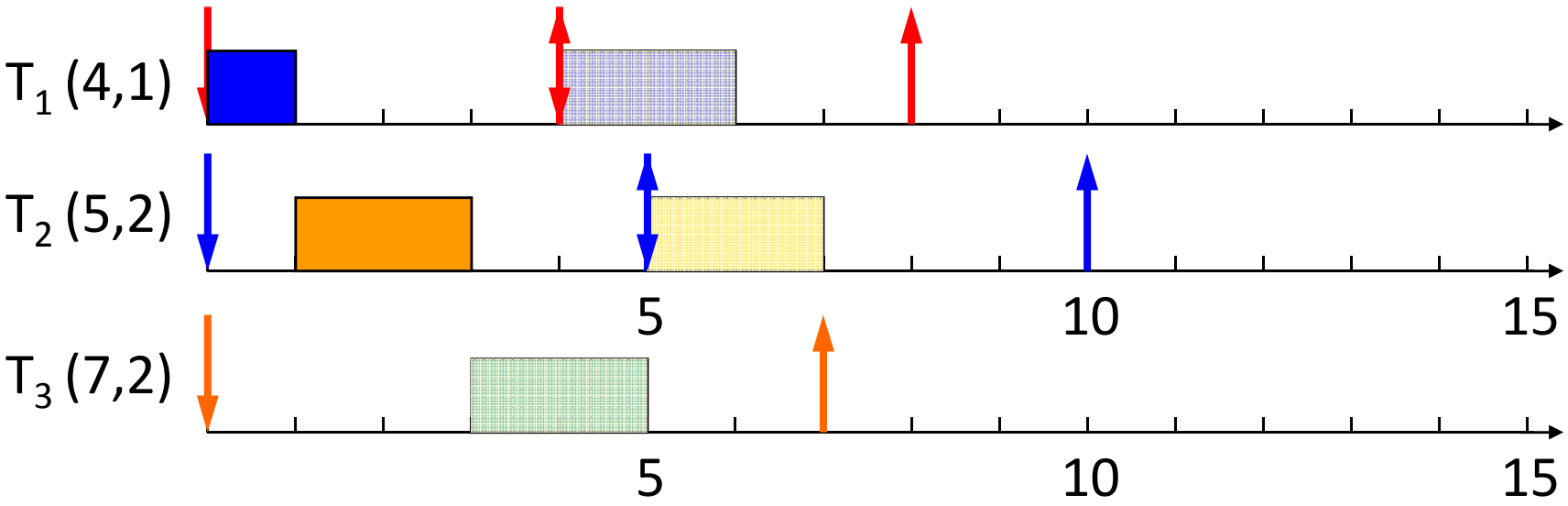
EDF (Earliest Deadline First)

- Task-ul cu deadline apropiat are prioritate mai mare
- Executa task-ul cu cel mai apropiat termen



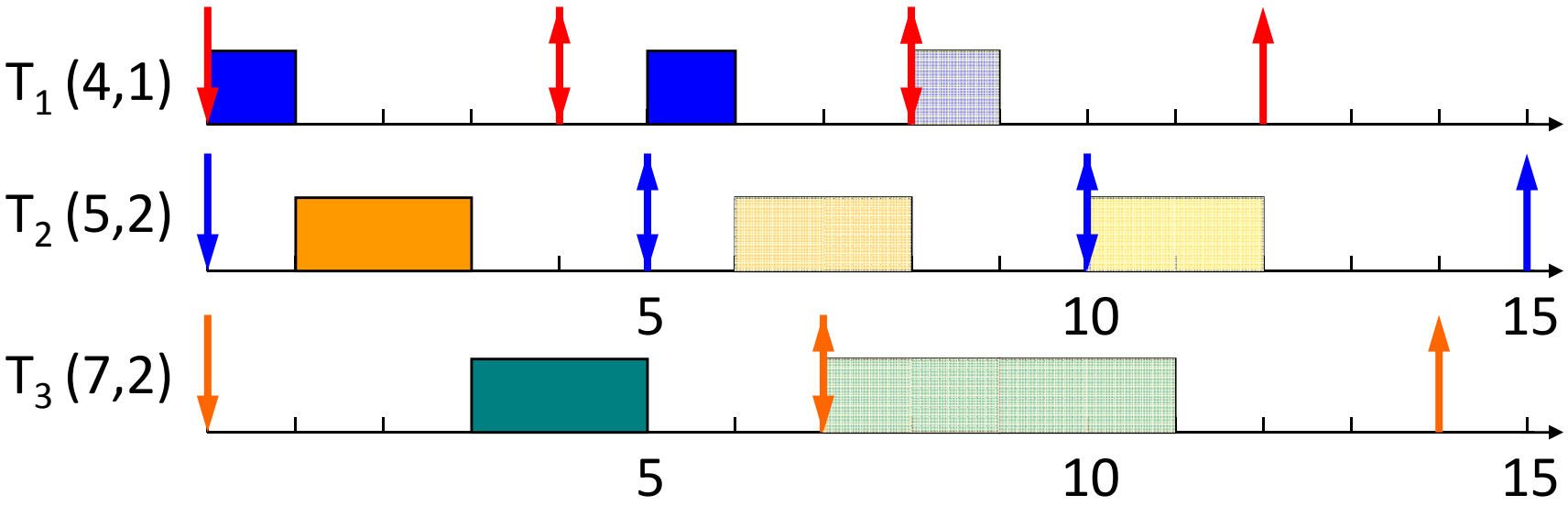
EDF (Earliest Deadline First)

- Executa task-ul cu cel mai apropiat termen



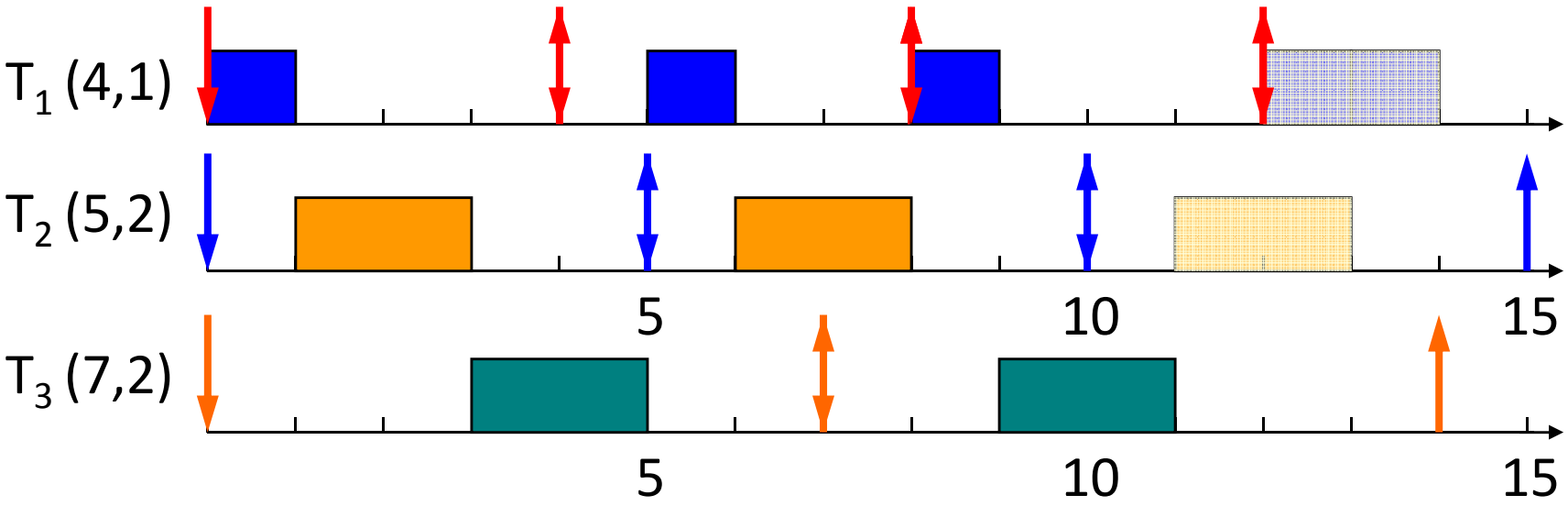
EDF (Earliest Deadline First)

- Executa task-ul cu cel mai apropiat termen



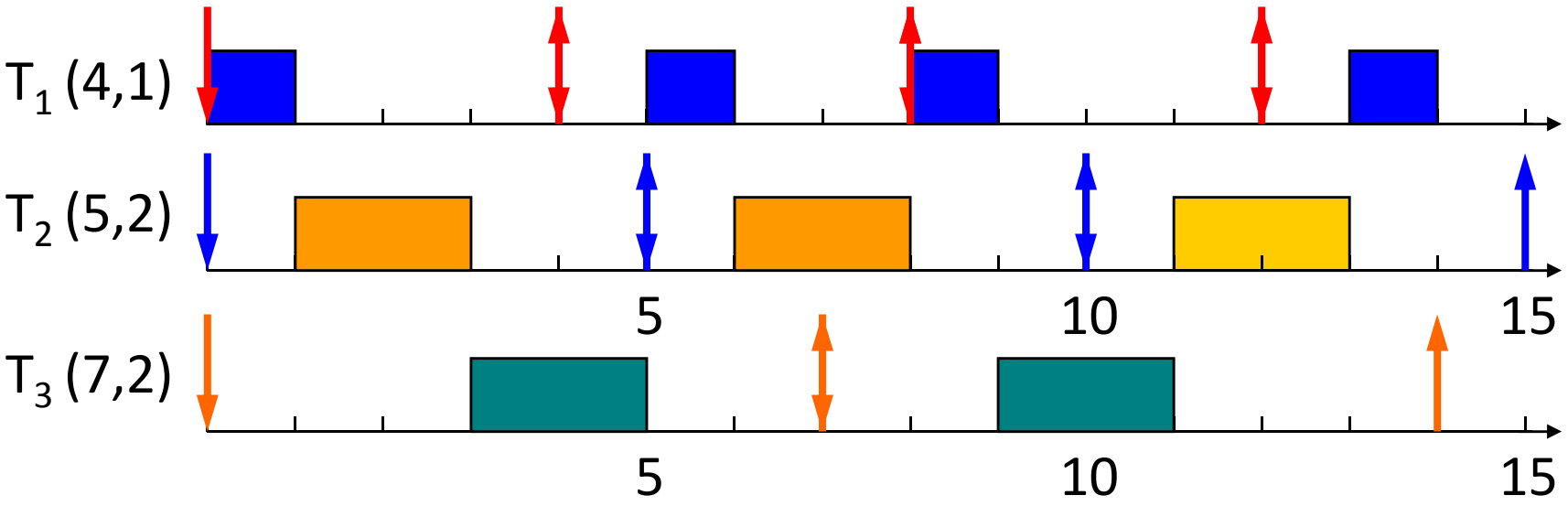
EDF (Earliest Deadline First)

- Executa task-ul cu cel mai apropiat termen



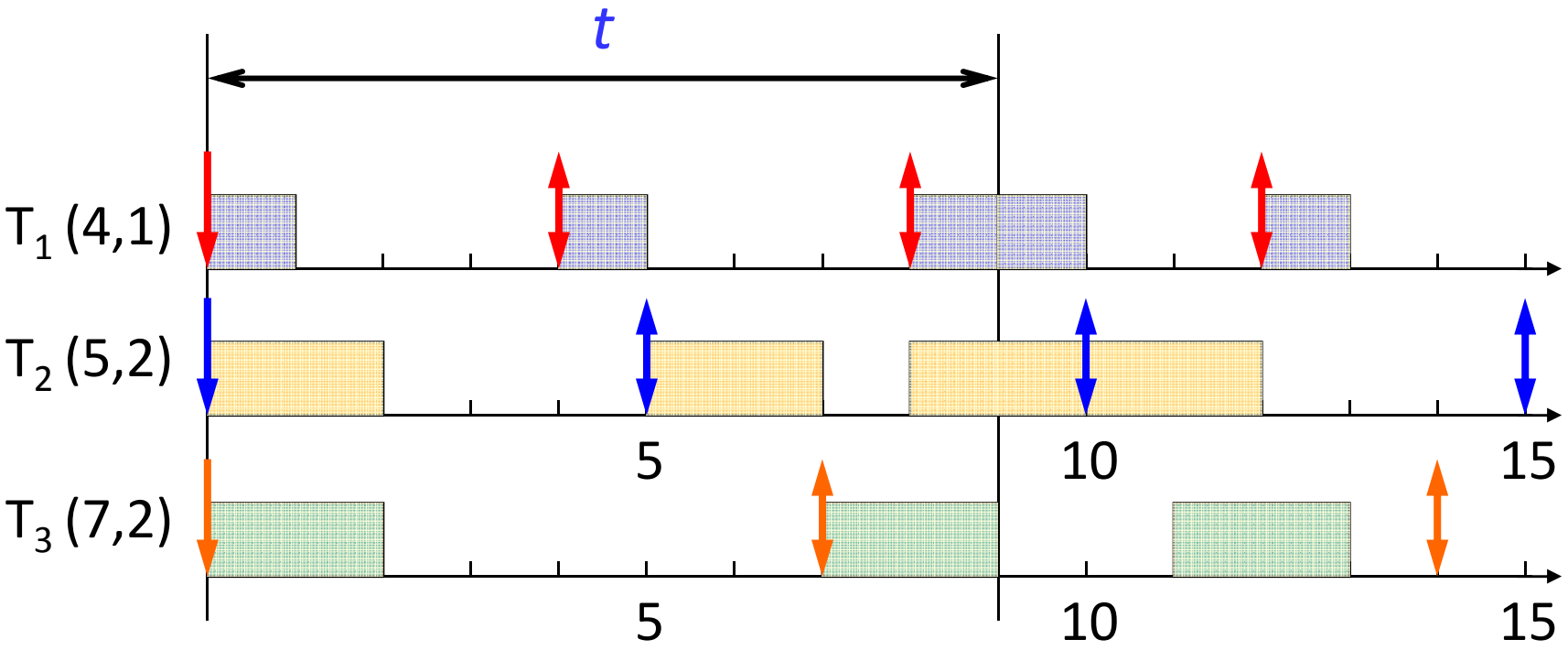
EDF (Earliest Deadline First)

- Algoritm de planificare optim
  - Daca exista o schema de planificare pentru un set de task-uri de timp real, EDF o va gasi



Limita de utilizare a procesorului

- Demand Bound Function : *dbf(t)*



EDF – Analiza planificabilitatii

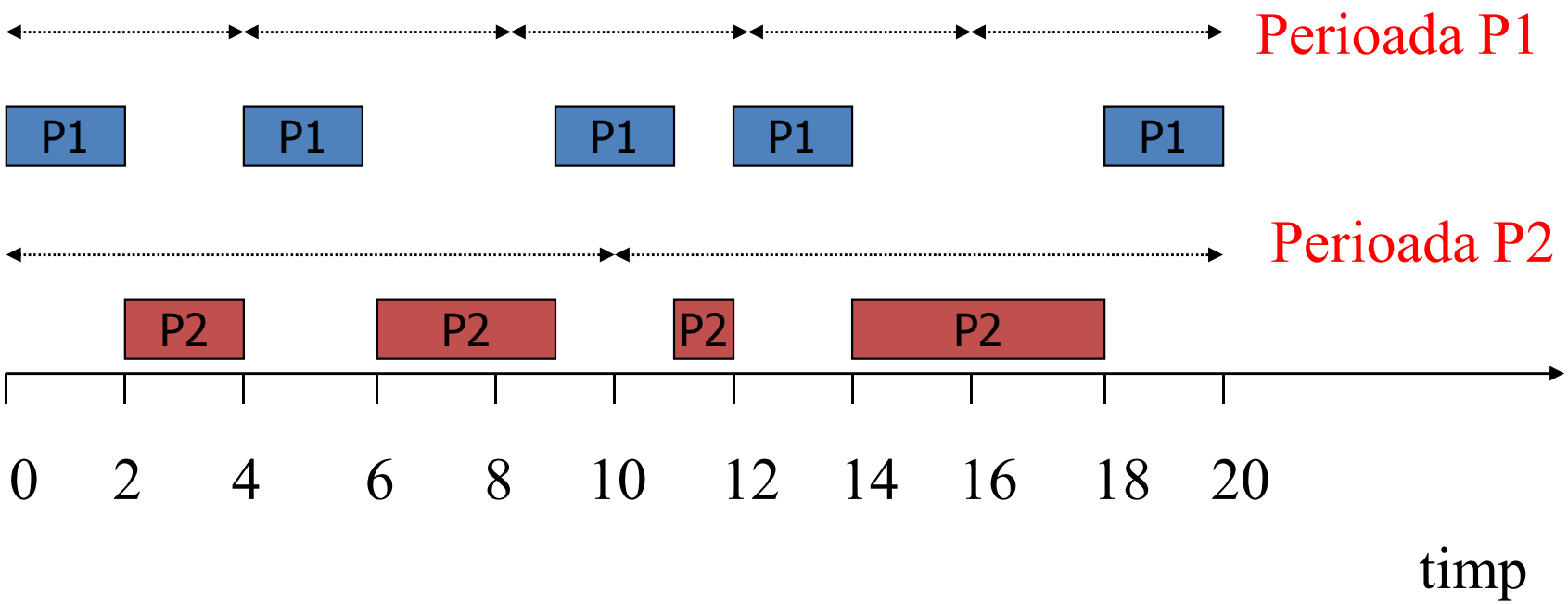
- Un sistem de timp real este planificabil cu EDF
  daca si numai daca **dbf(t) ≤ t** pentru toate intervalele de timp t

  Baruah et al.
  "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor", Journal of Real-Time Systems, 1990.

- Demand Bound Function : dbf(t)
  - Maximul utilizarii procesorului pe intervalul de timp t

Exemplu EDF

P1 = (2, c=4, d=4) P2 = (5, c=10, d=10)



Rezultatele EDF

- EDF este un algoritm optim de planificare
  - Daca exista o schema de planificare cu prioritati dinamice, EDF va produce o schema fezabila
- EDF va produce un scenariu de planificare fezabil daca U<=1
- Planificarea cu atribuire de prioritati dinamice este fezabila daca si numai daca if U<=1 (conditie necesara si suficienta)

EDF – Utilizarea procesorului

- Utilizarea pentru n procese este

$$U = \sum_{i=1}^{n} \frac{c_i}{p_i} \leq 1$$

- U<=1 pentru ca schema de planificare sa fie fezabila
- Exemplu

  P1: s=0, c=1, p=d=8
  P2: s=1, c=2, p=d=5
  P3: s=2, c=4, p=d=10

$$U = \frac{1}{8} + \frac{2}{5} + \frac{4}{10} = 0.925 = 92.5\%$$

Sistemul este planificabil.

Concluzii

- Rate Monotonic
  - Implementare mai simpla chiar si pentru sistemele fara suport explicit pentru constrangeri de timp (perioade, termene limita)
  - Previzibil pentru task-urile cu prioritatea cea mai mare
- EDF
  - Utilizarea maxima a procesorului
  - Functionare imprecisa la supraincarcare

- Pentru mai multe detalii: Buttazzo, “Rate monotonic vs. EDF: Judgement Day”, EMSOFT 2003.