Temps real

Programació concurrent i temps real

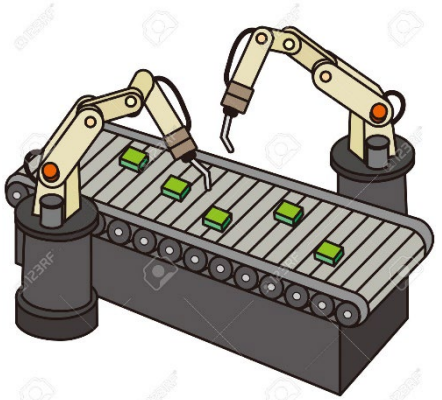
ESQUEMA

- **Introducció als sistemes de temps real**
  - Sistemes de temps real durs i tous (hard i soft)
- Planificació de feines o treballs
  - Algoritmes de planificació basats en rellotges
  - Algorismes de planificació basats en prioritats
    - Estàtiques
    - Dinàmiques
- Control d'accés als recursos
  - Algorismes
  - Implementació

Introducció als sistemes de temps real

- Laplante: *"Un sistema de temps real és un sistema en que els temps de resposta han de satisfer requisits explícits, i en cas de no respondre dins d'un interval acordat es poden produir conseqüències greus, fins i tot poder arribar al fracàs."*

- Exemple: Un robot que ha de recollir peces d'una cinta transportadora. Si arriba tard, la peça ja no hi serà. El procés d'arribar al lloc es correcte, no s'acompleix el requisit temporal i per tant s'ha fallat en l'acció.
- La importància d'un sistema en temps real no és la rapidesa en fer una tasca, la importància del temps real és assegurar que el temps definit per fer una tasca es complirà.



Introducció als sistemes de temps real
Classificació: Exigència temporal

- **Crítics: (durs) Hard real-time Systems**:
  - Totes les accions s'han de fer dins d'un límit específic.
  - Una resposta fora del termini pot causar unes conseqüències catastròfiques del sistema que controla.
    - Un exemple és el control de frenada d'un automòbil, un control de vol, una central nuclear...
- **Flexible: (tous) Soft real-time Systems**:
  - Poden tolerar un incompliment en el temps de resposta, amb la seva penalització pertinent.
    - Un exemple: L'intèrpret de comandes d'una interfície amb l'usuari, representacions gràfiques, el sistema GPS...
- **(forts) Firm real-time Systems**:
  - Es poden perdre terminis ocasionalment.
  - Una resposta fora de termini no tindrà cap valor pel sistema, però no causa cap dany.
    - Un exemple: Un sistema en temps real multimèdia, de vídeo,

Introducció als sistemes de temps real
Sistemes de temps real durs o crítics (hard)

- Un sistema de temps real dur garanteix que un treball o feina es completarà en un termini de temps especificat.
- Aquest sistema ha de garantir que tots els retards en el processament, l'entrada i sortida són limitades.
- El sistema no pot esperar indefinidament de manera que els sistemes de temps real dur solen ser molt limitats.
- Generalment no hi ha emmagatzematge secundari, com ara unitats de disc ja que un accés a disc pot trigar un temps variable en el procés.
- Alguns exemples d'un sistema de temps real dur són:
  - El programari que executa el pilot automàtic en un avió.
  - El programari d'imatges en un míssil.
  - El control d'una central nuclear.
  - Els marcapassos.

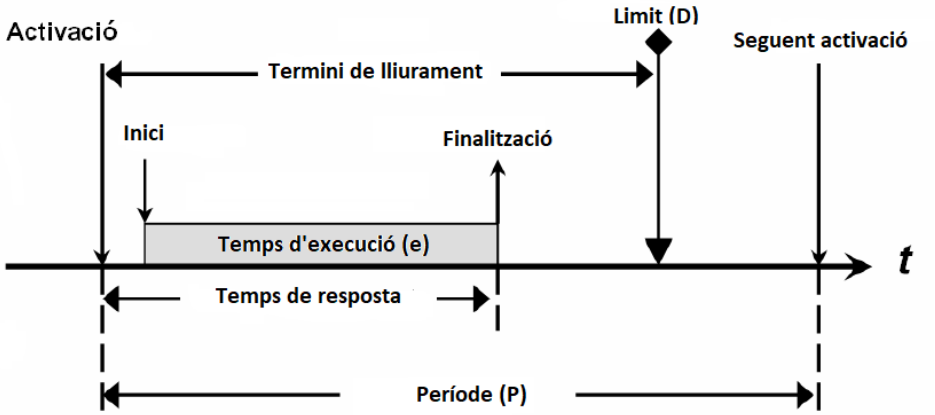
Introducció als sistemes de temps real
Sistemes de temps real tous o suaus (soft)

- Un sistema de temps real suau és una versió molt menys restrictiva d'un sistema de temps real dur.
- Un sistema de temps real suau no garanteix que una feina es completarà en un termini de temps especificat, però, intenta tot el possible per acabar la feina el més aviat possible.
- Si una feina crítica en temps real entra al sistema, el sistema operatiu pot assignar la màxima prioritat a aquesta tasca i que s'executi de forma contínua fins que es completi.
- Sistemes d'aquest tipus són:
  - Els sistemes multimèdia.
  - Els sistemes d'adquisició de dades.
  - Gestió d'un magatzem

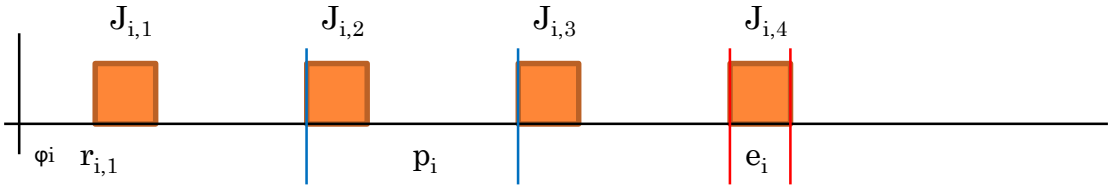
Introducció als sistemes de temps real
Classificació: Escala de temps

- Basats en el rellotge: S'invoca i executa repetidament a intervals constants a partir d'una crida inicial, tasques periòdiques. Solen ser de temps real dur
- Basats en esdeveniments: Les accions es realitzen a partir d'un esdeveniment. Per exemple, una acció comença quan es llegeix una dada del sensor, bàsicament son tasques aperiòdiques. Tenen un termini de finalització suau o tou o cap.
- Interactius: L'execució de les accions es fa en temps irregulars, per exemple, un operari inserint dades. Tasques esporàdiques. Normalment tenint uns terminis de finalització durs (Hart real time)

Introducció als sistemes de temps real
Tipus de tasques: Periòdiques



- Un conjunt de feines (treballs) que s'executen repetidament a intervals regulars de temps es poden considerar com una tasca periòdica
- Cada tasca periòdica T_i és una seqüència de feines $J_{i,1}, J_{i,2}, ..., J_{i,n}$
  - La fase d'una tasca T_i és el temps d'activació $r_{i,1}$ de la primera feina $J_{i,1}$ de la tasca. S'anomena com φ_i ("phi")
  - El període p_i de la tasca T_i és la llargada mínima de tots els intervals de temps entre els temps d'activació de les feines consecutives
  - El temps d'execució e_i d'una tasca T_i és el temps màxim d'execució de totes les feines de la tasca periòdica
  - El període i el temps d'execució de cada tasca periòdica en el sistema es coneixen sempre amb una precisió raonable



Introducció als sistemes de temps real
Característiques

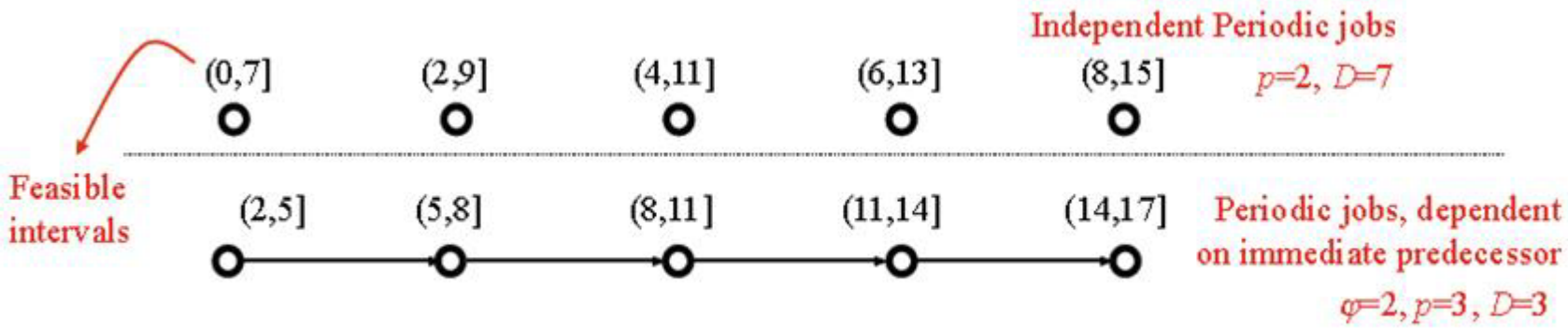
- Paràmetres funcionals
  - **Període**: Una tasca s'executa regularment cada interval de temps p_i. Cada període d'una tasca T_i, anomenat p_i, és una seqüència d'activacions (a_1, a_2, a_3, ..., a_n) cadascuna d'elles en un interval de temps.
  - **Termini de lliurament**: D_i, és l'interval de temps màxim que pot haver-hi entre l'instant d'activació i el de finalització. Normalment, és el període.
  - **Desfasament Inicial**: Senyala el desfasament que té l'inici de la primera activació de la tasca T_i respecte al temps d'inici i s'anomena ϕ_i.

- Paràmetres dependents del processador i del planificador
  - **Temps de còmput**. És el temps de CPU necessari, e_i, per completar l'execució en cadascuna de les activacions.
  - **Retard d'inici**. És el retard inicial que es correspon al temps que tarda l'activitat a ser executada ($r_{min} + [r^-, r^+]$).
  - **Temps de finalització**. És el temps en que una activitat d'una tasca T_i finalitza la seva execució.

Introducció als sistemes de temps real

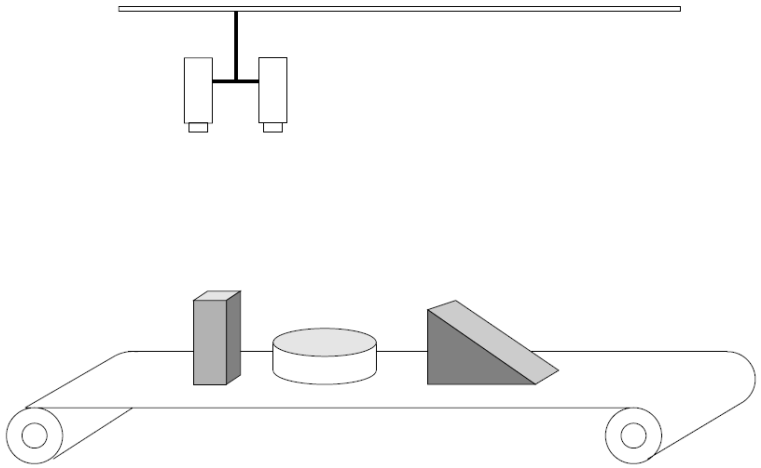
Gràfics de tasques

- Es poden representar les restriccions de precedència entre les feines en un conjunt J mitjançant un graf dirigit G = (J,<)
  - Cada node representa una feina, una aresta dirigida surt de J_i a J_k si J_i és un predecessor immediat de J_k



Introducció als sistemes de temps real

Gràfics de tasques: Exemple

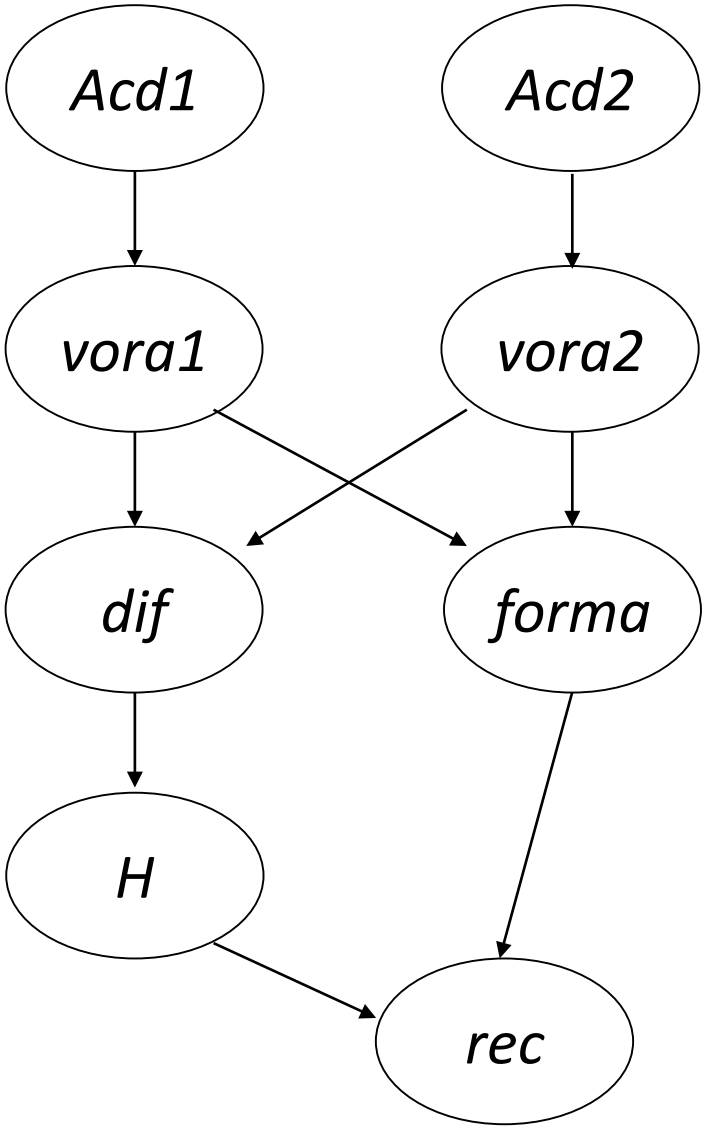


- Dues feines (una per a cada càmera) dedicades a l'adquisició de la imatge, l'objectiu és transferir la imatge des de la càmera a la memòria del processador (que s'identifiquen amb els noms ***acd1*** i ***acd2***).
- Dues feines (una per a cada imatge) dedicades al processament de la imatge de baix nivell (les operacions típiques realitzades en aquest nivell inclouen el filtratge digital, per a la reducció de soroll i detecció de contorns; podem identificar aquestes feines com ***vora1*** i ***vora2***)

- Una feina per extreure els trets bidimensionals dels contorns dels objectes (que es coneix com a ***forma***);
- Una feina per calcular les diferències dels píxels de les dues imatges (que es coneix com ***dif***);
- Una feina per determinar l'alçada de l'objecte a partir dels resultats obtinguts de la feina ***dif*** (que es coneix com ***H***);
- Una feina per realitzar el reconeixement final (aquesta feina integra les característiques geomètriques del contorn de l'objecte amb la informació d'altura i intenta fer identificar aquestes dades amb les emmagatzemades a la base de dades, es refereix com a ***rec***).

Introducció als sistemes de temps real

Gràfics de tasques: Exemple



Introducció als sistemes de temps real
Paràmetres funcionals

- Les tasques poden tenir prioritat, i en alguns casos es poden interrompre per una feina de més prioritat
  - Una feina és (*preemptable)* substituïble si la seva execució es pot interrompre
  - Una feina no és substituïble si s'ha d'executar completament una vegada s'ha iniciat
    - Moltes feines substituïbles tenen períodes durant els quals no es poden aturar, per exemple a l'hora d'accedir a certs recursos
  - La capacitat per anticipar-se a una tasca (o no) afecta a l'algorisme de planificació
  - El temps de canvi de context és el temps necessari per canviar de feina
    - Forma una sobrecàrrega que s'ha de tenir en compte al planificar les feines (treballs)
- La resposta a la pèrdua d'un temps límit pot variar
  - Algunes feines tenen components opcionals, que es poden ometre per estalviar temps (a costa d'un resultat de qualitat més pobre)
  - La utilitat dels resultats finals varia; algunes aplicacions toleren un cert retard, altres no

ESQUEMA

- Introducció als sistemes de temps real
  - Sistemes de temps real durs i tous (hard i soft)
- **Planificació de feines o treballs**
  - Algoritmes de planificació basats en rellotges
  - Algorismes de planificació basats en prioritats
    - Estàtiques
    - Dinàmiques
- Control d'accés als recursos
  - Algorismes
  - Implementació

Planificació de feines o treballs

- Planificar tasques i assignar els recursos segons un conjunt seleccionat d'algorismes de planificació i protocols de control d'accés als recursos
  - El planificador de tasques implementa aquests algoritmes
- Un planificador assigna específicament feines als processadors
- Un planificador és el distribuïdor de totes les feines del sistema als processadors disponibles.
- Un planificador vàlid satisfà les següents condicions:
  - Cada processador s'assigna a només un feina en un moment qualsevol
  - Cada feina s'assigna com a màxim a un processador en un moment qualsevol
  - Cap feina està prevista abans de la seva activació
  - La quantitat total de temps de processador assignat a cada feina és igual al seu termini màxim o temps d'execució actual
  - Tota la precedència i restriccions d'ús de recursos està satisfet

Planificació

- Un planificador és viable si cada feina compleix amb les seves limitacions temporals.
  - La taxa d'errors és el percentatge de feines que s'executen però que acaben massa tard
  - La taxa de pèrdues és el percentatge de feines que no s'executen
- Un algorisme de planificació de tasques de temps real dur, és òptim si l'algorisme de planificació sempre genera un planificador factible (viable) si el conjunt de feines planificades és viable
- Existeixen molts algorismes de planificació: l'objectiu principal d'aquesta part és la comprensió de la planificació en temps real

Planificadors

- La planificació consisteix en assignar un ordre d'execució a un conjunt de tasques que s'executen concurrentment

A la planificació hi ha 3 elements:

- Els Processadors.
- Els recursos
- Les tasques

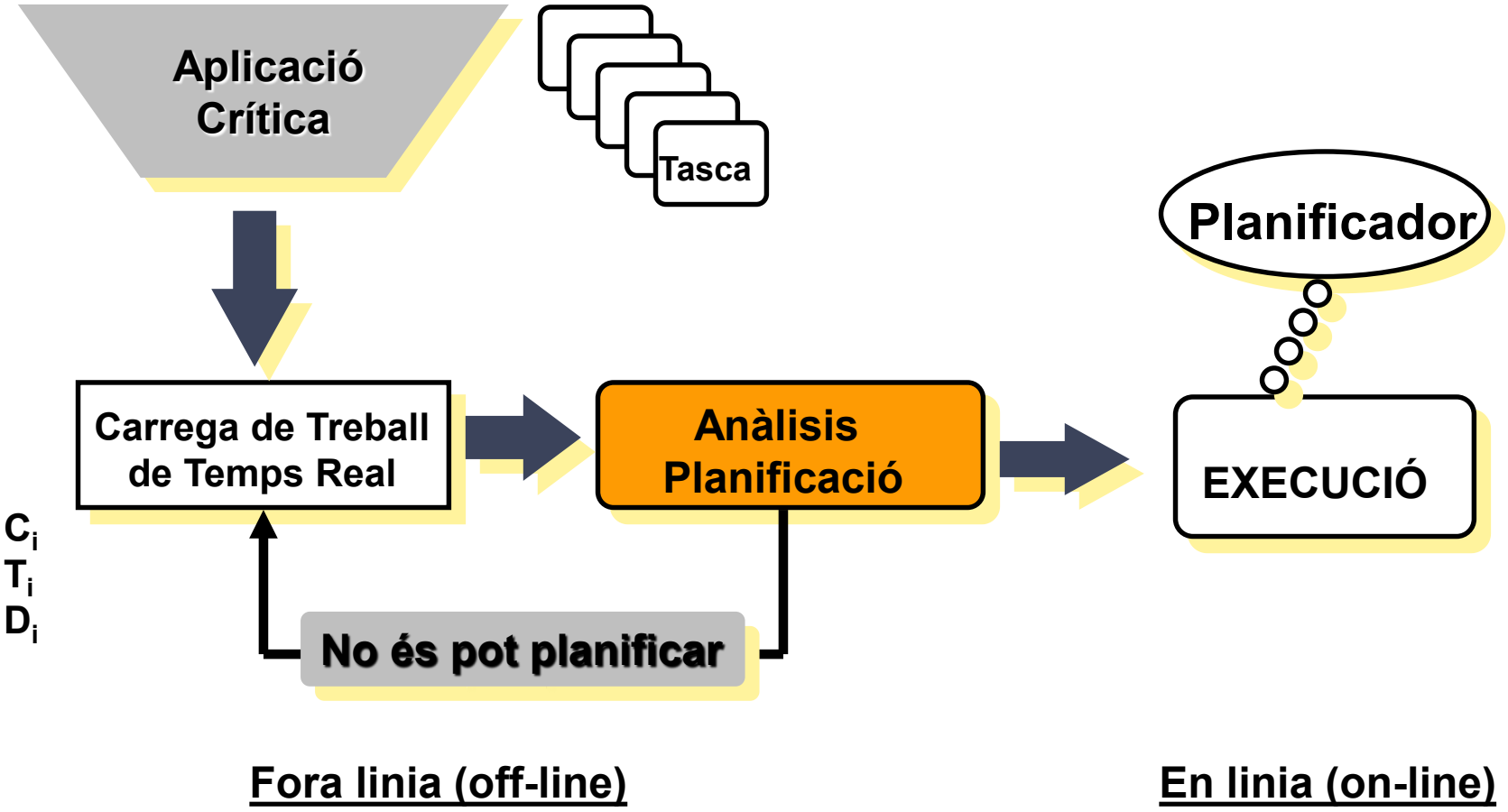
Planificador i Gestor de Recursos (Sistema Operatiu)

- Cada tasca és planificada per un algoritme de planificació i els diferents recursos s'assignen mitjançant un protocol d'accés als recursos.

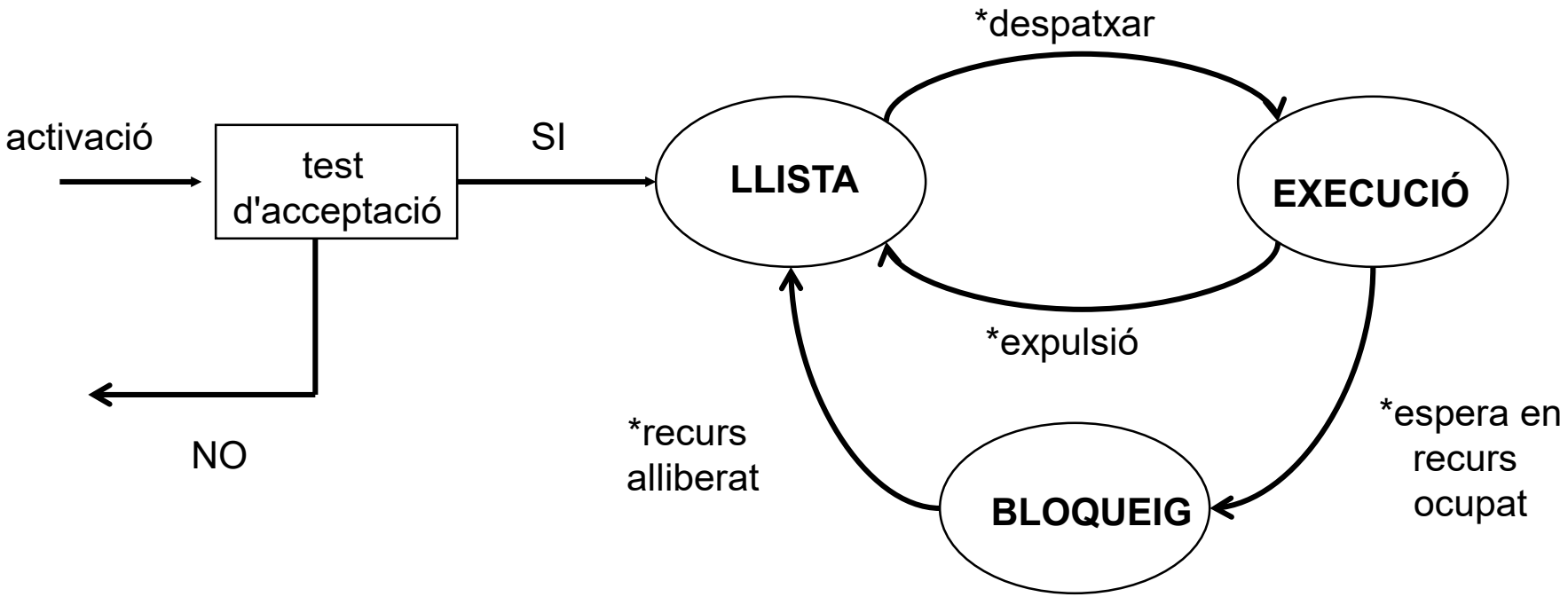
Planificadors
Mètodes

- **Planificar** l'ús dels recursos amb l'objectiu de **garantir e**ls **requisits temporals.**
- Un **mètode de planificació** té dos aspectes importants:
  - Un **algoritme de planificació** que determina l'ordre d'accés de les tasques als recursos del sistema
  - Un **mètode d'anàlisis** que permeti calcular el **comportament temporal** del sistema.
    - Per comprovar si els requisits temporals estan **garantits** en tots els casos possibles.
    - En general s'estudia el **pitjor comportament** possible.

Planificadors
Procés de planificació



Planificadors
Procés de planificació (Estats)



***decisions de planificació**

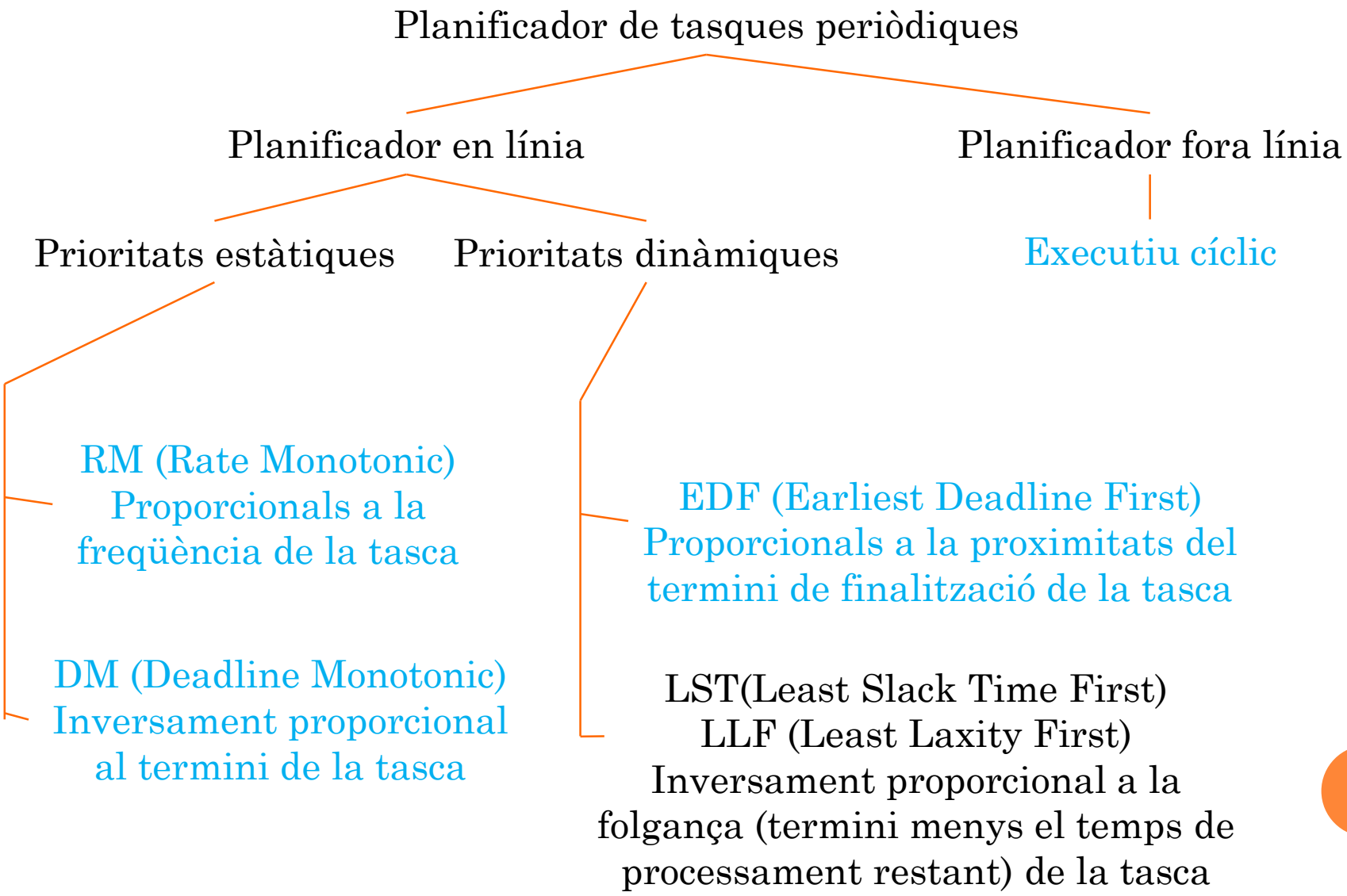
Planificadors: Temps de còmput

- Per realitzar un bon planificador s'ha de fer una estimació correcta del temps de còmput per les diferents tasques del sistema.
- Hi ha dues tècniques bàsiques:
  - Mesura del temps d'execució: És difícil de determinar el temps màxim, ja que pot ser complicat tenir en compte els efectes dels dispositius.
  - Anàlisis del codi executable: Descomposició del codi en blocs seqüencials. Calculant el temps d'execució de cada bloc, determinem el camí més llarg.

La utilització d'estructures amb temps no acotats (bucles no acotats, recursivitat, ...) dificulta l'estimació correcta d'aquest paràmetre.

Per més informació:

Planificadors
Algoritmes de planificació



Planificadors
Algoritmes de planificació

- Cíclics
  - S'utilitza principalment per a sistemes de temps real estricte en el què totes les propietats de totes les feines són conegudes en temps de disseny, de manera que es poden utilitzar les tècniques de programació fora línia
- Round-robin
  - S'utilitza principalment per a la planificació de tràfic en temps real d'alta velocitat, xarxes de commutació
- Prioritats
  - S'utilitza principalment per sistemes de temps real dinàmics amb una barreja d'activitats basades en el temps i basades en esdeveniments, on el sistema s'ha d'adaptar a les condicions canviants i esdeveniments

Planificadors: Cíclic

- Decisions sobre quines feines s'executen, quan es realitzen, en quins instants de temps específics
  - Aquests instants són elegits abans que el sistema inicii l'execució
  - En general, regularment espaiats, s'implementen utilitzant una interrupció d'un temporitzador periòdic (cicle secundari)
    - El planificador es desperta després de cada interrupció, planifica la feina a executar per al proper període, i després es bloqueja fins a la següent interrupció
- Normalment, en els sistemes cíclics:
  - Tots els paràmetres de les feines de temps real són fixes i coneguts
  - La planificació de les feines es calcula fora de línia i s'emmagatzema per al seu ús en temps d'execució, com a resultat, la sobrecàrrega de la planificació en temps d'execució es pot minimitzar
  - Simple i senzill, no és flexible

Planificació cíclica

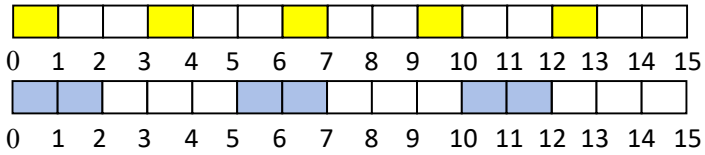
- La planificació cíclica consisteix en confeccionar un pla d'execució fixa.
- Funcionament:
  - Obtenir l'hiperperíode (cicle en el que es repeteix el comportament del sistema):
    - $H = mcm\ (p_i)$
  - Obtenir el període secundari: (molts cops coincideix amb el període més petit) -> Només si és harmònic.
    - $p_{Min} = min(p_i)$
  - Dividir l'hiperperíode pel període mínim $K = H/p_{Min}$ cicles secundaris: en cadascun d'ells es podran executar les feines corresponents a determinades tasques.

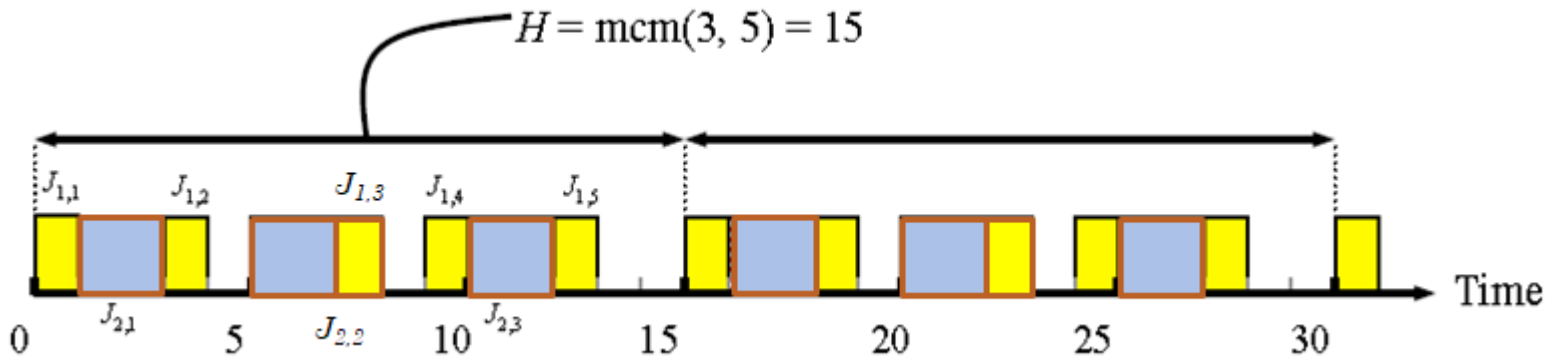
Planificadors cíclics

- El hiper-període d'un conjunt de tasques periòdiques és el mínim comú múltiple dels períodes:
  - H = mcm (p_i) per i = 1, 2, ..., n
  - Temps després del qual el patró del temps d'execució/activació de la feina es comença a repetir, anàlisi límit necessari
- exemple:

$T_1 : p_1 = 3, e_1 = 1$

$T_2 : p_2 = 5, e_2 = 2$





Planificadors cíclics: Utilització

- La relació $u_i = e_i/p_i$ és la utilització de tasca T_i
  - És la fracció de temps que manté ocupat el processador amb una tasca periòdica que té un període p_i i un temps d'execució de e_i
- La utilització total d'un sistema és la suma de les totes les utilitzacions de les tasques d'un sistema: $U = \Sigma u_i$
- Normalment, assumirem que el termini d'execució d'una feina de la tasca és igual al període de la tasca
  - De vegades pot ser més curt que el període, donant lloc als temps d'inactivitat
- En el món real, els sistemes que s'ajusten a aquest model son molt útils i és fàcil de raonar sobre aquestes tasques periòdiques

Planificació cíclica: Cicle secundari

m -> possibles cicles secundaris

- $m \leq \min(D_i)$; menor que tots els terminis de finalització
- $m \geq \max(e_i)$; més gran que tots els temps d'execució
- $\exists\ k : H = k \cdot m$; Ha de ser un divisor del cicle principal (Hiperperíode)
- $\forall i: 2 \cdot m - mcd(m, p_i) \leq D_i$
  - Garanteix que entre l'instant d'activació de cada procés i el seu termini de finalització hi hagi un cicle secundari complert.
  - $m - mcd(m, p_i)$ retard màxim entre l'activació i l'inici del següent cicle.

Podeu fer-vos un programa.

Ex:

T	e	p	D	U
A	1	15	14	1/15
B	2	20	26	2/20
C	3	22	22	3/22
				0,303

$H = mcm(15,20,22) = 3 \cdot 4 \cdot 5 \cdot 11 = 660$

$m \leq \min(14,26,22) \rightarrow m = 1..14$

$m \geq \max(1,2,3) \rightarrow m = 3..14$

$\exists\ k\ 660 = k \cdot m \rightarrow m = 3,4,5,6,10,11,12$

$2m - mcd(m,p_i) \leq D_i \rightarrow m = 3,4,5,6$

Per m = 3

Nº d'execucions de cada procés:

$n_{ei} = M/p_i$

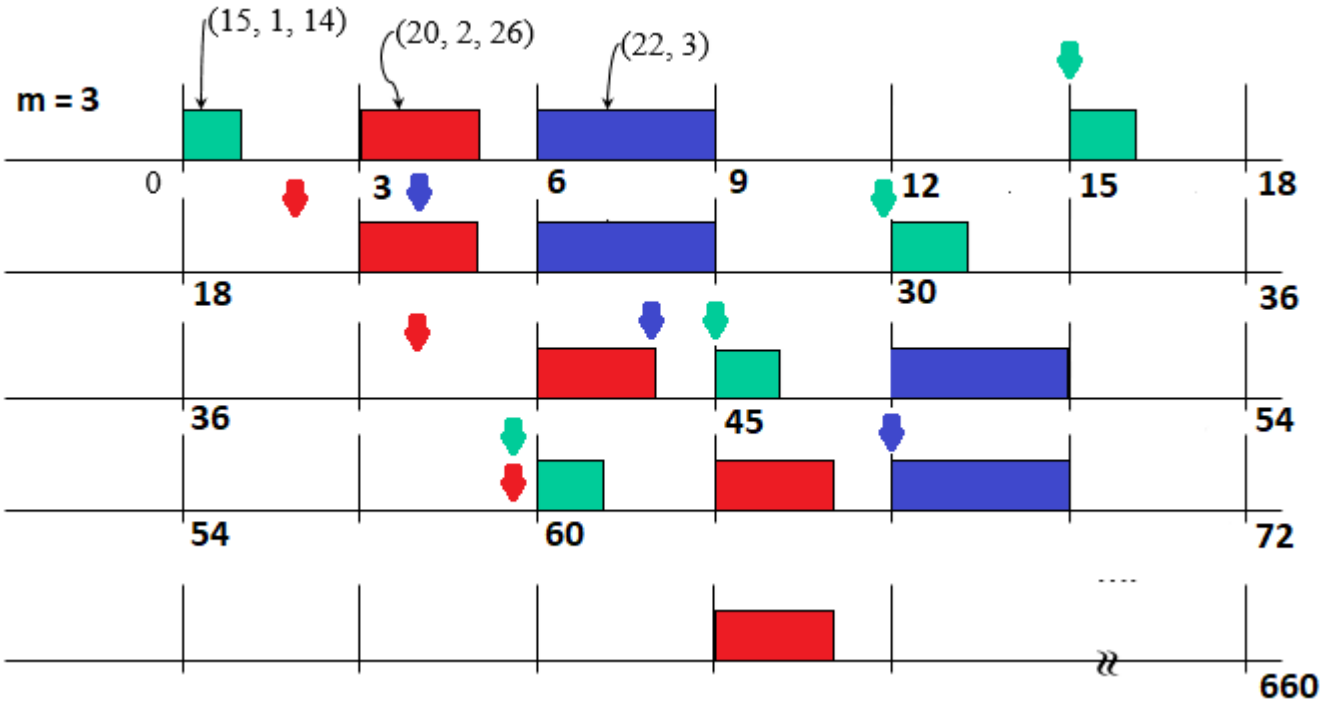
$n_{eA} = 660/15 = 44$

$n_{eA} = 660/20 = 33$

$n_{eA} = 660/22 = 30$

i	p_i	mcd	2m-mcd		D_i
A	15	3	3	<	14
B	20	1	5	<	26
C	22	1	5	<	22

K = H/m = 660/3 = 220 cicles secundaris



Per m = 4

Nº d'execucions de cada procés:

$n_{ei} = M/p_i$

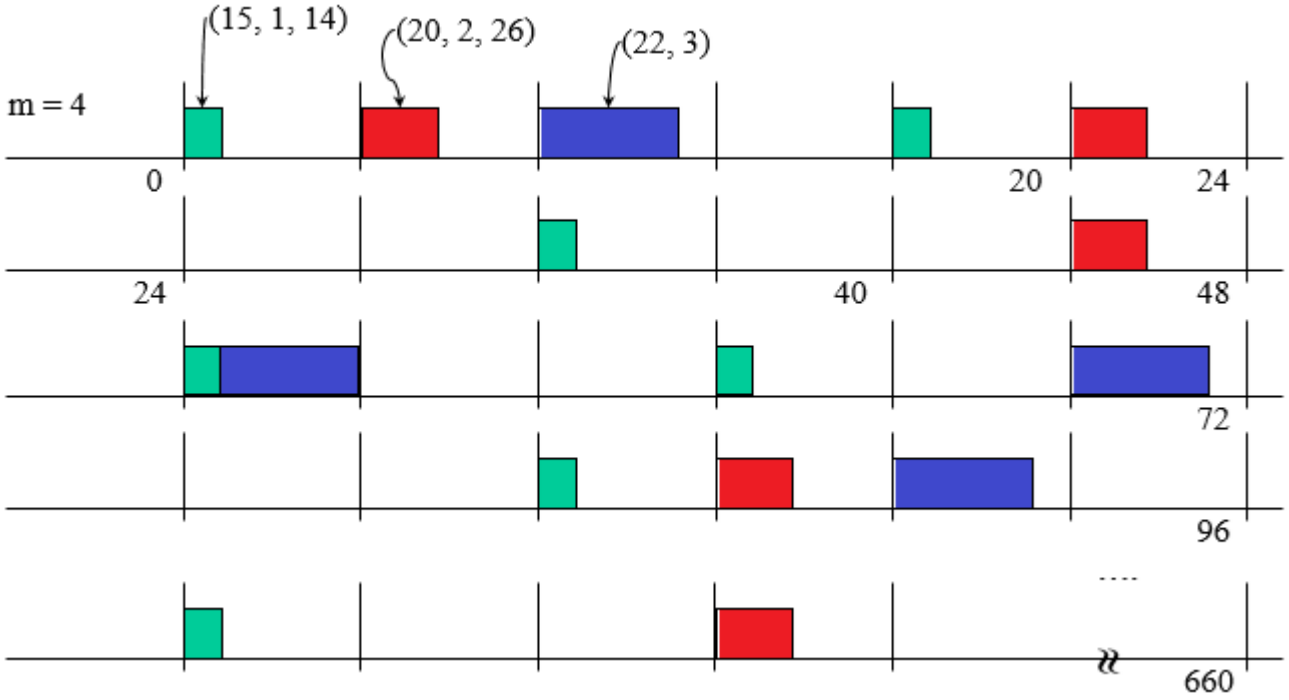
$n_{eA} = 660/15 = 44$

$n_{eA} = 660/20 = 33$

$n_{eA} = 660/22 = 30$

i	p_i	mcd	2m-mcd		D_i
A	15	1	7	<	14
B	20	4	4	<	26
C	22	2	6	<	22

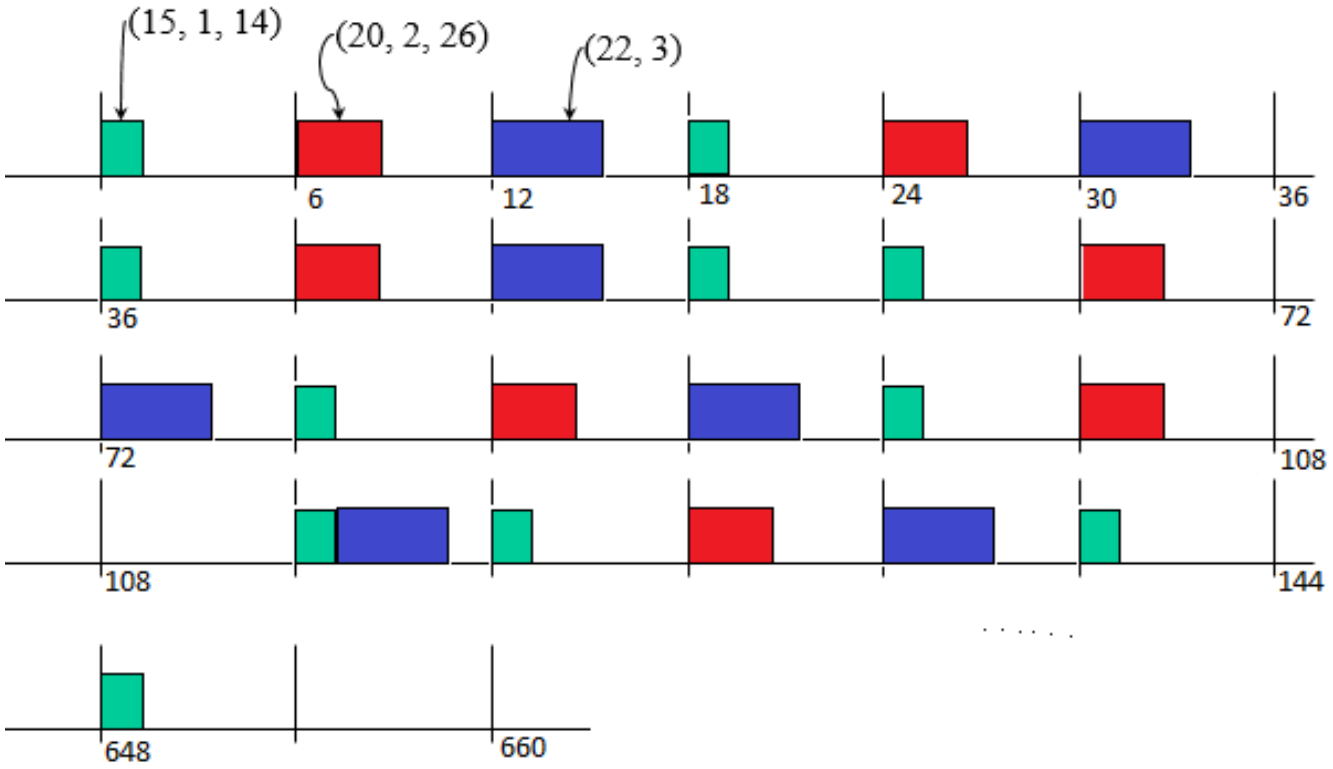
K = H/m = 660/4 = 165 cicles secundaris



Per m = 6

i	Ti	mcd	2m-mcd		Di
A	15	3	9	<	14
B	20	2	10	<	26
C	22	2	10	<	22

K = H/m = 660/6 = 110 cicles secundaris



~~Per m = 10~~

i	Ti	mcd	2m-mcd		Di
A	15	5	15	<	14
B	20	10	10	<	26
C	22	2	18	<	22



~~Per m = 11~~

i	Ti	mcd	2m-mcd		Di
A	15	1	21	<	14
B	20	1	21	<	26
C	22	11	18	<	22


~~Per m = 12~~

i	Ti	mcd	2m-mcd		Di
A	15	3	21	<	14
B	20	4	2	<	26
C	22	2	20	<	22


Planificació cíclica (Exemple)

Tasca	període	Temps d'execució
T1	25	10
T2	25	8
T3	50	5
T4	50	4
T5	100	2

- Hiperperíode (H) = mcm(p_i) = 100
- Hi ha 4 cicles secundaris de 25 unitats de temps

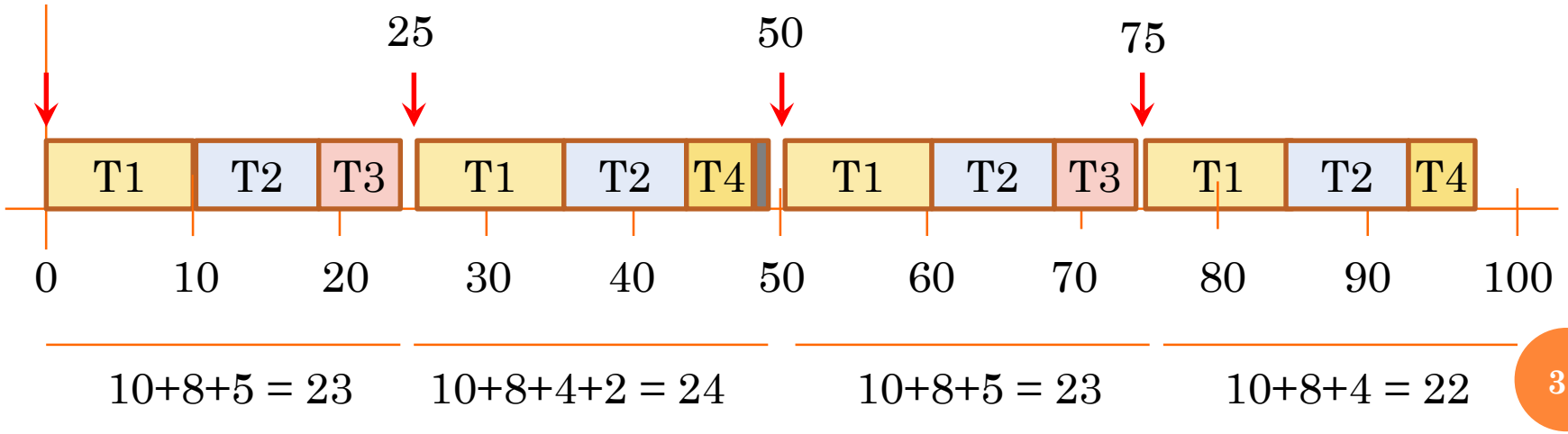
$m \leq \min(25,.., 100) \rightarrow m = 1..25$

$m \geq \max(10, 8,.., 2) \rightarrow m = 10..25$

$\exists\ k\ 100 = k \cdot m \rightarrow m = 10, 20, 25$

$2m - mcd(m, p_i) \leq D_i \rightarrow m = 10, 25$

Per m = 25 → 4 cicles secundaris



Planificació cíclica (Característiques)

- A la planificació cíclica no hi ha concurrència: cada cicle secundari és una seqüència de crides a les tasques.
- No es necessiten procediments d'exclusió mútua per compartir dades.
- No és necessari analitzar el comportament temporal: és correcte per disseny.
- Els períodes han de ser harmònics
- La planificació cíclica és fàcil de construir.
- És poc flexible i difícil de mantenir

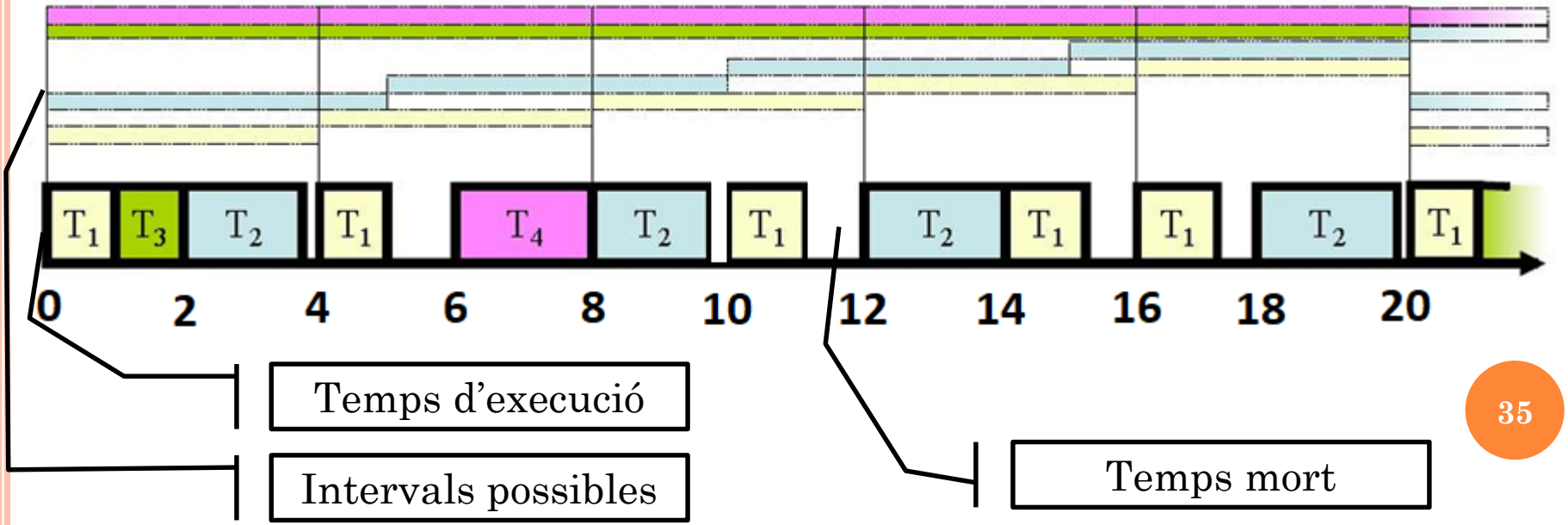
Planificació cíclica (Exemple)

- Considerem un sistema amb 4 tasques paròdiques independents:
  - T1 = (4, 1.0) (Temps d'execució)
  - T2 = (5, 1.8) (Període)
  - T3 = (20, 1.0)
  - T4 = (20, 2.0)
  - Hiper-període H = 20; mcm(4,5,20,20)
  - Es pot construir un programa estàtic arbitrari per complir amb tots els terminis

Cicle secundari

$m = 2$

És l'únic que compleix



Planificació cíclica (Exemple)

k	t_k	$T(t_k)$
0	0.0	T_1 i T_3
1	2.0	T_2
2	4.0	T_1
3	6.0	T_4
4	8.0	T_2
5	10.0	T_1
6	12.0	T_2
7	14.0	T_1
8	16.0	T_1
9	18.0	T_2

- Planificació calculada com una taula
- El sistema crea totes les tasques que s'han d'executar:
  - Assigna suficient memòria per al codi i les dades de cada tasca
  - Porta el codi executat per la tasca a la memòria
- El planificador de tasques posa el temporitzador de maquinari per interrompre a la primera decisió temporal, $t_k = 0$.
- En rebre una interrupció en t_k:
  - El planificador estableix el límit del temps d'interrupció a t_k +2
  - En cas de sobrepassar la tasca anterior, gestionar el fracàs
  - Si una tasca T (t_k) finalitza, i una tasca aperiòdica s'està esperant, iniciar-la.
  - En cas contrari, iniciar la següent feina de la tasca T (t_k) que executa

Planificació cíclica (Exemple)

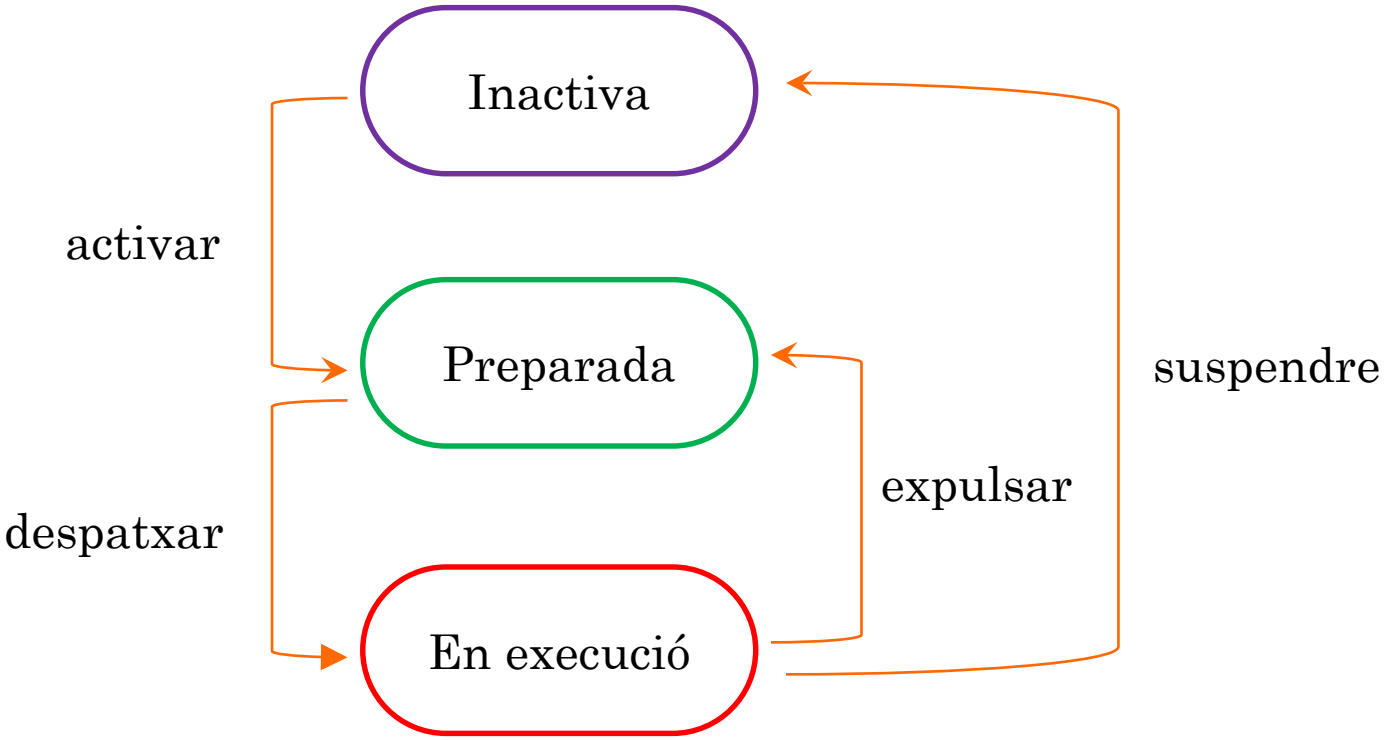
```
procedure Cyclic_Executive  is
   type  Cycle  is  mod  10;
   Frame  ;  Cycle  : =  0;

Interrupt_2;
   case Frame is
          when  0  =>  T1; T3;
          when  1  =>  T2;
          when  2  =>  T1;
          when  3  =>  T4;
          when  4  =>  T2;
          when  5  =>  T1;
          when  6  =>  T2;
          when  7  =>  T1;
          when  8  =>  T1;
          when  9  =>  T2;
   end case;
   Frame  : = Frame  +  1;
   end;

end Cyclic_Executive;
```

Planificació per prioritats

- Les tasques es modelen com a processos concurrents.
- Les tasques executables es van distribuint "dispatcher" en ordre de la prioritat.
- La distribució pot ser expulsiva (Preemptive Dispatching) o no expulsiva (non preemptive dispatching).



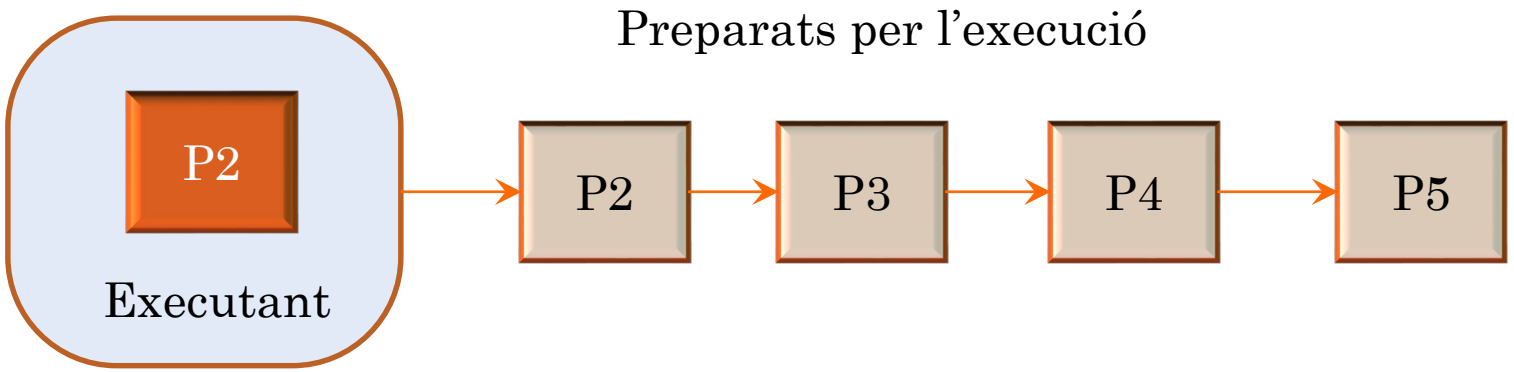
Planificadors Prioritat

Que ha de fer:

- Assignar prioritats a les tasques, basat en algun algoritme
- Prendre decisions de planificació de tasques basades en les prioritats, quan succeeixen esdeveniments com alliberaments i finalitzacions de feines
  - Els Algoritmes de planificació de tasques de prioritats son per esdeveniments
  - Les feines es col·loquen en una o més cues, per cada esdeveniment, i s'executa la feina preparada amb més prioritat
  - L'assignació de les feines a les cues de prioritat, juntament amb les regles de preferència, es defineix en un algorisme de planificació de prioritat
- Els algoritmes de prioritat prenent decisions locals òptimes per decidir quina és la tasca que s'ha d'executar.
  - Les decisions òptimes locals de planificació normalment no són òptimes globalment
  - Els algoritmes de prioritat mai deixen intencionalment cap recurs inactiu
    - Deixant de recurs inactiu no és localment òptim

Planificació per prioritats

- A cada tasca s'assigna una prioritat, en funció de la seva importància relativa.
- Es dona al processador la tasca amb més prioritat, expulsant la de menys prioritat

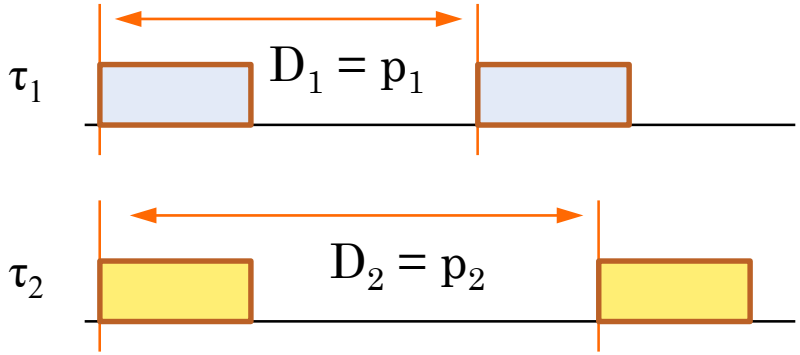


- Les prioritats poden ser:

- Fixes: Es calculen un sol cop, s'assignen a la tasca i es manté durant tota la seva vida
- Dinàmiques: Es calculen en temps d'execució, segons l'estat de la tasca i del sistema

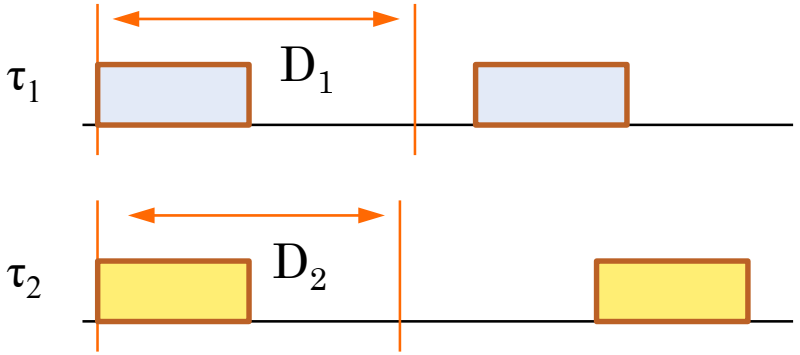
Planificació per prioritats: Fixes

- Es dos mètodes més coneguts:
  - RM (Rate Monotonic): Proporcional a la freqüència de la tasca
  - DM (Deadline Monotonic): Inversament proporcional al termini de la tasca
- Limiten l'ús del processador
- Garanteixen els terminis en sobrecàrrega
- Poden ser modificats per gestionar la sincronització entre tasques, tasques aperiòdiques, i altres situacions



Per RM → T1 més prioritari

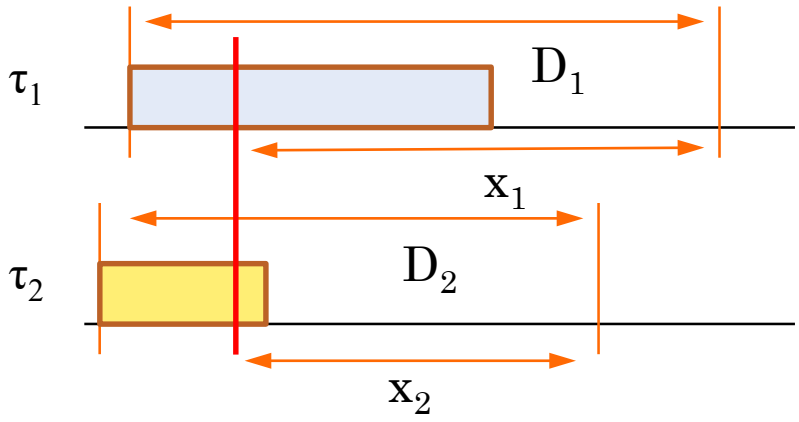
Per DM → T1 més prioritari



Per DM → T2 més prioritari

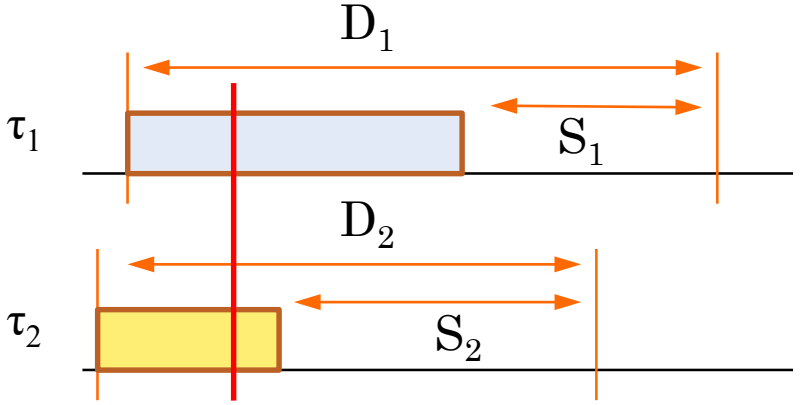
Planificació per prioritats: Dinàmiques

- Es dos mètodes més coneguts:
  - EDF (Earliest Deadline First): Proporcional a la proximitat del termini de la tasca
  - LLF (Least Laxity First): Inversament proporcional a la folgança (termini menys el temps de processament restant) de la tasca
- S'aconsegueix una millor utilització del processador
- El comportament en sobrecàrregues no és previsible
- No està ben resolta la interrelació entre tasques



Per EDF → T2 més prioritari

$x_2 < x_1$



Per LLF → T1 més prioritari

$S_1 < S_2$

Planificació per prioritats (RM)

Planificador Rate Monotonic RM

- Totes les tasques són periòdiques i independents unes de les altres
- El termini de finalització de cada tasca és igual al seu període. $\mathbf{D_i = p_i}$
- L'assignació de prioritats es farà de forma inversa al període. p↓ → P↑
- El planificador serà **expulsiu**
- El temps de canvi de context es considera menyspreable
- El conjunt de tasques és síncron

El planificador RM no és un pla planificador òptim en el cas general, però sí que ho és quan les tasques són periòdiques simples.

Planificació per prioritats (RM)

- Assignació per prioritats monòtones en freqüència (Rate Monotonic): consisteix en assignar major prioritat (estàtica) a les tasques de menor període.
  - Òbviament, si tractem de satisfer els terminis començant per aquelles tasques que són més crítiques, aconseguirem anar complint tots els terminis a què està sotmès el sistema.
- Aquest algoritme de planificació és òptim per a tasques periòdiques independents sense termini de finalització.
  - Si no és així, el sistema no serà planificable, independentment del algoritme de planificació utilitzat. Així mateix, si podem garantir els terminis d'un sistema de tasques amb un altre algorisme de planificació, també el podrem garantir amb el Rate Monotonic.
- El RM ha estat elegit com a algoritme de planificació en projectes de gran envergadura (NASA, IEEE Futurebus +, etc.).
- Encara que el RM és òptim, no sempre un conjunt de tasques (periòdiques independents sense termini de finalització) és planificable utilitzant el RM.
- Per esbrinar si ho són utilitzarem tres tècniques:
  - Per mitjà del test de garantia (condició suficient).
  - Per la cota hiperbòlica (condició suficient)
  - Mitjançant el càlcul del temps de resposta (condició necessària i suficient).

Planificació per prioritats :Algoritme RM

- *Definició*: Un conjunt de N tasques és **periòdic simple** quan per a qualsevol parell de tasques T_i, T_k amb $p_i < p_k$, llavors p_k és un nombre enter de vegades de p_i.
  - Per exemple: T1 = (4, 1.0); T2 = (20, 1.0) => p2 és 5 vegades p1

- ***Teorema***: Un sistema de tasques periòdiques simple, independents i interrompibles, on els seus terminis de finalització són majors o iguals que els seus períodes, és planificable en un processador d'acord amb l'algoritme RM, si i només si **el factor d'utilització total és menor o igual que un**.

Planificació per prioritats
Test de garantia (Factor d'utilització)

- Teorema (Liu & Layland [1973]):
- Un conjunt de N tasques periòdiques $\tau_1, \tau_2, \ldots, \tau_n$, amb $\mathbf{D_i = p_i}$, $1 \leq i \leq n$, és planificable per l'algoritme de planificació monòtona (RM) si es compleix:

$$U = \sum_{i=1}^{n} \left(\frac{e_i}{p_i}\right) \leq n * \left(2^{1/n} - 1\right), n = 1,2 \ \ldots$$

Test de garantia

$$U^*(n) = n\left(2^{1/n} - 1\right)$$

És el factor d'utilització del processador per n tasques, que assegura la planificabilitat

$$U^*(\infty) = U^* = ln\ 2 = 0{,}693$$

Qualsevol conjunt de feines periòdiques es pot planificar amb RMS si no utilitza el processador més del 69.3%

n	*U*(n)*
1	*1.0*
2	*0.828*
3	*0.780*
4	*0.756*

Planificació per prioritats
Exemple 1

$$u_i = e_i / p_i$$

T	p	e	P	U
T_1	30	10	3	
T_2	40	10	2	
T_3	50	12	1	

Test de garantia

$$U^*(n) = n\left(2^{1/n} - 1\right)$$

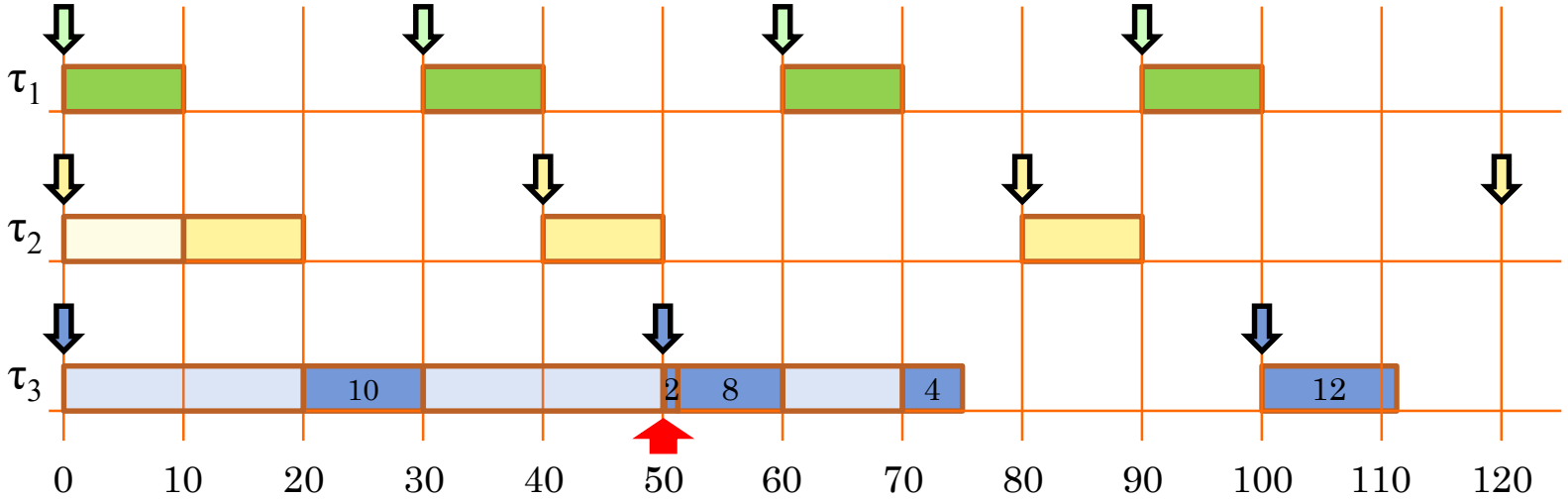
$$U^*(3) = 3(2^{1/3}-1) = 0{,}780$$

$$U = 0{,}823 > U^*(3) = 0{,}780$$

No compleix

No és periòdic simple

H = mcm (30, 40, 50) = 600



Fallada

Planificació per prioritats
Exemple 2

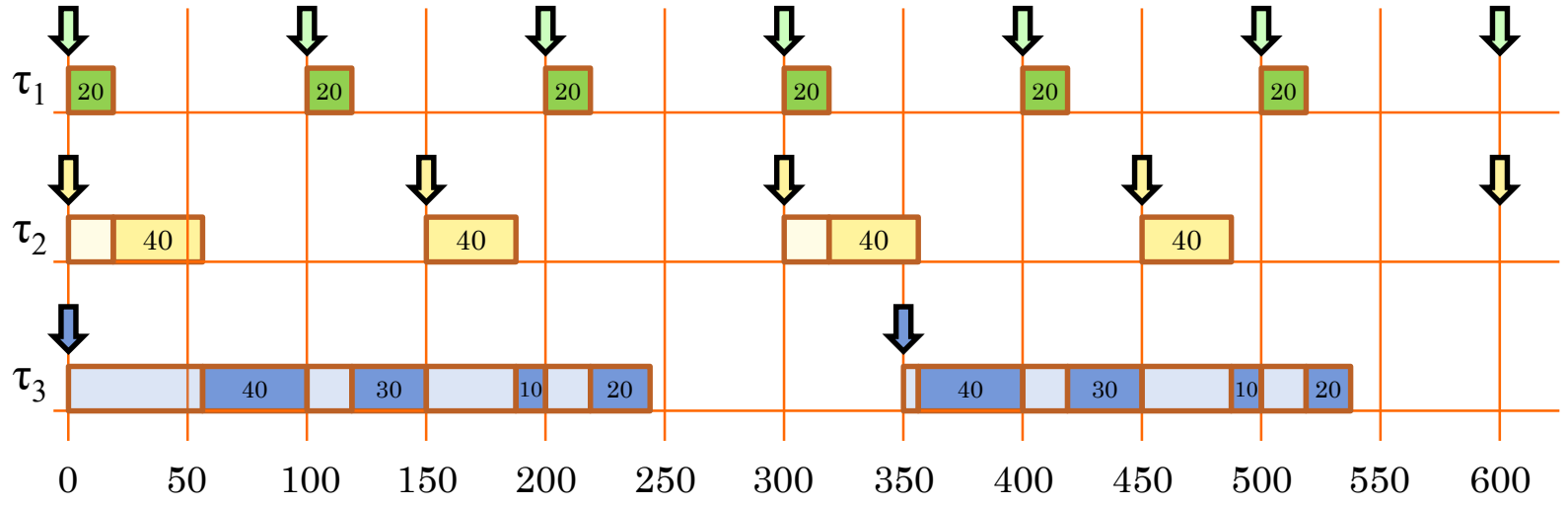
T	p	e	P	U
T_1	100	20	3	20/100 = 0,200
T_2	150	40	2	40/150 = 0,267
T_3	350	100	1	100/350 = 0,286
				0,753

$$u_i = e_i / p_i$$

U(3) = 0,780*
U = 0,753 < U(3)*

Complex

mcm (100, 150, 350) = 2100



Cota Hiperbòlica per RM

- Teorema (Bini & Buttazzo 2001). Defineix una cota hiperbòlica donant una condició de planificabilitat de suficiència.

Un conjunt de N tasques periòdiques τ_1, τ_2, ..., τ_n, on cada tasca τ_i es caracteritza amb una utilització de U_i. El conjunt de tasques T serà planificable per RM si compleix:

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

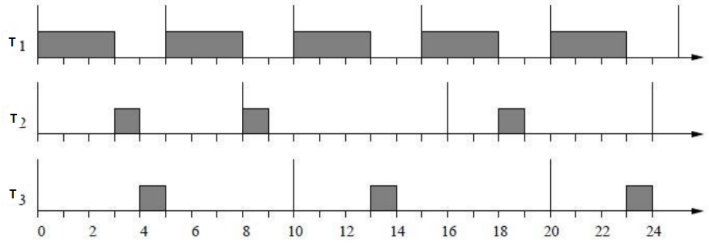
Tasca	e_i	p_i	P	U
T1	3	5	3	0,6
T2	1	8	2	0,125
T3	1	10	1	0,1

$$U = 0{,}825 > 0{,}78$$

No compleix el test de garantia

$$\prod_{i=1}^{3}(U_i + 1) = (0{,}6 + 1) \cdot (0{,}125 + 1) \cdot (0{,}1 + 1) = 1{,}6 \cdot 1{,}125 \cdot 1{,}1 = 1{,}98 < 2$$

Compleix el test de la cota hiperbòlica → Es planificable per RM



Cota Hiperbòlica per RM

Tasca	e_i	p_i	P	U
T1	1	4	3	0,25
T2	2	6	2	0,33
T3	3	10	1	0,3

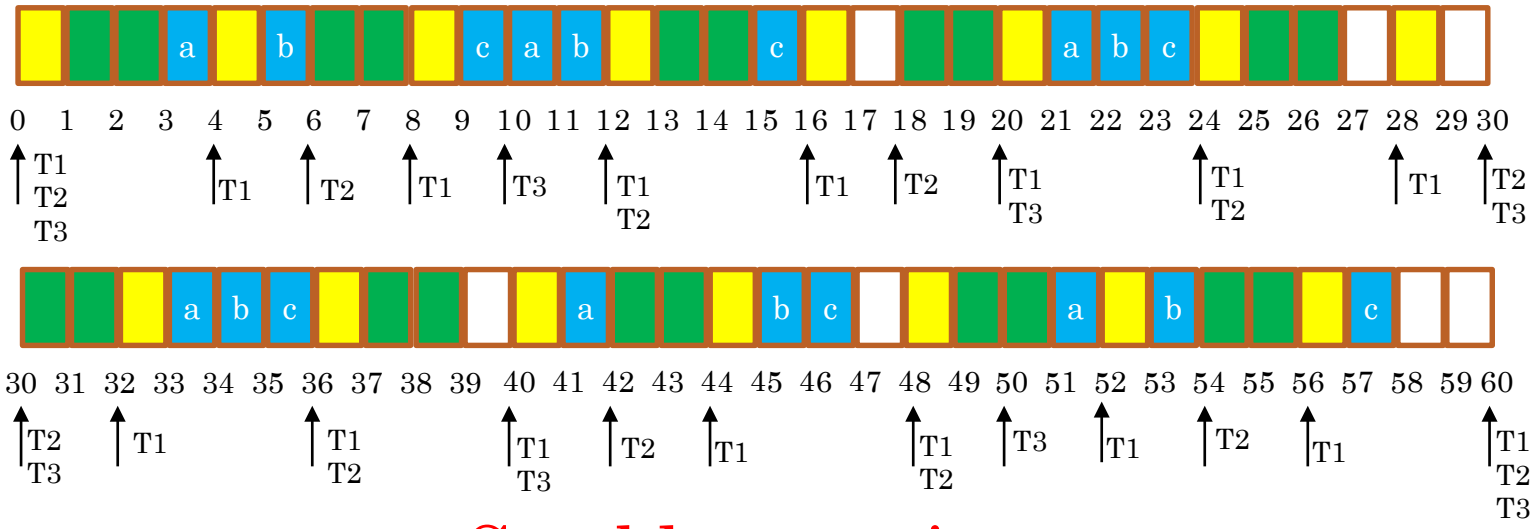
$$U = 0{,}883 > 0{,}78$$

No compleix el test de garantia.
Podem mirar la cota hiperbòlica

$$\prod_{i=1}^{3}(U_i + 1) = (0{,}25 + 1) \cdot (0{,}33 + 1) \cdot (0{,}3 + 1) = 1{,}25 \cdot 1{,}33 \cdot 1{,}3 = 2{,}16 > 2$$

No compleix el test de la cota hiperbòlica

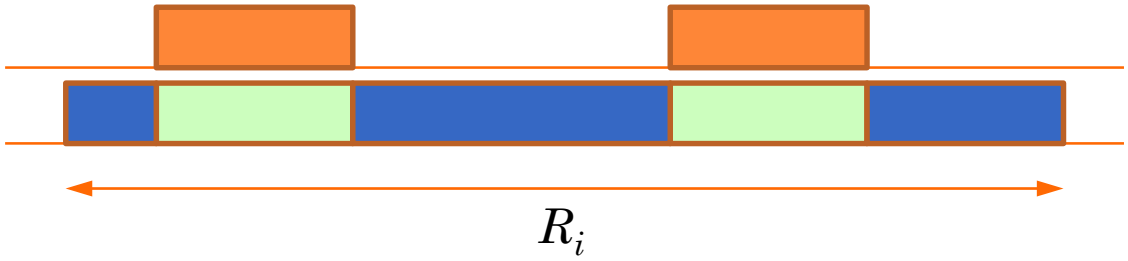
És planificable?



Sembla que si....

Planificació per prioritats
Temps de resposta

- El test de garantia basat en el factor d'utilització i la cota hiperbòlica tenen dos inconvenients molt importants:
  - Son inexactes. Donen una condició suficient però no necessària.
  - No es poden aplicar a un model de processos més general, en el que les prioritats no s'assignen per RM generalitzar-se per models de tasques més complexes
- El temps de resposta (R_i) d'una tasca T_i és la suma del seu temps d'execució e_i més les interferències que pot tenir.



- Un conjunt de tasques periòdiques independents serà planificable només si es compleix que:

$$R_i \leq p_i \qquad \text{per a cada tasca } T_i$$

- S'ha de calcular per a cada tasca el seu temps de resposta. Si compleix la condició el sistema serà planificable.

Planificació per prioritats Temps de resposta

- El temps de resposta de la tasca més prioritària, és: R = e
- El temps de resposta d'una tasca T_i, és: $R_i = e_i + I_i$

  on I_i és la interferència màxima que pot tenir la tasca T_i degut a tasques més prioritàries.
- I_i es donarà, quan totes les tasques amb més prioritat que T_i s'executen en el mateix instant que ho fa la tasca T_i (instant crític)
- La interferència màxima que pot tenir la tasca T_i degut a la tasca T_j, és:

Nombre d'activacions

$$I_{i,j} = \left\lceil \frac{R_i}{p_j} \right\rceil e_j$$

on $\lceil R_i/p_j \rceil$ (funció sostre) és el nombre de cops que s'executa una tasca T_j (de més prioritat que i) dins del interval R_i

- El total d'interferències que podrà tenir la tasca i serà (equació de [Joseph & Pandya 1986]):

$$I_i = \sum_{j \in ps(i)} I_{i,j} = \sum_{j \in ps(i)} \left\lceil \frac{R_i}{p_j} \right\rceil e_j$$

  - On *ps(i)* és el conjunt de tasques de mes prioritat que la tasca T_i

Planificació per prioritats Temps de resposta

- Substituint aquest valor a l'expressió del temps de resposta,

$$R_i = e_i + \sum_{j \in ps(i)} \left\lceil \frac{R_i}{p_j} \right\rceil e_j$$

Tot i que la formulació de l'equació de la interferència és exacta, el valor de la interferència segueix sent desconegut ja que està en funció de R_i que és just el que volem calcular.

L'equació final té als dos costats el valor R_i, però és difícil d'extreure a causa del operador sostre. En general hi haurà més d'un valor que doni solució a aquesta equació, però a nosaltres només ens interessarà el valor més petit. Tots són vàlids per al temps de resposta en el pitjor dels casos de la tasca T_i, però perquè es compleixin els terminis n'hi haurà prou que un sigui menor o igual que D_i (termini màxim d'execució) , i si n'hi ha algun sempre serà el menor.

Una manera senzilla de resoldre l'equació és per mitjà d'una relació de recurrència (mecanisme iteratiu).

Planificació per prioritats Temps de resposta

- Teorema ([Audsley et al. 1993]): Un conjunt de N tasques periòdiques independents sense termini de finalització, serà planificable si i només si es compleix que, per a cadascuna de les tasques T_i, es garanteixen els terminis de la l'equació:

$$\omega_i^{n+1} = e_i + \sum_{j \in ps(i)} \left\lceil \frac{\omega_i^n}{p_j} \right\rceil e_j$$

- On $\omega_i^0 = e_i$, $\omega_i^1 = e_i + \sum_{j \in ps(i)} \left\lceil \frac{\omega_i^0}{p_j} \right\rceil e_j$, ...

Que és una successió monòtonament creixent.

- Finalitza quan:
  - Per tot n: $\omega_i^n > D_i \Rightarrow$ Terminis no garantits
  - Per tot n: $\omega_i^{n+1} = \omega_i^n \Rightarrow$ Terminis garantits ($R_i = \omega_i^n$)

Si l'equació no té solució, la successió anirà creixent indefinidament (passarà per tasques de baixa prioritat si el factor d'utilització sobrepassa el 100 %). Si s'arriba a un valor superior al període de la tasca, es pot determinar que no complirà amb el seu límit temporal.

Temps de resposta per RM

Tasca	e_i	p_i	P	U
T1	1	4	3	0,25
T2	2	6	2	0,33
T3	3	10	1	0,3

$$R_1 = \omega_1 = e_1 = 1$$

$$\omega_2^0 = e_2 = 2$$

$$\omega_2^1 = e_2 + \left\lceil \frac{\omega_2^0}{p_1} \right\rceil \cdot e_1 = 2 + \left\lceil \frac{2}{4} \right\rceil \cdot 1 = 3 \quad \Leftarrow R_2$$

$$\omega_2^2 = e_2 + \left\lceil \frac{\omega_2^1}{p_1} \right\rceil \cdot e_1 = 2 + \left\lceil \frac{3}{4} \right\rceil \cdot 1 = 3$$

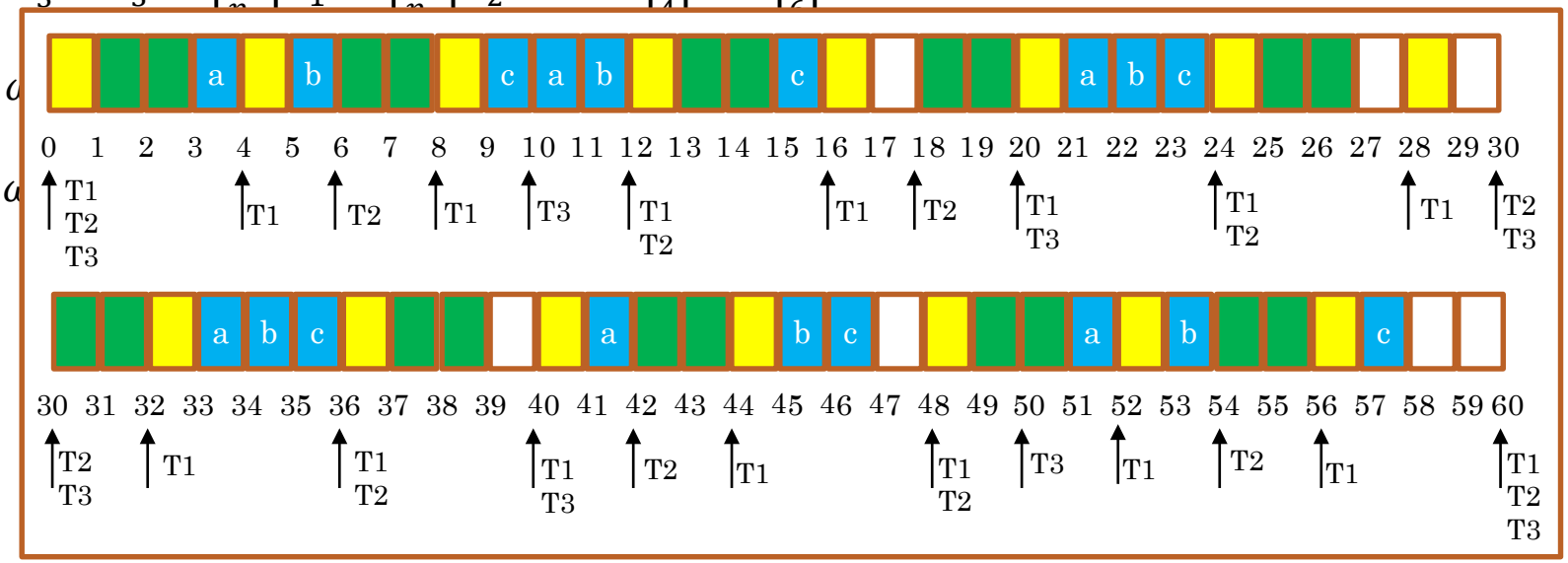
$$\omega_3^0 = e_3 = 3$$

$$\omega_3^1 = e_3 + \left\lceil \frac{\omega_3^0}{p_1} \right\rceil e_1 + \left\lceil \frac{\omega_3^0}{p_2} \right\rceil e_2 = 3 + \left\lceil \frac{3}{4} \right\rceil 1 + \left\lceil \frac{3}{6} \right\rceil 2 = 3 + 1 + 2 = 6$$

$$\omega_3^2 = e_3 + \left\lceil \frac{\omega_3^1}{p_1} \right\rceil e_1 + \left\lceil \frac{\omega_3^1}{p_2} \right\rceil e_2 = 3 + \left\lceil \frac{6}{4} \right\rceil 1 + \left\lceil \frac{6}{6} \right\rceil 2 = 3 + 2 + 2 = 7$$

$$\omega_3^3 = e_3 + \left\lceil \frac{\omega_3^2}{p_1} \right\rceil e_1 + \left\lceil \frac{\omega_3^2}{p_2} \right\rceil e_2 = 3 + \left\lceil \frac{7}{4} \right\rceil 1 + \left\lceil \frac{7}{6} \right\rceil 2 = 3 + 2 + 4 = 9$$

Es planificable



Planificació per prioritats Temps de resposta

- Problema

Tasques	Període (p)	Temps d'execució (e)	Prioritat (P)	Utilització (U)	
T1	7	3	3	3/7 =	0,42857
T2	12	3	2	3/12 =	0,25
T3	20	5	1	5/20 =	0,25
					0.92857

La condició de planificabilitat

Hiper-període (H) mcm(p_i)

H = $2^2 \cdot 3 \cdot 5 \cdot 7 = 420$

Test de garantia

$$U = \sum_{i=1}^{3} \left(\frac{e_i}{p_i}\right) = 0{,}92857 \leq 3 * \left(2^{1/3} - 1\right) = 0{,}779$$

No compleix

Temps de resposta

Tasques	p	R
T1	7	3
T2	12	6
T3	20	20

Cota hiperbòlica

$$\prod_{i=1}^{3} (U_i + 1) = 2{,}232 > 2$$

No compleix

Temps de resposta

$Tasca\ 1$ És la més prioritària. No té interferències d'altres tasques

$$R_1^0 = e_1 = 3$$

$Tasca\ 2$ Només té interferències de la tasca 1

$$\omega_2^0 = e_2 = 3$$

Ull. Funció sostre (ceil)

$$\omega_2^1 = e_2 + \left\lceil \frac{\omega_2^0}{p_1} \right\rceil \cdot e_1 = 3 + \left\lceil \frac{3}{7} \right\rceil \cdot 3 = 3 + 3 = 6$$

$$\omega_2^2 = e_2 + \left\lceil \frac{\omega_2^1}{p_1} \right\rceil \cdot e_1 = 3 + \left\lceil \frac{6}{7} \right\rceil \cdot 3 = 3 + 3 = 6$$

Estabilitzat. Ja no puja més. El temps de resposta màxim de la tasca 2 és de 6

El temps màxim per aquesta tasca, és de 12. Compleix

TEMPS DE RESPOSTA

Tasca 3 Té interferències de la tasca 1 i 2

$$\omega_3^0 = e_3 = 5$$

$$\omega_3^1 = e_3 + \left\lceil \frac{\omega_3^0}{p_1} \right\rceil e_1 + \left\lceil \frac{\omega_3^0}{p_2} \right\rceil e_2 = 5 + \left\lceil \frac{5}{7} \right\rceil 3 + \left\lceil \frac{5}{12} \right\rceil 3 = 5 + 3 + 3 = 11$$

$$\omega_3^2 = e_3 + \left\lceil \frac{\omega_3^1}{p_1} \right\rceil e_1 + \left\lceil \frac{\omega_3^1}{p_2} \right\rceil e_2 = 5 + \left\lceil \frac{11}{7} \right\rceil 3 + \left\lceil \frac{11}{12} \right\rceil 3 = 5 + 6 + 3 = 14$$

$$\omega_3^3 = e_3 + \left\lceil \frac{\omega_3^2}{p_1} \right\rceil e_1 + \left\lceil \frac{\omega_3^2}{p_2} \right\rceil e_2 = 5 + \left\lceil \frac{14}{7} \right\rceil 3 + \left\lceil \frac{14}{12} \right\rceil 3 = 5 + 6 + 6 = 17$$

$$\omega_3^4 = e_3 + \left\lceil \frac{\omega_3^3}{p_1} \right\rceil e_1 + \left\lceil \frac{\omega_3^3}{p_2} \right\rceil e_2 = 5 + \left\lceil \frac{17}{7} \right\rceil 3 + \left\lceil \frac{17}{12} \right\rceil 3 = 5 + 9 + 6 = 20$$

$$\omega_3^5 = e_3 + \left\lceil \frac{\omega_3^4}{p_1} \right\rceil e_1 + \left\lceil \frac{\omega_3^4}{p_2} \right\rceil e_2 = 5 + \left\lceil \frac{20}{7} \right\rceil 3 + \left\lceil \frac{20}{12} \right\rceil 3 = 5 + 9 + 6 = 20$$

Estabilitzat. Ja no puja més. El temps de resposta màxim de la tasca 3 és de 20

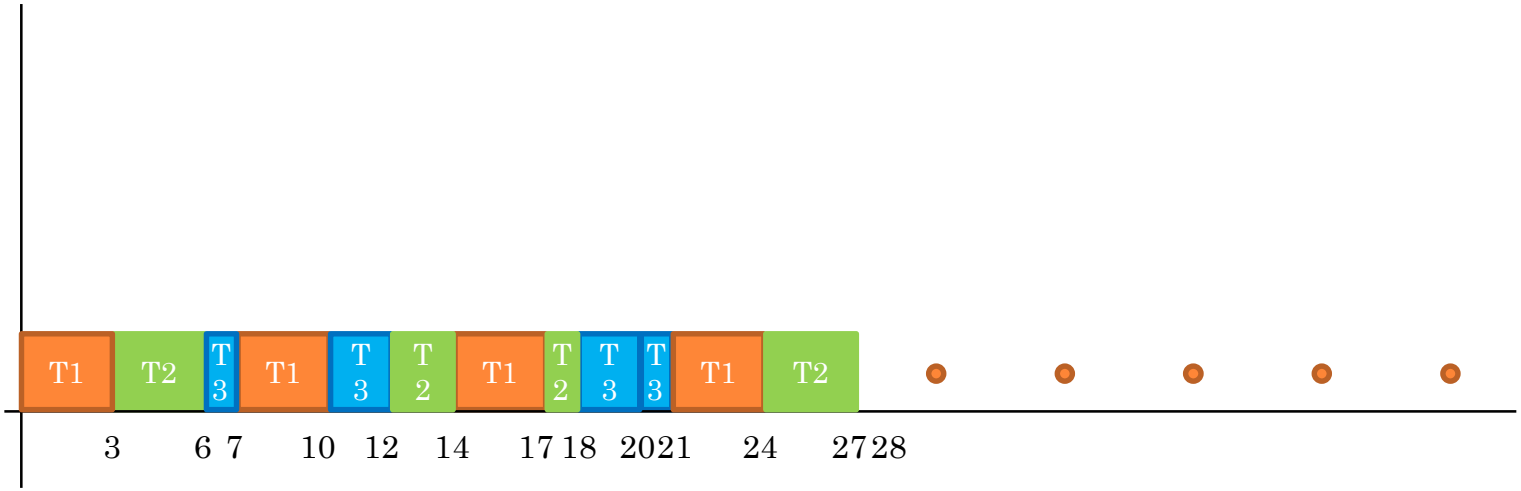
El temps màxim per aquesta tasca, és de 20. Compleix -> És planificable

Planificació per prioritats Temps de resposta

- Diagrama de Gantt

15/12

T1, T2 i T3



I1 I1 I1 I1 I1
I2 I2 I2
I3 I3

Planificació Deadline Monotonic (DM)

La política de planificació és idèntica a la RM, però els terminis de finalització poden ser diferents als períodes (**menors o iguals**).

$$D_i = P_i$$

L'assignació de prioritats es fa en l'ordre invers al termini de finalització.

Una tasca amb un termini de finalització més petit, té més prioritat

Com a test de garantia pel DM podem agafar el resultat aconseguit en el anàlisis del temps de resposta, però comparant-lo amb el termini de finalització de cada tasca enlloc del període.

Planificació DM

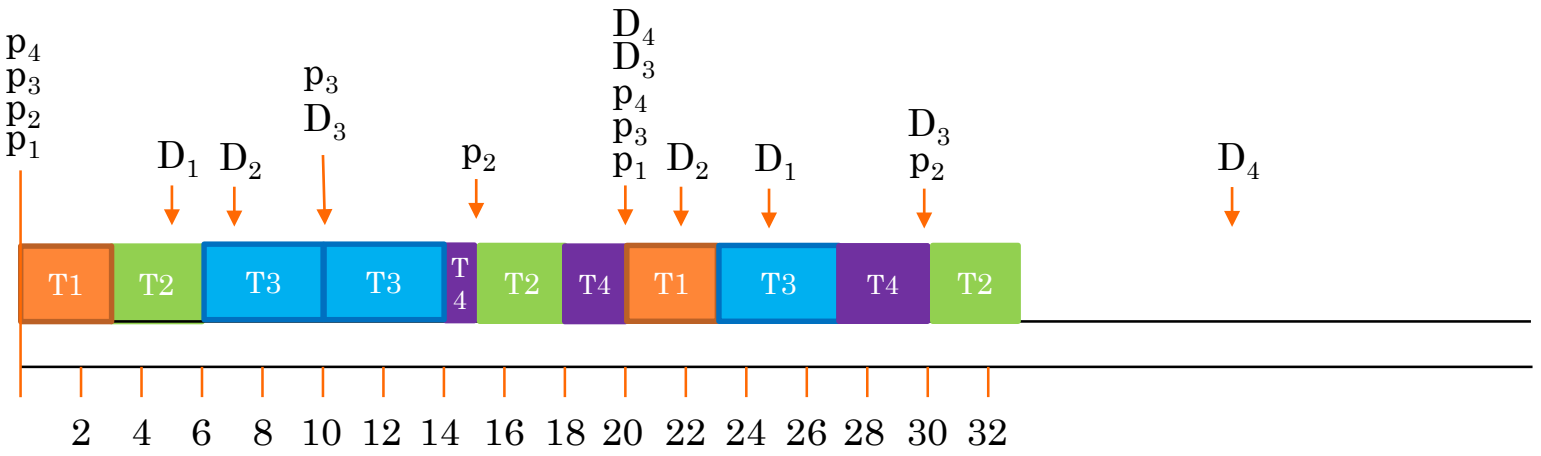
- Problema

Tasques	p	D	e	Pr	U	R
T1	20	5	3	4	0,15	3
T2	15	7	3	3	0,2	6
T3	10	10	4	2	0,4	10
T4	20	20	3	1	0,15	20
					0,9	

No es pot aplicar RM -> $D_i < p_i$

S'ha d'anar directament a l'anàlisi del temps de resposta

La condició de temps de resposta → Es compleix Es pot planificar



Planificació estàtica. Bloqueig

- Interacció entre processos
  - Per processos que no son independents
  - Accés a dades comuns (zones d'exclusió mútua)
  - Recursos compartits

Bloqueig: un procés es queda esperant que un altre de menys prioritat finalitzi una part del seu codi. Es produeix una inversió de prioritat.

Planificació estàtica. Bloqueig

- Exemple

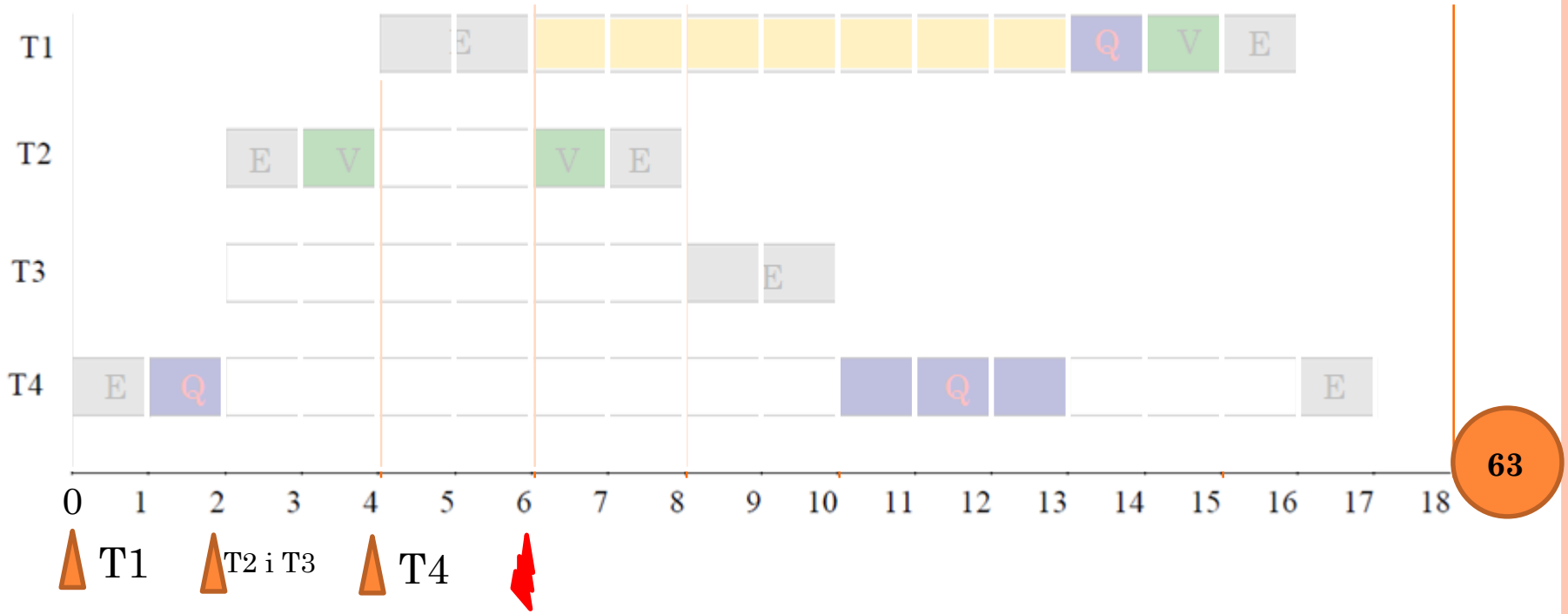
Tenim 4 tasques que poden accedir a 2 recursos compartits Q i V (en exclusió mútua)

Execució de V

Execució de Q

Bloquejat

Tasca	Prioritat	feines	Activació	Resposta
T1	4	EEQVE	4	12
T2	3	EVVE	2	6
T3	2	EE	2	8
T4	1	EQQQQE	0	17

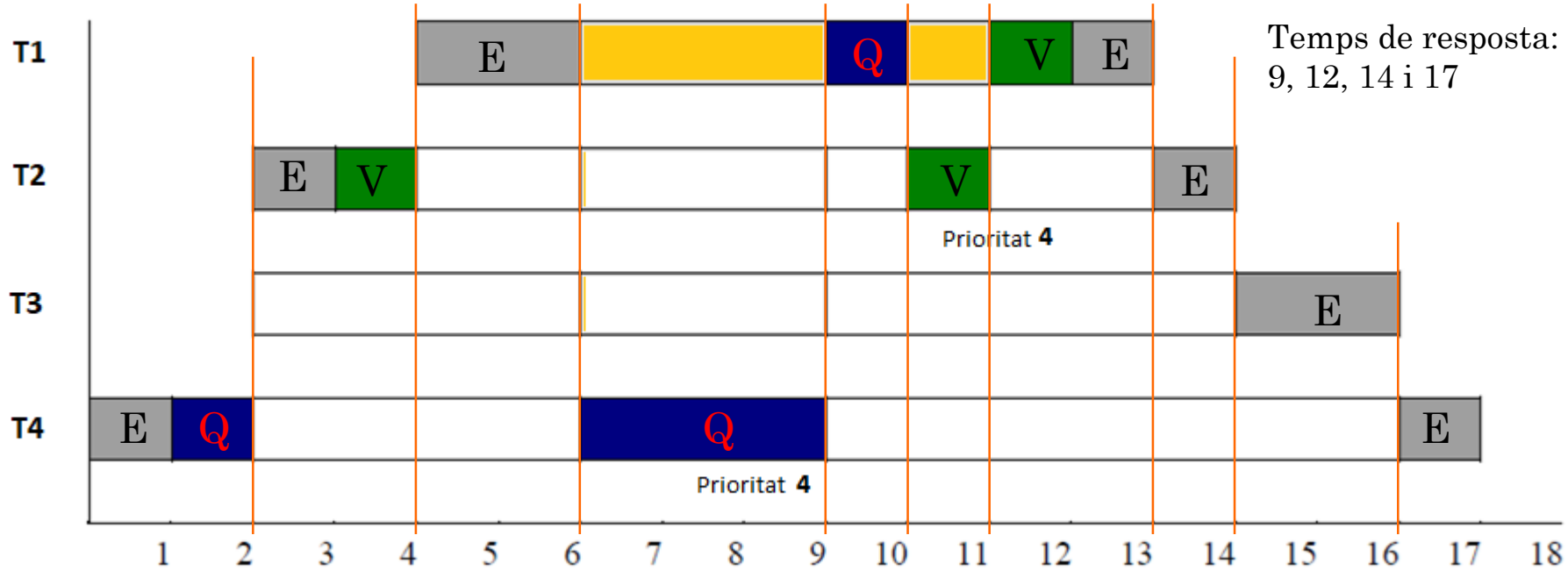


Planificació estàtica. Bloqueig: Herència de prioritat

Hi ha un problema d'inversió de prioritat. T1 queda bloquejat per T4, T2 i T3. El problema es presenta per utilitzar recursos compartits i mantenir un esquema de prioritats fixa.

Una mesura per limitar el seu efecte és l'ús de l'herència de prioritats: Si un procés A es bloqueja esperant que un altre procés B finalitzi algun càlcul, llavors B agafa la prioritat d'A.

En l'exemple anterior quan T1 es bloquegés esperant que T4 acabi d'usar Q es produiria l'herència de prioritats per la qual T4 passaria a prendre la prioritat de T1, i tindria una prioritat més alta que T2 i T3. Amb aquesta regla la prioritat d'un procés és la màxima entre la seva pròpia prioritat i la d'altres processos que depenguin d'ell



Planificació estàtica. Herència de prioritat

$$R_i = e_i + B_i + \sum_{j \in ps(i)} \left\lceil \frac{R_i}{p_j} \right\rceil e_j$$

ps = major prioritat
lp = menor prioritat

- On B_i és el màxim temps de bloqueig que pot tenir el procés i.
- Una tasca es pot bloquejar com a màxim:
  - Un cop per cada recurs que utilitza
  - Un cop per cada tasca de menys prioritat

$$B_i = \sum_{j \in 1}^{K} \acute{u}s(j,i) * C_k$$

$$\omega_i^{n+1} = e_i + B_i + \sum_{j \in ps(i)} \left\lceil \frac{\omega_i^n}{p_j} \right\rceil e_j$$

On

K és el nombre seccions critiques (recursos compartits)

ús(j,i) val 1 si el recurs *j* s'utilitza al menys per un procés amb una prioritat menor que la del procés *i*, i al menys, per un procés amb una prioritat més gran o igual que la del procés *i*. 0 en cas contrari.

C_k és el temps d'execució del recurs o secció critica *r en el pitjor dels casos.*

$B_1 = C_{4,Q} + C_{3,Q} + C_{2,Q} + C_{4,V} + C_{3,V} + C_{2,V} = 4+0+0+0+0+2 = 6$

$B_2 = C_{4,Q} + C_{3,Q} + C_{4,V} + C_{3,V} = 4 + 0 + 0 + 0 = 4$

$B_3 = C_{4,Q} + C_{4,V} = 4 + 0 = 4$

$B_4 = 0$

- Una tasca es pot bloquejar per recursos que no accedeix (T2)
- Una tasca es pot bloquejar encara que no accedeixi a recursos compartits (T3)
- La tasca de menys prioritat (T4) no es bloqueja mai

Sostre de prioritats Immediat (ICPP)

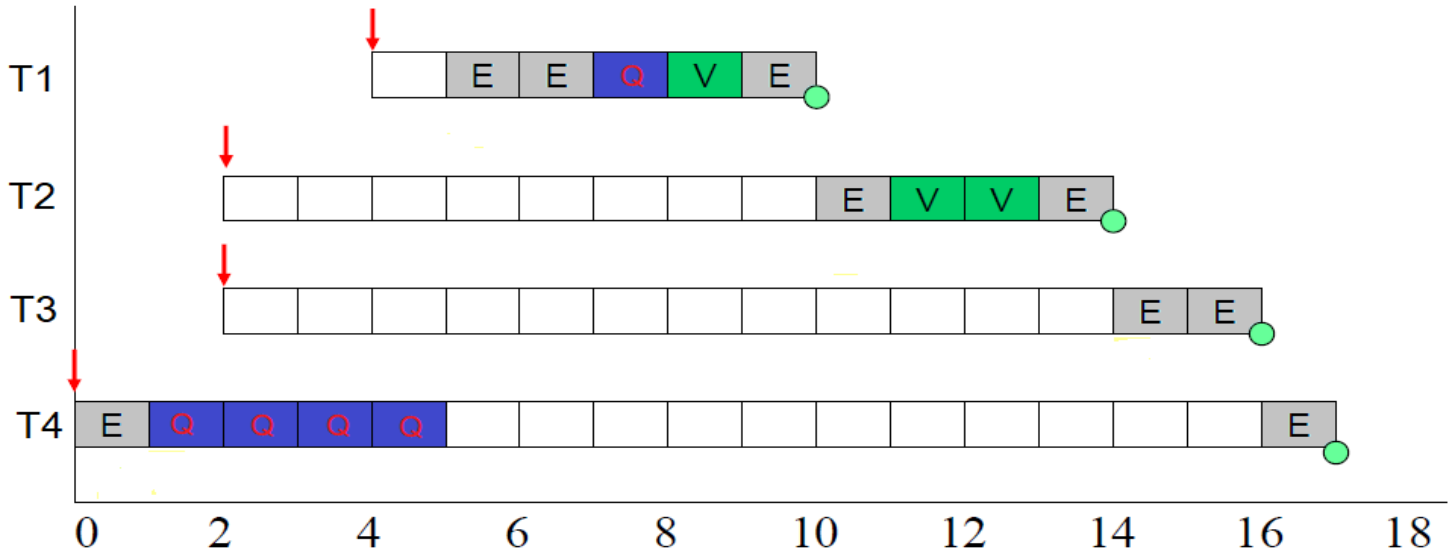
Consisteix en:

- Cada procés té una prioritat estàtica assignada.
- Cada recurs compartit té una prioritat estàtica assignada, que és la màxima de les prioritats de les tasques que l'utilitzen.
- Un procés té una prioritat dinàmica que és el màxim entre la seva prioritat i la prioritat de qualsevol recurs que tingui bloquejat.
  - Consisteix: Un procés només pot utilitzar un recurs si la seva prioritat dinàmica és mes gran que el sostre de tots els recursos utilitzats per altres processos.
  - Exemple: Si un procés p_1 utilitza un recurs r_1 amb un sostre de prioritat Q i un altre procés p_2 amb una prioritat P_2, $P_2 \leq Q$, vol utilitzar un recurs r_2, llavors el procés p_2 també es bloqueja.

Temps màxim de bloqueig: $B_i = \max_{j=1:K} us(j,i)C(j)$

K son les seccions critiques (recursos compartits)
i el procés que s'està analitzant
us(j,i) = {0,1}
1 si el recurs j s'utilitza en algun procés de prioritat menor que i
0 en cas contrari
C(j) és el temps d'execució de la secció critica j

Sostre de prioritats Immediat (ICPP)



A t=1, el procés T4 actualitza la seva prioritat dinàmica a 4 (utilitza el valor del sostre de prioritat de el recurs Q). Per aquest motiu, tot i que en els instants de temps 2 i 4, s'activen processos més prioritats, el procés T4 no s'expulsa.

És important considerar quan es produeix la reducció efectiva de la prioritat dinàmica d'un procés. Per exemple, el procés T4 retornarà a la seva prioritat per defecte (estàtica) just després d'alliberar el recurs Q (t = 5).

S'ha millorat el temps de resposta dels processos, ja que es respecte la prioritat de cadascun d'ells.

Sostre de prioritats original (CPP)

- El temps d'execució és el mateix
- Les diferències:
  - ICPP és més fàcil d'implementar. No cal fer un seguiment de quines tasques estan bloquejades.
  - ICPP genera menys canvis de context. El bloqueig es produeix abans de començar l'execució.
  - ICPP genera menys canvis de prioritats en el sistema
  - CPP modifica la prioritat només quan es necessari
- ICPP s'anomena Priority Protect Protocol en POSIX i Priority Ceiling Emulation en Real-Time Java

EXEMPLE:

Tasca	p	e	D	R1	R2	R3	R4	P
A	80	10	80	3	-	-	5	2
B	150	20	150	2	2	1	-	1
C	100	10	15	1	1	-	-	4
D	500	12	30	2	-	4	-	3

1) S'ha d'establir la prioritat

RM o DM?

No es pot utilitzar RM degut a que hi ha terminis de finalització menors que el període.

Exemple: Herència de prioritat

Tasca	p	e	D	R1	R2	R3	R4	P	B
A	80	10	80	3	-	-	5	2	5
B	150	20	150	2	2	1	-	1	0
C	100	10	15	1	1	-	-	4	9
D	500	12	30	2	-	4	-	3	8

2) Temps de bloqueig

$B_B = 0$

$B_A = C_{b,r1} + C_{b,r2} + C_{b,r3} = 2 + 2 + 1 = 5$

$B_D = C_{b,r1} + C_{a,r1} + C_{b,r2} + C_{b,r3} = 2 + 3 + 2 + 1 = 8$

$B_C = C_{d,r1} + C_{b,r1} + C_{a,r1} + C_{b,r2} = 2 + 2 + 3 + 2 = 9$

EXEMPLE

Tasca	p	e	D	R1	R2	R3	R4	P	B	R
A	80	10	80	3	-	-	5	2	5	
B	150	20	150	2	2	1	-	1	0	
C	100	10	15	1	1	-	-	4	9	19
D	500	12	30	2	-	4	-	3	8	

3) Temps de resposta

$R_C = e_C + B_C = 10 + 9 = 19$

$$R_C = 19 > D_C = 15$$

No compleix. Fa una fallada. No es pot planificar.

No cal continuar

Exemple: Sostre de prioritat

Tasca	p	e	D	R1	R2	R3	R4	P
A	80	10	80	3	-	-	5	2
B	150	20	150	2	2	1	-	1
C	100	10	15	1	1	-	-	4
D	500	12	30	2	-	4	-	3

1) Buscar el sostre de prioritats

$TP_{r1} = \max\{P_A,P_B,P_C,P_D\} = \max\{2,1,4,3\} = 4$

$TP_{r2} = \max\{P_B,P_C\} = \max\{1,4\} = 4$

$TP_{r3} = \max\{P_B,P_D\} = \max\{1,3\} = 3$

$TP_{r4} = \max\{P_A\} = 2$

Exemple: Sostre de prioritat

Tasca	p	e	D	R1	R2	R3	R4	P	B
A	80	10	80	3	-	-	5	2	2
B	150	20	150	2	2	1	-	1	0
C	100	10	15	1	1	-	-	4	
D	500	12	30	2	-	4	-	3	

Recordeu que aquest valor ve determinat pel temps màxim d'ús d'un recurs per part un procés, considerant el conjunt de processos amb menor prioritat que P_i accedint a recursos amb un sostre de prioritat igual o major a P_i.

2) Temps de bloqueig

$B_B = 0$

$B_A = \max\{C_{B,R1}, C_{B,R2}, C_{B,R3}\} = \max\{2, 2, 1\} = 2$

Recurs	Prioritat
R1	4
R2	4
R3	3
R4	2

EXEMPLE: SOSTRE DE PRIORITAT

Tasca	p	e	D	R1	R2	R3	R4	P	B
A	80	10	80	3	-	-	5	2	2
B	150	20	150	2	2	1	-	1	0
C	100	10	15	1	1	-	-	4	
D	500	12	30	2	-	4	-	3	3

La prioritat de R4 (2) és menor que la prioritat del procés D (3)

Recordeu que aquest valor ve determinat pel temps màxim d'ús d'un recurs per part un procés, considerant el conjunt de processos amb menor prioritat que P_i accedint a recursos amb un sostre de prioritat igual o major a P_i.

2) Temps de bloqueig

$B_B = 0$

$B_A = \max\{C_{B,R1}, C_{B,R2}, C_{B,R3}\} = \max\{2, 2, 1\} = 2$

$B_D = \max\{C_{A,R1}, C_{B,R1}, C_{B,R2}, C_{B,R3}\} = \max\{3,2,2,1\} = 3$

Recurs	Prioritat
R1	4
R2	4
R3	3
R4	2

EXEMPLE: SOSTRE DE PRIORITAT

Tasca	p	e	D	R1	R2	R3	R4	P	B
A	80	10	80	3	-	-	5	2	2
B	150	20	150	2	2	1	-	1	0
C	100	10	15	1	1	-	-	4	3
D	500	12	30	2	-	4	-	3	3

La prioritat de R4 (2) i R3 (3) és menor que la prioritat del procés C (4)

Recordeu que aquest valor ve determinat pel temps màxim d'ús d'un recurs per part un procés, considerant el conjunt de processos amb menor prioritat que P_i accedint a recursos amb un sostre de prioritat igual o major a P_i.

2) Temps de bloqueig

$B_B = 0$

$B_A = \max\{C_{B,R1}, C_{B,R2}, C_{B,R3}\} = \max\{2, 2, 1\} = 2$

$B_D = \max\{C_{A,R1}, C_{B,R1}, C_{B,R2}, C_{B,R3}\} = \max\{3,2,2,1\} = 3$

$B_C = \max\{C_{D,R1}, C_{A,R1}, C_{B,R1}, C_{B,R2}, \} = \max\{2,3,2,2\} = 3$

Recurs	Prioritat
R1	4
R2	4
R3	3
R4	2

EXEMPLE: SOSTRE DE PRIORITAT

Tasca	p	e	D	R1	R2	R3	R4	P	B	R
A	80	10	80	3	-	-	5	2	2	34
B	150	20	150	2	2	1	-	1	0	52
C	100	10	15	1	1	-	-	4	3	15
D	500	12	30	2	-	4	-	3	3	25

2) Temps de resposta

Es pot planificar

$R_C = e_C + B_C = 10 + 3 = 13 < D_C = 15$

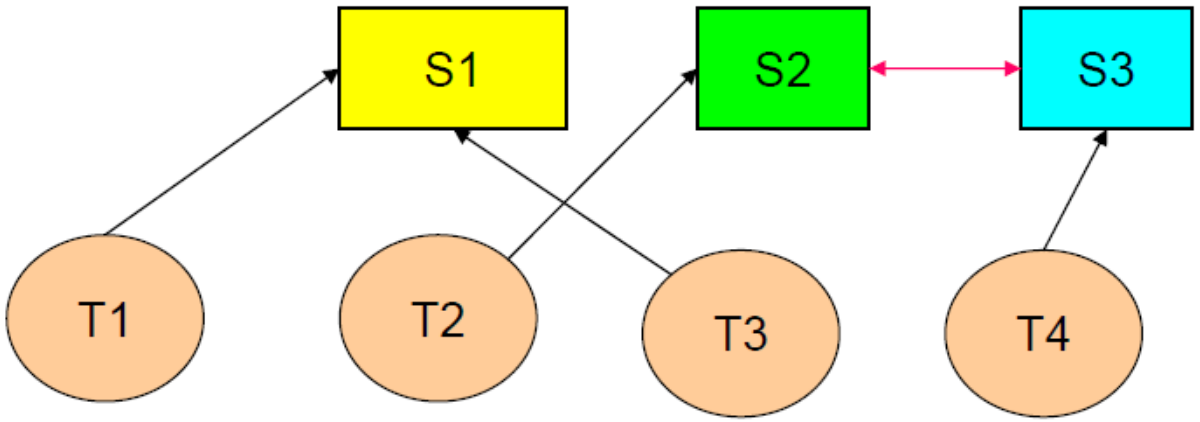
$R_D = e_D + I_D + B_D$

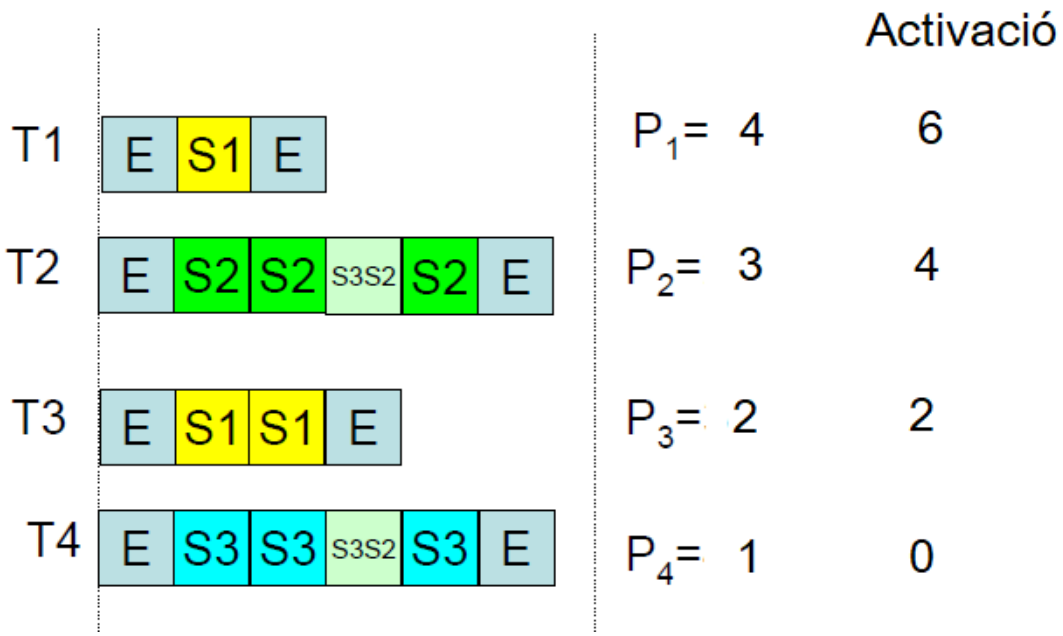
$$\omega_D^0 = e_D + B_C = 12 + 3 = 15$$

$$\omega_D^1 = e_D + \left\lceil \frac{\omega_D^0}{p_C} \right\rceil e_C + B_C = 12 + \left\lceil \frac{15}{100} \right\rceil 10 + 3 = 12 + 10 + 3 = 25$$

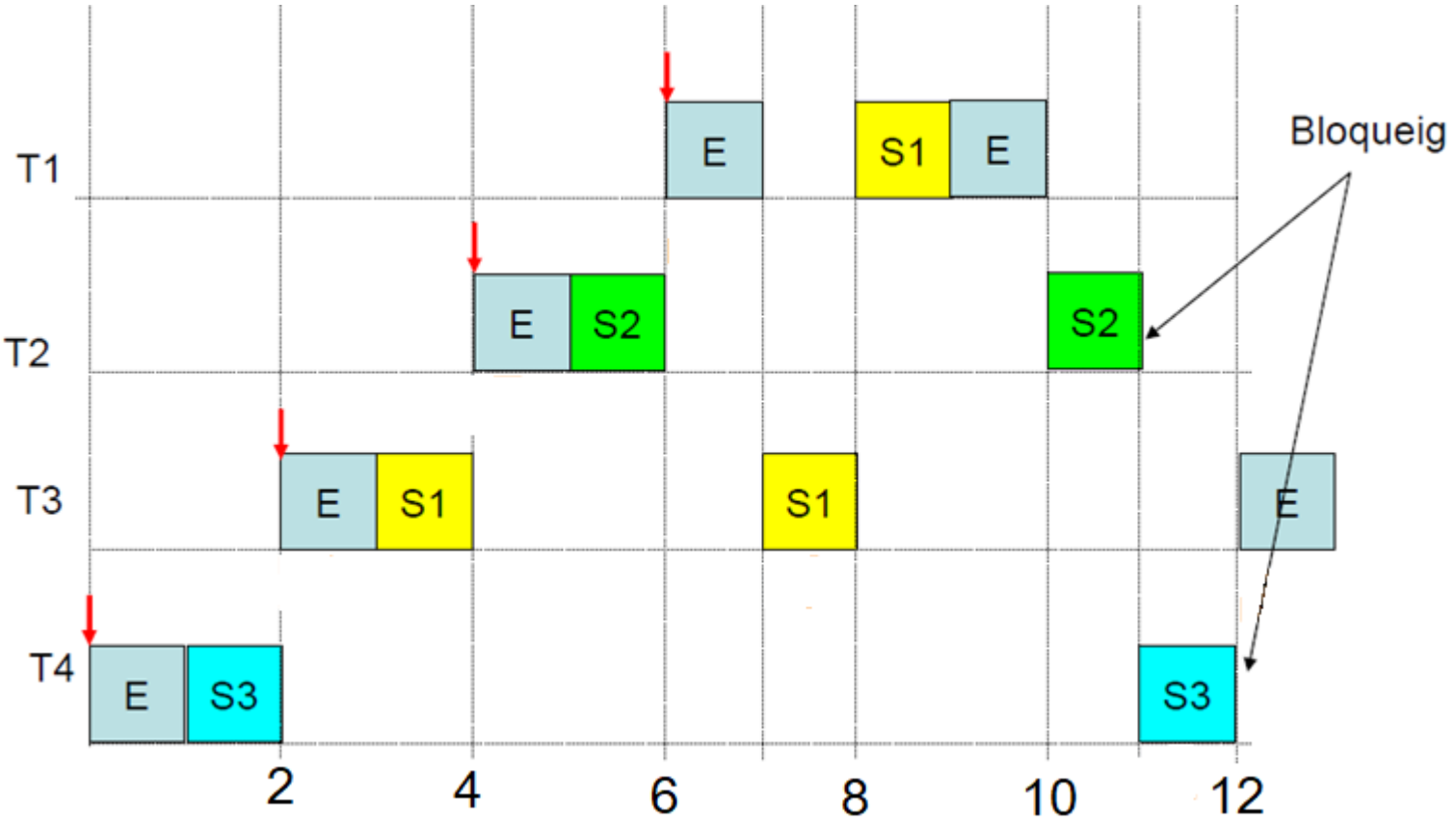
$R_D = 25$

Exemple

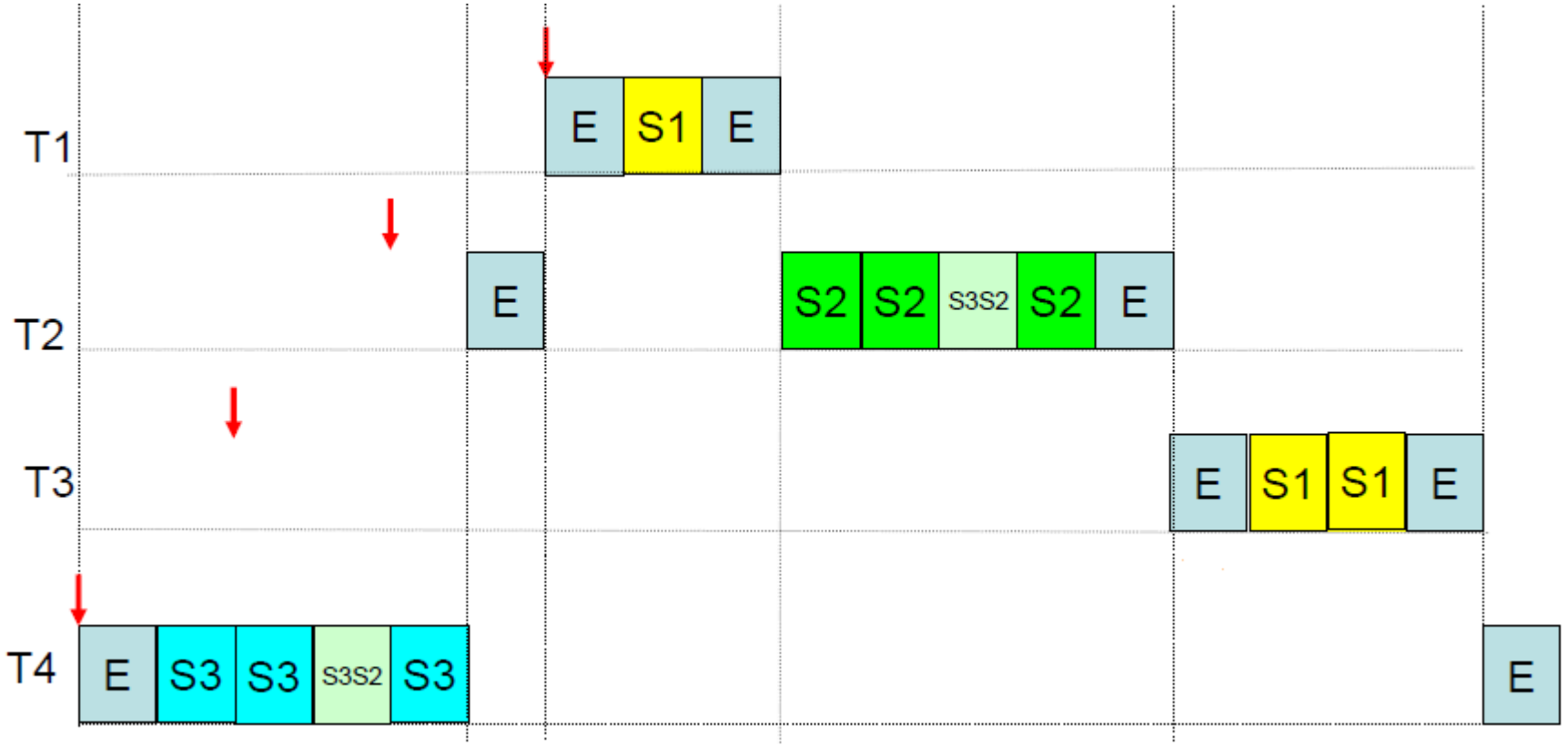

S1
S2
S3
T1
T2
T3
T4


Activació
T1
E S1 E
P1= 4
6
T2
E S2 S2 S3S2 S2 E
P2= 3
4
T3
E S1 S1 E
P3= 2
2
T4
E S3 S3 S3S2 S3 E
P4= 1
0

Herència de prioritat



Sostre de prioritat Immediat (ICPP)

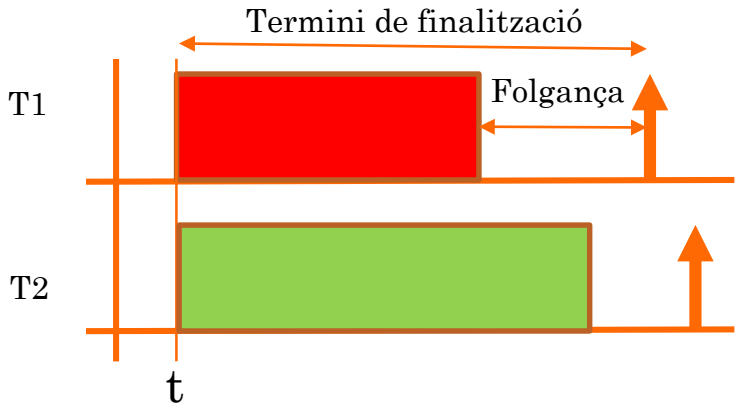


Planificadors amb prioritats dinàmiques

Els planificadors dinàmics són planificadors basats en prioritats, però es caracteritzen perquè les prioritats no són estàtiques durant tota la vida del sistema sinó que varien en el temps seguint un algoritme que defineix el planificador.

S'utilitzen dos planificadors dinàmics:

- EDF (Earliest Deadline First). S'assignen les prioritats donant preferència a les tasques amb termini de finalització més proper.
- LLF (Least Laxity First). S'assignen les prioritats donant preferència a les tasques amb menys folgança de la tasca.



EDF -> Més prioritari T1

LLF-> Més prioritari T2

Planificadors amb prioritats dinàmiques

Un planificador dinàmic es pot expressar definint la tasca en execució T_x en cada moment de la manera següent:

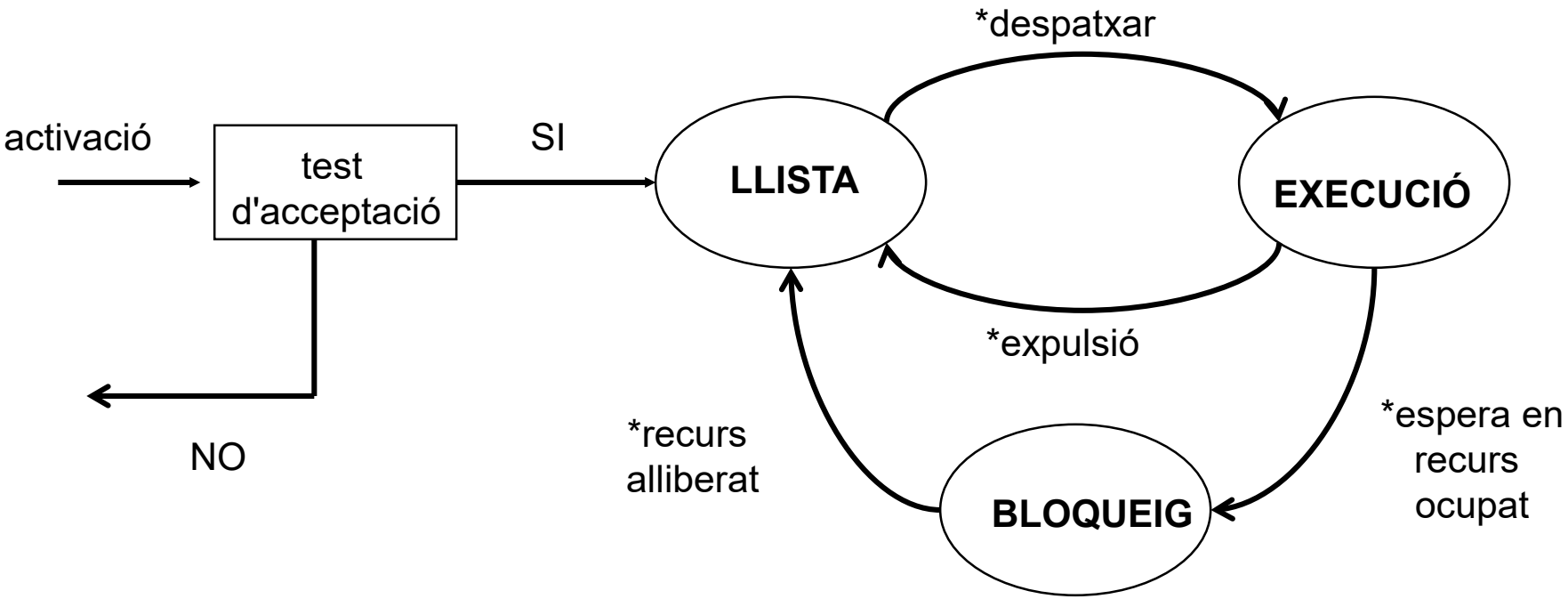
T_x executant-se $\leftrightarrow \forall$ x, i $\in$ Actives $f(T_x, t) \geq f(T_i, t)$

Si la funció f no depèn de t, es tractarà un planificador estàtic.

En el planificador EDF la funció *f* és tal que dóna més prioritat a les tasques amb distància absoluta al termini d'execució més petit.

- La prioritat de cada tasca s'assigna en el moment de l'activació. La implementació no té un cost excessiu, no gaire més que el DM.

Planificadors
Procés de planificació (Estats)



***decisions de planificació**

Planificadors amb prioritats dinàmiques

Avantatges:

- És òptim. Aconsegueix la màxima utilització del processador. Si aquest no aconsegueix planificar un conjunt de tasques, tampoc ho aconseguirà altre planificador.
- Precisa d' un temps de càlcul (overhead) no excessiu.

Inconvenients:

- És poc estable davant sobrecàrregues transitòries. Si una tasca que pren l'execució consumeix molta CPU, pot afectar tasques més importants.
- És difícil de portar a la pràctica. Pocs Sistemes Operatius comercials l'implementen.

Actualment se sol utilitzar és Sistemes de Temps Real no estricte (soft real time), sobretot per la seva novetat i la seva poca estabilitat.

Planificadors amb prioritats dinàmiques

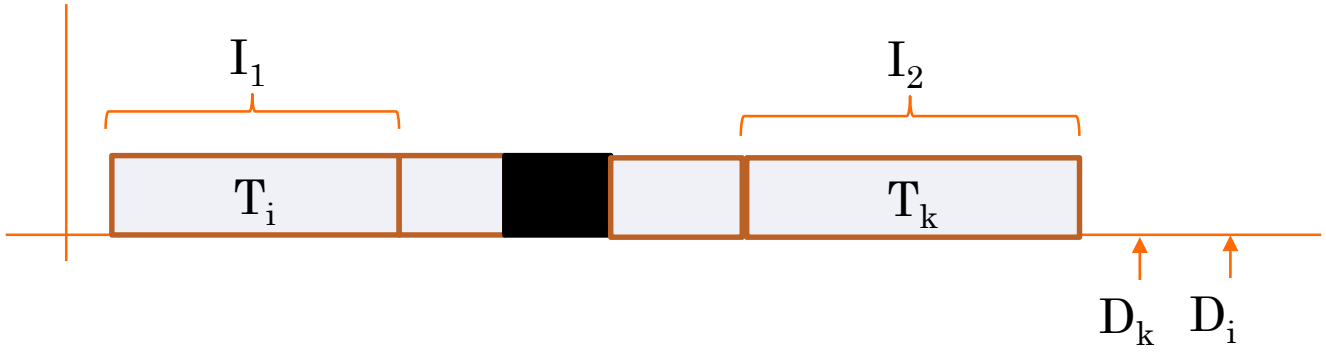
Planificador òptim:

- Teorema: El planificador EDF pot produir una planificació vàlida en un sol processador d'un conjunt de tasques independents (T) amb expulsió permesa i amb terminis d'execució arbitraris, si i només si T té planificacions vàlides.

Demostració:

La prova es basa en comprovar que qualsevol planificador vàlid per T es pot transformar en un planificador EDF.

Suposem que dues tasques T_i i T_k són planificats en els intervals I_1 i I_2 respectivament. A més, el termini d'execució D_i per T_i és posterior al termini D_k per T_k. però I_1 és anterior a I_2.



Planificadors amb prioritats dinàmiques

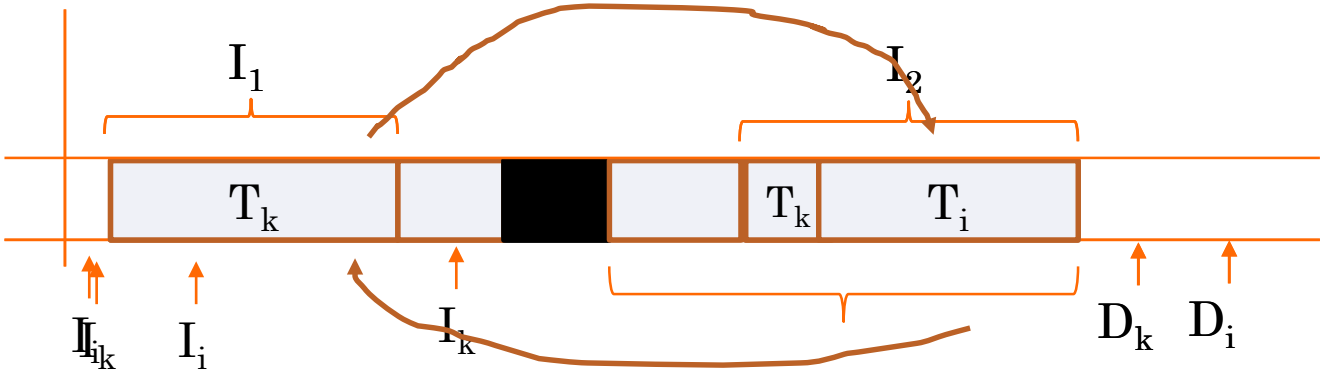
Poden succeir dues coses:

a) Que l'instant d'activació de T_k sigui posterior al final de I_1. En aquest cas T_k no pot executar-se abans de T_i i, per tant, el planificador compleix les regles del EDF.

b) Que l'instant d'activació de T_k sigui anterior al final de I_1. Aquest és l'únic cas que s'ha de considerar.

Sense perdre generalitat, podem assumir que l'activació de T_k es produeix abans del començament de I_1 (si no és així, n'hi hauria prou amb considerar I_1 a partir del moment de l'activació).

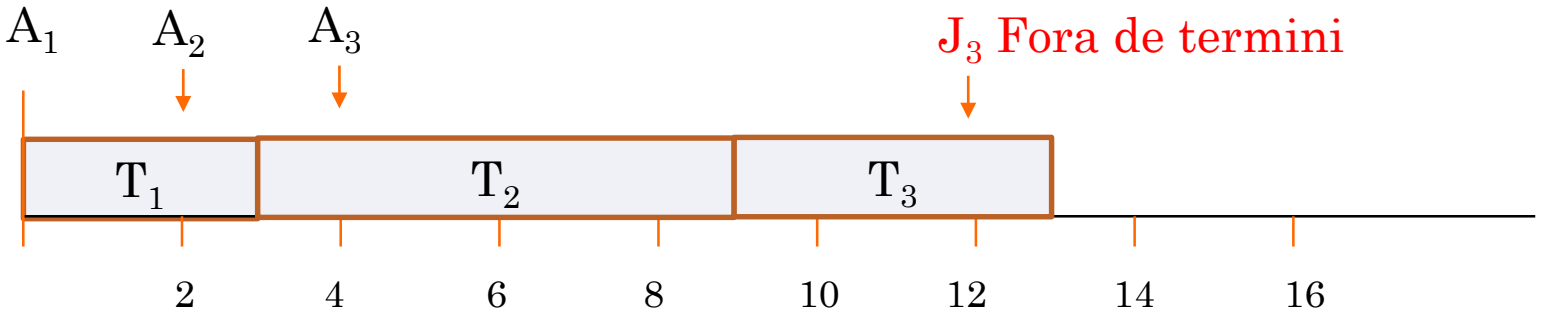
Per transformar aquest planificador, commutem T_i i T_k. En concret, si I_1 és més curt que I_2, tal com es mostra a la figura, es mou la porció de T_k que cap en I_1 sobre I_1, i T_i es mou per complet al final de I_2. Després d'aquest canvi, el sistema continua sent planificable.



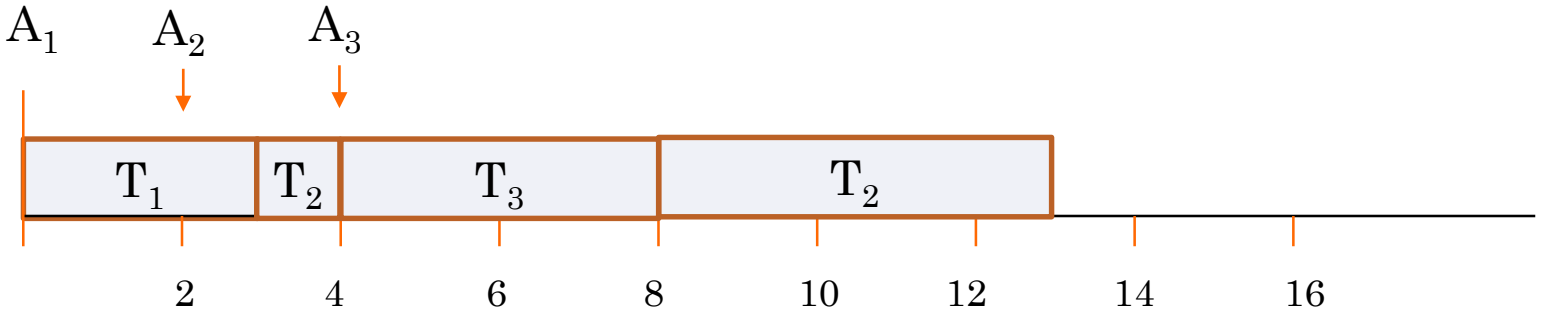
Planificadors amb prioritats dinàmiques

S'ha suposat que les tasques s'havien de poder interrompre.

Considerem les tasques T1, T2 i T3 amb instants d'activació 0, 2 i 4, temps d'execució 3, 6 i 4 i límits d'execució 10, 14 i 12



Hi ha una planificació possible:



Planificadors amb prioritats dinàmiques

Test de planificabilitat

EDF és un algoritme òptim. Un conjunt de tasques amb els terminis de finalització igual als períodes, una condició suficient i necessària pot ser:

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

- Per un sistema amb terminis de finalització restringits ($D_i < p_i$) es pot utilitzar la condició de suficiència basada en la densitat:

$$D = \sum_{i=1}^{n} \frac{e_i}{D_i} \leq 1 \ \ (D = densitat)$$

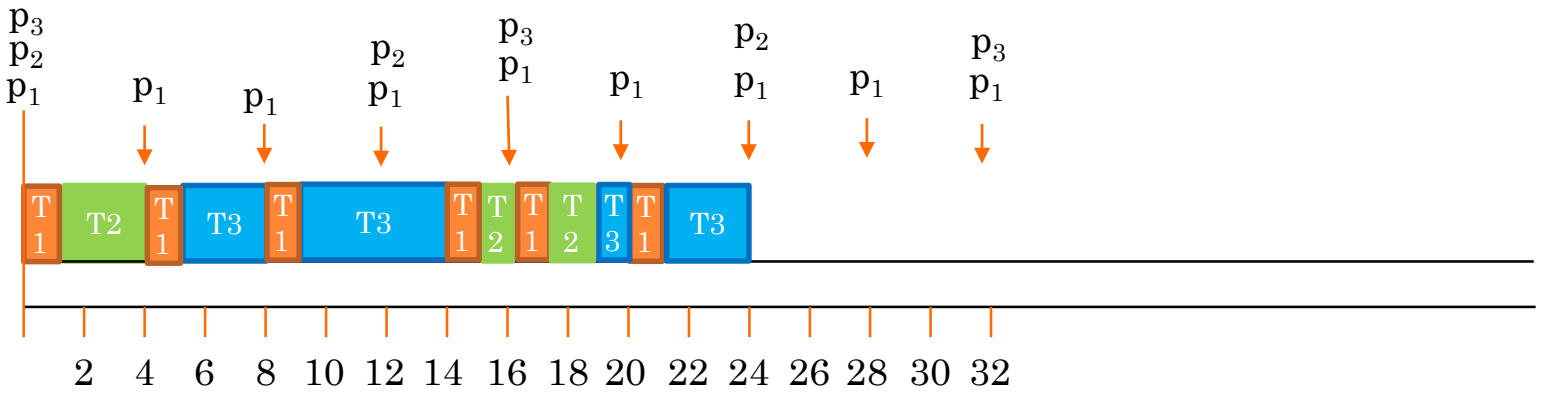
Planificadors amb prioritats dinàmiques

- Problema

Tasques	P (=D)	e	U	R
T1	4	1	0,25	
T2	12	3	0,25	
T3	16	8	0,50	
			1	

- Comportament de la tasca 2

A l'instant 0 (crític per la tasca 2), només interfereix amb la tasca 1



PLANIFICADORS: EDF. TEST: CÒMPUT SOL·LICITAT

- Analitza el comportament de totes les tasques en el seu conjunt, enlloc de calcular el pitjor temps de resposta per a cadascuna de les tasques.
- Si les tasques son periòdiques i activades simultàniament en el temps t = 0, llavors el planificadors es repeteix cada "híper-període" H. I la condició:$\forall L \in D > 0 \rightarrow g(0,L) \leq L$, només s'ha de verificar pels valors ≤ H, on:

S'analitza el comportament de totes les tasques en el seu conjunt

La funció g(0,L) representa la quantitat de còmput que s'ha de donar perquè cap tasca del conjunt perdi un termini de lliurament.

No es pot sobrepassar la capacitat del processador entre l'interval 0 i L

$$D = \{d_k | d_k \leq \min(H, L^*)\}$$

$$H = mcm(p_1, p_2, \ldots., p_n)$$

$$L^* = \frac{\sum_{i=1}^{n}(p_i - D_i)U_i}{1 - U}$$

$$g(0,L) = \sum_{i=1}^{n} \left\lfloor \frac{L + p_i - D_i}{p_i} \right\rfloor e_i$$

$$d_k = (k-1)p_i + D_i$$

PLANIFICADORS: EDF. TEST: CÒMPUT SOL·LICITAT

- Analitza el comportament de totes les tasques en el seu conjunt, enlloc de calcular el pitjor temps de resposta per a cadascuna de les tasques.
- Aquest test, es basa en comprovar que la funció:

$$H(t) = \sum_{i=1}^{n} \left\lceil \frac{t + p_i - D_i}{p_i} \right\rceil e_i$$

compleix la condició: $\forall t < L, H(t) \leq t$

- H(t) representa la quantitat de còmput de temps que es necessita perquè cap tasca perdi un termini de finalització. Senyala que aquest temps de còmput no pot superar la capacitat del processador.
- Límit de comprovació L:

$$L = \{\min(L_a, L_b)\}$$

L_b : Durada de l'interval ocupat: Es calcula amb la relació de recurrència:

$$L_a = \frac{\sum_{i=1}^{n}(p_i - D_i)U_i}{1 - U}$$

L_a : límit per la utilització de les tasques

$$W^0 = \sum_{i=1}^{n} e_i \qquad W^{m+1} = \sum_{i=1}^{n} \left\lceil \frac{W^m}{p_i} \right\rceil e_i$$

$L_b = W^m$, quan $W^{m+1} = W^m$

Punts de comprovació: Només s'ha de mirar si compleix en els instants en que finalitzen els temps absoluts de les tasques, ja que son els únics instants en que la funció *H(t)* pot canviar.

- Limita el càlcul de la funció *H(t)*. Només s'ha de fer per tots els instants de temps possibles entre 0 i L. Entre [0,L]

$$P = \{d_k | d_k = (k-1)p_i + D_i < L, k \in \mathbb{N}\} \qquad \forall t < P, H(t) \leq t$$

EDF: EXEMPLE

	e_i	D_i	p_i
T1	2	4	6
T2	2	5	8
T3	3	7	9

$$U = \frac{2}{6}+\frac{2}{8}+\frac{3}{9}=\frac{11}{12} \qquad D = \frac{2}{4}+\frac{2}{5}+\frac{3}{7}=\frac{186}{140}=1.328$$

$$H = mcm(6,8,9) = 72$$

$$L_a = \frac{\sum_{i=1}^{3}(p_i - D_i)U_i}{1-U} = \frac{(6-4)\frac{2}{6}+(8-5)\frac{2}{8}+(9-7)\frac{3}{9}}{1-\frac{11}{12}} = 25$$

$$W^0 = \sum_{i=1}^{3} e_i = 2+2+3 = 7$$

$$W^1 = \sum_{i=1}^{3}\left\lceil\frac{W^0}{p_i}\right\rceil e_i = \left\lceil\frac{7}{6}\right\rceil 2+\left\lceil\frac{7}{8}\right\rceil 2+\left\lceil\frac{7}{9}\right\rceil 3 = 4+2+3 = 9$$

$$W^2 = \sum_{i=1}^{3}\left\lceil\frac{W^1}{p_i}\right\rceil e_i = \left\lceil\frac{9}{6}\right\rceil 2+\left\lceil\frac{9}{8}\right\rceil 2+\left\lceil\frac{9}{9}\right\rceil 3 = 4+4+3 = 11$$

$$W^3 = \sum_{i=1}^{3}\left\lceil\frac{W^2}{p_i}\right\rceil e_i = \left\lceil\frac{11}{6}\right\rceil 2+\left\lceil\frac{11}{8}\right\rceil 2+\left\lceil\frac{11}{9}\right\rceil 3 = 4+4+6 = 14$$

$$W^4 = \sum_{i=1}^{3}\left\lceil\frac{W^3}{p_i}\right\rceil e_i = \left\lceil\frac{14}{6}\right\rceil 2+\left\lceil\frac{14}{8}\right\rceil 2+\left\lceil\frac{14}{9}\right\rceil 3 = 6+4+6 = 16$$

$$W^5 = \sum_{i=1}^{3}\left\lceil\frac{W^4}{p_i}\right\rceil e_i = \left\lceil\frac{16}{6}\right\rceil 2+\left\lceil\frac{16}{8}\right\rceil 2+\left\lceil\frac{16}{9}\right\rceil 3 = 6+4+6 = 16$$

$$L_b = 16$$

$$L = \{\min(16,25)\} = 16$$

$$d_k = (k-1)p_i + D_i$$

Termini de finalització de les tasques

	T1	T2	T3
$d_1 = D_i$	4	5	7
$d_2 = p_i + D_i$	10	13	16
$d_3 = 2\,p_i + D_i$	16	~~21~~	~~25~~
$d_4 = 3\,p_i + D_i$	~~22~~	-	-

$$D = \{4,5,7,10,13,16\}$$

S'ha de verificar per tots els valors de D que g(0,L) ≤ L

EDL: Exemple

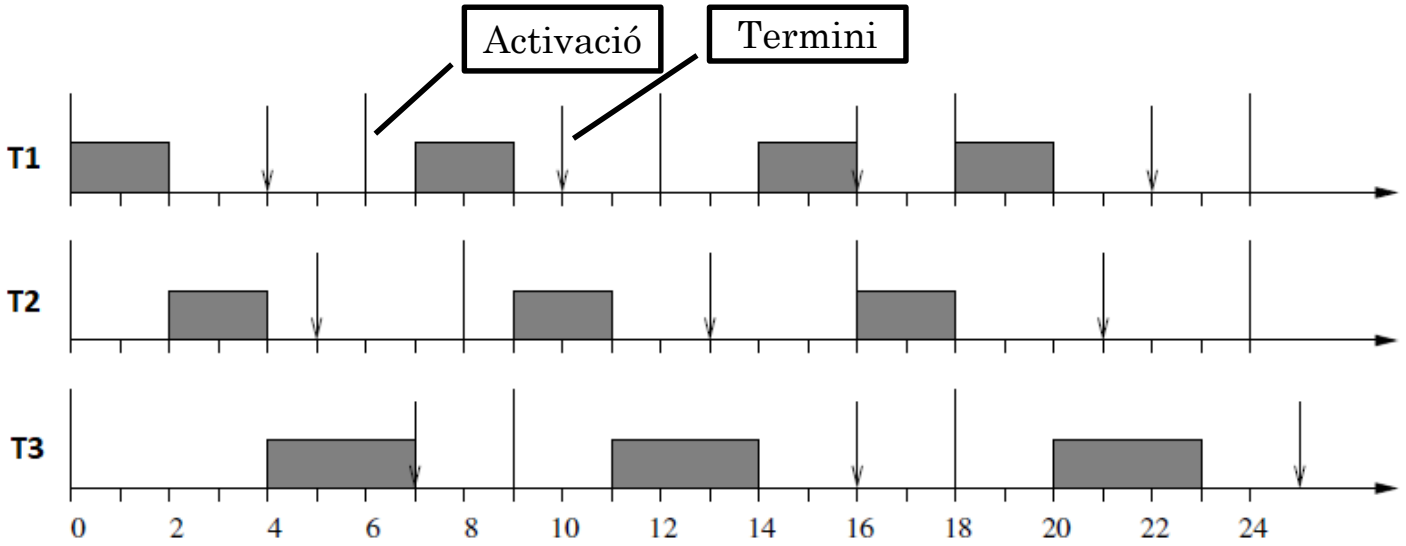
$$H(4) = \sum_{i=1}^{3} \left\lfloor \frac{4 + p_i - D_i}{p_i} \right\rfloor \cdot e_i$$

$$= \left\lfloor \frac{4+6-4}{6} \right\rfloor 2 + \left\lfloor \frac{4+8-5}{8} \right\rfloor 2 + \left\lfloor \frac{4+9-7}{9} \right\rfloor 3$$

$$= \lfloor 1 \rfloor 2 + \left\lfloor \frac{7}{8} \right\rfloor 2 + \left\lfloor \frac{6}{9} \right\rfloor 3 = 2 + 0 + 0 = 2$$

L	H(t)	resultat
4	H(4) = 2	2 ≤ 4 ☺
5	H(5) = 4	4 ≤ 5 ☺
7	H(7) = 7	7 ≤ 7 ☺
10	H(10) = 9	9 ≤ 10 ☺
13	H(13) = 11	11 ≤ 13 ☺
16	H(16) = 16	16 ≤ 16 ☺
~~21~~	~~g(0,21) = 18~~	~~18 ≤ 21 ☺~~
~~22~~	~~g(0,22) = 20~~	~~20 ≤ 22 ☺~~
~~25~~	~~g(0,25) = 23~~	~~23 ≤ 25 ☺~~

Es pot planificar. És factible



Temps d'execució

- En totes les polítiques de planificació que s'han descrit, se suposa que es coneix el temps d'execució (e_i) en el pitjor dels casos per a totes les tasques. Aquest temps és el temps màxim de CPU que pot necessitar una tasca quan s'ha activat.
- L'estimació del temps d'execució en el pitjor dels casos es pot obtenir utilitzant mètodes de mesura i d'anàlisis.
  - Problema pels mètodes de mesura -> no es pot assegurar que sigui realment el pitjor cas.
  - Inconvenient dels mètodes d'anàlisis -> es necessita un model del processador, incloent les memòries cau, la segmentació, estats d'espera de la memòria, ..

Temps d'execució

- Activitats dels mètodes d'anàlisis:
  1. Descomposta el codi de les tasques i el representa amb un graf de blocs bàsics (trossos de codi sense iteracions ni condicions)
  2. Analitza el codi màquina de cada bloc bàsic i utilitza el model del processador per estimar el seu pitjor temps d'execució.
- Un cop tenim els temps d'execució dels blocs bàsics, el graf es pot simplificar.
  - Per exemple: L'execució condicional entre dos blocs bàsics es pot reduir a un únic bloc, prenent es més gran dels dos temps.
  - Les iteracions es podem simplificar a un únic bloc bàsic si coneixem un límit pel nombre d'iteracions.
  - Es poden utilitzar tècniques de reducció de gràfics més sofisticades si es disposa de suficient informació sobre la semàntica dels blocs.

Temps d'execució

Per exemple:

```
for i in 1 .. 10 loop
   if Cond then
        A - Bloc bàsic de cost 100
   else
        B - Bloc bàsic de cost 10
   end if
end loop
```

- Si no tenim més informació, el cost total d'aquesta construcció serà de: 10 * 100 més el cost que afegeix la iteració. Per exemple : 1005.
- Potser és possible, deduir mitjançant un anàlisis del codi, que la condició que produeix l'execució d'A, es pugui donar com a molt en tres ocasions. En aquest cas, obtenim un valor del cost menys pessimista, que seria de 375 (3 * 100 + 7 * 10 + 5).
- En un anàlisi del cost d'execució en el pitjor dels casos, hi ha restriccions:
  - Les iteracions i la recursivitat ha d'estar acotada

COM PODEM MESURAR EL COST COMPUTACIONAL D'UN ALGORISME?

Sense tenir cap coneixement podríem establir un sistema molt simple, lògic i senzill. Si nosaltres li donem un temps d'execució a cada operació elemental i comptem el nombre d'operacions elementals que es donen en un algoritme en concret, podrem conèixer a priori el cost temporal d'execució.

Un exemple senzill. Suposem el següent algorisme:

```
I1: Per i des de 1 fins n fer
I2:         Pmin: = i;
I3:         Per j des i + 1 fins a n fer
I4:                     Si a[j] < a[Pmin] llavors Pmin: = j fsi
            fper
I5:         intercanviar (a [i], a[Pmin]);
    fper
```

si volem mesurar el seu cost computacional (en temps) en funció de la mida de les dades "n" s'han d'establir una sèrie de constants. Cada constant tindrà assignada una certa quantitat de cicles de CPU (temps d'execució)

Com podem mesurar el cost computacional d'un algorisme?

ta: temps d'una instrucció d'assignació d'enters.
tc: temps d'una instrucció de comparació d'enters.
tu: temps d'una instrucció d'increment d'un sencer.
tv: temps d'una instrucció d'accés a un element d'un vector.

```
I1: Per i des de 1 fins n fer
I2:         Pmin: = i;
I3:         Per j des i + 1 fins a n fer
I4:                     Si a[j] < a[Pmin] llavors Pmin: = j fsi
            fper
I5:         intercanviar (a [i], a[Pmin]);
    fper
```

Analitzem ara el cost de cada instrucció del nostre algorisme:

- I1: ta + (n-1) · tu + n · tc
- I2 : (n-1) · ta
- I3 (per a un i): ta + (n-i) · tu + (n-i + 1) · tc
- I4 (per a un i):
  - Millor cas = (n - i) · (2 · tv + tc)
  - Pitjor cas = (n - i) · (2 · tv + tc) + (n - i) · ta
- I5: (n-1) · (4 · tv + 3 · ta)

- I4 és una instrucció condicional -> no ens queda més remei que establir un criteri de selecció.
- És interessant esbrinar quina és la cota superior del consum temporal de l'algorisme
  - Podem examinar el cost computacional de l'algorisme en el pitjor dels casos, que seria en el cas que sempre es compleix la condició de I4.
- Així doncs, l'expressió final que ens indica el cost computacional de l'algorisme en el cas més desfavorable és:

$$t_a + nt_c + \sum_{i=1}^{n-1} \left(5t_a + t_c + 4t_v + t_u + (n-i)(t_u + 2t_v + t_a + 2t_c)\right)$$

$$O(n^2)$$

L'EFICIÈNCIA DELS ALGORITMES

- **A l'exemple anterior:**
  - **El temps d'execució de l'algorisme depèn de 3 factors**
    - **La grandària del conjunt de dades d'entrada (n)**
    - **El contingut de les dades d'entrada:**

      Oscil·la entre T_{min} i T_{max}
    - **El codi generat per un compilador i un computador en concret**
  - **De forma simplificada: la fórmula de l'anàlisi total dona lloc a dos polinomis de la forma:**

$$T_{min} = A\ n^2 - B\ n + C$$

$$T_{max} = A'\ n^2 + B'\ n + C'$$

L'ordre és quadràtic

L'eficiència dels algoritmes. En general

- **Adoptarem el criteri d'estimar el cost computacional en el pitjor dels casos**
  - **En el cas mitjà de vegades es fa difícil conèixer la distribució que segueixen les dades d'entrada**
- **El tercer factor que influeix en el cost computacional -> adoptarem el criteri d'ignorar-lo. Criteri Asimptòtic (constant multiplicativa)**
- **En resum, l'important a considerar serà la dependència del primer factor. La grandària del conjunt de dades d'entrada (*n*)**

Retorn