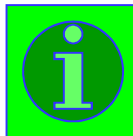


Scheduling di processi in presenza di risorse condivise

Eugenio Faldella

Dipartimento di Informatica - Scienza e Ingegneria
Scuola di Ingegneria e Architettura, Università di Bologna



eugenio.faldella@unibo.it
<http://www.ing.unibo.it>

PROTOCOLLI DI ACCESSO A RISORSE CONDIVISE ...

Modello generale di riferimento

- sistema di elaborazione monoprocesso;
- N processi P_1, \dots, P_N periodici o sporadici;
- strategia di schedulazione del tipo preemptive, priority-driven;
- priorità "nominale" dei processi definita:
 - 1) staticamente (in accordo all'algoritmo RMPO o DMPO),
 - 2) dinamicamente (in accordo all'algoritmo EDF);
- ogni processo completa l'esecuzione di ciascun job senza autosospendersi;
- M risorse logiche o fisiche condivise R_1, \dots, R_M , ciascuna con un numero finito u_k ($k = 1, \dots, M$) di unità indistinguibili:
 - a) una sola unità per risorsa,
 - b) una o più unità per risorsa.

... PROTOCOLLI DI ACCESSO A RISORSE CONDIVISE

Protocolli esaminati ed ipotesi considerate (1 / 2 - a / b):

■ Non-Preemptive Critical Section Protocol	✓	✓	✓	✓
■ Priority Inheritance Protocol	✓		✓	
■ Priority Ceiling Protocol	✓		✓	
■ Immediate Priority Ceiling Protocol	✓		✓	
■ Preemption Ceiling Protocol		✓	✓	
■ Stack Resource Policy	✓	✓	✓	✓

IL PROTOCOLLO "NON-PREEMPTIVE CRITICAL SECTION" (NPCS) [MOK (83)] ...

Nessun processo in esecuzione all'interno di una sezione critica può subire preemption da parte di altri processi.

A tal fine è sufficiente associare ad ogni processo P_i ($i = 1..N$), indipendentemente dalla sua priorità "nominale" p_i , una priorità "corrente"

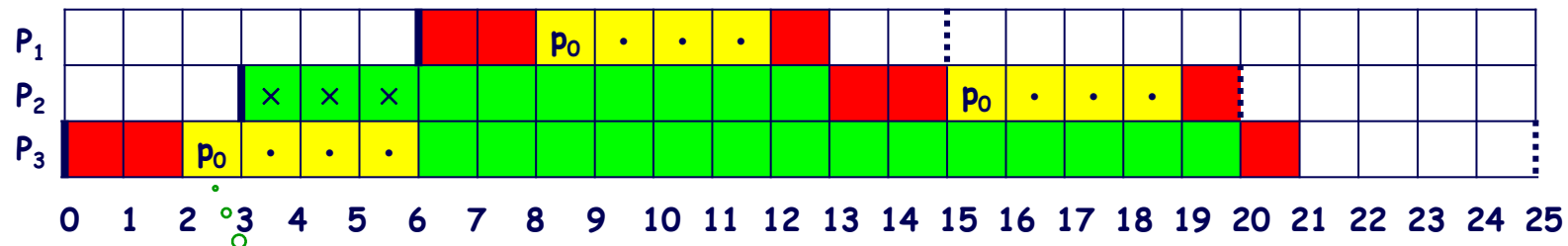
$$\pi_i = p_0 > \max \{p_1, p_2, \dots, p_N\}$$

allorché inizia l'esecuzione di una sezione critica, e di nuovo associare ad esso la sua priorità nominale p_i allorché termina l'esecuzione della sezione critica.

... IL PROTOCOLLO NPCS ...

Non implica la conoscenza a priori delle risorse utilizzate dai processi.

		priorità	a [t.u.]	d [t.u.]	C [t.u.]	
A_1	P_1	p_1 (max)	6	15	7	
	P_2	p_2	3	20	7	
	P_3	p_3 (min)	0	25	7	

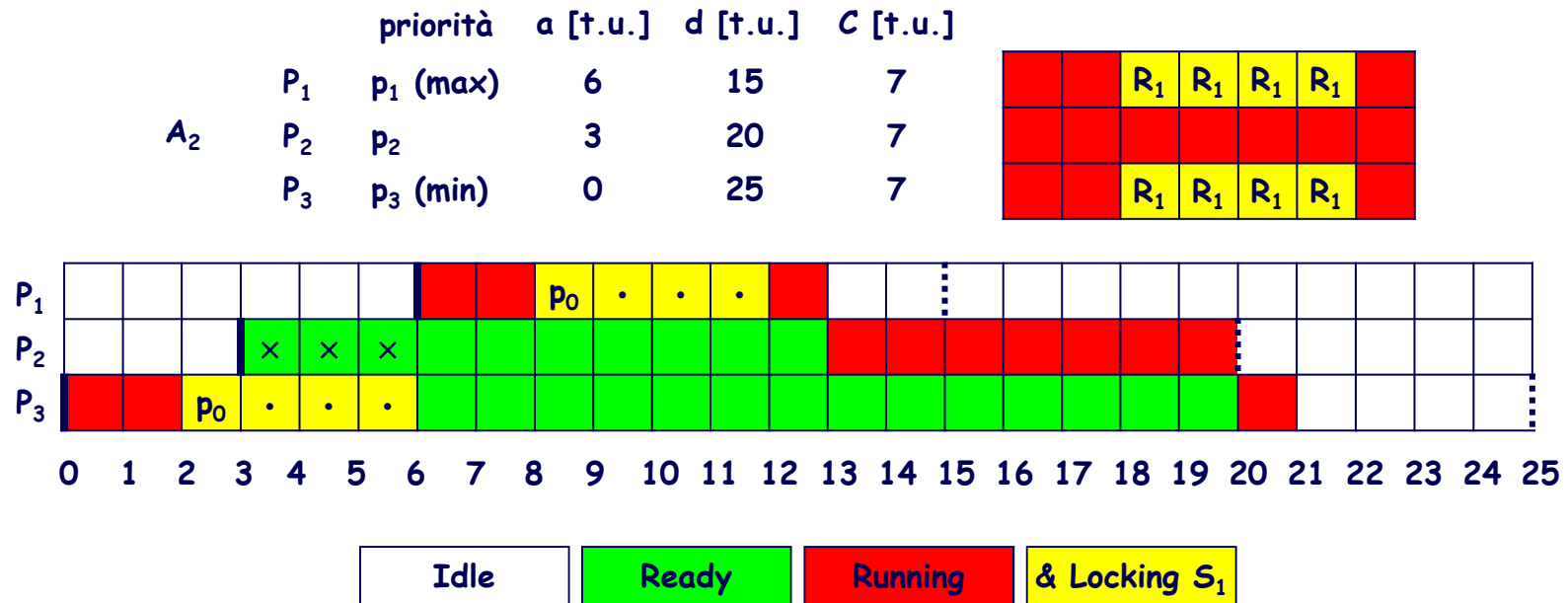


(°) Per ogni processo P_i ($\forall i$), la priorità corrente π_i è esplicitamente evidenziata solo se differisce dalla priorità nominale p_i ("." denota più sinteticamente l'ultimo valore di π_i indicato).



... IL PROTOCOLLO NPCS ...

Previene inversioni di priorità incontrollate.



... IL PROTOCOLLO NPCS ...

Previene la concatenazione di blocchi
(un processo può essere bloccato al più una volta).

		priorità	a [t.u.]	d [t.u.]	C [t.u.]	
A_3	P_1	p_1 (max)	6	15	7	
	P_2	p_2	3	20	7	
	P_3	p_3 (min)	0	25	7	

Legend: Idle Ready Running & Locking S_1 & Locking S_2

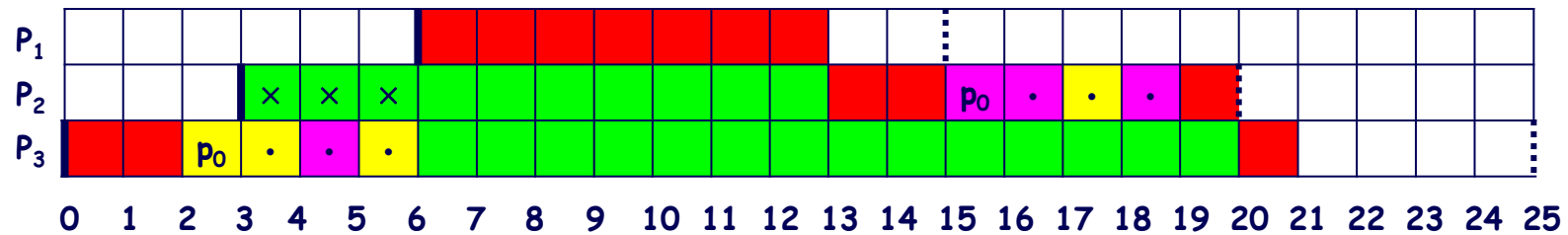
... IL PROTOCOLLO NPCS ...

Previene situazioni di deadlock.

		priorità	a [t.u.]	d [t.u.]	C [t.u.]		
	P ₁	p ₁ (max)	6	15	7		
A ₄	P ₂	p ₂	3	20	7		
	P ₃	p ₃ (min)	0	25	7		

		R ₂	R ₂	R ₁	R ₂	
		R ₁	R ₁	R ₂	R ₁	

!?



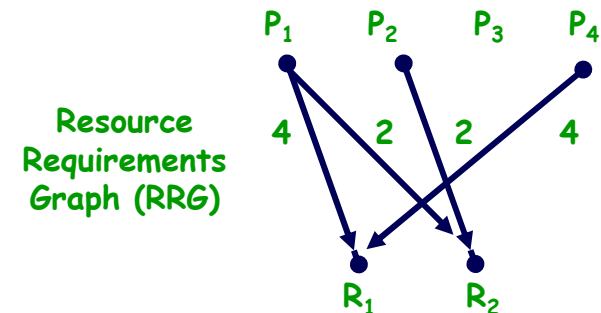
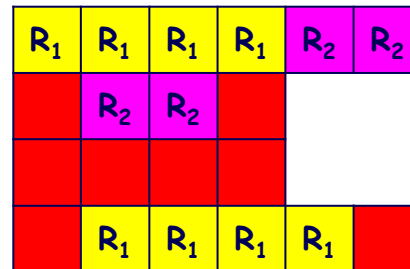
... IL PROTOCOLLO NPCS ...

Nei sistemi a priorità statica che utilizzano la strategia di schedulazione RMPO o DMPO, il massimo tempo di blocco B_i di un processo P_i avente priorità p_i è uguale al tempo di esecuzione Z_{jk} della più lunga sezione critica relativa ad una qualsiasi risorsa R_k utilizzata da un qualunque processo P_j di priorità $p_j < p_i$:

$$B_i = \max_{j,k} \{ Z_{jk} \mid p_j < p_i \} \quad \forall i$$

Qualche esempio, facendo riferimento alla tipica notazione utilizzata per descrivere parametri temporali [t.u.] e sezioni critiche di ogni processo: $P_j (\phi_j, T_j, C_j, D_j; [R_k; Z_{jk}] \dots [R_m; Z_{jm} [R_n; Z_{jn}]])$

A_5 { $P_1 (6, 15, 6, 15; [R_1;4] [R_2;2])$
 $P_2 (4, 20, 4, 20; [R_2;2])$
 $P_3 (2, 50, 4, 50)$
 $P_4 (0, 100, 6, 100; [R_1;4])$



Strategia
di scheduling
RMPO



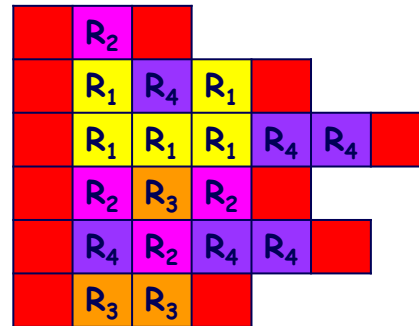
priorità		R_1	R_2
p_1 (max)	P_1	4	2
p_2	P_2		2
p_3	P_3		
p_4 (min)	P_4	4	

Il massimo tempo di blocco ([t.u.])
di ciascun processo risulta:
 $B_1 = B_2 = B_3 = 4, B_4 = 0.$

... IL PROTOCOLLO NPCS ...

Applicazione A_6

P_1 (9, 30, 3, 30; $[R_2;1]$)
 P_2 (8, 40, 5, 40; $[R_1;1]$ $[R_4;1]$ $[R_1;1]$)
 P_3 (6, 50, 7, 50; $[R_1;3]$ $[R_4;2]$)
 P_4 (4, 60, 5, 60; $[R_2;1]$ $[R_3;1]$ $[R_2;1]$)
 P_5 (2, 70, 6, 70; $[R_4;1]$ $[R_2;1]$ $[R_4;2]$)
 P_6 (0, 80, 4, 80; $[R_3;2]$)



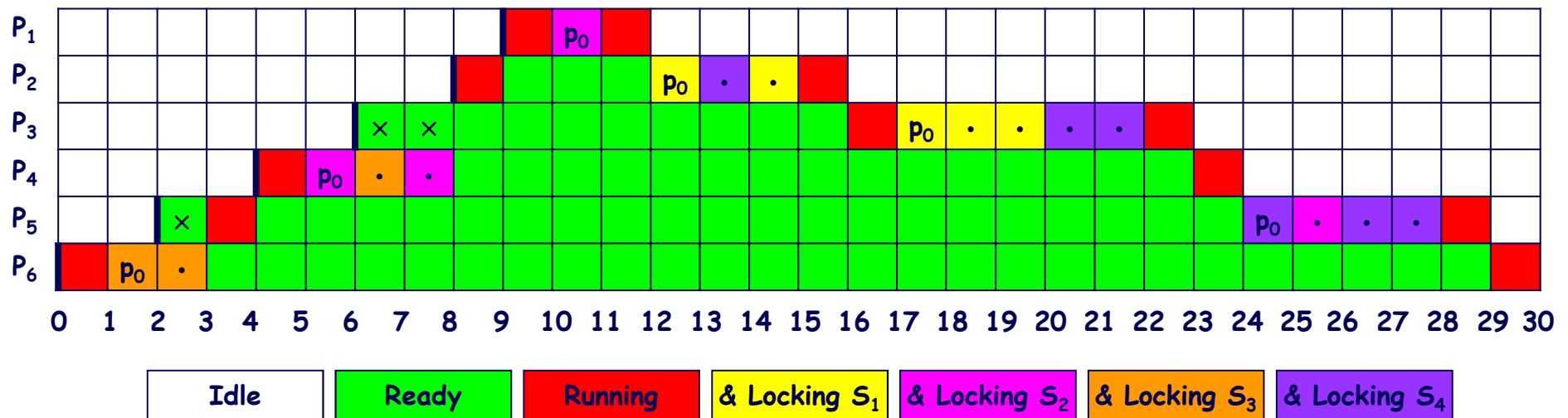
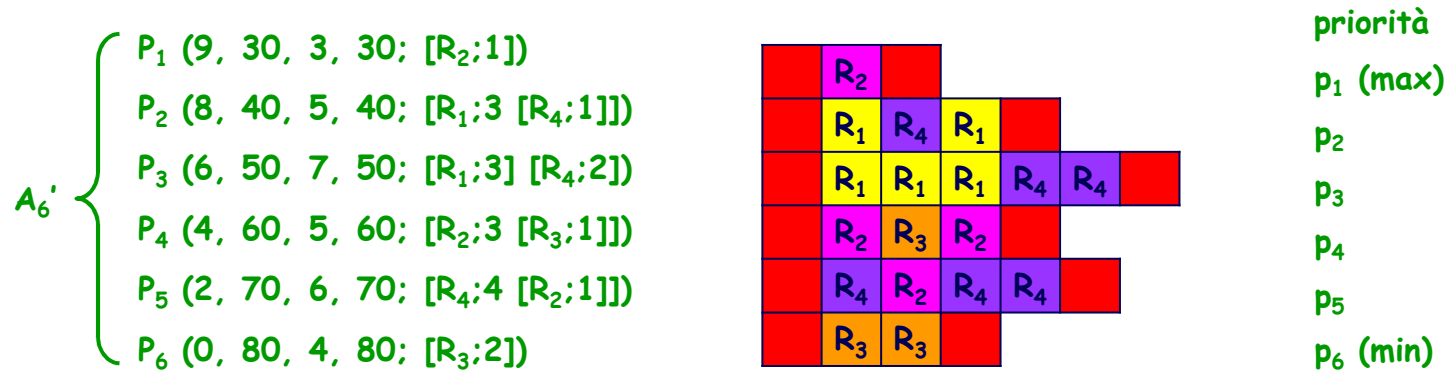
Applicazione A_6'

P_1 (9, 30, 3, 30; $[R_2;1]$)
 P_2 (8, 40, 5, 40; $[R_1;3]$ $[R_4;1]$)
 P_3 (6, 50, 7, 50; $[R_1;3]$ $[R_4;2]$)
 P_4 (4, 60, 5, 60; $[R_2;3]$ $[R_3;1]$)
 P_5 (2, 70, 6, 70; $[R_4;4]$ $[R_2;1]$)
 P_6 (0, 80, 4, 80; $[R_3;2]$)

A_6' differisce da A_6 per la modalità di accesso alle risorse da parte dei tre processi P_2 , P_4 e P_5 : sezioni critiche annidate con rilascio delle risorse secondo l'ordine LIFO.

priorità		R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	
p_1 (max)	P_1		1				1			$B_1 = 3 / 4$
p_2	P_2	1			1	3			1	$B_2 = 3 / 4$
p_3	P_3	3			2	3			2	$B_3 = 2 / 4$
p_4	P_4		1	1			3	1		$B_4 = 2 / 4$
p_5	P_5		1		2		1		4	$B_5 = 2 / 2$
p_6 (min)	P_6			2				2		$B_6 = 0 / 0$
		A_6				A_6'				A_6 / A_6'

... IL PROTOCOLLO NPCS ...



Nell'ipotesi che a_i ($i = 1, \dots, 4$) $> a_5 > a_6 = \forall$,

P_5 subisce il massimo tempo di blocco $B_5 = 2$ t.u. se $a_5 = a_6 + 1 + \varepsilon$,

P_i ($i = 1, \dots, 4$) subisce il massimo tempo di blocco $B_i = 4$ t.u. se $a_5 < a_6 + 1$ (o $a_5 > a_6 + 3$) e $a_i = a_5 + 1 + \varepsilon$,
con $\varepsilon \rightarrow 0$.

... IL PROTOCOLLO NPCS ...

Nei sistemi a priorità dinamica che utilizzano la strategia di schedulazione EDF, un processo P_i può subire un blocco da parte di un processo P_j solo se:

- il processo P_j è in esecuzione allorché si attiva P_i , ovvero

$$r_j < r_i;$$

- la priorità nominale di P_i è maggiore di quella di P_j , ovvero

$$r_i + D_i = d_i < d_j = r_j + D_j.$$

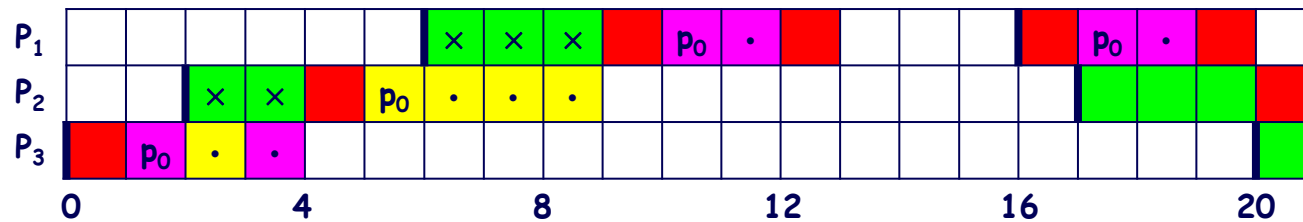
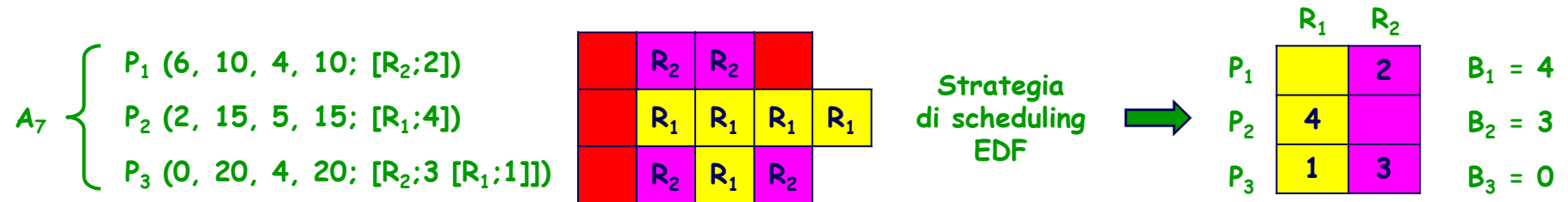
Entrambe le condizioni sono soddisfatte solo se $D_j > D_i$.

Il massimo tempo di blocco B_i di un processo P_i con deadline relativa D_i è pertanto dato dal tempo di esecuzione Z_{jk} della più lunga sezione critica relativa a qualunque risorsa R_k utilizzata da un processo P_j con deadline relativa $D_j > D_i$:

$$B_i = \max_{j,k} \{ Z_{jk} \mid D_j > D_i \} \quad \forall i$$

... IL PROTOCOLLO NPCS ...

Con riferimento alla seguente applicazione, il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:



Nell'ipotesi che $a_1 > a_2 > a_3 = \forall$,

P₂ subisce il massimo tempo di blocco B₂ = 3 t.u. se $a_2 = a_3 + 1 + \varepsilon$,

P₁ subisce il massimo tempo di blocco B₁ = 4 t.u. se $a_2 < a_3 + 1$ (o $a_2 > a_3 + 4$) e $a_1 = a_2 + 1 + \varepsilon$,

con $\varepsilon \rightarrow 0$.

... IL PROTOCOLLO NPCS

Protocollo molto semplice,
efficiente in sistemi in cui tutti (o quasi) i processi competono per tutte (o quasi)
le risorse, con sezioni critiche tutte di breve durata.

Una limitazione fondamentale:
se una risorsa (o più se annidate) è occupata da un processo,
tutte le altre risultano inaccessibili e tutti i processi di priorità superiore bloccati,
anche in assenza di condizioni conflittuali.



IL PROTOCOLLO "PRIORITY INHERITANCE" (PI) [CORNHILL et al. (87)] ...

Regole di schedulazione dei processi, di allocazione delle risorse, di gestione delle priorità dei processi allorché detengono risorse:

La schedulazione dei processi (pronti) è operata in base alle relative priorità correnti. La priorità corrente π_i di un processo P_i ($\forall i$) coincide con:

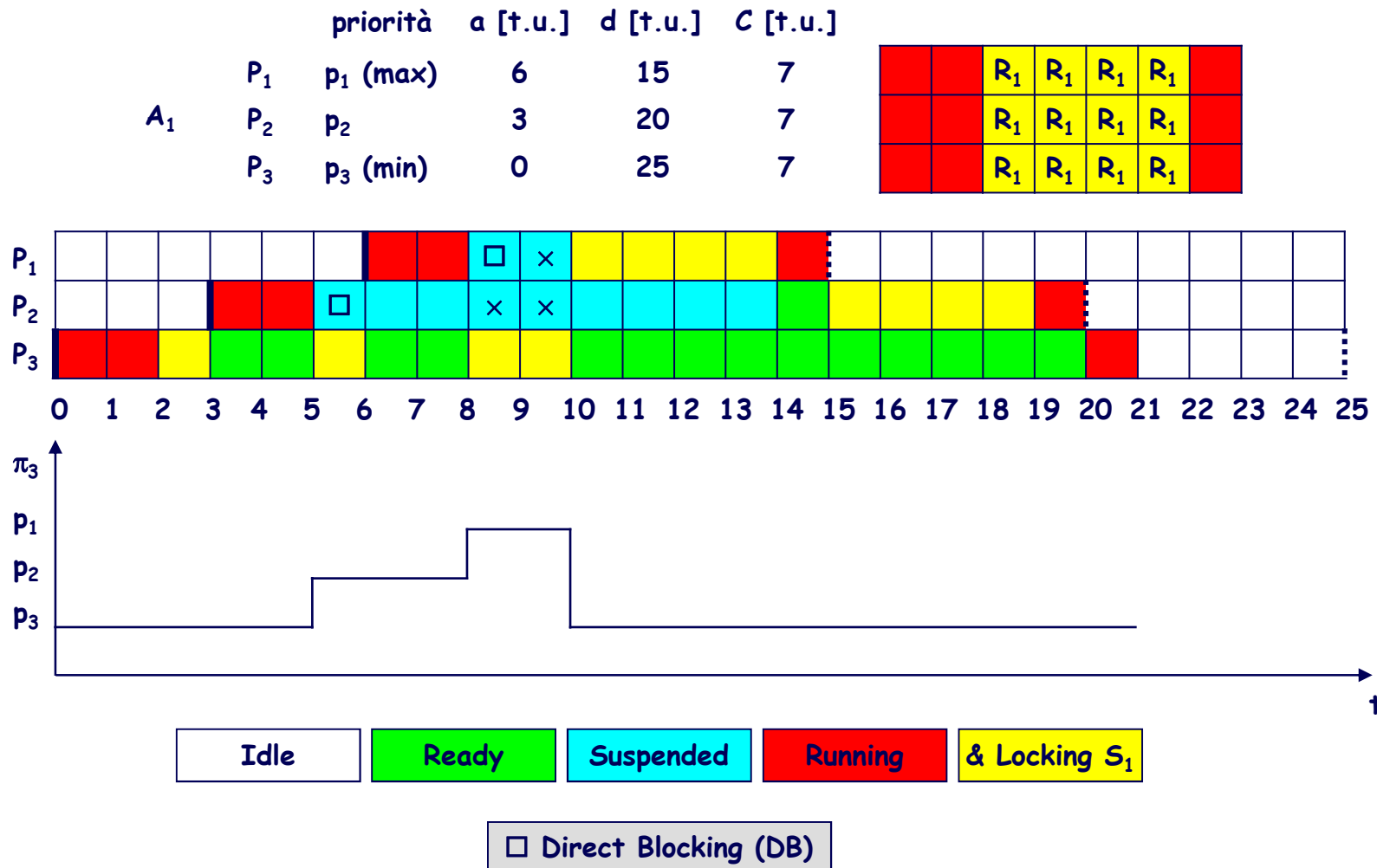
- la corrispondente priorità nominale p_i nel caso in cui P_i non detenga alcuna risorsa richiesta da processi di priorità superiore;
- la più alta fra le priorità correnti dei processi da esso bloccati in caso contrario.

Ad un processo è negato l'accesso ad una risorsa solo se essa è già occupata. In tal caso la priorità corrente del processo bloccato viene ereditata ^(°) dal processo che detiene la risorsa.

^(°) L'ereditarietà della priorità è transitiva: se un processo P_H di priorità p_H (High priority) è bloccato da un processo P_M di priorità p_M (Medium priority) e P_M a sua volta è bloccato da un processo P_L di priorità p_L (Low priority), allora P_L eredita la priorità p_H di P_H via P_M . Tale forma di ereditarietà si può manifestare solo in presenza di sezioni critiche annidate.

... IL PROTOCOLLO PI ...

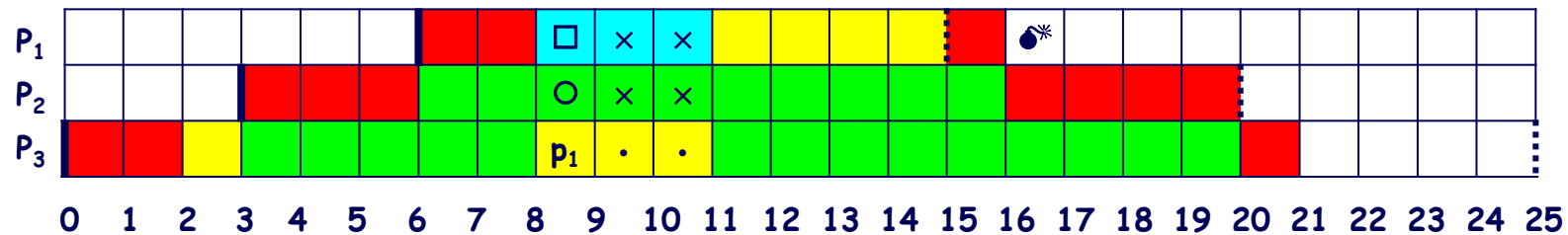
Non implica la conoscenza a priori delle risorse utilizzate dai processi.



... IL PROTOCOLLO PI ...

Previene inversioni di priorità incontrollate
(grazie al meccanismo della ereditarietà della priorità).

		priorità	a [t.u.]	d [t.u.]	C [t.u.]	
	P ₁	p ₁ (max)	6	15	7	
A ₂	P ₂	p ₂	3	20	7	
	P ₃	p ₃ (min)	0	25	7	

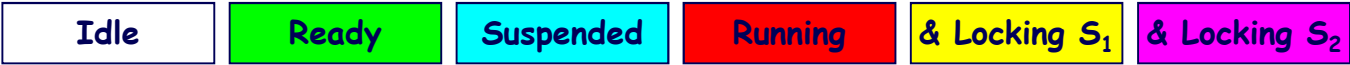
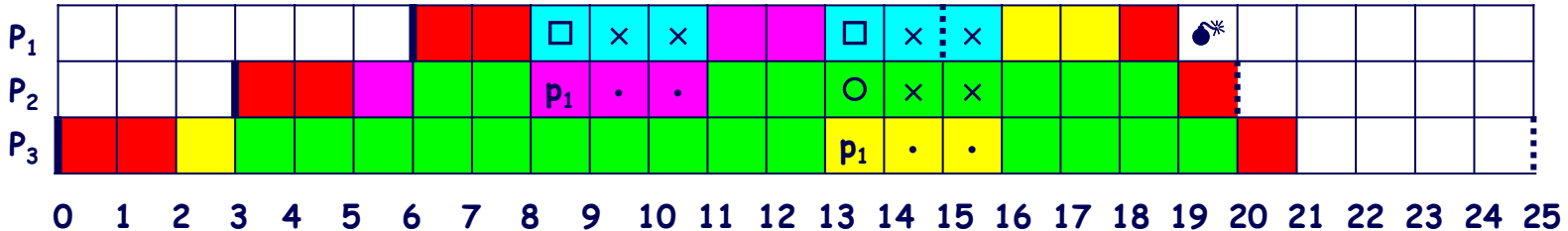


○ Indirect (Push-Through) Blocking (IB)

... IL PROTOCOLLO PI ...

Non previene la concatenazione di blocchi.

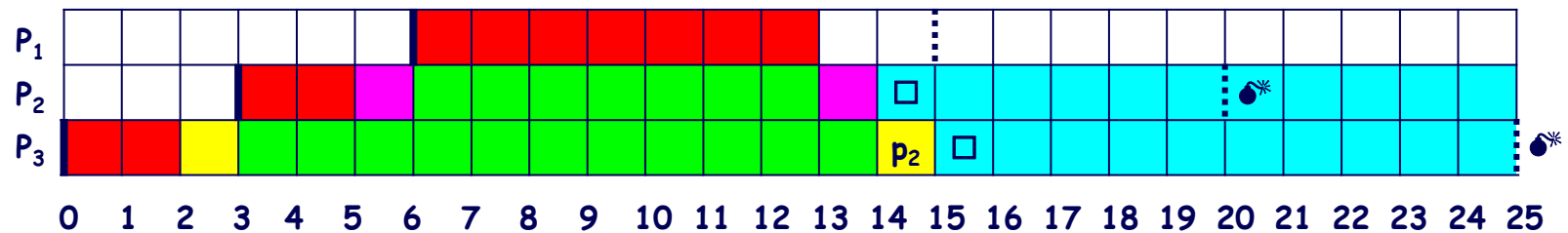
		priorità	a [t.u.]	d [t.u.]	C [t.u.]							
A ₃	P ₁	p ₁ (max)	6	15	7			R ₂	R ₂	R ₁	R ₁	
	P ₂	p ₂	3	20	7			R ₂	R ₂	R ₂	R ₂	
	P ₃	p ₃ (min)	0	25	7			R ₁	R ₁	R ₁	R ₁	



... IL PROTOCOLLO PI ...

Non previene situazioni di deadlock.

		priorità	a [t.u.]	d [t.u.]	C [t.u.]
	P ₁	p ₁ (max)	6	15	7
A ₄	P ₂	p ₂	3	20	7
	P ₃	p ₃ (min)	0	25	7



... IL PROTOCOLLO PI ...

(1) Sezioni critiche non annidate

Indicato con

$$PC_k = \max_j \{p_j \mid P_j \text{ usa } R_k\} \quad \forall k$$

il tetto di priorità (Priority Ceiling) di ciascuna risorsa R_k ($k = 1, \dots, M$), coincidente con la massima fra le priorità nominali dei processi che ad essa possono accedere,

il numero massimo di blocchi, diretti o indiretti, in cui può incorrere il processo P_i ($i = 1, \dots, N$) di priorità nominale p_i è dato da:

$$n_i = \min(l_i, r_i),$$

essendo l_i il numero di processi di priorità nominale $< p_i$ che accedono ad almeno una risorsa con tetto di priorità $\geq p_i$, e r_i il numero di (semafori che regolamentano gli accessi a) risorse con tetto di priorità $\geq p_i$ utilizzate da almeno un processo con priorità nominale $< p_i$.

... IL PROTOCOLLO PI ...

A_6 {

 P_1 (9, 30, 3, 30; $[R_2;1]$)

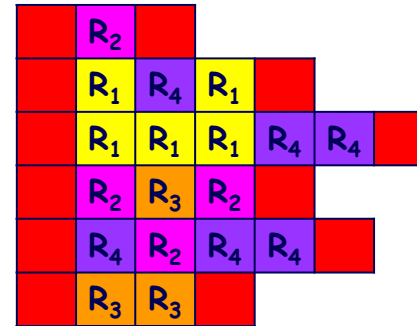
 P_2 (8, 40, 5, 40; $[R_1;1]$ $[R_4;1]$ $[R_1;1]$)

 P_3 (6, 50, 7, 50; $[R_1;3]$ $[R_4;2]$)

 P_4 (4, 60, 5, 60; $[R_2;1]$ $[R_3;1]$ $[R_2;1]$)

 P_5 (2, 70, 6, 70; $[R_4;1]$ $[R_2;1]$ $[R_4;2]$)

 P_6 (0, 80, 4, 80; $[R_3;2]$)



l	r	n
2	1	1
3	3	3
2	2	2
2	3	2
1	1	1
0	0	0

p_i, PC_k

p_1 (max)

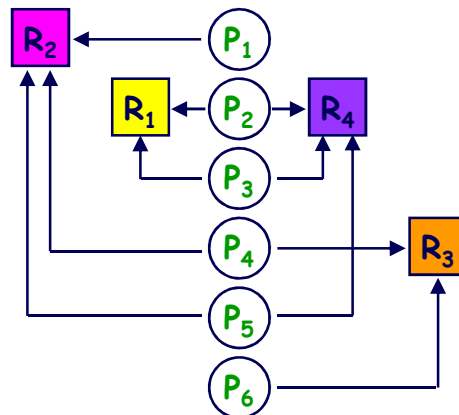
p_2

p_3

p_4

p_5

p_6 (min)



$P_1: l_1 = 2$ (P_4, P_5), $r_1 = 1$ (R_2), $n_1 = 1$ (DB)

(la risorsa condivisa può essere detenuta da un solo processo di priorità inferiore).

$P_2: l_2 = 3$ (P_3, P_4, P_5), $r_2 = 3$ (R_1, R_2, R_4), $n_2 = 3$ (DB, IB, DB).

$P_3: l_3 = 2$ (P_4, P_5), $r_3 = 2$ (R_2, R_4), $n_3 = 2$ (IB, DB).

$P_4: l_4 = 2$ (P_5, P_6), $r_4 = 3$ (R_2, R_3, R_4), $n_4 = 2$ (DB/IB, DB)
(ogni processo di priorità inferiore può detenere al più una risorsa).

$P_5: l_5 = 1$ (P_6), $r_5 = 1$ (R_3), $n_5 = 1$ (IB).

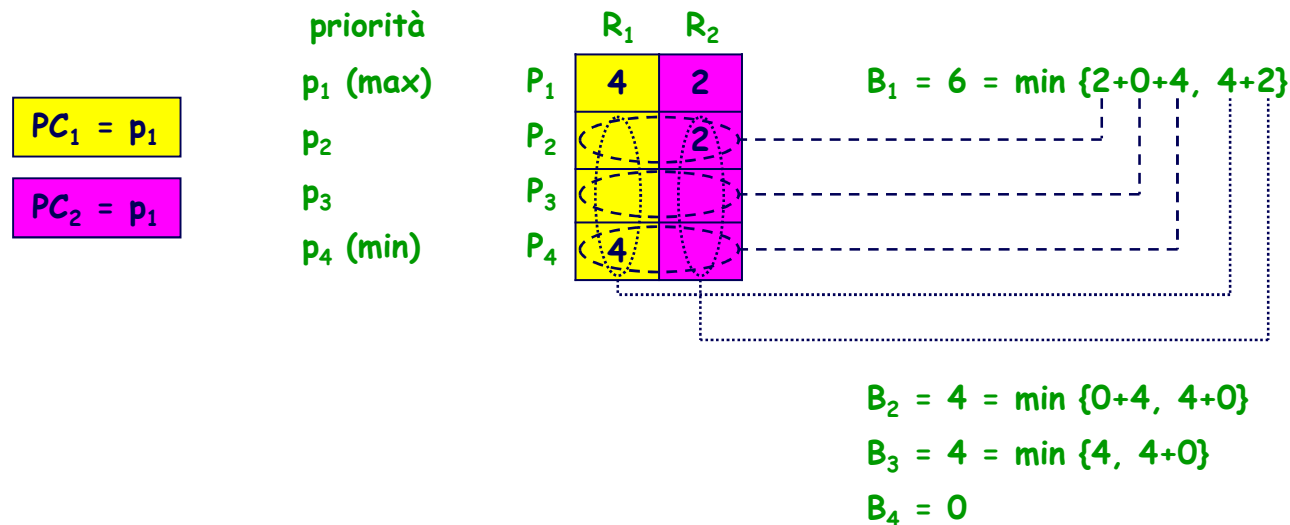
$P_6: l_6 = 0$, $r_6 = 0$, $n_6 = 0$.

... IL PROTOCOLLO PI ...

Un "upper bound" del massimo tempo di blocco B_i di ciascun processo P_i ($i = 1, \dots, N$) può essere identificato molto semplicemente mediante il seguente algoritmo (complessità computazionale $O(r_i l_i^2)$):

$$B_i = \min \left\{ \sum_{j | p_j < p_i} \max_k \{Z_{jk} \mid PC_k \geq p_i\}, \sum_{k | PC_k \geq p_i} \max_j \{Z_{jk} \mid p_j < p_i\} \right\} \quad \forall i$$

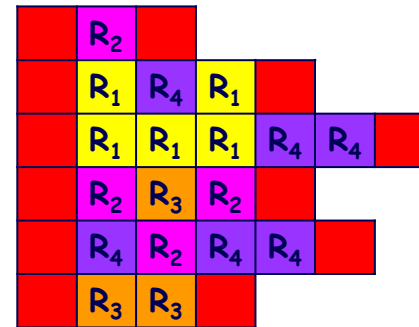
Con riferimento all'applicazione A_5 , il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:



... IL PROTOCOLLO PI ...

Con riferimento all'applicazione A_6 , il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:

$P_1 (9, 30, 3, 30; [R_2;1])$
 $P_2 (8, 40, 5, 40; [R_1;1] [R_4;1] [R_1;1])$
 $P_3 (6, 50, 7, 50; [R_1;3] [R_4;2])$
 $P_4 (4, 60, 5, 60; [R_2;1] [R_3;1] [R_2;1])$
 $P_5 (2, 70, 6, 70; [R_4;1] [R_2;1] [R_4;2])$
 $P_6 (0, 80, 4, 80; [R_3;2])$



$PC_1 = p_2$

$PC_2 = p_1$

$PC_3 = p_4$

$PC_4 = p_2$

priorità

p_1 (max)

p_2

p_3

p_4

p_5

p_6 (min)

	R_1	R_2	R_3	R_4
P_1		1		
P_2	1			1
P_3	3			2
P_4		1	1	
P_5		1		2
P_6			2	

$$B_1 = 1 = \min \{1+1, 1\}$$

$$B_2 = 6 = \min \{3+1+2, 3+1+2\}$$

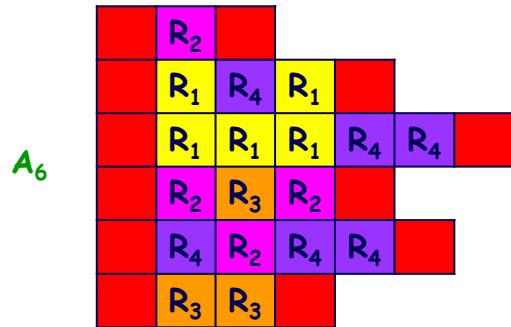
$$B_3 = 3 = \min \{1+2, 1+2\}$$

$$B_4 = 4 = \min \{2+2, 1+2+2\}$$

$$B_5 = 2 = \min \{2, 2\}$$

$$B_6 = 0$$

... IL PROTOCOLLO PI ...



$$B_1 = 1$$

$$B_2 = 6$$

$$B_3 = 3$$

$$B_4 = 4$$

$$B_5 = 2$$

$$B_6 = 0$$

I processi subiscono il massimo tempo di blocco se ad esempio:

$$a_6 = \forall,$$

$$a_5 = a_6 + 1 + \varepsilon,$$

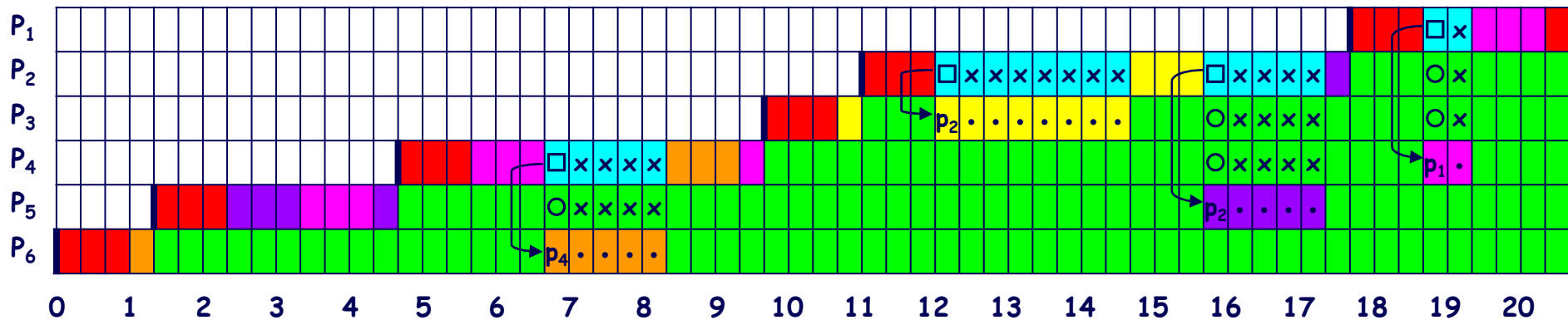
$$a_4 = a_5 + 3 + \varepsilon,$$

$$a_3 = a_4 + 5,$$

$$a_2 = a_3 + 1 + \varepsilon,$$

$$a_1 > a_2, a_1 < a_2 + 10 - 2\varepsilon,$$

con $\varepsilon \rightarrow 0$.



... IL PROTOCOLLO PI ...

In generale l'algoritmo fornisce una valutazione conservativa, in quanto nel calcolo di:

$$B_i = \min \left\{ \sum_{j | p_j < p_i} \max_k \{Z_{jk} \mid PC_k \geq p_i\}, \sum_{k | PC_k \geq p_i} \max_j \{Z_{jk} \mid p_j < p_i\} \right\} \quad \forall i$$

① il contributo
derivante da più
processi può riguardare
l'accesso ad una stessa
risorsa

② il contributo
derivante dall'accesso
a più risorse può
riguardare uno stesso
processo

Con riferimento all'applicazione A_5' , il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:

$PC_1 = p_1$
$PC_2 = p_1$

priorità

p_1 (max)
p_2
p_3
p_4 (min)

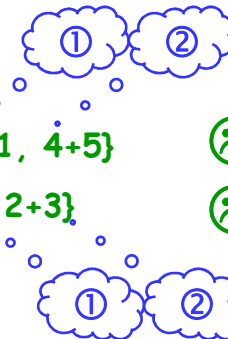
	R_1	R_2
P_1	6	7
P_2	4	5
P_3	2	3
P_4		1

$B_1 = 9 = \min \{5+3+1, 4+5\}$

$B_2 = 4 = \min \{3+1, 2+3\}$

$B_3 = 1 = \min \{1, 1\}$

$B_4 = 0$



... IL PROTOCOLLO PI ...

Metodo meno conservativo, applicabile anche in presenza di sezioni critiche annidate, di complessità computazionale esponenziale (RAJKUMAR [91]):

1. Si costruisce un albero ad un livello (simpleton) per ogni processo P_j di priorità $p_j < p_i$, in cui ogni ramo rappresenta una risorsa R_k utilizzata da P_j che può comportare il blocco di P_i . Il peso di ogni ramo è $\max \{Z_{jk}\}$ (nel caso di sezioni critiche annidate si considera il tempo di esecuzione della sezione critica più esterna).
2. Si costruisce l'albero finale componendo ordinatamente tutti gli alberi ad un livello, partendo da quello corrispondente al processo P_j di priorità massima e collegando via via ciascun singleton a tutte le foglie dell'albero in costruzione.
3. Si esplorano tutti i percorsi che conducono dalla radice alle foglie dell'albero finale, calcolando per ciascuno di essi la somma dei pesi associati ai vari rami (nel caso siano presenti più rami corrispondenti ad una stessa risorsa si considera soltanto il contributo del ramo di peso maggiore). Il massimo tempo di blocco B_i di P_i coincide con il più grande di tali valori.

... IL PROTOCOLLO PI ...

Calcolo di B_1

Con riferimento all'applicazione A_5'

priorità		R_1	R_2
p_1 (max)	P_1	6	7
p_2	P_2	4	5
p_3	P_3	2	3
p_4 (min)	P_4		1

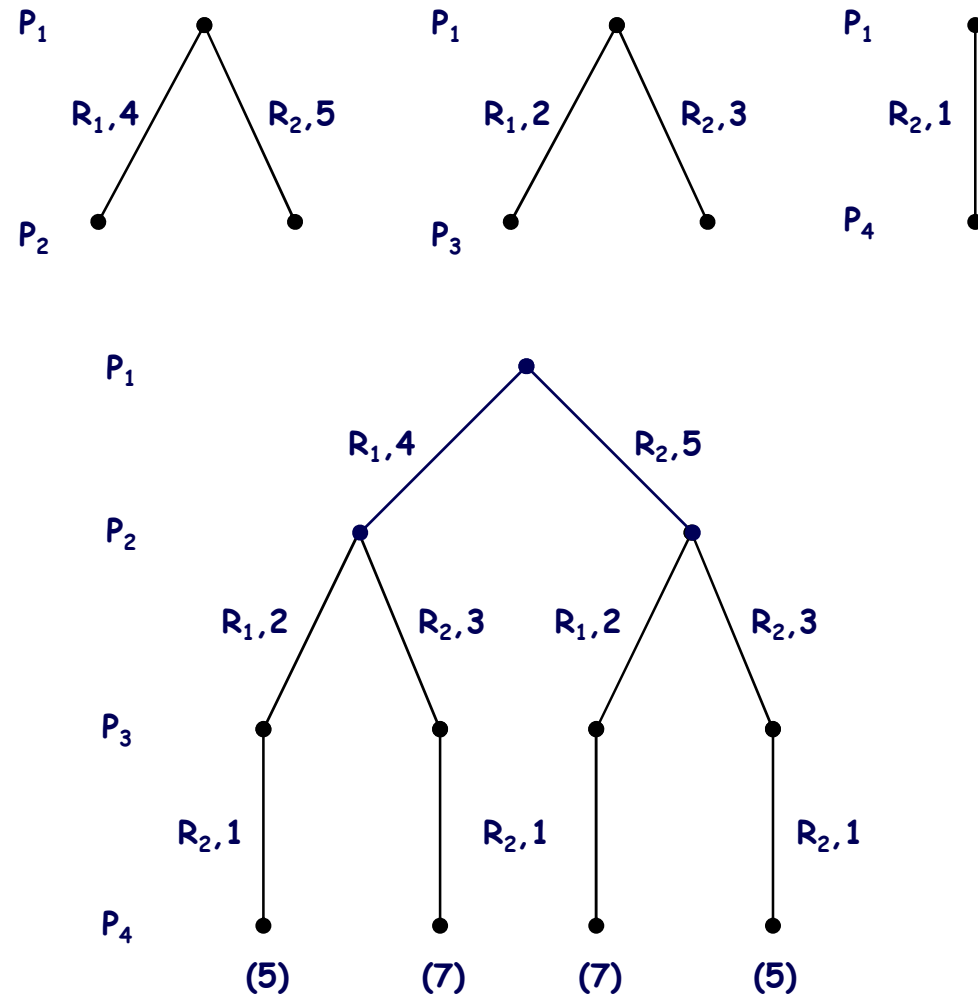
il massimo tempo di blocco ([t.u.])
di ciascun processo risulta:

$$B_1 = 7$$

$$B_2 = 3$$

$$B_3 = 1$$

$$B_4 = 0$$



... IL PROTOCOLLO PI ...

In alternativa si può ricorrere alla costruzione dell'albero di ricerca duale.

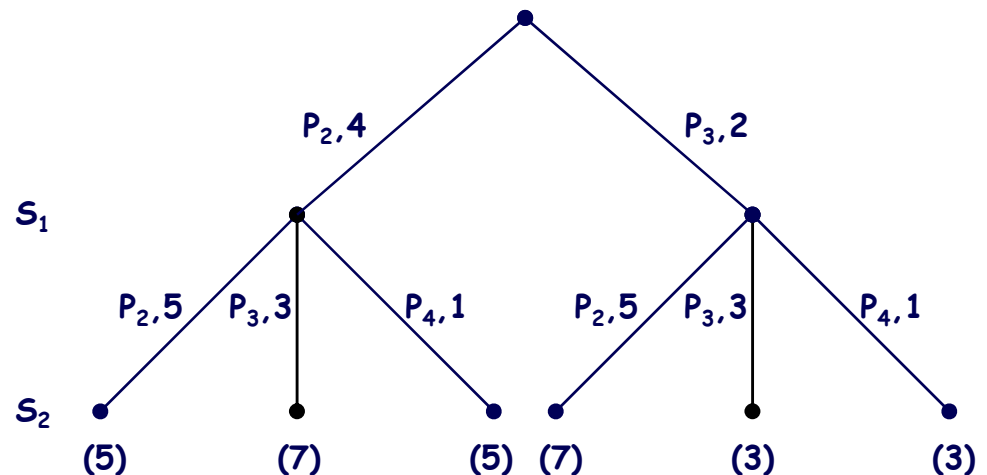
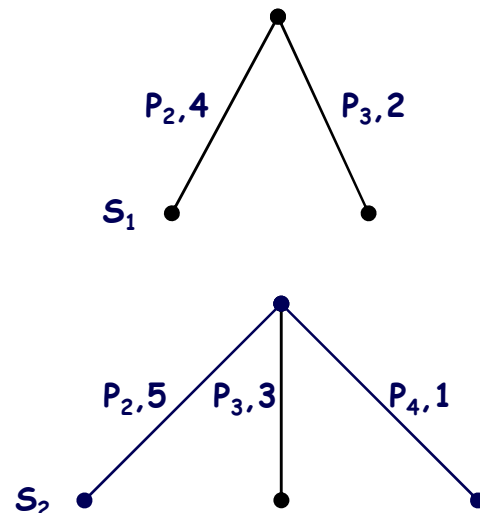
I simpleton sono associati ai semafori che possono bloccare P_i . I rami di un simpleton rappresentano i processi che accedono al corrispondente semaforo. Il peso del ramo corrispondente al processo P_j nell'albero associato al semaforo S_k coincide con il tempo di esecuzione Z_{jk} della più lunga sezione critica di P_j protetta dal semaforo S_k .

La costruzione dell'albero finale, così come l'attribuzione di un peso ad ogni percorso che conduce dalla radice ad una foglia, procede in maniera analoga. In particolare, se in un percorso sono presenti più rami corrispondenti ad uno stesso processo, si considera soltanto il contributo del ramo di peso maggiore.

A_5' : calcolo di B_1

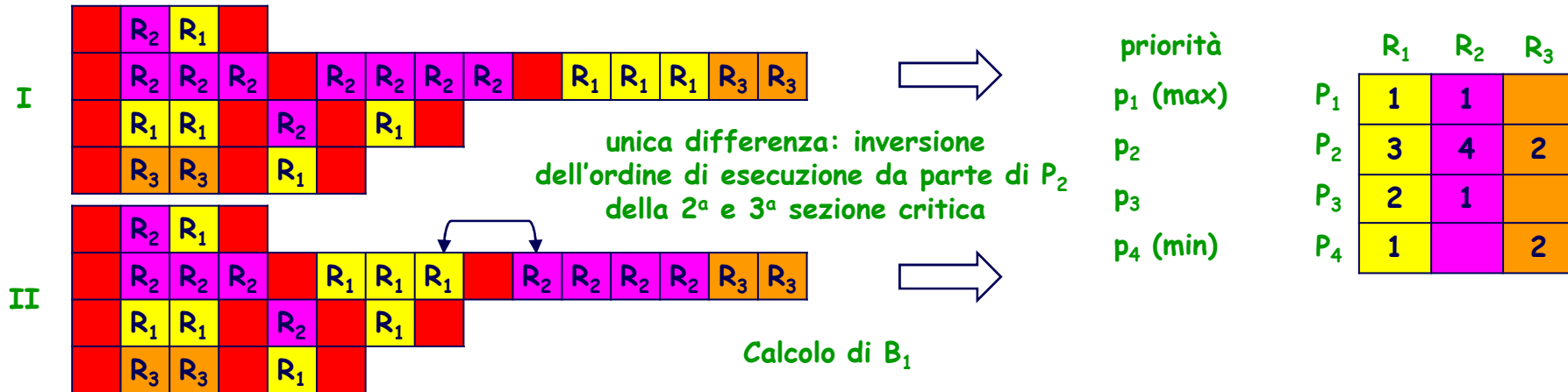
	R_1	R_2
P_1	6	7
P_2	4	5
P_3	2	3
P_4		1

$B_1 = 7$

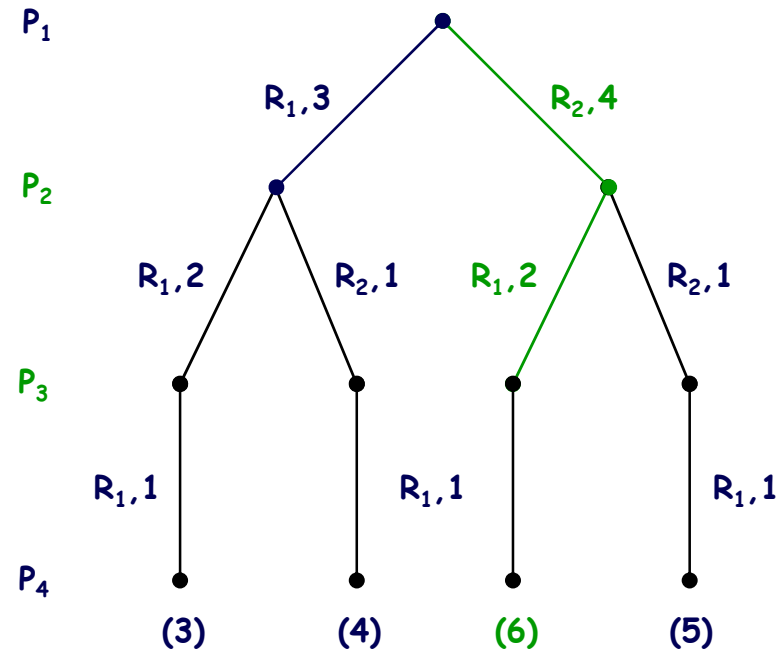
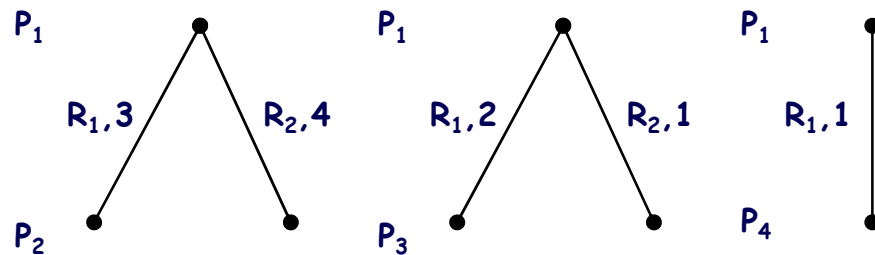


... IL PROTOCOLLO PI ...

Rajkumar: still an upper bound

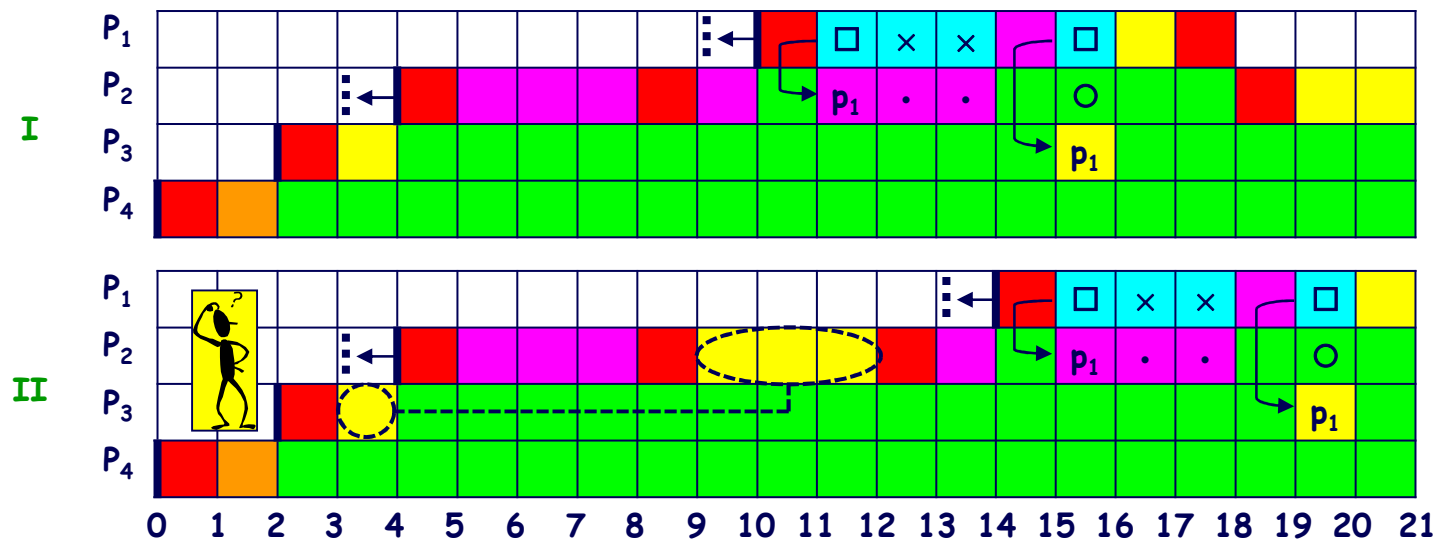
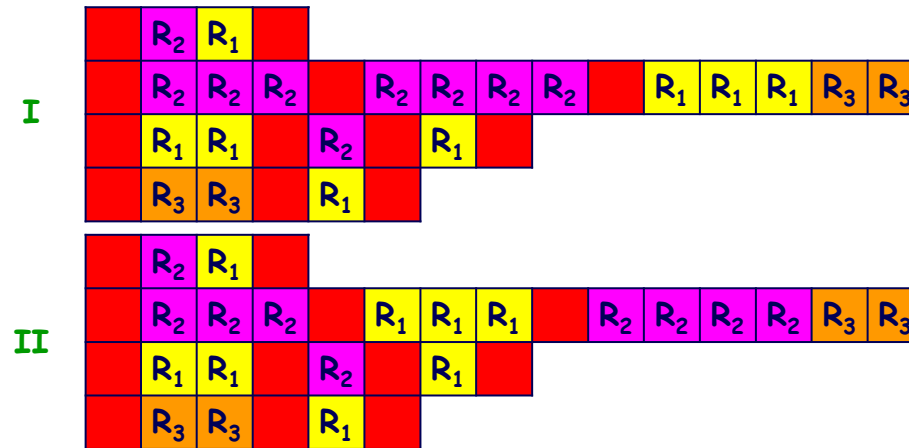


$B_1 = \min \{4+2+1, 3+4\} = 7$ ☹️



... IL PROTOCOLLO PI ...

I e II: $B_1 = 6$!?



... IL PROTOCOLLO PI ...

Binary Linear Programming^(*) Models [FALDELLA, LORETI (2020)]

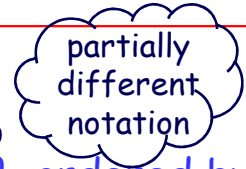
Base Model: starting from the same assumptions, it allows to compute the blocking bounds identifiable through the Rajkumar's algorithm with polynomial - rather than exponential - complexity.

Refined Model: it allows to compute the worst-case blocking times - rather than just upper bounds - for any task set in any possible operating condition.

^(*) Linear Programming (LP) is a class of optimization problems that involves only linear functions as regards both the objective and the constraints. Integer Linear Programming (ILP) problems are particular LP problems that envisage the set of decision variables in the solution to be composed of integer values only. A subclass of ILP problems is represented by Binary Linear Programming (BLP) problems, where the values of the decision variables are further bounded to be in $\{0, 1\}$.

... IL PROTOCOLLO PI ...

Base Model (M_1)



Let us consider a uniprocessor system and a set of N tasks $\tau = \{T_1, \dots, T_n, \dots, T_N\}$, ordered by descending nominal priority $\{P_1, \dots, P_n, \dots, P_N\}$ (i.e. $P_l < P_h \forall l > h$), which share a set of M resources $\{\ell_1, \dots, \ell_m, \dots, \ell_M\}$ guarded by M distinct binary semaphores $\{S_1, \dots, S_m, \dots, S_M\}$. Tasks can be periodic (with unknown periods and initial release offsets), or sporadic. For the sake of simplicity, we suppose that each task has a unique priority. *Every task may access any resource more than once during its execution. All resource accesses are not nested, and tasks do not self-suspend.*

The problem of computing for all task phasings the worst-case blocking time B_n that each task T_n ($n=1, 2, \dots, N-1$) may experience is indeed complex, even in the case of non-nested critical sections.

To this end we consider, for each lower-priority task T_l ($1 \leq n < l \leq N$), the set $\gamma_{n,l}$ of the *longest critical sections* that may block (either directly or via push-through) T_n :

$$\gamma_{n,l} = \{Z_{l,m} \mid PC(\ell_m) \geq P_n\}.$$

We then associate a binary decision variable $x_{l,m}$ to every $Z_{l,m}$ belonging to the set $\gamma_n = \bigcup_{l>n} \gamma_{n,l}$. The role of each $x_{l,m}$ is to discriminate, on the basis of the value that the BLP solver will assign to it, whether the duration $L_{l,m}$ of the corresponding longest critical section $Z_{l,m}$ does contribute ($x_{l,m}=1$) or not ($x_{l,m}=0$) to the maximum blocking time B_n .

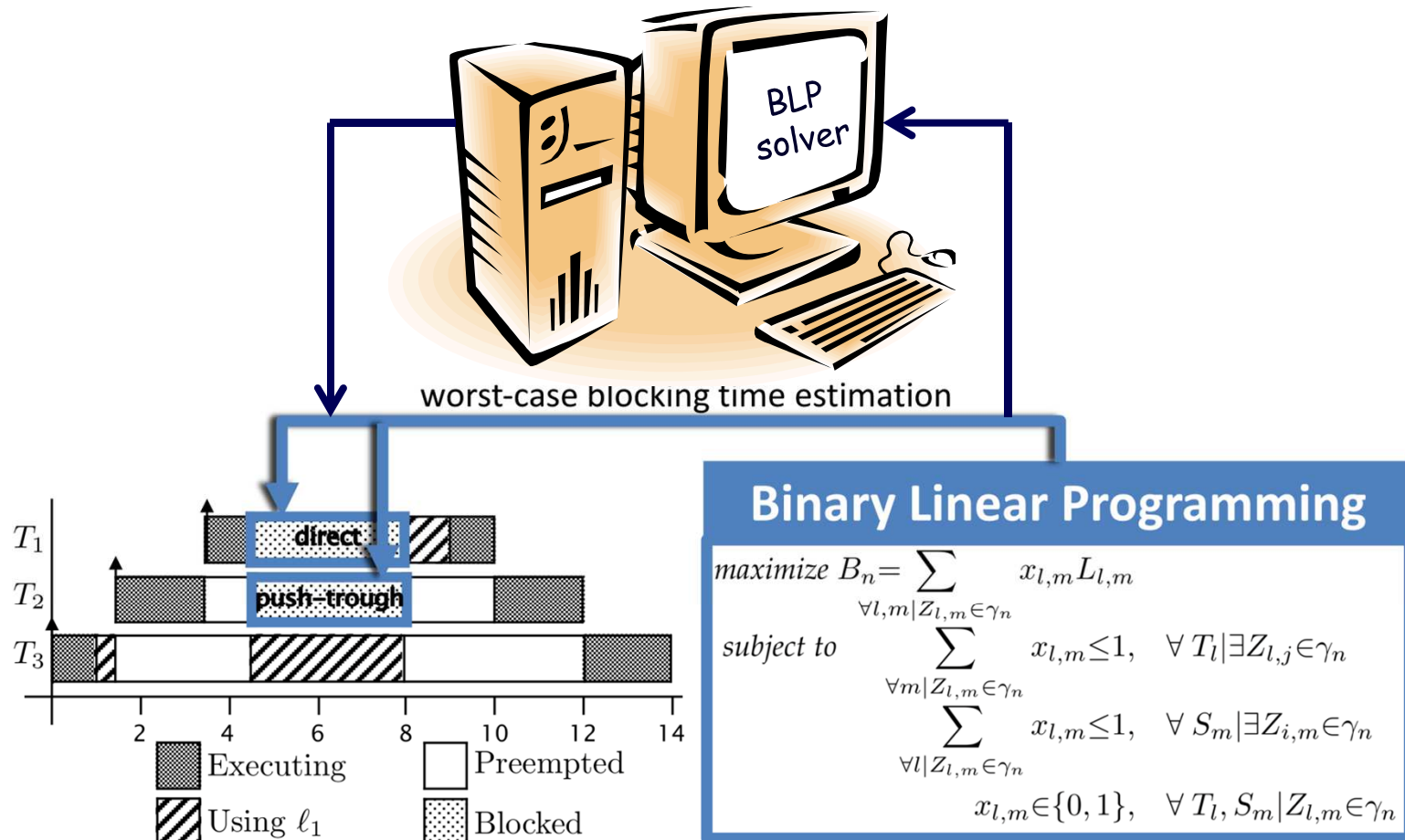
According to the PI protocol's rules, the subset of critical sections in γ_n that *most adversely* contribute to the blocking of T_n must be selected in such a way to be compliant with the two following constraints:

... IL PROTOCOLLO PI ...

C_1 : select at most one of the critical sections belonging to the same lower-priority task T_i ;

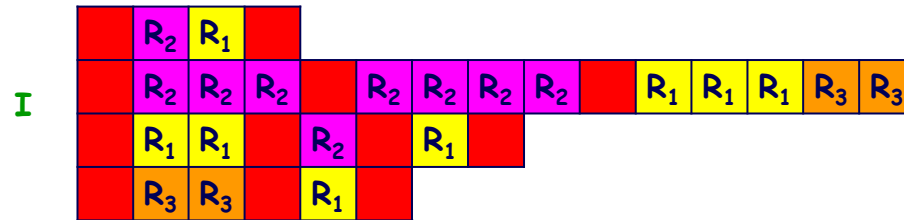
C_2 : select at most one of the critical sections guarded by the same semaphore S_m .

Therefore, the problem of computing the worst-case blocking time B_n of task T_n for all task phasings can be straightforwardly modeled through BLP as follows:



... IL PROTOCOLLO PI ...

Calcolo di B_1



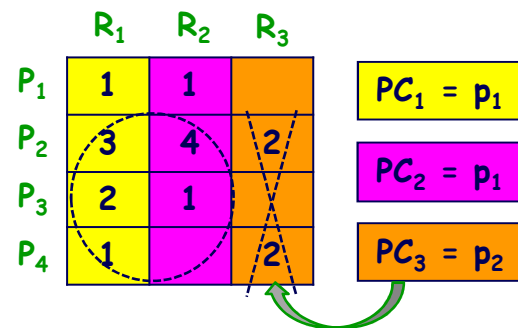
priorità

p_1 (max)

p_2

p_3

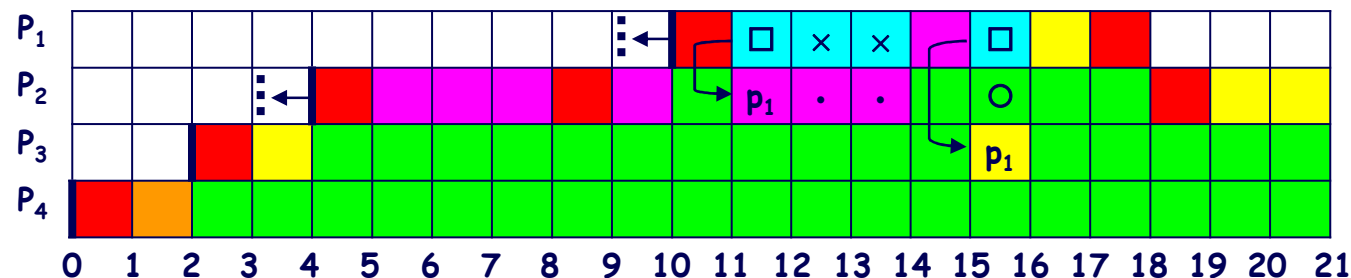
p_4 (min)



Given the complete set of the longest critical sections in lower-priority tasks that may contribute to B_1 ($\gamma_{1,2} = \{Z_{2,1}, Z_{2,2}\}$, $\gamma_{1,3} = \{Z_{3,1}, Z_{3,2}\}$, $\gamma_{1,4} = \{Z_{4,1}\} \rightarrow \gamma_1 = \{Z_{2,1}, Z_{2,2}, Z_{3,1}, Z_{3,2}, Z_{4,1}\}$), the BLP model to determine B_1 is defined as follows:

$$\begin{aligned}
 &\text{maximize } B_1 = 3x_{2,1} + 4x_{2,2} + 2x_{3,1} + x_{3,2} + x_{4,1} \\
 &\text{subject to } x_{2,1} + x_{2,2} \leq 1 \quad (i) \\
 &\quad \quad \quad x_{3,1} + x_{3,2} \leq 1 \\
 &\quad \quad \quad x_{4,1} \leq 1 \\
 &\quad \quad \quad x_{2,1} + x_{3,1} + x_{4,1} \leq 1 \quad (ii) \\
 &\quad \quad \quad x_{2,2} + x_{3,2} \leq 1 \\
 &x_{l,m} \in \{0,1\}, \forall T_l, S_m \mid Z_{l,m} \in \gamma_1
 \end{aligned}$$

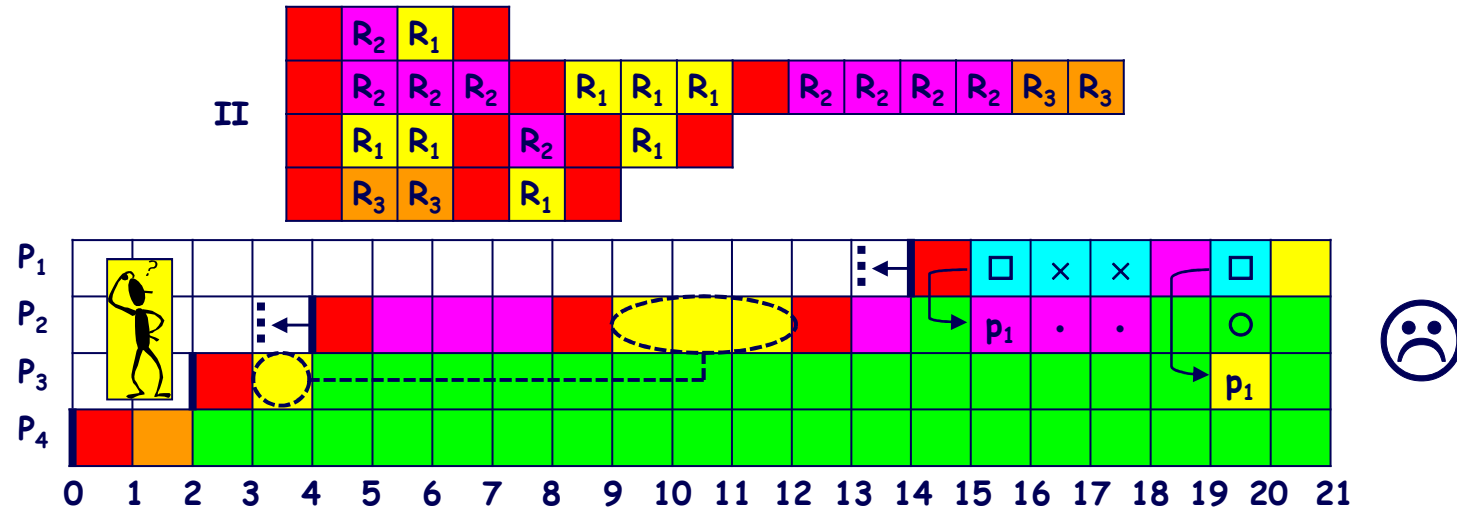
Applying one of the ILP solving methods (e.g., cutting planes, branch-and-bound, branch-and-cut, etc.), the optimal solution of this problem is found to be: $x_{2,1} = 0$, $x_{2,2} = 1$, $x_{3,1} = 1$, $x_{3,2} = 0$, $x_{4,1} = 0$. Analogously to Rajkumar's outcome, $B_1 = L_{2,2} + L_{3,1} = 6$.



... IL PROTOCOLLO PI ...

Refined Model (M_2)

In general, constraints C_1 and C_2 are not enough to precisely compute the worst-case blocking time B_n of task T_n . A third condition has to be taken into account: a task cannot be executing any critical section preceded by another critical section involving a resource that is held by a lower-priority task.



In order to formalize this condition, we need to consider in lower-priority tasks *all the critical sections - not just the longest ones - that may block T_n , as well as the order in which they are executed.*



extended notation

... IL PROTOCOLLO PI ...

$z_{n,k}$ denotes the k -th ($k=1,2,\dots$) critical section executed by task T_n ($n=1,2,\dots,N$). The semaphore guarding the resource accessed during $z_{n,k}$ is denoted by $\sigma_{n,k}$ and the duration of $z_{n,k}$ by $d_{n,k}$. $Z_n = \{z_{n,k} \mid k=1,2,\dots\}$ ($n=1,2,\dots,N$) denotes the set of the critical sections of the task T_n sorted by execution order - i.e., for any $k', k'' \mid k' < k''$, the execution of $z_{n,k'}$ precedes that of $z_{n,k''}$ ($z_{n,k'} < z_{n,k''}$).

$\Psi_{n,l}$ ($1 \leq n < l \leq N$) denotes the set of semaphores with priority ceiling greater than or equal to the priority of the task T_n which can be acquired by a lower-priority task T_l .

$\Psi_{n,*}$ denotes the set of all semaphores with a priority ceiling greater than or equal to the priority of the task T_n which can be acquired by any lower-priority task: $\Psi_{n,*} = \bigcup_{l>n} \Psi_{n,l}$.

$\Phi_{n,l,m}$ ($1 \leq n < l \leq N, S_m \in \Psi_{n,l}$) denotes the set of critical sections guarded by semaphore S_m in the lower-priority task T_l which can block the task T_n : $\Phi_{n,l,m} = \{z_{l,k} \mid \sigma_{l,k} = S_m \in \Psi_{n,l}\}$.

$\varphi_{l,m}$ denotes the critical section among those guarded by the semaphore S_m that is first executed by the task T_l : $\varphi_{l,m} = z_{l,k'} \in \Phi_{n,l,m} \mid z_{l,k'} < z_{l,k''} \forall z_{l,k''} \in \Phi_{n,l,m}, k' \neq k''$.

$\Phi_{n,l,*}$ ($1 \leq n < l \leq N$) denotes the set of critical sections of task T_l that can block the higher-priority task T_n : $\Phi_{n,l,*} = \{z_{l,k} \mid \sigma_{l,k} \in \Psi_{n,l}\} = \bigcup_{m \mid S_m \in \Psi_{n,l}} \Phi_{n,l,m}$.

$\Phi_{n,*,m}$ ($1 \leq n < N, S_m \in \Psi_{n,*}$) denotes the set of the critical sections guarded by the semaphore S_m in all the lower-priority tasks that can block T_n : $\Phi_{n,*,m} = \bigcup_{l>n} \Phi_{n,l,m}$.

$\Phi_{n,*,*}$ ($1 \leq n < N$) denotes the overall set of critical sections in the lower-priority tasks which can block T_n : $\Phi_{n,*,*} = \bigcup_{l>n} \Phi_{n,l,*} = \bigcup_{m \mid S_m \in \Psi_{n,*}} \Phi_{n,*,m}$.

$\Theta_{n,l,m}$ ($1 \leq n < l \leq N, S_m \in \Psi_{n,l}$) denotes the subset of the critical sections in $\Phi_{n,l,*}$ which are guarded by a semaphore other than S_m and executed after $\varphi_{l,m}$: $\Theta_{n,l,m} = \{z_{l,k} \in \Phi_{n,l,*} \mid \sigma_{l,k} \neq S_m, \varphi_{l,m} < z_{l,k}\}$.

... IL PROTOCOLLO PI ...

Aiming to refine the computation of the maximum blocking time B_n of a task T_n over all tasks phasings, we associate a binary decision variable $x_{l,k}$ to every critical section $z_{l,k} \in \Phi_{n,*,*}$. As in model M_1 , the role of each $x_{l,k}$ is to discriminate, on the basis of the value that the solver will assign to it, whether the duration $d_{l,k}$ of $z_{l,k}$ does contribute ($x_{l,k}=1$) or not ($x_{l,k}=0$) to the maximum blocking time B_n .

The model must be reformulated to express, besides C_1 and C_2 with the new notation, the set of constraints originating from the third condition:

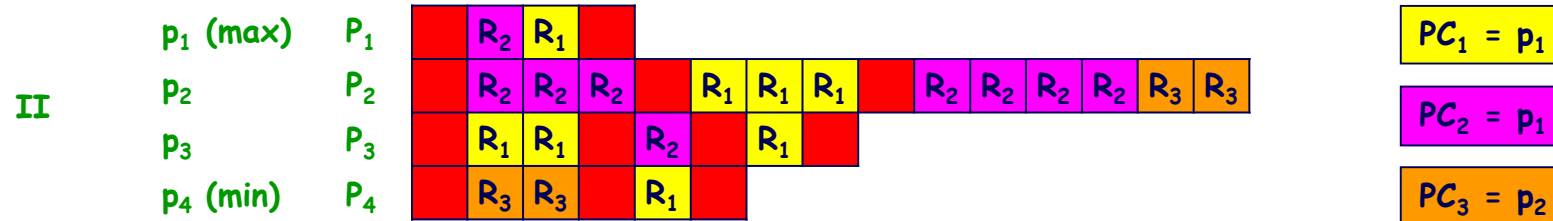
C_3 : for each pair (T_l, S_m) such that T_l can block T_n on a critical section guarded by the semaphore S_m , select either one of the critical sections following $\phi_{l,m}$, or one of those involving S_m in tasks with priority lower than T_l .

Therefore, the problem of computing the worst-case blocking time B_n of task T_n for all task phasings can be modeled through BLP as follows:

$$\begin{array}{ll}
 \text{maximize} & B_n = \sum_{z_{l,k} \in \Phi_{n,*,*}} x_{l,k} d_{l,k} \\
 \text{subject to } \forall T_l \mid l > n: & \sum_{z_{l,k} \in \Phi_{n,l,*}} x_{l,k} \leq 1, \text{ (i)} \\
 \forall S_m \in \Psi_{n,*}: & \sum_{z_{l,k} \in \Phi_{n,*,m}} x_{l,k} \leq 1, \text{ (ii)} \\
 \forall T_l, S_m \mid n < l < N \wedge S_m \in \Psi_{n,l}: & \sum_{z_{l,k} \in \Theta_{n,l,m}} x_{l,k} + \sum_{z_{v,k} \in \cup_{v>l} \Phi_{n,v,m}} x_{v,k} \leq 1, \text{ (iii)} \\
 \forall T_l \mid l > n: & x_{l,k} \in \{0,1\}.
 \end{array}$$

... IL PROTOCOLLO PI ...

Calcolo di B_1 priorità

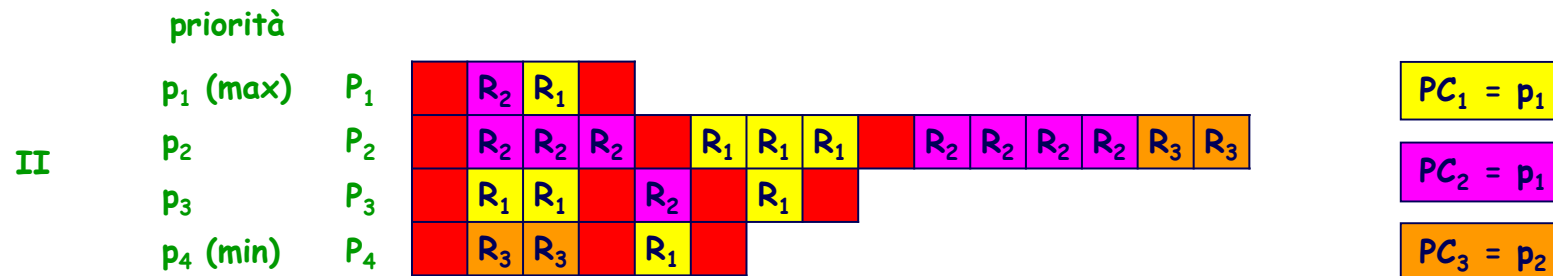


$Z_1 = \{z_{1,1}, z_{1,2}\}$	$\sigma_{1,1} = S_2 \quad \sigma_{1,2} = S_1$	$d_{1,1} = 1 \quad d_{1,2} = 1$
$Z_2 = \{z_{2,1}, z_{2,2}, z_{2,3}, z_{2,4}\}$	$\sigma_{2,1} = S_2 \quad \sigma_{2,2} = S_1 \quad \sigma_{2,3} = S_2 \quad \sigma_{2,4} = S_3$	$d_{2,1} = 3 \quad d_{2,2} = 3 \quad d_{2,3} = 4 \quad d_{2,4} = 2$
$Z_3 = \{z_{3,1}, z_{3,2}, z_{3,3}\}$	$\sigma_{3,1} = S_1 \quad \sigma_{3,2} = S_2 \quad \sigma_{3,3} = S_1$	$d_{3,1} = 2 \quad d_{3,2} = 1 \quad d_{3,3} = 1$
$Z_4 = \{z_{4,1}, z_{4,2}\}$	$\sigma_{4,1} = S_3 \quad \sigma_{4,2} = S_1$	$d_{4,1} = 2 \quad d_{4,2} = 1$

In order to compute the worst-case blocking time of the first task, all the critical sections in the lower-priority tasks should be taken into account except those guarded by the semaphore S_3 , because it has a priority ceiling lower than the priority of the selected task. Hence:

$\Psi_{1,2} = \{S_1, S_2\}$	$\Psi_{1,3} = \{S_1, S_2\}$	$\Psi_{1,4} = \{S_1\}$
$\Psi_{1,*} = \{S_1, S_2\}$		
$\Phi_{1,2,1} = \{z_{2,2}\}$	$\Phi_{1,2,2} = \{z_{2,1}, z_{2,3}\}$	
$\Phi_{1,3,1} = \{z_{3,1}, z_{3,3}\}$	$\Phi_{1,3,2} = \{z_{3,2}\}$	
$\Phi_{1,4,1} = \{z_{4,2}\}$		
$\Phi_{1,2,*} = \{z_{2,1}, z_{2,2}, z_{2,3}\}$	$\Phi_{1,3,*} = \{z_{3,1}, z_{3,2}, z_{3,3}\}$	$\Phi_{1,4,*} = \{z_{4,2}\}$
$\Phi_{1,*1} = \{z_{2,2}, z_{3,1}, z_{3,3}, z_{4,2}\}$	$\Phi_{1,*2} = \{z_{2,1}, z_{2,3}, z_{3,2}\}$	
$\Phi_{1,*,*} = \{z_{2,1}, z_{2,2}, z_{2,3}, z_{3,1}, z_{3,2}, z_{3,3}, z_{4,2}\}$		

... IL PROTOCOLLO PI ...



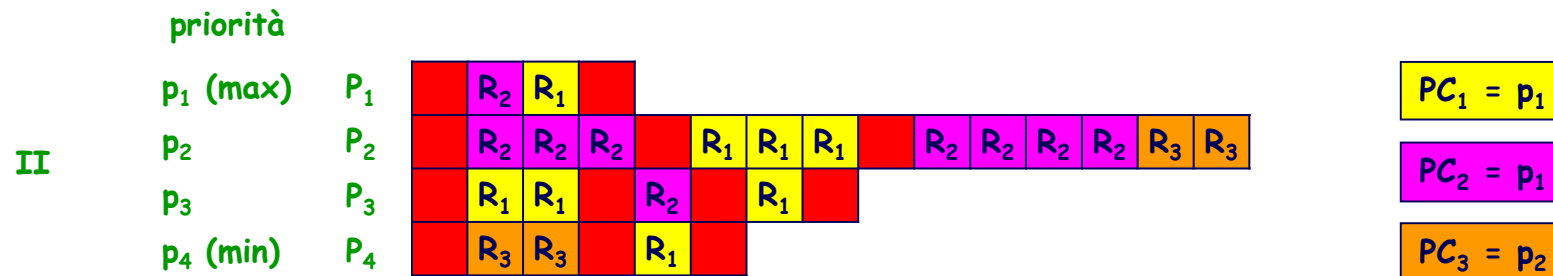
A binary decision variable is first associated to each critical section in $\Phi_{1,*,*} = \{z_{2,1}, z_{2,2}, z_{2,3}, z_{3,1}, z_{3,2}, z_{3,3}, z_{4,2}\}$; $x_{2,1}, x_{2,2}, x_{2,3}, x_{3,1}, x_{3,2}, x_{3,3}, x_{4,2}$. Then, in order to write the set of constraints (iii), the relevant sets of critical sections for each lower-priority task T_l ($1 < l < N = 4$) and each semaphore $S_m \in \Psi_{1,l}$ are identified:

$$\begin{aligned}
 l = 2, \Psi_{1,2} = \{S_1, S_2\}: & \quad \varphi_{2,1} = z_{2,2} \quad \Theta_{1,2,1} = \{z_{2,3}\} \quad \bigcup_{v>2} \Phi_{1,v,1} = \{z_{3,1}, z_{3,3}, z_{4,2}\} \\
 & \quad \varphi_{2,2} = z_{2,1} \quad \Theta_{1,2,2} = \{z_{2,2}\} \quad \bigcup_{v>2} \Phi_{1,v,2} = \{z_{3,2}\} \\
 l = 3, \Psi_{1,3} = \{S_1, S_2\}: & \quad \varphi_{3,1} = z_{3,1} \quad \Theta_{1,3,1} = \{z_{3,2}\} \quad \bigcup_{v>3} \Phi_{1,v,1} = \{z_{4,2}\} \\
 & \quad \varphi_{3,2} = z_{3,2} \quad \Theta_{1,3,2} = \{z_{3,3}\} \quad \bigcup_{v>3} \Phi_{1,v,2} = \emptyset
 \end{aligned}$$

The BLP model to determine B1 is therefore defined as follows:

$$\begin{aligned}
 & \text{maximize } B_1 = 3 x_{2,1} + 3 x_{2,2} + 4 x_{2,3} + 2 x_{3,1} + x_{3,2} + x_{3,3} + x_{4,2} \\
 & \text{subject to} \quad x_{2,1} + x_{2,2} + x_{2,3} \leq 1 \quad (i) \\
 & \quad \quad \quad x_{3,1} + x_{3,2} + x_{3,3} \leq 1 \\
 & \quad \quad \quad x_{4,2} \leq 1 \\
 & \quad \quad \quad x_{2,2} + x_{3,1} + x_{3,3} + x_{4,2} \leq 1 \quad (ii) \\
 & \quad \quad \quad x_{2,1} + x_{2,3} + x_{3,2} \leq 1 \\
 & \quad \quad \quad x_{2,3} + x_{3,1} + x_{3,3} + x_{4,2} \leq 1 \quad (iii) \\
 & \quad \quad \quad x_{2,2} + x_{3,2} \leq 1 \\
 & \quad \quad \quad x_{3,2} + x_{4,2} \leq 1 \\
 & \quad \quad \quad x_{3,3} \leq 1 \\
 & \quad x_{l,k} \in \{0,1\}, \forall 1 < l \leq 4
 \end{aligned}$$

... IL PROTOCOLLO PI ...



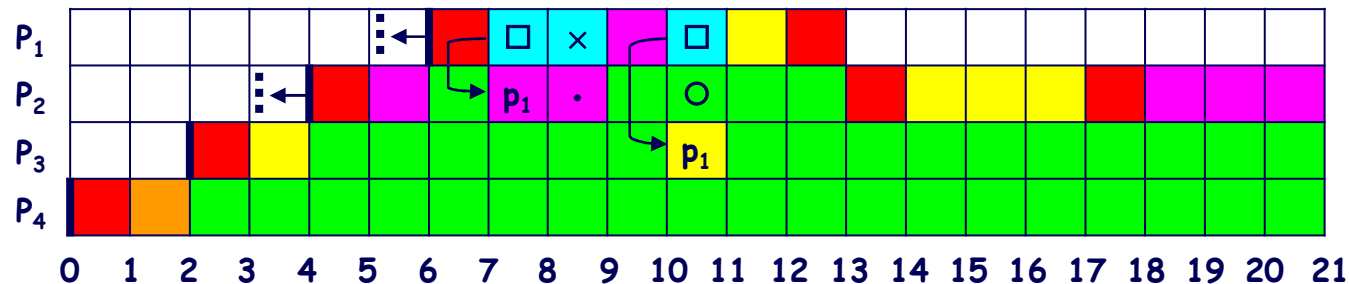
Applying one of the ILP solving methods, the optimal solution is found to be:

$$x_{2,1} = 1, x_{2,2} = 0, x_{2,3} = 0, x_{3,1} = 1, x_{3,2} = 0, x_{3,3} = 0, x_{4,2} = 0.$$

Therefore:

$$B_1 = d_{2,1} + d_{3,1} = 5.$$

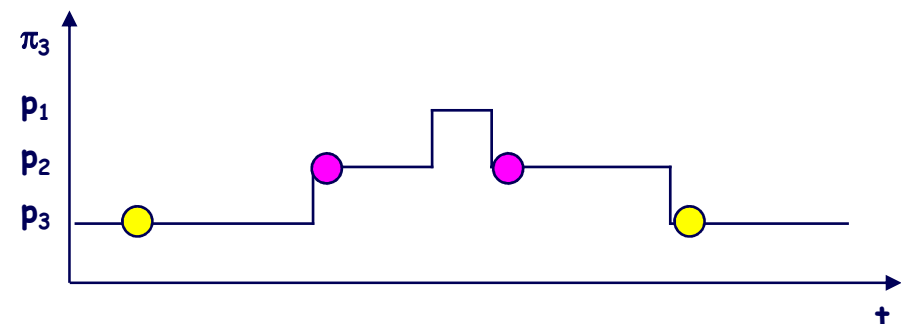
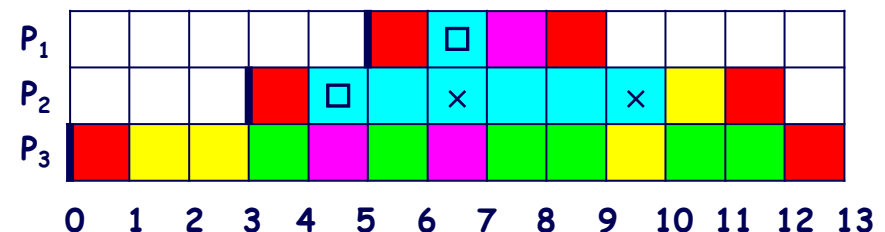
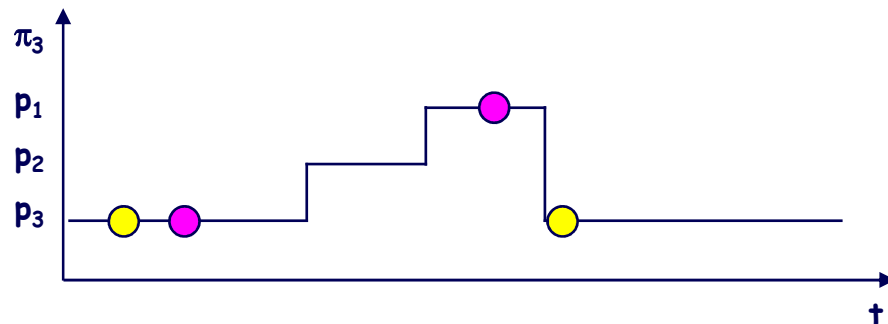
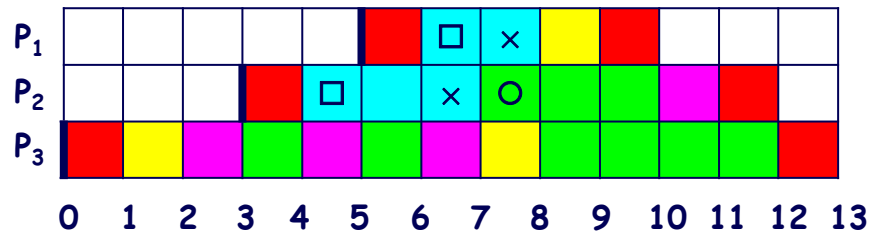
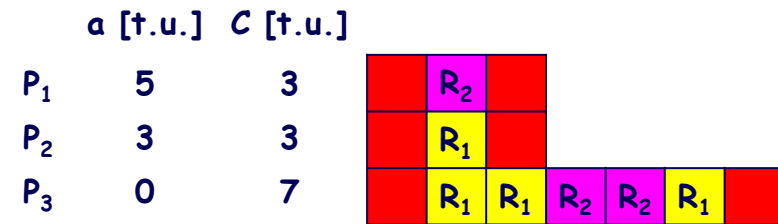
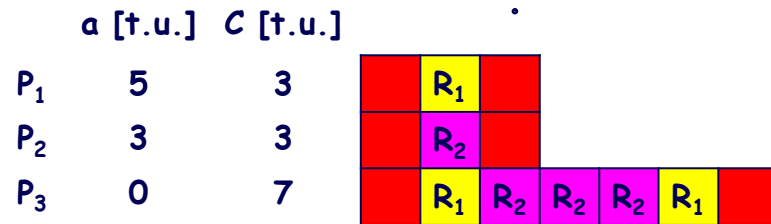
The figure below shows the blocking chain that causes B_1 .



... IL PROTOCOLLO PI ...

(2) Sezioni critiche annidate

La priorità che un processo assume allorché rilascia una risorsa
 non necessariamente coincide con
 la priorità che il processo aveva all'atto dell'acquisizione della risorsa.



... IL PROTOCOLLO PI ...

L'ereditarietà della priorità è transitiva.

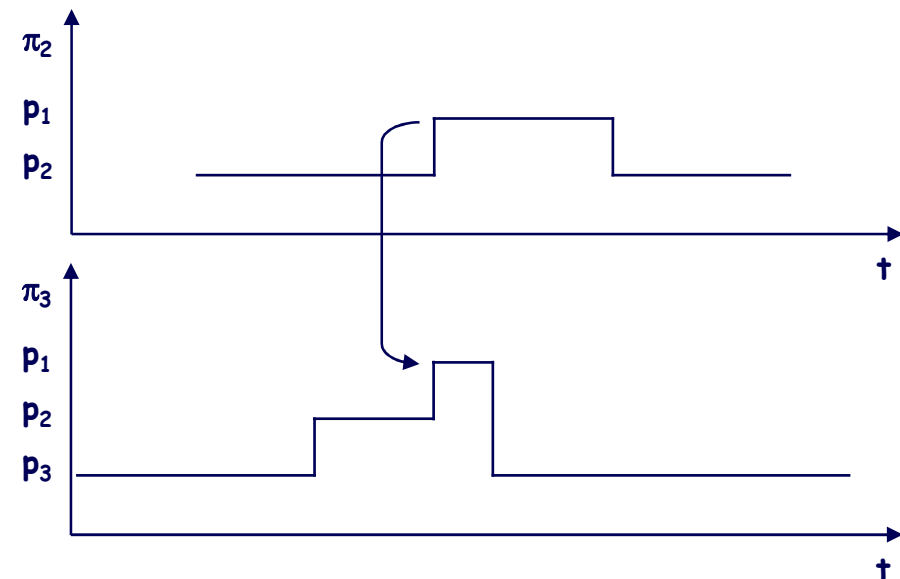
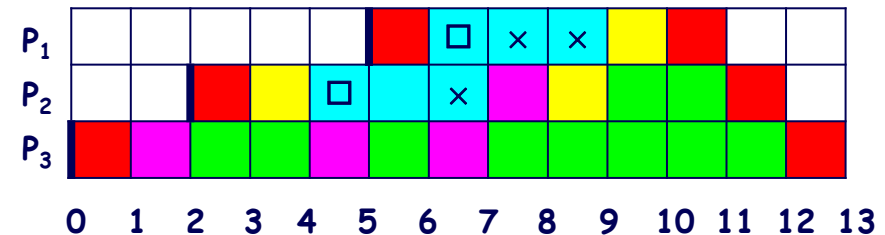
	a [t.u.]	C [t.u.]						
P ₁	5	3	<table><tr><td></td><td>R₁</td><td></td></tr></table>		R ₁			
	R ₁							
P ₂	2	5	<table><tr><td></td><td>R₁</td><td>R₂</td><td>R₁</td><td></td></tr></table>		R ₁	R ₂	R ₁	
	R ₁	R ₂	R ₁					
P ₃	0	5	<table><tr><td></td><td>R₂</td><td>R₂</td><td>R₂</td><td></td></tr></table>		R ₂	R ₂	R ₂	
	R ₂	R ₂	R ₂					

	R ₁	R ₂
P ₁	1	
P ₂	3	1
P ₃		3

$$PC_1 = p_1$$

$$PC_2 = p_2$$

“transitive blocking”:
un processo P_i di priorità p_i può subire un blocco da parte di un processo P_j di priorità p_j < p_i anche se P_j accede soltanto a risorse R_k con PC_k < p_i



Il processo P₁ subisce il massimo tempo di blocco B₁ = 6 t.u. se:

$$a_3 = \forall, a_2 = a_3 + 1 + \varepsilon, a_1 = a_2 + 1 + \varepsilon, \text{ con } \varepsilon \rightarrow 0.$$

... IL PROTOCOLLO PI ...

Un processo P_i di priorità p_i può essere bloccato da un processo P_j di priorità $p_j < p_i$ se P_j condivide una risorsa con un processo che ha o può ereditare una priorità $\geq p_i$.

Un semaforo S_k può bloccare un processo P_i di priorità p_i se ad S_k accedono un processo di priorità $< p_i$ ed un processo che ha o può ereditare una priorità $\geq p_i$.

Un processo P_i può essere bloccato al più $n_i = \min(l_i, r_i)$ volte, essendo l_i e r_i il numero, rispettivamente, di processi e di distinti semafori che possono bloccare P_i .

Un "upper bound" del massimo tempo di blocco B_i di ciascun processo P_i ($i = 1, \dots, N$) può essere identificato con il metodo di Rajkumar.

Con riferimento all'applicazione A_6'

priorità

p_1 (max)

p_2

p_3

p_4

p_5

p_6 (min)

P_1		R_2					
P_2		R_1	R_4	R_1			
P_3		R_1	R_1	R_1	R_4	R_4	
P_4		R_2	R_3	R_2			
P_5		R_4	R_2	R_4	R_4		
P_6		R_3	R_3				

il massimo tempo di blocco ([t.u.])
di ciascun processo risulta:

$B_1 = 5$

$B_2 = 12$

$B_3 = 9$

$B_4 = 6$

$B_5 = 2$

$B_6 = 0$

... IL PROTOCOLLO PI ...

Applicazione A_6' : calcolo di B_2

priorità

p_1 (max)

p_2

p_3

p_4

p5

p_6 (min)

P_1

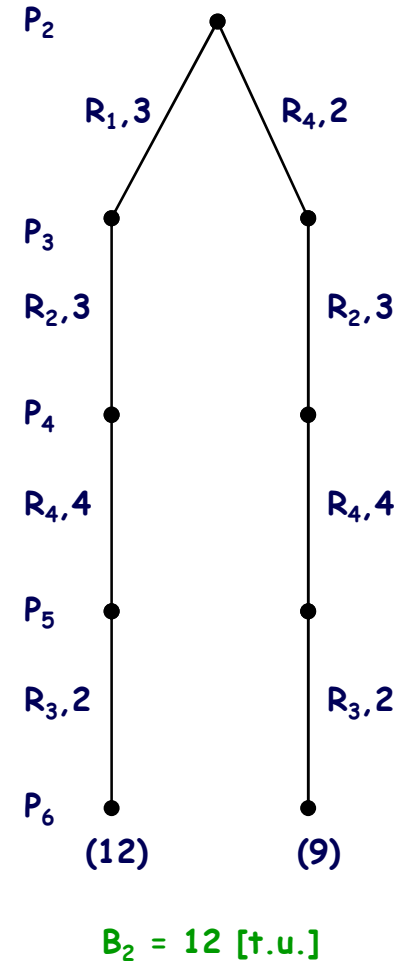
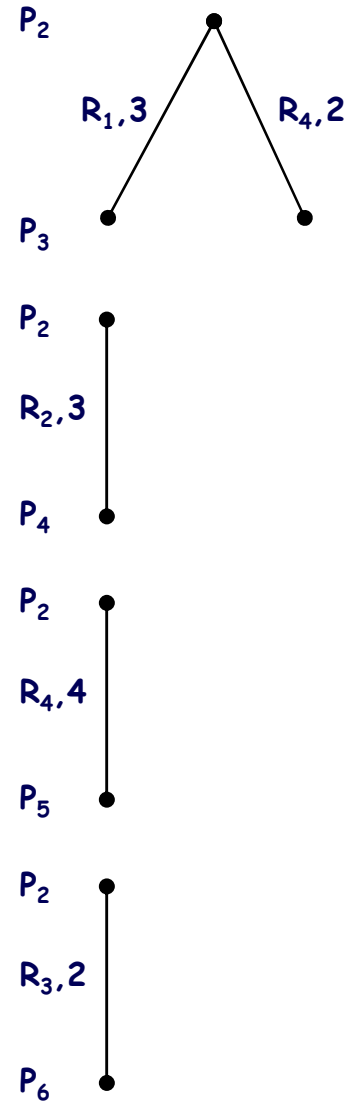
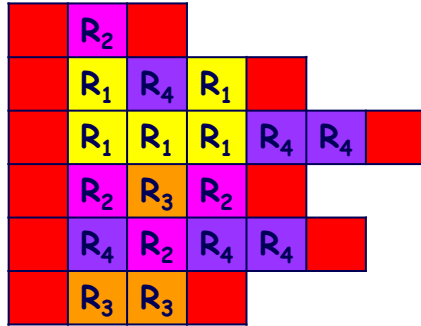
P_2

P_3

P_4

P_5

P_6



... IL PROTOCOLLO PI ...

Applicazione A_6' : calcolo di B_2 tramite costruzione dell'albero di ricerca duale

priorità

p_1 (max)

p_2

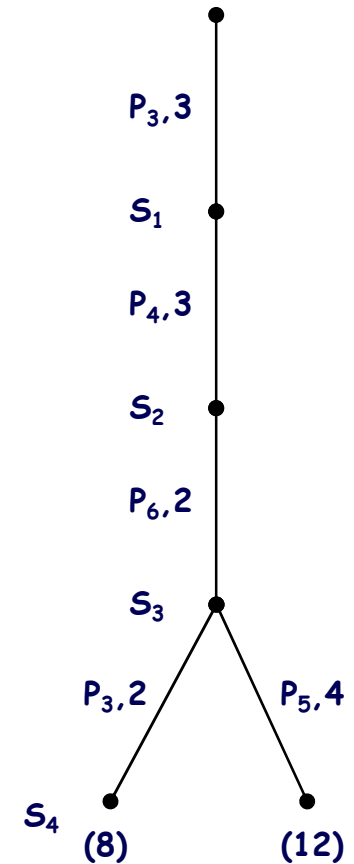
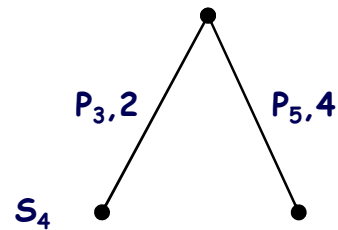
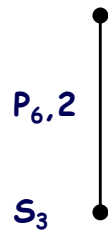
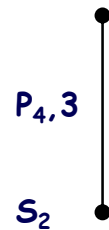
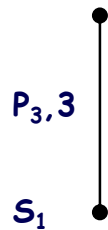
p_3

p_4

p_5

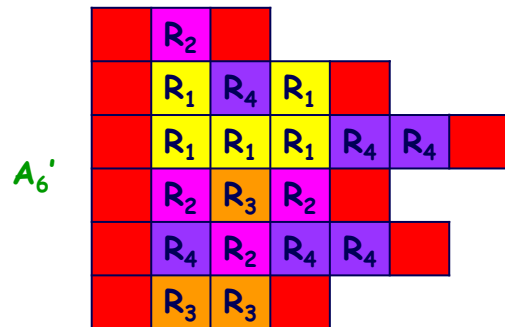
p_6 (min)

P_1		R_2					
P_2		R_1	R_4	R_1			
P_3		R_1	R_1	R_1	R_4	R_4	
P_4		R_2	R_3	R_2			
P_5		R_4	R_2	R_4	R_4		
P_6		R_3	R_3				



$B_2 = 12$ [t.u.]

... IL PROTOCOLLO PI



$$B_1 = 5$$

$$B_2 = 12$$

$$B_3 = 9$$

$$B_4 = 6$$

$$B_5 = 2$$

$$B_6 = 0$$

I processi subiscono il massimo tempo di blocco se:

$$a_6 = \forall,$$

$$a_5 = a_6 + 1 + \varepsilon,$$

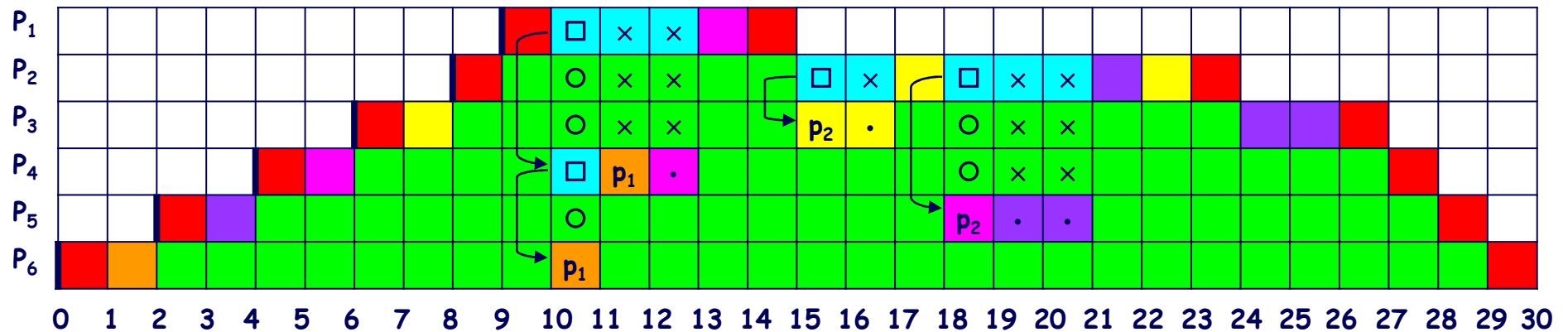
$$a_4 = a_5 + 1 + \varepsilon,$$

$$a_3 = a_4 + 1 + \varepsilon,$$

$$a_2 = a_3 + 1 + \varepsilon,$$

$$a_1 > a_2, a_1 < a_2 + 6 - 2\varepsilon,$$

con $\varepsilon \rightarrow 0$.



IL PROTOCOLLO "PRIORITY CEILING" (PC) [SHA, RAJKUMAR, LEHOCZKY (90)] ...

Regole di schedulazione dei processi, di allocazione delle risorse, di gestione delle priorità dei processi allorché detengono risorse:

La schedulazione dei processi (pronti) è operata in base alle relative priorità correnti. La priorità corrente π_i di un processo P_i ($\forall i$) coincide con:

- la corrispondente priorità nominale p_i nel caso in cui P_i non detenga alcuna risorsa richiesta da processi di priorità superiore;
- la più alta fra le priorità correnti dei processi da esso bloccati in caso contrario.

L'accesso ad una risorsa è negato se la priorità corrente del processo che ne fa richiesta non è maggiore del tetto di priorità del sistema

$$\Pi\Gamma_s = \max_k \{ PC_k \mid R_k \text{ é in uso} \}$$

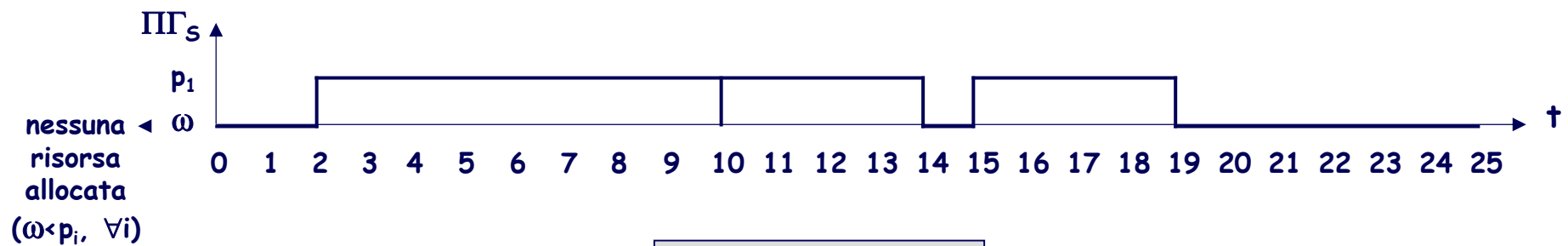
coincidente con il più elevato tetto di priorità delle risorse al momento allocate, a meno che tale processo detenga già la risorsa (le risorse) il cui tetto di priorità coincide con $\Pi\Gamma_s$.

Se ad un processo è negato l'accesso ad una risorsa, la sua priorità corrente viene ereditata dal processo che ne ha causato il blocco.

... IL PROTOCOLLO PC ...

Implica la conoscenza a priori delle risorse utilizzate dai processi
(onde poterne determinare il tetto di priorità).

		priorità	a [t.u.]	d [t.u.]	C [t.u.]		
A ₁	P ₁	p ₁ (max)	6	15	7	<div><div></div><div></div><div>R₁</div><div>R₁</div><div>R₁</div><div>R₁</div><div></div></div>	<div>PC₁ = p₁</div>
	P ₂	p ₂	3	20	7	<div><div></div><div></div><div>R₁</div><div>R₁</div><div>R₁</div><div>R₁</div><div></div></div>	
	P ₃	p ₃ (min)	0	25	7	<div><div></div><div></div><div>R₁</div><div>R₁</div><div>R₁</div><div>R₁</div><div></div></div>	

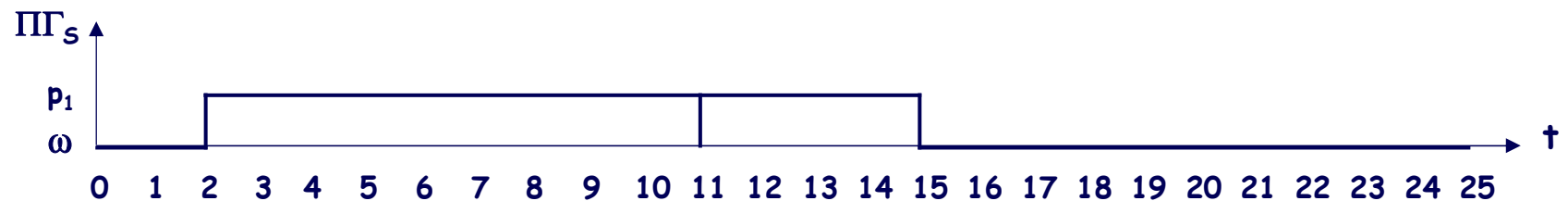


∇ Ceiling Blocking (CB)

... IL PROTOCOLLO PC ...

Previene inversioni di priorità incontrollate
(grazie al meccanismo della ereditarietà della priorità).

		priorità	a [t.u.]	d [t.u.]	C [t.u.]		
A ₂	P ₁	p ₁ (max)	6	15	7		<div>PC₁ = p₁</div>
	P ₂	p ₂	3	20	7		
	P ₃	p ₃ (min)	0	25	7		

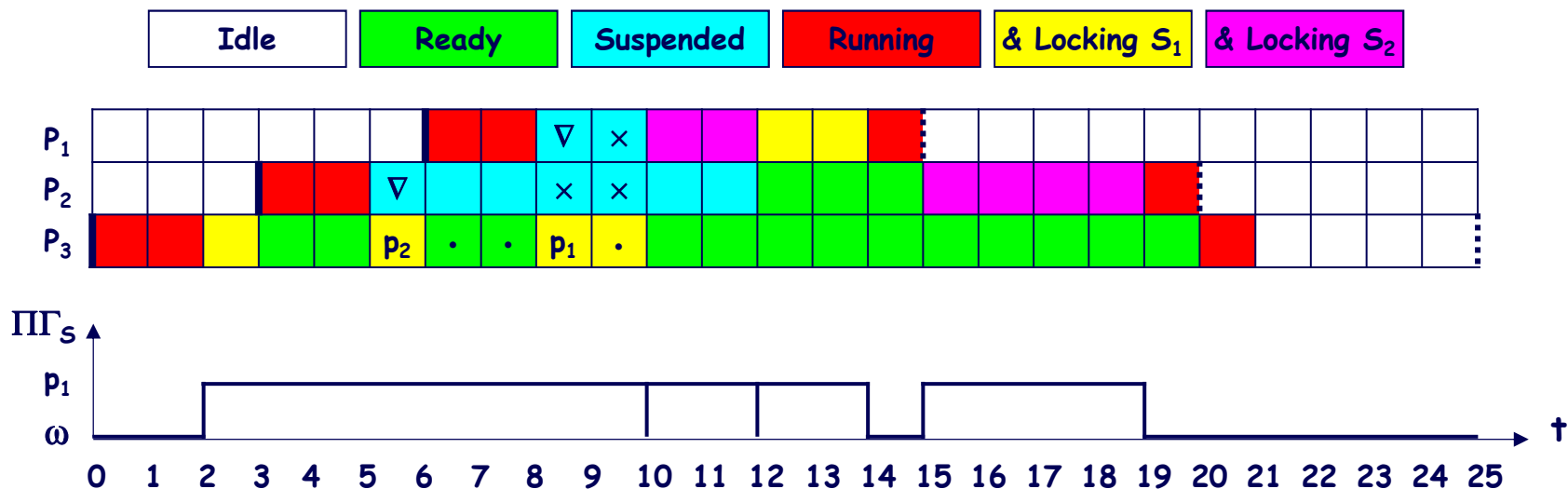


... IL PROTOCOLLO PC ...

Previene la concatenazione di blocchi

(un processo P_i può essere bloccato solo all'atto dell'accesso alla prima risorsa da esso utilizzata, e ciò se contestualmente risulta $p_i \leq \Pi\Gamma_S$. Ottenuto l'accesso allorché $p_i > \Pi\Gamma_S$, P_i può completare l'esecuzione senza subire ulteriori blocchi, poiché tutte le risorse di cui necessita, o possono essere in seguito richieste da processi di priorità superiore, sono libere (processi di priorità inferiore, anche se detengono risorse, non possono ereditare una priorità $> p_i$)).

		priorità	a [t.u.]	d [t.u.]	C [t.u.]		
A ₃	P ₁	p ₁ (max)	6	15	7		PC ₁ = p ₁
	P ₂	p ₂	3	20	7		PC ₂ = p ₁
	P ₃	p ₃ (min)	0	25	7		



... IL PROTOCOLLO PC ...

Previene situazioni di deadlock.

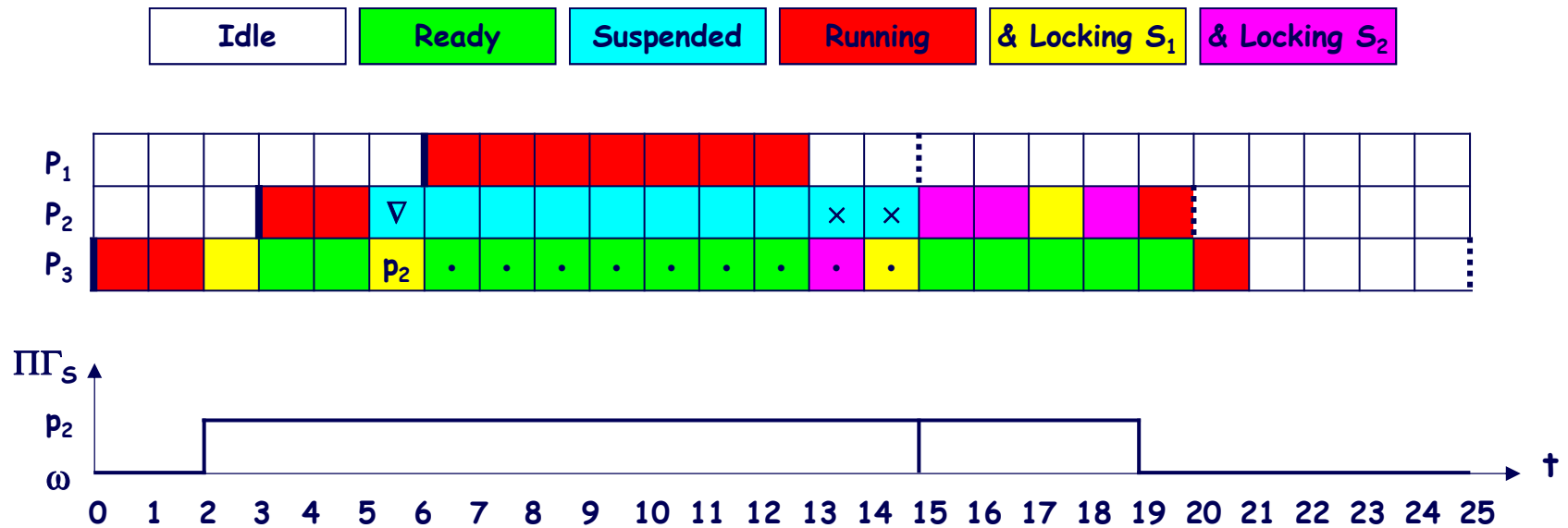
		priorità	a [t.u.]	d [t.u.]	C [t.u.]		
A_4	P_1	p_1 (max)	6	15	7		
	P_2	p_2	3	20	7		
	P_3	p_3 (min)	0	25	7		

		R_2	R_2	R_1	R_2	
		R_1	R_1	R_2	R_1	

!?

$PC_1 = p_2$

$PC_2 = p_2$



... IL PROTOCOLLO PC ...

Il massimo tempo di blocco B_i di un processo P_i di priorità p_i è uguale al tempo di esecuzione Z_{jk} della più lunga sezione critica relativa a qualunque risorsa R_k condivisa da un processo di priorità $p_j < p_i$ e da un processo di priorità $\geq p_i$:

$$B_i = \max_{j,k} \{ Z_{jk} \mid p_j < p_i, PC_k \geq p_i \} \quad \forall i$$

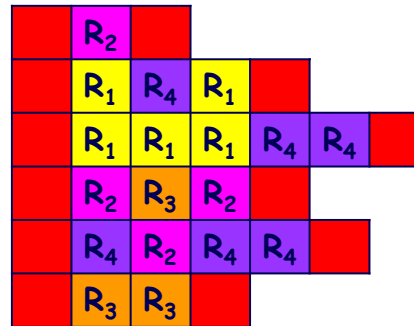
Con riferimento all'applicazione A_5 , il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:

	priorità		R_1	R_2	
$PC_1 = p_1$	p_1 (max)	P_1	4	2	$B_1 = 4 = \max \{2, 0, 4\}$
	p_2	P_2		2	$B_2 = 4 = \max \{0, 4\}$
$PC_2 = p_1$	p_3	P_3			$B_3 = 4$
	p_4 (min)	P_4	4		$B_4 = 0$

... IL PROTOCOLLO PC ...

Applicazione A_6

- $P_1 (9, 30, 3, 30; [R_2;1])$
- $P_2 (8, 40, 5, 40; [R_1;1] [R_4;1] [R_1;1])$
- $P_3 (6, 50, 7, 50; [R_1;3] [R_4;2])$
- $P_4 (4, 60, 5, 60; [R_2;1] [R_3;1] [R_2;1])$
- $P_5 (2, 70, 6, 70; [R_4;1] [R_2;1] [R_4;2])$
- $P_6 (0, 80, 4, 80; [R_3;2])$



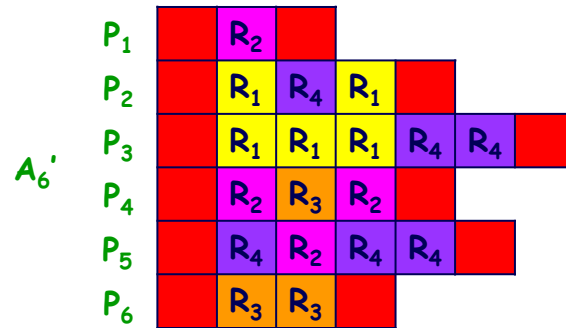
Applicazione A_6'

- $P_1 (9, 30, 3, 30; [R_2;1])$
- $P_2 (8, 40, 5, 40; [R_1;3] [R_4;1])$
- $P_3 (6, 50, 7, 50; [R_1;3] [R_4;2])$
- $P_4 (4, 60, 5, 60; [R_2;3] [R_3;1])$
- $P_5 (2, 70, 6, 70; [R_4;4] [R_2;1])$
- $P_6 (0, 80, 4, 80; [R_3;2])$

priorità			R ₁	R ₂	R ₃	R ₄	R ₁	R ₂	R ₃	R ₄	
PC ₁ = p ₂	p ₁ (max)	P ₁		1				1			B ₁ = 1 / 3
PC ₂ = p ₁	p ₂	P ₂	1			1	3			1	B ₂ = 3 / 4
	p ₃	P ₃	3			2	3			2	B ₃ = 2 / 4
PC ₃ = p ₄	p ₄	P ₄		1	1			3	1		B ₄ = 2 / 4
	p ₅	P ₅		1		2		1		4	B ₅ = 2 / 2
PC ₄ = p ₂	p ₆ (min)	P ₆			2				2		B ₆ = 0 / 0
			A ₆				A ₆ '				A ₆ / A ₆ '

Il massimo tempo di blocco di P_1, P_2, P_3, P_4 è sensibilmente inferiore rispetto a quello derivante dall'applicazione del protocollo PI ($B_1 = 1/5, B_2 = 6/12, B_3 = 3/9, B_4 = 4/6$).

... IL PROTOCOLLO PC



$PC_1 = p_2$

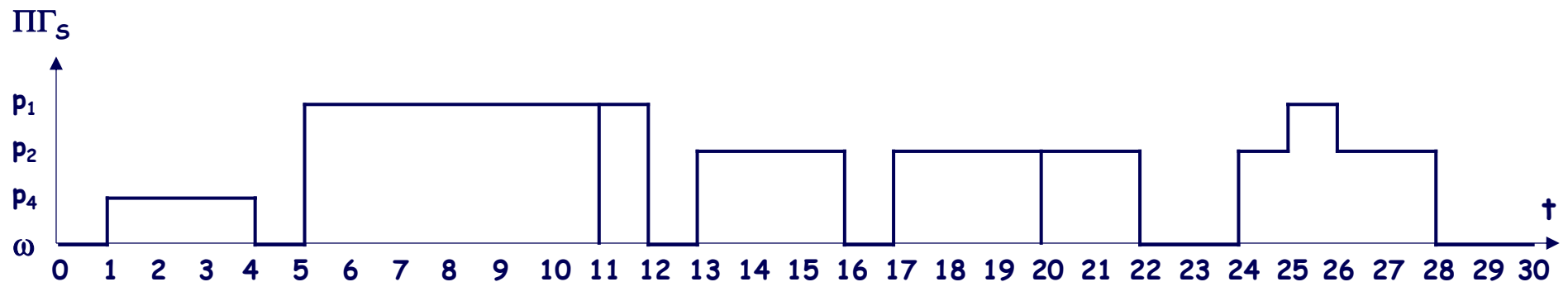
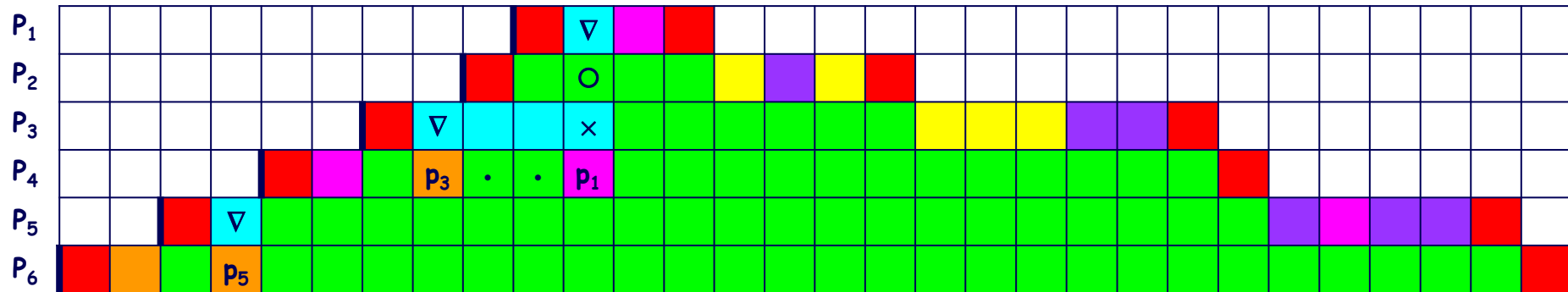
$PC_2 = p_1$

$PC_3 = p_4$

$PC_4 = p_2$

Idle	Ready	Suspended	Running
& Locking S_1	& Locking S_2	& Locking S_3	& Locking S_4

Il tempo di risposta di P_1, P_2, P_3, P_4 è sensibilmente inferiore rispetto a quello derivante dall'applicazione del protocollo PI (15 → 13, 24 → 17, 27 → 23, 28 → 24).



IL PROTOCOLLO "IMMEDIATE (STACK-BASED) PRIORITY CEILING" (IPC/SBPC) [BAKER (91)]

Regole di schedulazione dei processi, di allocazione delle risorse, di gestione delle priorità dei processi allorché detengono risorse:

La schedulazione dei processi (pronti) è operata in base alle relative priorità correnti. La priorità corrente π_i di un processo P_i ($\forall i$) coincide con:

- la corrispondente priorità nominale p_i nel caso in cui P_i non detenga alcuna risorsa;
- il massimo tetto di priorità delle risorse in suo possesso in caso contrario.

Processi con la stessa priorità corrente sono schedulati secondo la politica FIFO.

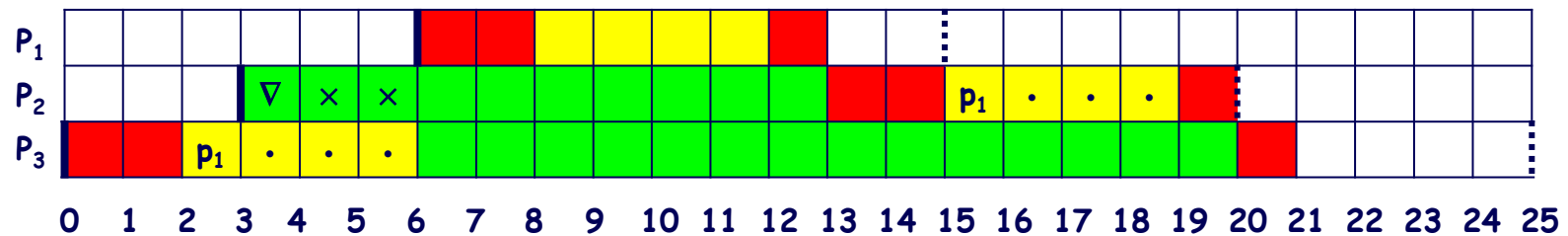
Una risorsa viene sempre allocata al processo che ne fa richiesta.

Un processo P_i di priorità nominale p_i può all'attivazione causare la preemption del processo P_j in esecuzione soltanto se $p_i > \pi_j$, ovvero se contestualmente sono libere sia tutte le risorse di cui P_i necessita, sia tutte quelle che potranno essere in seguito richieste da un qualunque processo P_k di priorità nominale $p_k > p_i$.

... IL PROTOCOLLO IPC ...

Implica la conoscenza a priori delle risorse utilizzate dai processi
(onde poterne determinare il tetto di priorità).

		priorità	a [t.u.]	d [t.u.]	C [t.u.]		
A_1	P_1	p_1 (max)	6	15	7		$PC_1 = p_1$
	P_2	p_2	3	20	7		
	P_3	p_3 (min)	0	25	7		



Idle Ready Running & Locking S_1

... IL PROTOCOLLO IPC ...

Previene inversioni di priorità incontrollate.

		priorità	a [t.u.]	d [t.u.]	C [t.u.]
A_2	P_1	p_1 (max)	6	15	7
	P_2	p_2	3	20	7
	P_3	p_3 (min)	0	25	7

$PC_1 = p_1$

Legend: Idle (white), Ready (green), Running (red), & Locking S_1 (yellow)

... IL PROTOCOLLO IPC ...

Previene la concatenazione di blocchi

(un processo può essere bloccato al più una volta, prima che abbia inizio la sua esecuzione).

Process Information:

	priorità	a [t.u.]	d [t.u.]	C [t.u.]
P_1	p_1 (max)	6	15	7
P_2	p_2	3	20	7
P_3	p_3 (min)	0	25	7

Process States and Locks:

- $PC_1 = p_1$ (Yellow box)
- $PC_2 = p_1$ (Pink box)

Timeline (0 to 25 t.u.):

- Process 1 (P_1):** Ready (green) from t.u. 6 to 12, Running (red) from t.u. 12 to 15, Idle (white) from t.u. 15 to 25.
- Process 2 (P_2):** Ready (green) from t.u. 3 to 12, Running (red) from t.u. 12 to 15, Idle (white) from t.u. 15 to 25.
- Process 3 (P_3):** Ready (green) from t.u. 0 to 12, Running (red) from t.u. 12 to 15, Idle (white) from t.u. 15 to 25.

Locks:

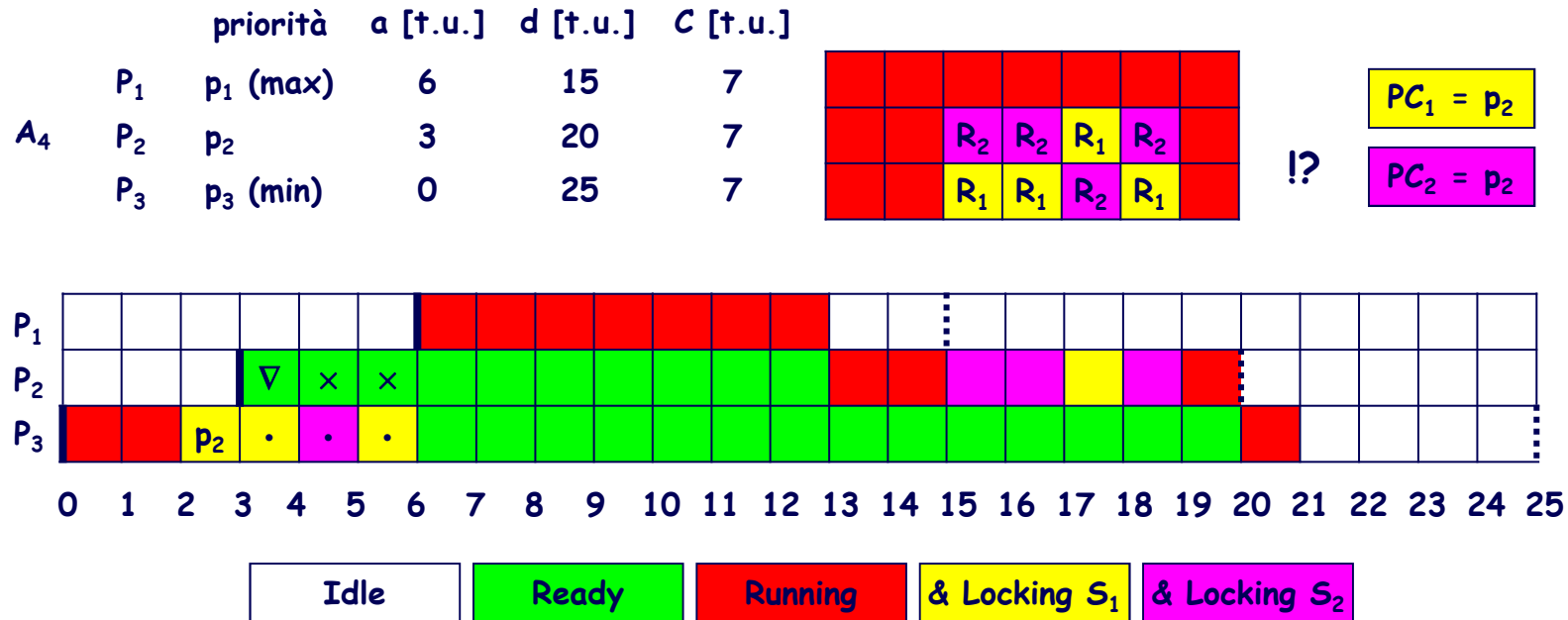
- S_1 (Yellow):** Locked from t.u. 12 to 15.
- S_2 (Pink):** Locked from t.u. 15 to 20.

Come nel caso del protocollo PC, il massimo tempo di blocco B_i di un processo P_i di priorità p_i è uguale al tempo di esecuzione Z_{jk} della più lunga sezione critica relativa a qualunque risorsa R_k condivisa da un processo di priorità $p_j < p_i$ e da un processo di priorità $\geq p_i$:

$$B_i = \max_{j,k} \{Z_{jk} \mid p_j < p_i, PC_k \geq p_i\} \quad \forall i$$

... IL PROTOCOLLO IPC ...

Previene situazioni di deadlock.



... IL PROTOCOLLO IPC ...

A_6' {

 $P_1 (9, 30, 3, 30; [R_2;1])$

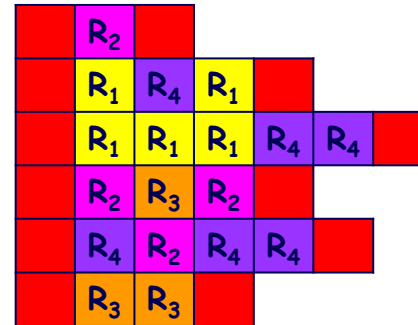
 $P_2 (8, 40, 5, 40; [R_1;3 [R_4;1]])$

 $P_3 (6, 50, 7, 50; [R_1;3] [R_4;2])$

 $P_4 (4, 60, 5, 60; [R_2;3 [R_3;1]])$

 $P_5 (2, 70, 6, 70; [R_4;4 [R_2;1]])$

 $P_6 (0, 80, 4, 80; [R_3;2])$



priorità

p_1 (max)

p_2

p_3

p_4

p_5

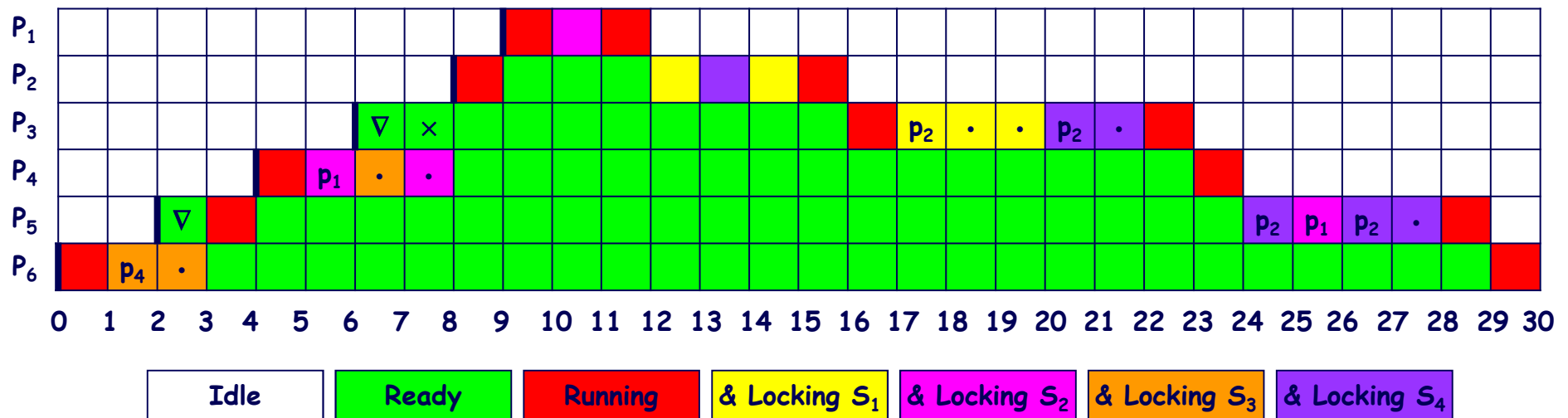
p_6 (min)

$PC_1 = p_2$

$PC_2 = p_1$

$PC_3 = p_4$

$PC_4 = p_2$



Il tempo di risposta di P_1 e P_2 migliora rispetto all'applicazione del protocollo PC ($13 \rightarrow 12, 17 \rightarrow 16$).

... IL PROTOCOLLO IPC ...

Una modalità alternativa di implementazione del protocollo

Regole di gestione del tetto di priorità del sistema, di schedulazione dei processi e di allocazione delle risorse:

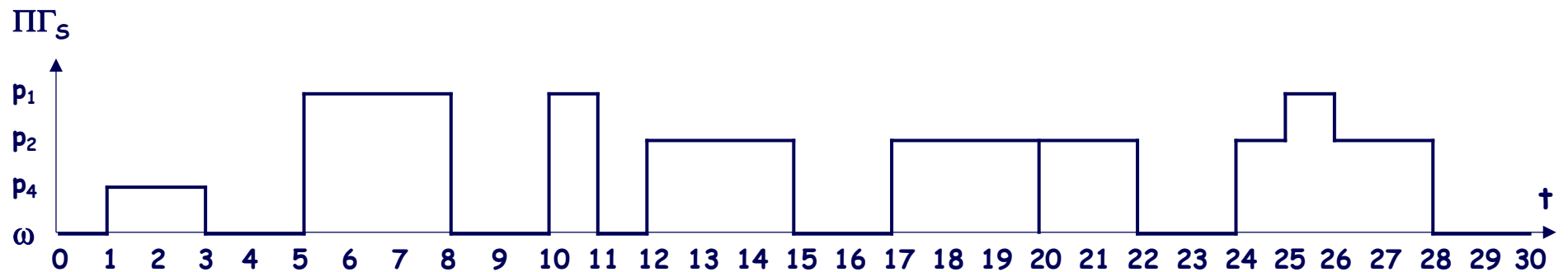
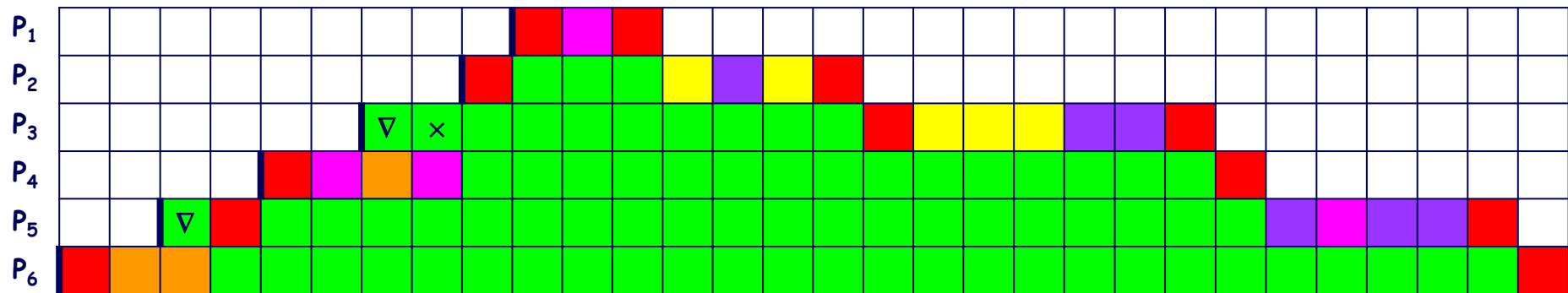
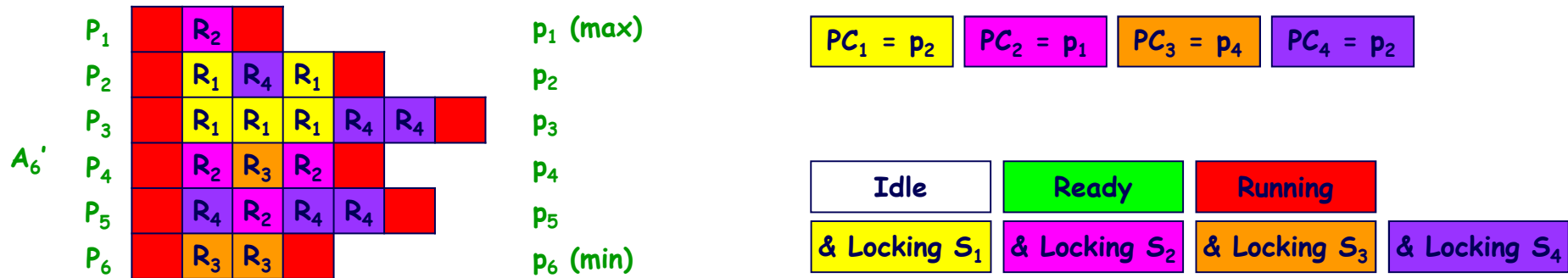
Il tetto di priorità del sistema $\Pi\Gamma_S$ è aggiornato ogni qual volta viene allocata o rilasciata una risorsa.

Le priorità correnti dei processi coincidono con le corrispondenti priorità nominali.

La schedulazione dei processi è operata in base alle relative priorità, nel rispetto del seguente vincolo: il processo pronto a massima priorità, se contraddistinto da una priorità maggiore di quella del processo in esecuzione, può causarne la preemption solo allorché la sua priorità risulta maggiore di $\Pi\Gamma_S$.

Una risorsa viene sempre allocata al processo che ne fa richiesta.

... IL PROTOCOLLO IPC ...



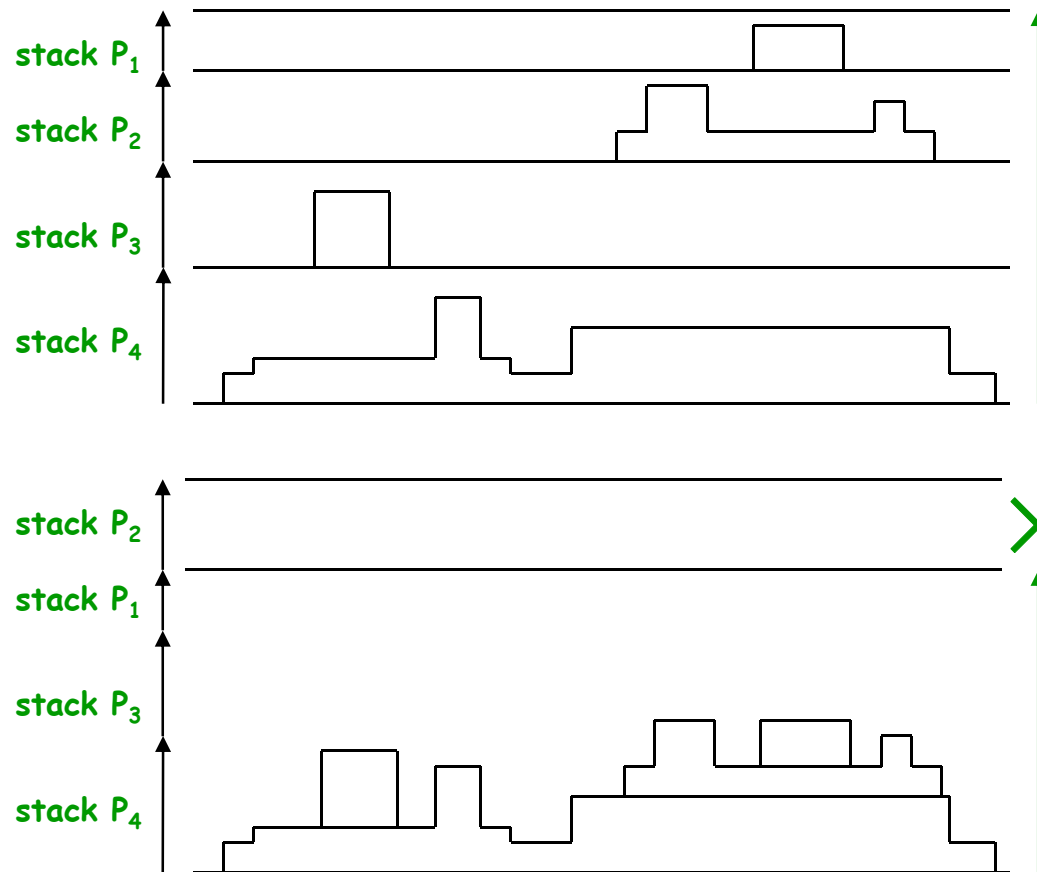
... IL PROTOCOLLO IPC ...

IPC vs. PC:

- ☹️ processi che non condividono risorse o che prevedono accessi condizionati tramite statement del tipo “if (condition) {lock (S_k), ..., unlock (S_k)}” possono comunque essere soggetti a blocchi preventivi → maggiori tempi di risposta;
- 😊 semplicità realizzativa;
- 😊 riduzione del numero di cambi di contesto (da 4 a 2), in quanto un processo non può subire un blocco dopo che è iniziata la sua esecuzione → minore overhead;
- 😊 possibilità di condivisione della risorsa “stack” → minore capacità di memoria richiesta allo scopo da parte di processi con la stessa priorità.

... IL PROTOCOLLO IPC

Quattro processi P_1, P_2, P_3, P_4 , di priorità, rispettivamente, $p_1 > p_2 = p_3 > p_4$



Qualora ogni processo abbia il suo spazio riservato di stack, di dimensione compatibile con la massima utilizzazione prevista, la dimensione totale della stack è uguale alla somma delle dimensioni delle singole stack.

Un processo, una volta in esecuzione, può subire preemption da parte di processi di priorità superiore, ma non più un blocco da parte di processi di priorità inferiore; processi con la stessa priorità non possono essere in esecuzione contemporaneamente: un unico spazio di stack condiviso dai processi.

Un sistema con k livelli di priorità implica una stack di dimensione uguale alla somma delle k maggiori dimensioni di stack necessarie su ogni livello (non per ogni processo).

Esempio: 100 processi, ciascuno con una utilizzazione massima di stack pari a 1 Kb, uniformemente distribuiti su 10 livelli di priorità. La dimensione totale della stack si riduce da 100 Kb a 10 Kb (- 90%).

IL PROTOCOLLO PREEMPTION CEILING ...

- E' applicabile in sistemi a priorità dinamica e "livelli di preemption" statici, ovvero in quei sistemi in cui i potenziali conflitti di accesso alle risorse condivise da parte di un insieme di processi non cambiano nel tempo.

Il livello di preemption ("preemption level", pl) di un job riflette la possibilità che esso ha di causare/subire la preemption di/da altri job. Se $pl_j < pl_i$, J_j può subire preemption da J_i (e quindi J_j può concorrere al blocco di J_i), ma J_i non può subire preemption da J_j (e quindi J_i non può concorrere al blocco di J_j).

In generale il livello di preemption pl_i di un job J_i è funzione del suo istante di attivazione r_i e della sua priorità p_i .

L'assegnamento dei livelli di preemption ai job deve essere operato nel rispetto della seguente "condizione di validità": se J_i è attivato dopo J_j ($r_i > r_j$) e J_i ha una priorità maggiore di J_j ($p_i > p_j$), allora J_i deve avere un livello di preemption maggiore di quello di J_j ($pl_i > pl_j$).

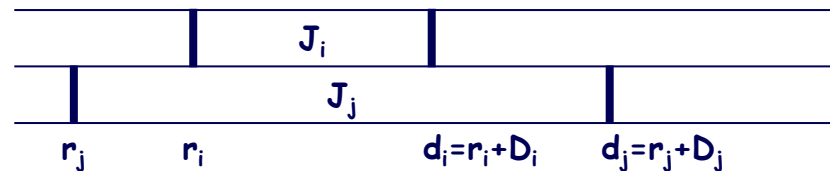
Ricadono in questa categoria i sistemi che operano la schedulazione dei processi in accordo alla strategia EDF, con livelli di preemption definiti come segue:

$$\text{se } D_j > D_i, \text{ allora } pl_j < pl_i \ (\forall i, j)$$

... IL PROTOCOLLO PREEMPTION CEILING ...

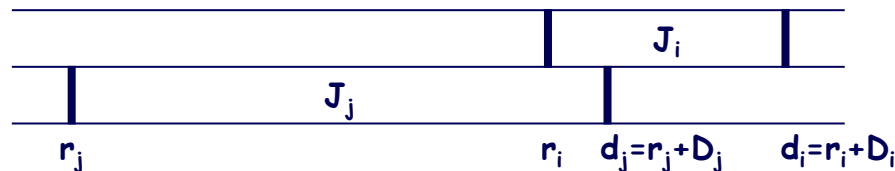
La condivisione di risorse tra due processi P_i e P_j aventi deadline relativa D_i e D_j , rispettivamente, non implica, qualora siano schedulati in accordo alla strategia EDF, che ciascuno di essi incorra, allorché in esecuzione con priorità massima, in una situazione di blocco imputabile all'altro.

$$D_j > D_i \rightarrow \boxed{pl_j < pl_i}$$



$$r_j < r_i, d_i < d_j \rightarrow \boxed{p_i > p_j}$$

J_j is preempted by $J_i \rightarrow J_i$ may be blocked by J_j



$$r_j < r_i, d_j < d_i \rightarrow \boxed{p_j > p_i}$$

J_j is not preempted by $J_i \rightarrow J_j$ can not be blocked by J_i

... IL PROTOCOLLO PREEMPTION CEILING ...

- Sfrutta regole analoghe a quelle previste dal protocollo PC in sistemi a priorità statica, con l'unica variante che l'allocazione di una risorsa è operata non più in base all'esito del confronto fra la priorità corrente del processo che ne fa richiesta ed il tetto di priorità del sistema, bensì fra il livello di preemption del processo ed il "tetto di preemption del sistema", definito come segue:

$$\Pi\Lambda_s = \max_k \{ PL_k \mid R_k \text{ é in uso} \}$$

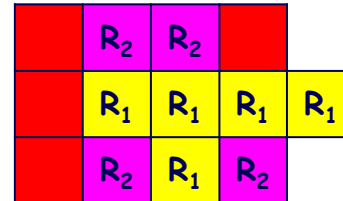
dove PL_k rappresenta il "preemption ceiling" associato staticamente alla risorsa R_k , coincidente con il massimo livello di preemption dei processi che ad essa possono accedere:

$$PL_k = \max_j \{ pl_j \mid P_j \text{ usa } R_k \} \quad \forall k$$

- Presenta le stesse proprietà del protocollo PC in sistemi a priorità statica, ovvero:
 - implica la conoscenza a priori delle risorse utilizzate dai processi;
 - previene inversioni di priorità incontrollate;
 - previene la concatenazione di blocchi (un processo può essere bloccato al più una volta, all'atto dell'accesso alla prima risorsa utilizzata);
 - previene situazioni di deadlock.

... IL PROTOCOLLO PREEMPTION CEILING ...

$A_7 \left\{ \begin{array}{l} P_1 (6, 10, 4, 10; [R_2; 2]) \\ P_2 (2, 15, 5, 15; [R_1; 4]) \\ P_3 (0, 20, 4, 20; [R_2; 3] [R_1; 1]) \end{array} \right.$



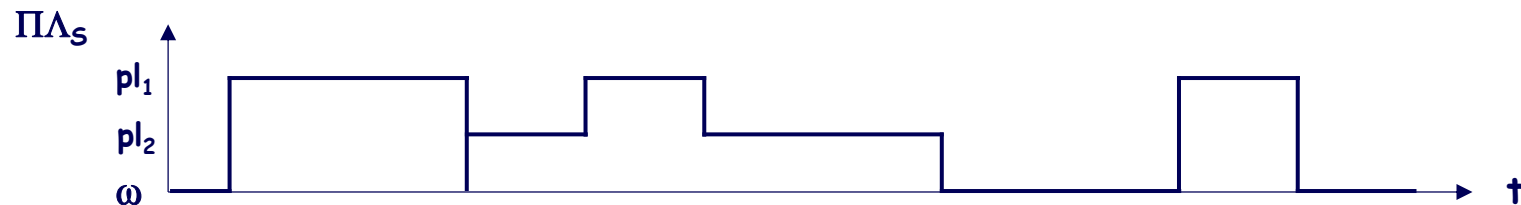
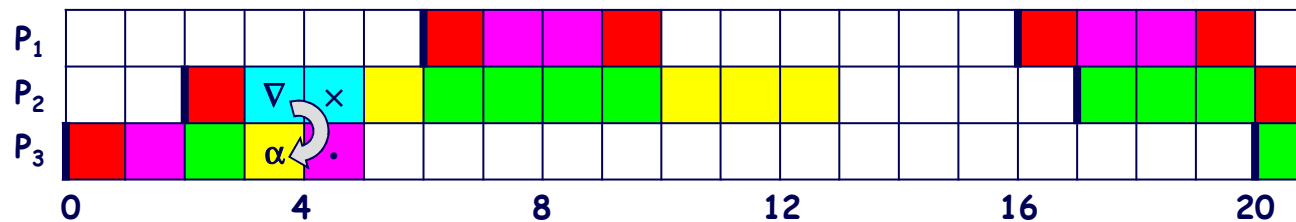
pl_1 (max)

pl_2

pl_3 (min)

$PL_1 = pl_2$

$PL_2 = pl_1$



... IL PROTOCOLLO PREEMPTION CEILING ...

Il massimo tempo di blocco B_i di un processo P_i con livello di preemption pl_i è uguale al tempo di esecuzione Z_{jk} della più lunga sezione critica relativa all'accesso da parte di un qualunque processo P_j con livello di preemption $pl_j < pl_i$ ad una qualunque risorsa R_k con tetto di preemption $PL_k > pl_j$:

$$B_i = \max_{j,k} \{ Z_{jk} \mid pl_j < pl_i, PL_k > pl_j \} \quad \forall i$$

Con riferimento all'applicazione A_7 , il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:

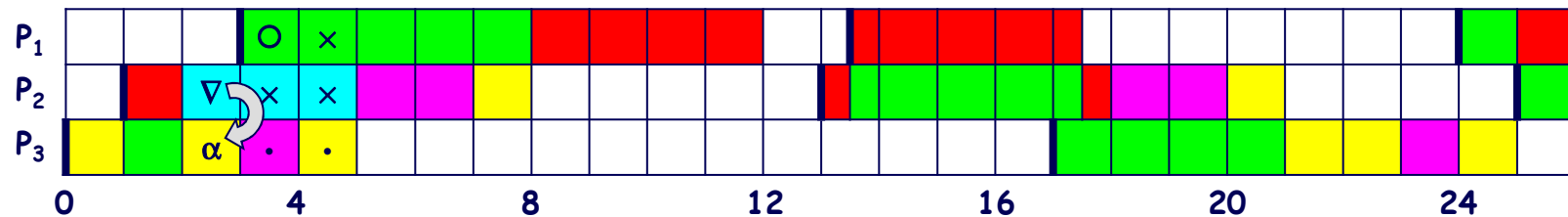
A_7 $\left\{ \begin{array}{l} P_1 (6, 10, 4, 10; [R_2;2]) \\ P_2 (2, 15, 5, 15; [R_1;4]) \\ P_3 (0, 20, 4, 20; [R_2;3 [R_1;1]]) \end{array} \right.$		<table> <tr><td></td><td>R_2</td><td>R_2</td><td></td></tr> <tr><td></td><td>R_1</td><td>R_1</td><td>R_1</td></tr> <tr><td></td><td>R_2</td><td>R_1</td><td>R_2</td></tr> </table>		R_2	R_2			R_1	R_1	R_1		R_2	R_1	R_2	pl_1 (max)	$PL_1 = pl_2$
		R_2	R_2													
		R_1	R_1	R_1												
	R_2	R_1	R_2													
			pl_2													
			pl_3 (min)	$PL_2 = pl_1$												

	R_1	R_2	
P_1		2	$B_1 = 3$
P_2	4		$B_2 = 3$
P_3	1	3	$B_3 = 0$

... IL PROTOCOLLO PREEMPTION CEILING ...

Con riferimento all'applicazione A_8 , il massimo tempo di blocco ($[t.u.]$) di ciascun processo risulta:

A_8	$P_1 (3, 10.5, 4, 10.5)$		$pl_1 (max)$	$PL_1 = pl_2$	P_1	R_1	R_2	$B_1 = 4$
	$P_2 (1, 12, 4, 12; [R_2;2] [R_1;1])$		pl_2	$PL_2 = pl_2$	P_2	1	2	$B_2 = 4$
	$P_3 (0, 17, 4, 17; [R_1;4] [R_2;1])$		$pl_3 (min)$		P_3	4	1	$B_3 = 0$

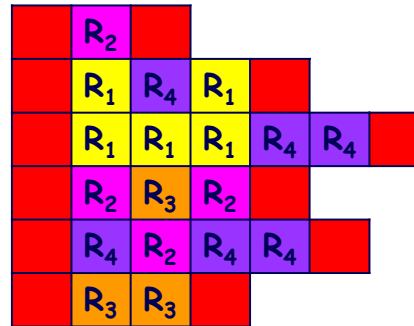


P_1 può essere bloccato da P_3 (in maniera indiretta) ma non da P_2 , in quanto P_2 può subire preemption solo da P_1 e P_1 non necessita delle risorse eventualmente detenute da P_2 .

... IL PROTOCOLLO PREEMPTION CEILING

Applicazione A_6

P_1 (9, 30, 3, 30; $[R_2;1]$)
 P_2 (8, 40, 5, 40; $[R_1;1]$ $[R_4;1]$ $[R_1;1]$)
 P_3 (6, 50, 7, 50; $[R_1;3]$ $[R_4;2]$)
 P_4 (4, 60, 5, 60; $[R_2;1]$ $[R_3;1]$ $[R_2;1]$)
 P_5 (2, 70, 6, 70; $[R_4;1]$ $[R_2;1]$ $[R_4;2]$)
 P_6 (0, 80, 4, 80; $[R_3;2]$)



Applicazione A_6'

P_1 (9, 30, 3, 30; $[R_2;1]$)
 P_2 (8, 40, 5, 40; $[R_1;3]$ $[R_4;1]$)
 P_3 (6, 50, 7, 50; $[R_1;3]$ $[R_4;2]$)
 P_4 (4, 60, 5, 60; $[R_2;3]$ $[R_3;1]$)
 P_5 (2, 70, 6, 70; $[R_4;4]$ $[R_2;1]$)
 P_6 (0, 80, 4, 80; $[R_3;2]$)

$$PL_1 = pl_2$$

$$PL_2 = pl_1$$

$$PL_3 = pl_4$$

$$PL_4 = pl_2$$

pl_1 (max)

pl_2

pl_3

pl_4

pl_5

pl_6 (min)

	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_1		1				1		
P_2	1			1	3			1
P_3	3			2	3			2
P_4		1	1			3	1	
P_5		1		2		1		4
P_6			2				2	

A_6

A_6'

$$B_1 = 3 / 4$$

$$B_2 = 3 / 4$$

$$B_3 = 2 / 4$$

$$B_4 = 2 / 4$$

$$B_5 = 2 / 2$$

$$B_6 = 0 / 0$$

A_6 / A_6'

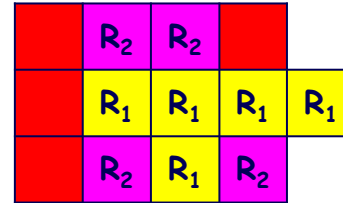
STACK RESOURCE POLICY [BAKER (91)] ...

- Sfrutta regole analoghe a quelle previste dal protocollo SBPC in sistemi a priorità statica, con l'unica variante che l'esecuzione di un processo ha luogo solo se:
 - ha la priorità massima fra i processi pronti e priorità maggiore del processo in esecuzione,
 - il suo livello di preemption è strettamente maggiore del tetto di preemption del sistema.

- Presenta le stesse proprietà del protocollo SBPC in sistemi a priorità statica, ovvero:
 - implica la conoscenza a priori delle risorse utilizzate dai processi;
 - previene inversioni di priorità incontrollate;
 - previene la concatenazione di blocchi (un processo può essere bloccato al più una volta, prima che abbia inizio la sua esecuzione);
 - previene situazioni di deadlock.

... STACK RESOURCE POLICY ...

$A_7 \left\{ \begin{array}{l} P_1 (6, 10, 4, 10; [R_2; 2]) \\ P_2 (2, 15, 5, 15; [R_1; 4]) \\ P_3 (0, 20, 4, 20; [R_2; 3] [R_1; 1]) \end{array} \right.$



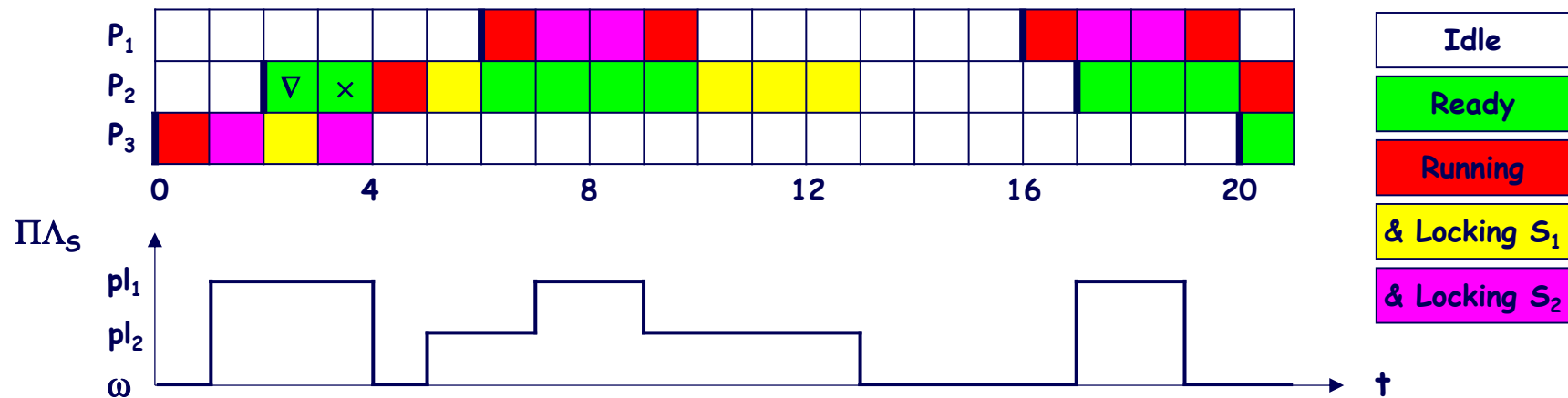
pl_1 (max)

pl_2

pl_3 (min)

$PL_1 = pl_2$

$PL_2 = pl_1$



Per quanto concerne il massimo tempo di blocco B_i di un qualunque processo P_i , valgono le stesse considerazioni illustrate per il protocollo Preemption Ceiling, ovvero:

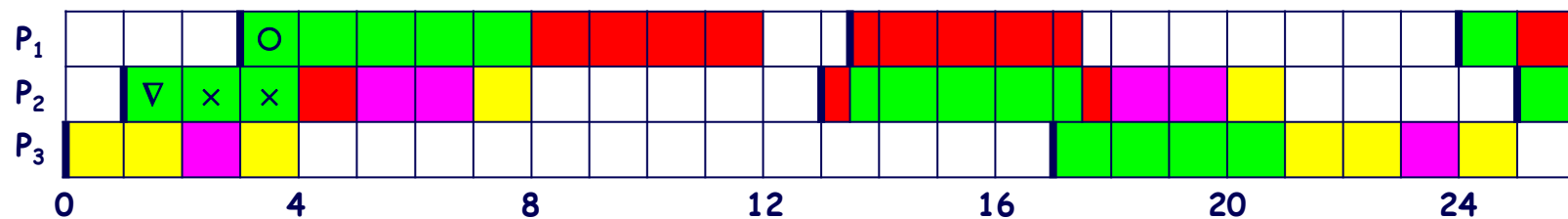
$$B_i = \max_{j,k} \{ Z_{jk} \mid pl_j < pl_i, PL_k > pl_j \} \quad \forall i$$

		R_1	R_2	
A_7 :	P_1		2	$B_1 = 3$
	P_2	4		$B_2 = 3$
	P_3	1	3	$B_3 = 0$

... STACK RESOURCE POLICY ...

Con riferimento all'applicazione A_8 , il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:

A_8 {	P_1 (3, 10.5, 4, 10.5)		pl_1 (max)	$PL_1 = pl_2$	P_1	R_1	R_2	$B_1 = 4$
	P_2 (1, 12, 4, 12; $[R_2;2]$ $[R_1;1]$)		pl_2		P_2	1	2	$B_2 = 4$
	P_3 (0, 17, 4, 17; $[R_1;4]$ $[R_2;1]$)		pl_3 (min)	$PL_2 = pl_2$	P_3	4	1	$B_3 = 0$



... STACK RESOURCE POLICY ...

Il protocollo, previa ridefinizione dei tetti di preemption delle risorse, è facilmente estendibile al caso in cui i processi condividano risorse con più unità.

A tal fine, indicato con

- u_k ($u_k \geq 1$) il numero di unità disponibili per ogni risorsa R_k ($k = 1, \dots, M$),
 - n_{jk} il numero massimo di unità di R_k richiesto da ciascun processo P_j ($j = 1, \dots, N$),
 - v_k ($k = 0, \dots, u_k$) il numero di unità di R_k libere nel generico istante t ,
- è sufficiente aggiornare il tetto di preemption del sistema $\Pi\Lambda_S$ come segue:

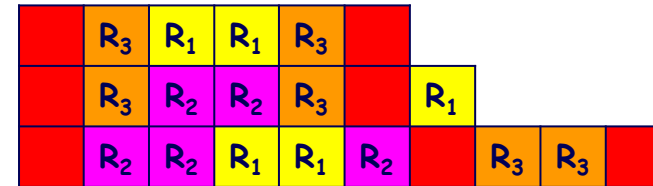
$$\Pi\Lambda_S = \max_k \{ \Pi\Lambda_k(v_k) \}$$

dove $\Pi\Lambda_k(v_k)$ rappresenta il tetto di preemption di R_k , dinamicamente variabile nel tempo in funzione di v_k e coincidente con il massimo livello di preemption dei processi che richiedono più di v_k unità di R_k (con il valore ω qualora nessun processo richieda più di v_k unità):

$$\Pi\Lambda_k(v_k) = \max_j \{ pl_j \mid n_{jk} > v_k \} \quad \forall k$$

... STACK RESOURCE POLICY ...

$$A_9 \left\{ \begin{array}{l} P_1 (4, 20, 6, 20; [R_3, 1; 4 [R_1, 1; 2]]) \\ P_2 (2, 24, 7, 24; [R_3, 3; 4 [R_2, 2]] [R_1, 2; 1]) \\ P_3 (0, 28, 10, 28; [R_2, 5 [R_1, 3; 2]] [R_3, 1; 2]) \end{array} \right.$$

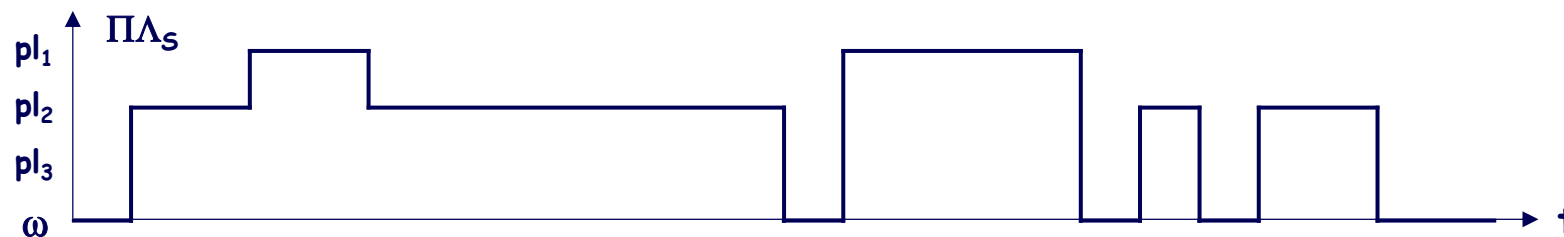
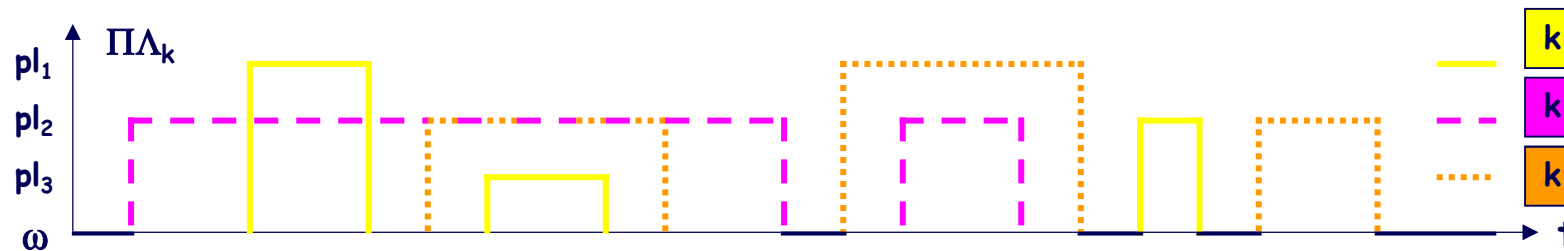
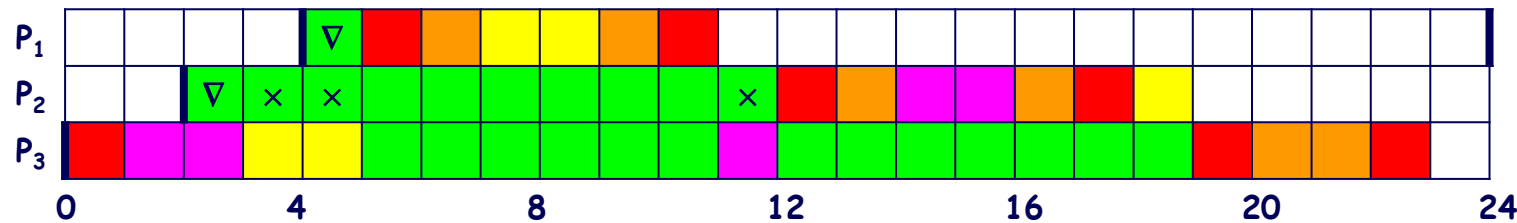


pl₁ (max)

pl₂

pl₃ (min)

k	u _k	n _{1k}	n _{2k}	n _{3k}	ΠΛ _k (v _k =0)	ΠΛ _k (v _k =1)	ΠΛ _k (v _k =2)	ΠΛ _k (v _k =3)
1	3	1	2	3	pl ₁	pl ₂	pl ₃	ω
2	1	0	1	1	pl ₂	ω	-	-
3	3	1	3	1	pl ₁	pl ₂	pl ₂	ω



... STACK RESOURCE POLICY ...

Il massimo tempo di blocco B_i di un processo P_i con livello di preemption pl_i è uguale al tempo di esecuzione Z_{jk} della più lunga sezione critica relativa all'accesso da parte di un qualunque processo P_j con livello di preemption $pl_j < pl_i$ ad una qualunque risorsa R_k con tetto di preemption massimo $\Pi\Lambda_k(0) > pl_j$:

$$B_i = \max_{j,k} \{ Z_{jk} \mid pl_j < pl_i, \Pi\Lambda_k(0) > pl_j \} \quad \forall i$$

Con riferimento all'applicazione A_9 , il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:

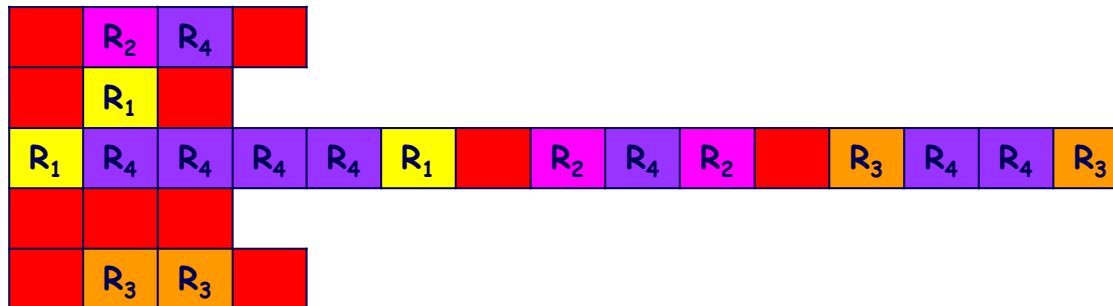
{	P_1 (4, 20, 6, 20; [R_3 ,1;4 [R_1 ,1;2]])	<div style="display: inline-block; width: 20px; height: 20px; background-color: red; border: 1px solid black;"></div> <div style="display: inline-block; width: 20px; height: 20px; background-color: orange; border: 1px solid black; text-align: center; color: black;">R_3</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: yellow; border: 1px solid black; text-align: center; color: black;">R_1</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: yellow; border: 1px solid black; text-align: center; color: black;">R_1</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: orange; border: 1px solid black; text-align: center; color: black;">R_3</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: red; border: 1px solid black;"></div>		pl_1 (max)
	P_2 (2, 24, 7, 24; [R_3 ,3;4 [R_2 ;2]] [R_1 ,2;1])	<div style="display: inline-block; width: 20px; height: 20px; background-color: red; border: 1px solid black;"></div> <div style="display: inline-block; width: 20px; height: 20px; background-color: orange; border: 1px solid black; text-align: center; color: black;">R_3</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: purple; border: 1px solid black; text-align: center; color: black;">R_2</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: purple; border: 1px solid black; text-align: center; color: black;">R_2</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: orange; border: 1px solid black; text-align: center; color: black;">R_3</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: red; border: 1px solid black;"></div> <div style="display: inline-block; width: 20px; height: 20px; background-color: yellow; border: 1px solid black; text-align: center; color: black;">R_1</div>		pl_2
	P_3 (0, 28, 10, 28; [R_2 ;5 [R_1 ,3;2]] [R_3 ,1;2])	<div style="display: inline-block; width: 20px; height: 20px; background-color: red; border: 1px solid black;"></div> <div style="display: inline-block; width: 20px; height: 20px; background-color: purple; border: 1px solid black; text-align: center; color: black;">R_2</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: purple; border: 1px solid black; text-align: center; color: black;">R_2</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: yellow; border: 1px solid black; text-align: center; color: black;">R_1</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: yellow; border: 1px solid black; text-align: center; color: black;">R_1</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: purple; border: 1px solid black; text-align: center; color: black;">R_2</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: red; border: 1px solid black;"></div> <div style="display: inline-block; width: 20px; height: 20px; background-color: orange; border: 1px solid black; text-align: center; color: black;">R_3</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: orange; border: 1px solid black; text-align: center; color: black;">R_3</div> <div style="display: inline-block; width: 20px; height: 20px; background-color: red; border: 1px solid black;"></div>		pl_3 (min)

		R_1	R_2	R_3	
P_1	2		4		$B_1 = 5$
P_2	1	2	4		$B_2 = 5$
P_3	2	5	2		$B_3 = 0$

k	1	2	3
$\Pi\Lambda_k(0)$	pl_1	pl_2	pl_1

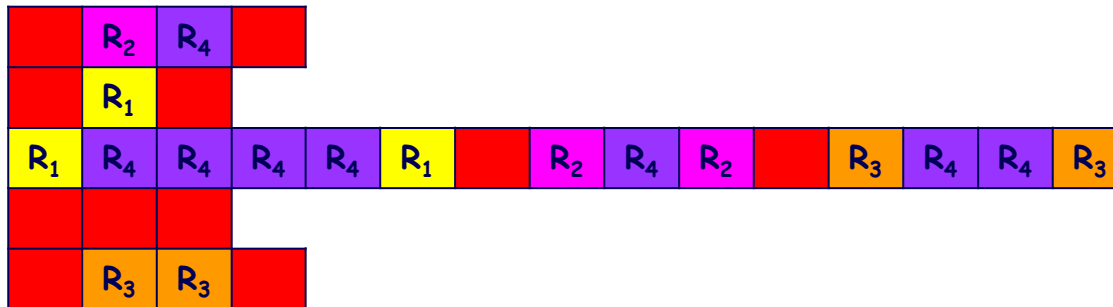
... STACK RESOURCE POLICY ...

{	P_1	(8, 25, 4, 25; [R ₂ ,1;1] [R ₄ ;1])	pl_1 (max)
	P_2	(6, 30, 3, 30; [R ₁ ,1;1])	pl_2
	P_3	(4, 35, 15, 35; [R ₁ ,2;6 [R ₄ ;4]] [R ₂ ,2;3 [R ₄ ;1]] [R ₃ ;4 [R ₄ ;2]])	pl_3
	P_4	(2, 45, 3, 40)	pl_4
	P_5	(0, 55, 4, 45; [R ₃ ;2])	pl_5 (min)



k	u_k	n_{1k}	n_{2k}	n_{3k}	n_{4k}	n_{5k}	$\Pi\Lambda_k(v_k=0)$	$\Pi\Lambda_k(v_k=1)$	$\Pi\Lambda_k(v_k=2)$
1	2		1	2			pl_2	pl_3	ω
2	2	1		2			pl_1	pl_3	ω
3	1			1		1	pl_3	ω	-
4	1	1		1			pl_1	ω	-

... STACK RESOURCE POLICY ...



pl_1 (max)

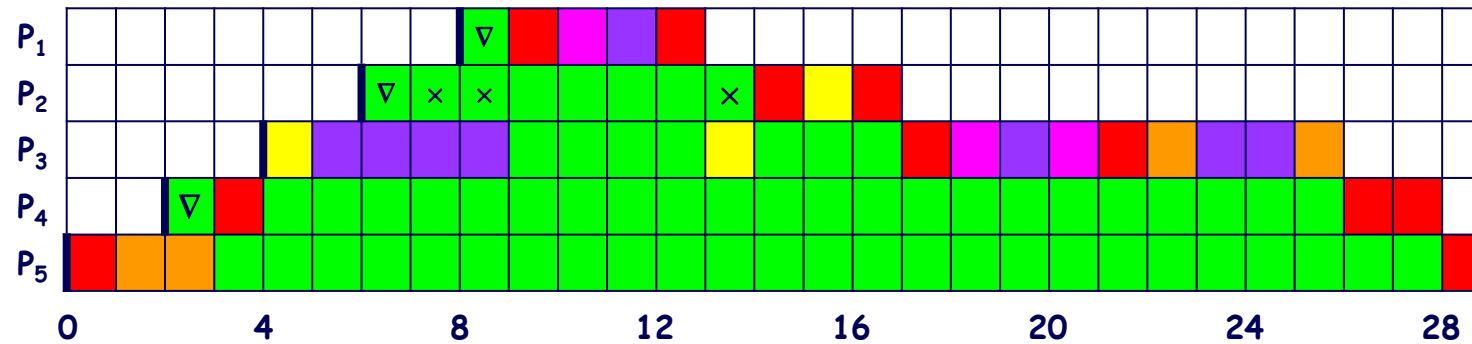
pl_2

pl_3

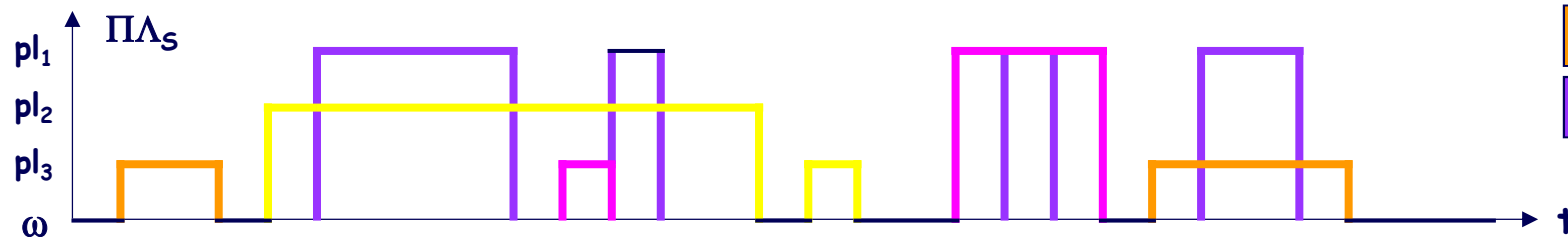
pl_4

pl_5 (min)

k	u_k	n_{1k}	n_{2k}	n_{3k}	n_{4k}	n_{5k}	$\Pi\Lambda_k(v_k=0)$	$\Pi\Lambda_k(v_k=1)$	$\Pi\Lambda_k(v_k=2)$
1	2		1	2			pl_2	pl_3	ω
2	2	1		2			pl_1	pl_3	ω
3	1			1		1	pl_3	ω	-
4	1	1		1			pl_1	ω	-

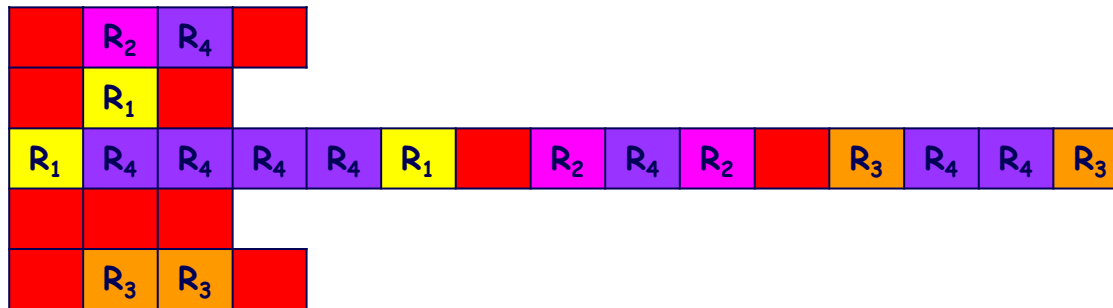


Idle
Ready
Running
& Locking S_1
& Locking S_2
& Locking S_3
& Locking S_4



... STACK RESOURCE POLICY

A_{10}	P_1 (8, 25, 4, 25; $[R_2,1;1]$ $[R_4;1]$)	pl_1 (max)
	P_2 (6, 30, 3, 30; $[R_1,1;1]$)	pl_2
	P_3 (4, 35, 15, 35; $[R_1,2;6]$ $[R_4;4]$ $[R_2,2;3]$ $[R_4;1]$ $[R_3;4]$ $[R_4;2]$)	pl_3
	P_4 (2, 45, 3, 40)	pl_4
	P_5 (0, 55, 4, 45; $[R_3;2]$)	pl_5 (min)



Il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:

k	1	2	3	4
$\Pi\Lambda_k(0)$	pl_2	pl_1	pl_3	pl_1

	R_1	R_2	R_3	R_4
P_1		1		1
P_2	1			
P_3	6	3	4	4
P_4				
P_5			2	

$B_1 = 6$
$B_2 = 6$
$B_3 = 2$
$B_4 = 2$
$B_5 = 0$

ANALISI DI SCHEDULABILITÀ ...

□ Sistemi a priorità statica

In presenza di risorse condivise, condizione sufficiente affinché un insieme di N processi periodici $P_1 (T_1, C_1), \dots, P_N (T_N, C_N)$, ordinati per periodo crescente (priorità decrescente), sia schedulabile con RMPO è che:

$$\forall i, 1 \leq i \leq N, \sum_{k=1}^i \frac{C_k}{T_k} + \frac{B_i}{T_i} \leq U_{\text{RMPO}}(i) = \begin{cases} 1 & \text{a) se i periodi sono in} \\ & \text{relazione armonica} \\ i(2^{1/i} - 1) & \text{b) in caso contrario} \end{cases}$$

o più semplicemente (un unico test in luogo di N), ma in maniera ancora più conservativa:

$$\sum_{k=1}^{N-1} \frac{C_k}{T_k} + \max \left\{ \frac{C_N}{T_N}, \frac{B_1}{T_1}, \dots, \frac{B_{N-1}}{T_{N-1}} \right\} \leq U_{\text{RMPO}}(N) = \begin{cases} 1 & \text{a)} \\ N(2^{1/N} - 1) & \text{b)} \end{cases}$$

... ANALISI DI SCHEDULABILITÀ ...

Esempio: l'applicazione A_5 risulta schedulabile con RMPO (in base al 1° criterio, non al 2°), qualunque sia il protocollo di accesso a risorse condivise utilizzato (NPCSP, PIP, PCP, IPCP).

			priorità		B ₁	B ₂	B ₃	B ₄
{	A ₅	P ₁ (6, 15, 6, 15; [R ₁ :4] [R ₂ :2])	p ₁ (max)	NPCSP	4	4	4	0
		P ₂ (4, 20, 4, 20; [R ₂ :2])	p ₂	PIP	6	4	4	0
		P ₃ (2, 50, 4, 50)	p ₃	PCP	4	4	4	0
		P ₄ (0, 100, 6, 100; [R ₁ :4])	p ₄ (min)	IPCP	4	4	4	0

Infatti, con riferimento a PIP (caso più sfavorevole), si ha:

$$A_5: C_1/T_1 + C_2/T_2 + C_3/T_3 + B_1/T_1 = 6/15 + 4/20 + 4/50 + 6/15 = 1.08 > U_{RMPO}(4) = 0.756 \quad \text{NO}$$

$$P_1: C_1/T_1 + B_1/T_1 = 6/15 + 6/15 = 0.8 \leq U_{RMPO}(1) = 1 \quad \text{OK}$$

$$P_2: C_1/T_1 + C_2/T_2 + B_2/T_2 = 6/15 + 4/20 + 4/20 = 0.8 \leq U_{RMPO}(2) = 0.828 \quad \text{OK}$$

$$P_3: C_1/T_1 + C_2/T_2 + C_3/T_3 + B_3/T_3 = 6/15 + 4/20 + 4/50 + 4/50 = 0.76 \leq U_{RMPO}(3) = 0.779 \quad \text{OK}$$

$$P_4: C_1/T_1 + C_2/T_2 + C_3/T_3 + C_4/T_4 = 6/15 + 4/20 + 4/50 + 6/100 = 0.74 \leq U_{RMPO}(4) = 0.756 \quad \text{OK}$$

... ANALISI DI SCHEDULABILITÀ ...

In presenza di risorse condivise, condizione necessaria e sufficiente affinché un insieme di N processi periodici e/o sporadici $P_1(T_1, C_1, D_1), \dots, P_N(T_N, C_N, D_N)$, ordinati per deadline relativa crescente (priorità decrescente), sia schedulabile con DMPO (con RMPO se $D_i = T_i, \forall i$) è che (algoritmo di Audsley):

$$R_i = C_i + B_i + I_i(R_i) \leq D_i \quad \forall i$$

$$R_i^0 = C_i + B_i \quad \forall i, \quad R_i^n = C_i + B_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_i^{n-1}}{T_k} \right\rceil C_k \quad i \neq 1, \quad n = 1, 2, \dots$$

Esempio: la seguente applicazione,

			priorità				B ₁	B ₂	B ₃	B ₄
{	A ₅ [*]	P ₁ (6, 15, 6, 15; [R ₁ ;4] [R ₂ ;2])	p ₁ (max)	NPCSP			4	4	4	0
		P ₂ (4, 20, 4, 20; [R ₂ ;2])	p ₂	PIP			6	4	4	0
		P ₃ (2, 50, 4, 30)	p ₃	PCP			4	4	4	0
		P ₄ (0, 100, 6, 50; [R ₁ ;4])	p ₄ (min)	IPCP			4	4	4	0

che differisce da A₅ soltanto per quanto concerne la deadline ([t.u.]) dei processi P₃ e P₄ (D₃=30 (anziché 50), D₄=50 (anziché 100)), risulta schedulabile con DMPO, qualunque sia il protocollo di accesso a risorse condivise utilizzato (NPCSP, PIP, PCP, IPCP).

... ANALISI DI SCHEDULABILITÀ ...

Infatti, con riferimento al protocollo più sfavorevole (PIP), si ha:

$$R_1^0 = 6 + 6 = 12$$

$$R_1^1 = 12 \leq D_1 = 15$$

OK

$$R_2^0 = 4 + 4 = 8$$

$$R_2^1 = 8 + 6 = 14$$

$$R_2^2 = 14 \leq D_2 = 20$$

OK

$$R_3^0 = 4 + 4 = 8$$

$$R_3^1 = 8 + 6 + 4 = 18$$

$$R_3^2 = 8 + 12 + 4 = 24$$

$$R_3^3 = 8 + 12 + 8 = 28$$

$$R_3^4 = 28 \leq D_3 = 30$$

OK

$$R_4^0 = 6$$

$$R_4^1 = 6 + 6 + 4 + 4 = 20$$

$$R_4^2 = 6 + 12 + 4 + 4 = 26$$

$$R_4^3 = 6 + 12 + 8 + 4 = 30$$

$$R_4^4 = 30 \leq D_4 = 50$$

OK

... ANALISI DI SCHEDULABILITÀ ...

□ Sistemi a priorità dinamica

In presenza di risorse condivise, condizione sufficiente affinché un insieme di N processi periodici $P_1(T_1, C_1), \dots, P_N(T_N, C_N)$, ordinati per periodo crescente, sia schedulabile con EDF è che:

$$\forall i, 1 \leq i \leq N, \sum_{k=1}^i \frac{C_k}{T_k} + \frac{B_i}{T_i} \leq U_{\text{EDF}} = 1$$

o più semplicemente (un unico test in luogo di N), ma in maniera ancora più conservativa:

$$\sum_{k=1}^{N-1} \frac{C_k}{T_k} + \max \left\{ \frac{C_N}{T_N}, \frac{B_1}{T_1}, \dots, \frac{B_{N-1}}{T_{N-1}} \right\} \leq 1$$

... ANALISI DI SCHEDULABILITÀ ...

Esempio: l'applicazione A_7 risulta schedulabile con EDF (in base al 1° criterio, non al 2°), qualunque sia il protocollo di accesso a risorse condivise utilizzato (NPCSP, PCP, SRP).

			B ₁	B ₂	B ₃
A ₇ {	P ₁ (6, 10, 4, 10; [R ₂ ;2])	NPCSP	4	3	0
	P ₂ (2, 15, 5, 15; [R ₁ ;4])	PCP	3	3	0
	P ₃ (0, 20, 4, 20; [R ₂ ;3 [R ₁ ;1]])	SRP	3	3	0

Infatti, con riferimento al protocollo più sfavorevole (NPCSP), si ha:

$$A_7: C_1/T_1 + C_2/T_2 + B_1/T_1 = 4/10 + 5/15 + 4/10 = 1.13 > 1 \quad \text{NO}$$

$$P_1: C_1/T_1 + B_1/T_1 = 4/10 + 4/10 = 0.8 \leq 1 \quad \text{OK}$$

$$P_2: C_1/T_1 + C_2/T_2 + B_2/T_2 = 4/10 + 5/15 + 3/15 = 0.93 \leq 1 \quad \text{OK}$$

$$P_3: C_1/T_1 + C_2/T_2 + C_3/T_3 = 4/10 + 5/15 + 4/20 = 0.93 \leq 1 \quad \text{OK}$$

... ANALISI DI SCHEDULABILITÀ ...

In presenza di risorse condivise, condizione sufficiente affinché un insieme di N processi periodici e/o sporadici $P_1(T_1, C_1, D_1), \dots, P_N(T_N, C_N, D_N)$, ordinati per deadline relativa crescente, sia schedulabile con EDF è che:

$$\forall i, 1 \leq i \leq N, \sum_{k=1}^i \frac{C_k}{D_k} + \frac{B_i}{D_i} \leq 1$$

Esempio: la seguente applicazione,

			B ₁	B ₂	B ₃
A ₇ [*] {	P ₁ (6, 10, 4, 9; [R ₂ ;2])	NPCSP	4	3	0
	P ₂ (2, 15, 5, 15; [R ₁ ;4])	PCP	3	3	0
	P ₃ (0, 20, 4, 16; [R ₂ ;3 [R ₁ ;1]])	SRP	3	3	0

che differisce da A_7 soltanto per quanto concerne la deadline ([t.u.]) dei processi P_1 e P_3 ($D_1=9$ (anziché 10), $D_3=16$ (anziché 20)), non è detto che sia schedulabile con EDF, qualunque sia il protocollo di accesso a risorse condivise utilizzato (NPCSP, PCP, SRP). Infatti:

$$P_1: C_1/D_1 + B_1/D_1 = 4/9 + 4(3)/9 = 0.889(0.778) \leq 1 \quad \text{OK}$$

$$P_2: C_1/D_1 + C_2/D_2 + B_2/D_2 = 4/9 + 5/15 + 3/15 = 0.978 \leq 1 \quad \text{OK}$$

$$P_3: C_1/D_1 + C_2/D_2 + C_3/D_3 = 4/9 + 5/15 + 4/16 = 1.028 > 1 \quad \text{NO}$$

... ANALISI DI SCHEDULABILITÀ ...

In presenza di risorse condivise, condizione necessaria e sufficiente affinché un insieme di N processi periodici e/o sporadici $P_1(T_1, C_1, D_1), \dots, P_N(T_N, C_N, D_N)$, ordinati per deadline relativa crescente, sia schedulabile con EDF è che (approccio "processor demand"):

$$\forall i, 1 \leq i \leq N, \sum_{k=1}^i \left(\left\lfloor \frac{t - D_k}{T_k} \right\rfloor + 1 \right) C_k + \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) B_i \leq t$$

$$\forall t \in \{d_{ij} \mid d_{ij} = (j-1)T_i + D_i, d_{ij} < \min(BI, t^*), 1 \leq i \leq N, j \geq 1\}$$

Esempio: l'applicazione

$A_7^* \left\{ \begin{array}{l} P_1 (6, 10, 4, 9; [R_2; 2]) \\ P_2 (2, 15, 5, 15; [R_1; 4]) \\ P_3 (0, 20, 4, 16; [R_2; 3 [R_1; 1]]) \end{array} \right.$			B_1	B_2	B_3
		NPCSP	4	3	0
		PCP	3	3	0
		SRP	3	3	0

risulta schedulabile con EDF, qualunque sia il protocollo di accesso a risorse condivise utilizzato (NPCSP, PCP, SRP). Infatti, con riferimento al protocollo più sfavorevole (NPCSP), si ha:

... ANALISI DI SCHEDULABILITÀ

$$BI^0 = 4 + 5 + 4 = 13$$

$$BI^1 = \lceil 13/10 \rceil 4 + \lceil 13/15 \rceil 5 + \lceil 13/20 \rceil 4 = 17$$

$$BI^2 = \lceil 17/10 \rceil 4 + \lceil 17/15 \rceil 5 + \lceil 17/20 \rceil 4 = 22$$

$$BI^3 = \lceil 22/10 \rceil 4 + \lceil 22/15 \rceil 5 + \lceil 22/20 \rceil 4 = 30$$

$$BI^4 = \lceil 30/10 \rceil 4 + \lceil 30/15 \rceil 5 + \lceil 30/20 \rceil 4 = 30 = BI^3$$

$$BI = 30 > t^* = (0,4 + 0,8) / (1 - 0,93) = 18$$

$$\mathcal{D} \cap \mathcal{D}^* \equiv \{9, 15, 16\}$$

$$\forall i, 1 \leq i \leq N, \sum_{k=1}^i \left(\left\lfloor \frac{t - D_k}{T_k} \right\rfloor + 1 \right) C_k + \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) B_i \leq t$$

	k=1	k=2	k=3	i=1	i=2	i=3	i=1	i=2	i=3
t	①	②	③	④	⑤	⑥	①+④	①+②+⑤	①+②+③+⑥
9	4	0	0	4	0	0	8	4	4
15	4	5	0	4	3	0	8	12	9
16	4	5	4	4	3	0	8	12	13

