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 monoprocessore;
- N processi P_1, …, 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, …,R_M, ciascuna con un numero finito u_k (k = 1, …, 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"

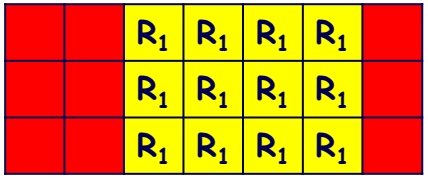
$$\pi_i = p_0 > \max \{p_1, p_2, \ldots, p_N\}$$

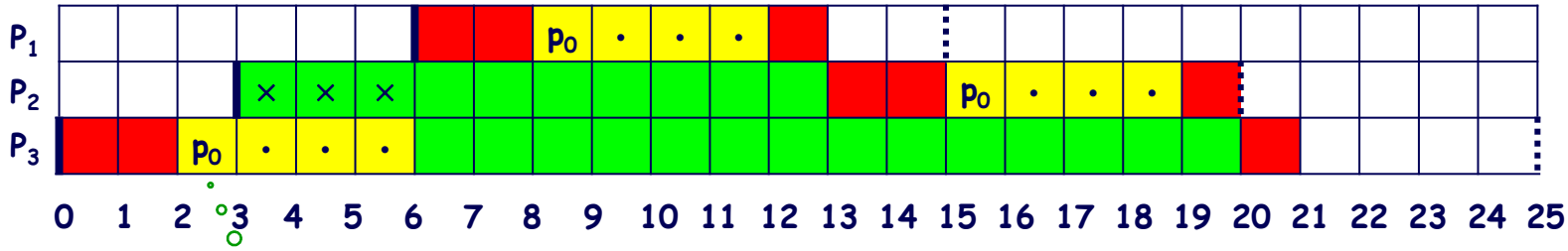
allorché inizia l'esecuzione di una sezione critica, e di nuovo associare ad esso la sua priorità nominale p_i allorché termina l'esecuzione della sezione critica.

… IL PROTOCOLLO NPCS …

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

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





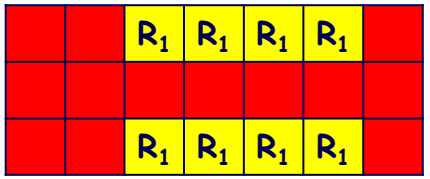
(°) Per ogni processo P_i ($\forall$ i), la priorità corrente π_i è esplicitamente evidenziata solo se differisce dalla priorità nominale p_i (“.” denota più sinteticamente l’ultimo valore di π_i indicato).

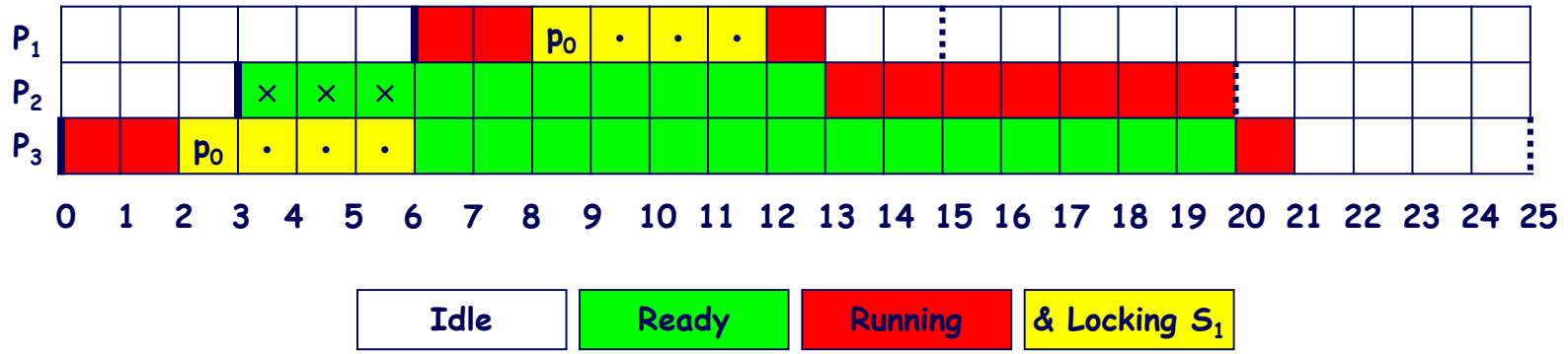
Idle | Ready | Running | & Locking S_1

… IL PROTOCOLLO NPCS …

Previene inversioni di priorità incontrollate.

		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

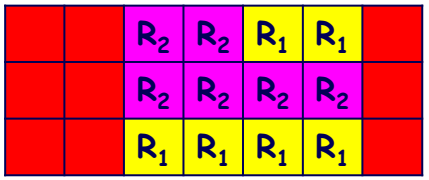


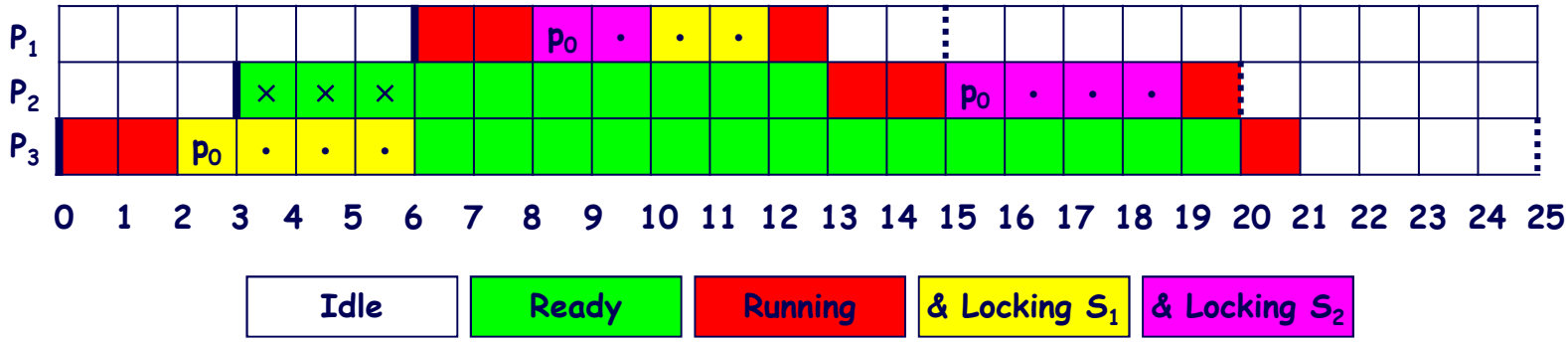


… IL PROTOCOLLO NPCS …

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

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

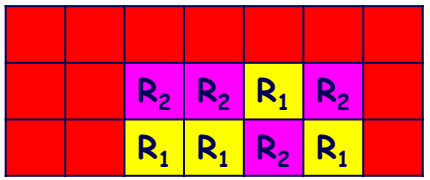




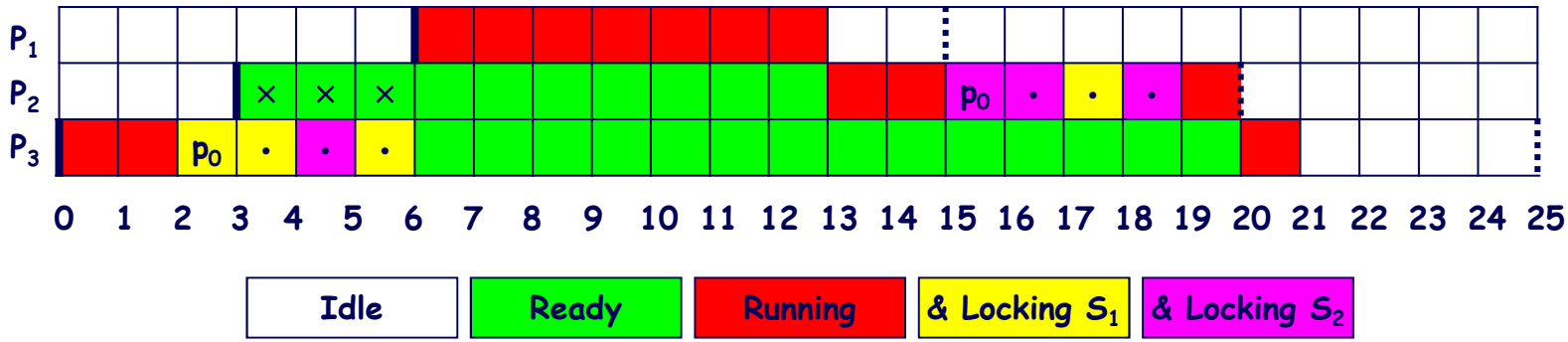
… IL PROTOCOLLO NPCS …

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



!?



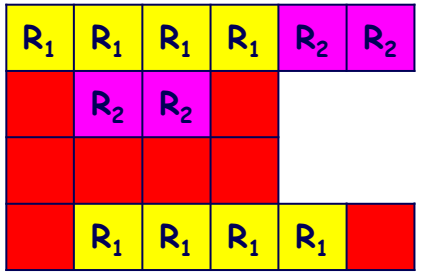
… IL PROTOCOLLO NPCS …

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 $p_j < p_i$:

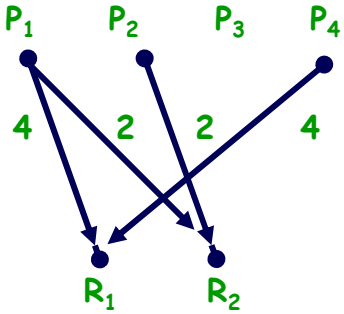
$$B_i = \max_{j,k} \{ Z_{jk} \mid p_j < p_i \} \quad \forall i$$

Qualche esempio, facendo riferimento alla tipica notazione utilizzata per descrivere parametri temporali [t.u.] e sezioni critiche di ogni processo: P_j (ϕ_j, T_j, C_j, D_j; [R_k;Z_{jk}] … [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])



Resource Requirements Graph (RRG)



Strategia di scheduling RMPO

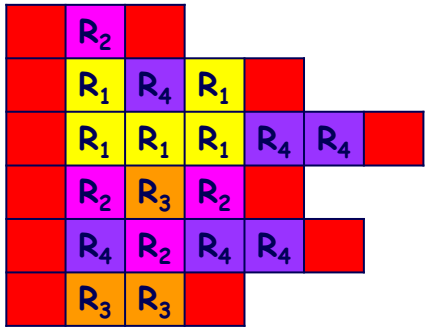
priorità		R_1	R_2
p_1 (max)	P_1	4	2
p_2	P_2		2
p_3	P_3		
p_4 (min)	P_4	4	

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])

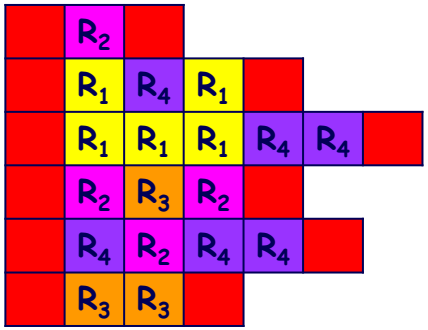
A_6' differisce da A_6 per la modalità di accesso alle risorse da parte dei tre processi P_2, P_4 e P_5: sezioni critiche annidate con rilascio delle risorse secondo l'ordine LIFO.

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'

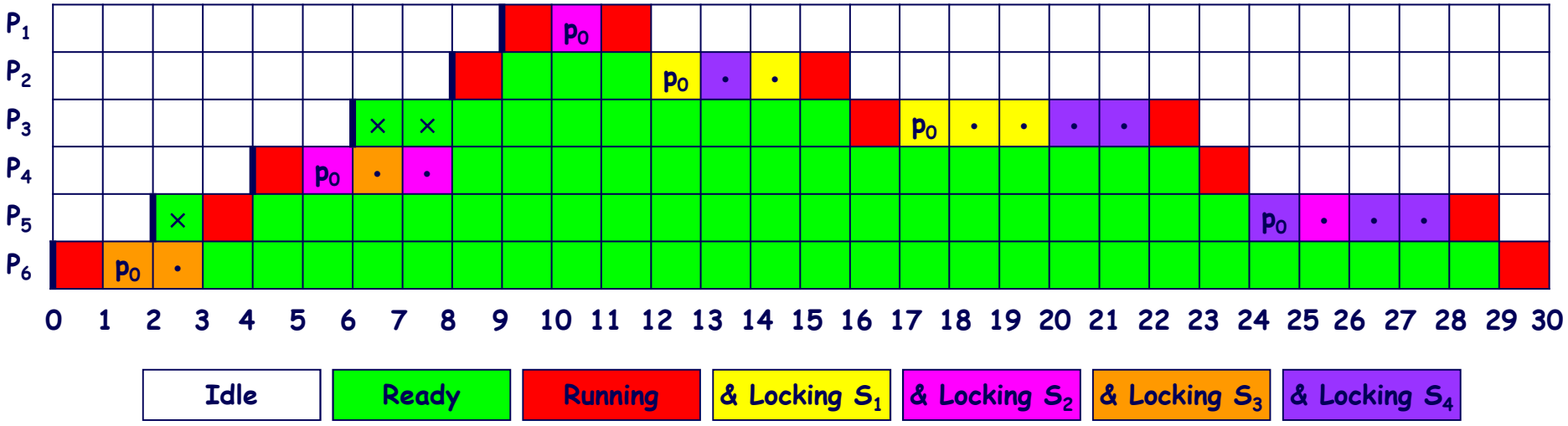
… IL PROTOCOLLO NPCS …

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)



Nell'ipotesi che a_i (i = 1, …, 4) > a_5 > a_6 = ∀,

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

P_i (i = 1, …, 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 $\varepsilon \to 0$.

… 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 $D_j > D_i$.

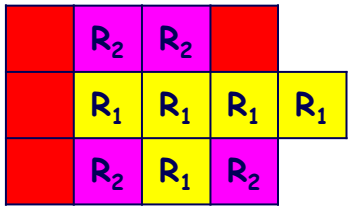
Il massimo tempo di blocco B_i di un processo P_i con deadline relativa D_i è pertanto dato dal tempo di esecuzione Z_{jk} della più lunga sezione critica relativa a qualunque risorsa R_k utilizzata da un processo P_j con deadline relativa $D_j > D_i$:

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

… IL PROTOCOLLO NPCS …

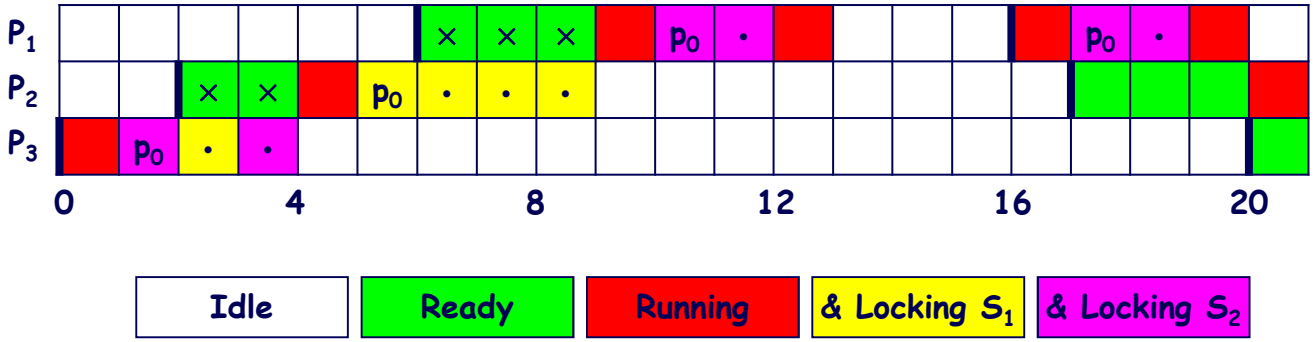
Con riferimento alla seguente applicazione, il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:

A_7 {
- 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]])



Strategia di scheduling EDF

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



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

P_2 subisce il massimo tempo di blocco $B_2 = 3$ t.u. se $a_2 = a_3 + 1 + \varepsilon$,

P_1 subisce il massimo tempo di blocco $B_1 = 4$ 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.

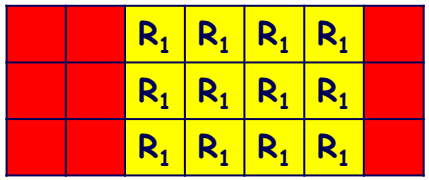
Ad un processo è negato l'accesso ad una risorsa solo se essa è già occupata. In tal caso la priorità corrente del processo bloccato viene ereditata (°) dal processo che detiene la risorsa.

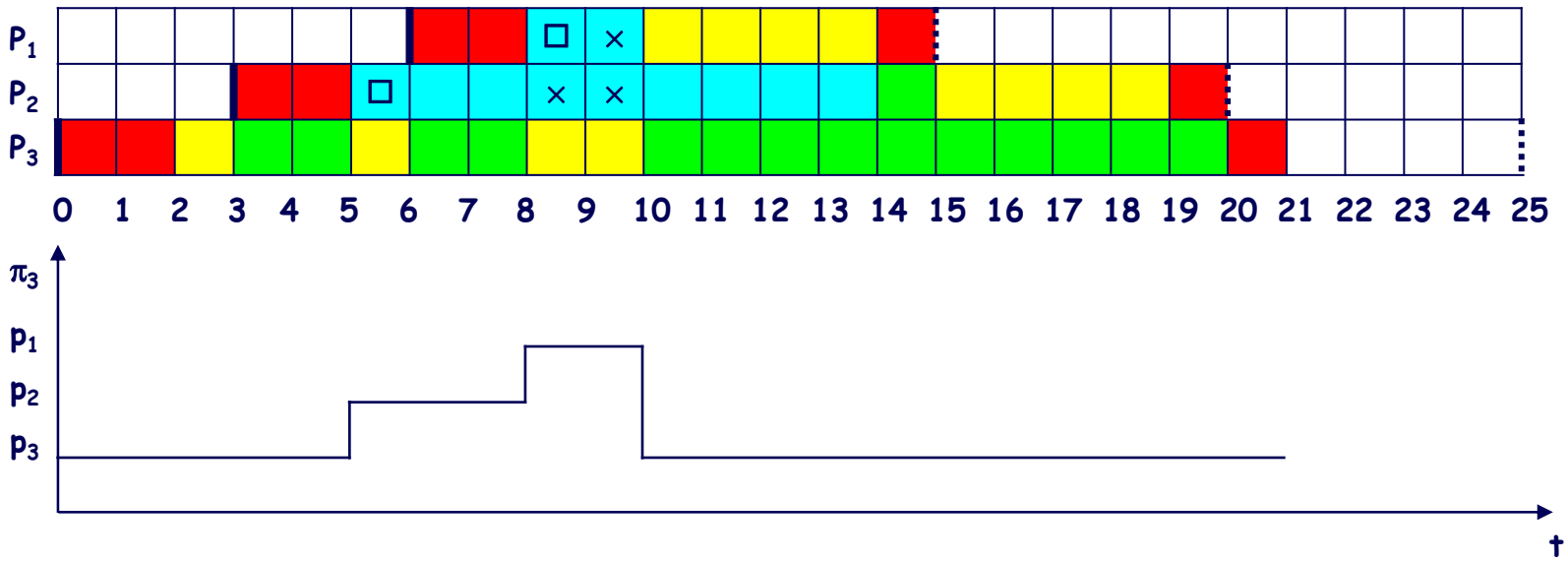
(°) L'ereditarietà della priorità è transitiva: se un processo P_H di priorità p_H (High priority) è bloccato da un processo P_M di priorità p_M (Medium priority) e P_M a sua volta è bloccato da un processo P_L di priorità p_L (Low priority), allora P_L eredita la priorità p_H di P_H via P_M. Tale forma 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.

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

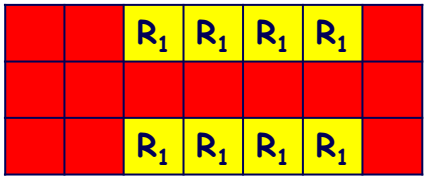


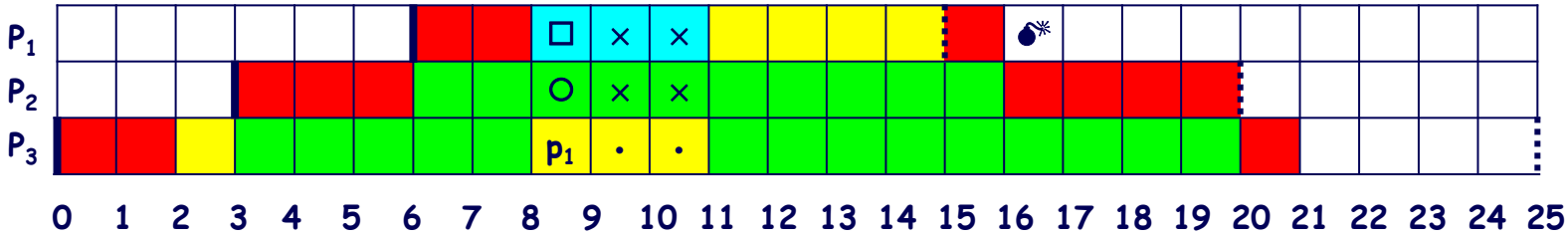


… IL PROTOCOLLO PI …

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





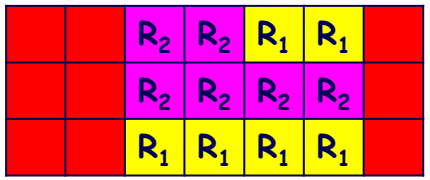
Idle | Ready | Suspended | Running | & Locking S_1

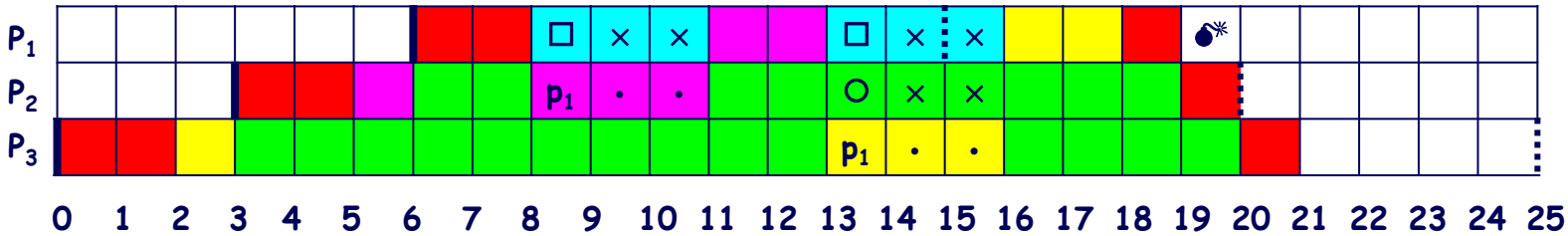
○ Indirect (Push-Through) Blocking (IB)

… IL PROTOCOLLO PI …

Non previene la concatenazione di blocchi.

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



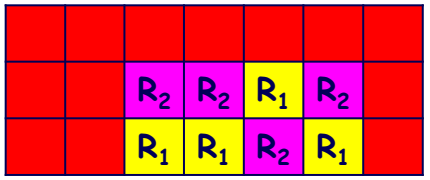


Idle | Ready | Suspended | Running | & Locking S_1 | & Locking S_2

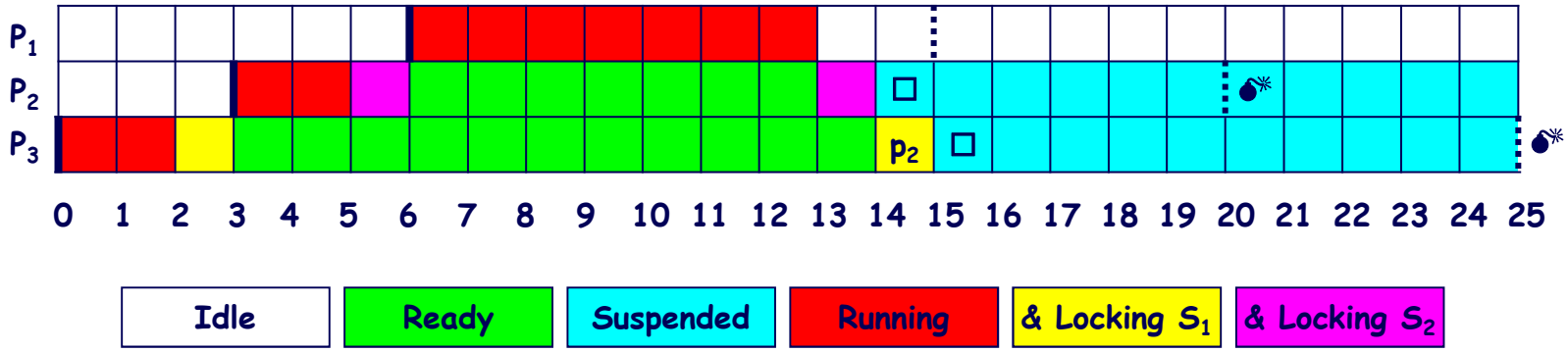
… IL PROTOCOLLO PI …

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 ($k = 1, \ldots, M$), coincidente con la massima fra le priorità nominali dei processi che ad essa possono accedere,

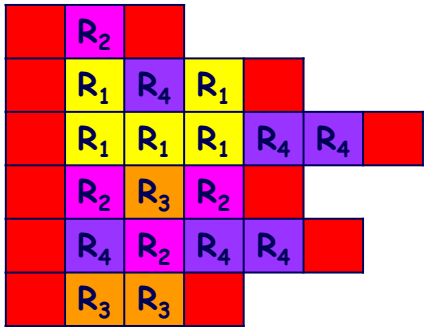
il numero massimo di blocchi, diretti o indiretti, in cui può incorrere il processo P_i ($i = 1, \ldots, N$) di priorità nominale p_i è dato da:

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

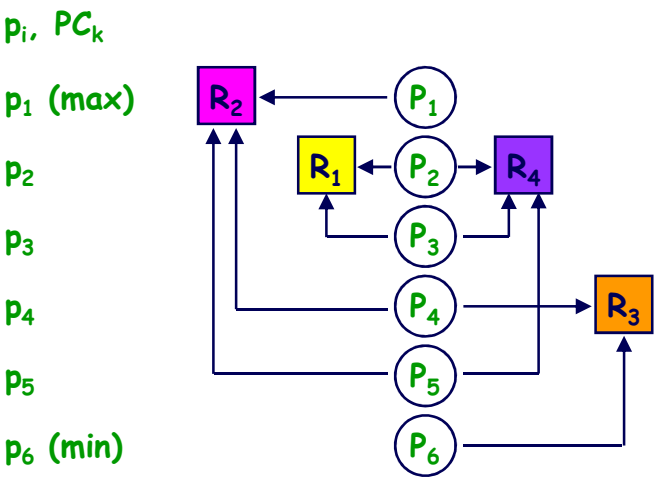
essendo l_i il numero di processi di priorità nominale $< p_i$ che accedono ad almeno una risorsa con tetto di priorità $\geq p_i$, e r_i il numero di (semafori che regolamentano gli accessi a) risorse con tetto di priorità $\geq p_i$ utilizzate da almeno un processo con priorità nominale $< p_i$.

… 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_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).

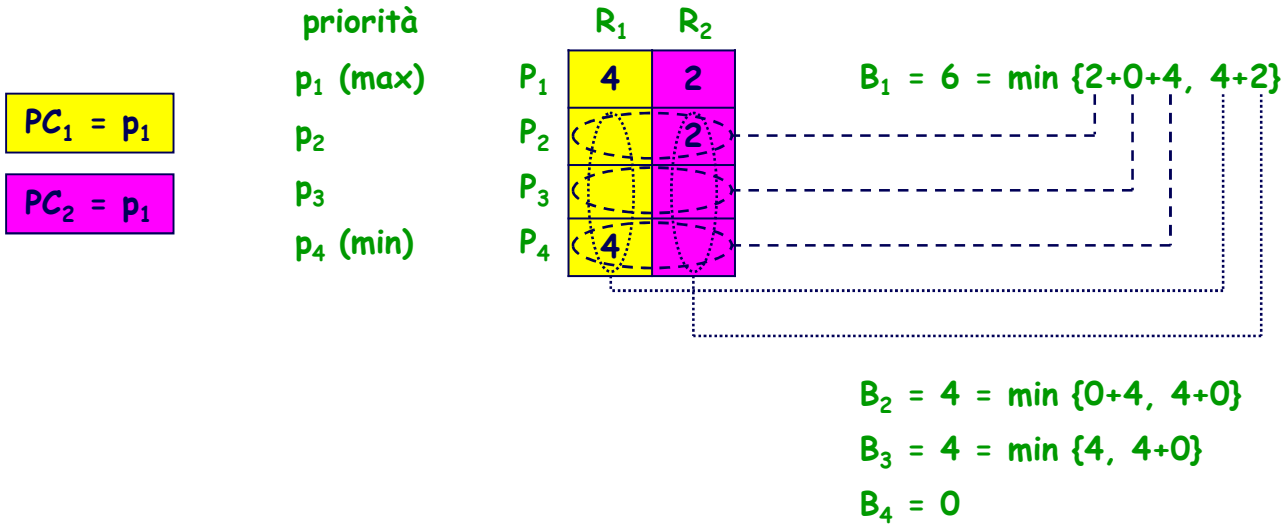
P_6: l_6 = 0, r_6 = 0, n_6 = 0.

… IL PROTOCOLLO PI …

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

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

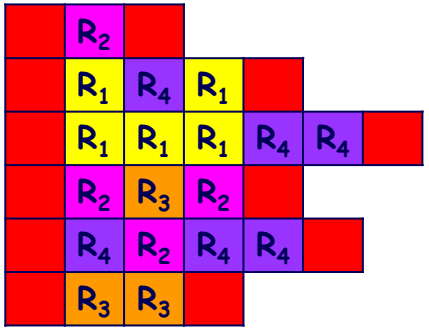
Con riferimento all'applicazione A_5, il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:



… 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])

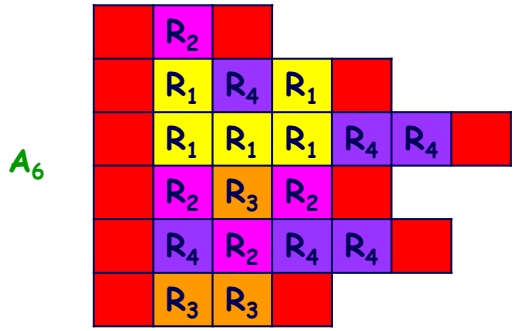


$PC_1 = p_2$
$PC_2 = p_1$
$PC_3 = p_4$
$PC_4 = p_2$

priorità		R_1	R_2	R_3	R_4
p_1 (max)	P_1		1		
p_2	P_2	1			1
p_3	P_3	3			2
p_4	P_4		1	1	
p_5	P_5		1		2
p_6 (min)	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:

$a_6 = \forall,$

$a_5 = a_6 + 1 + \varepsilon,$

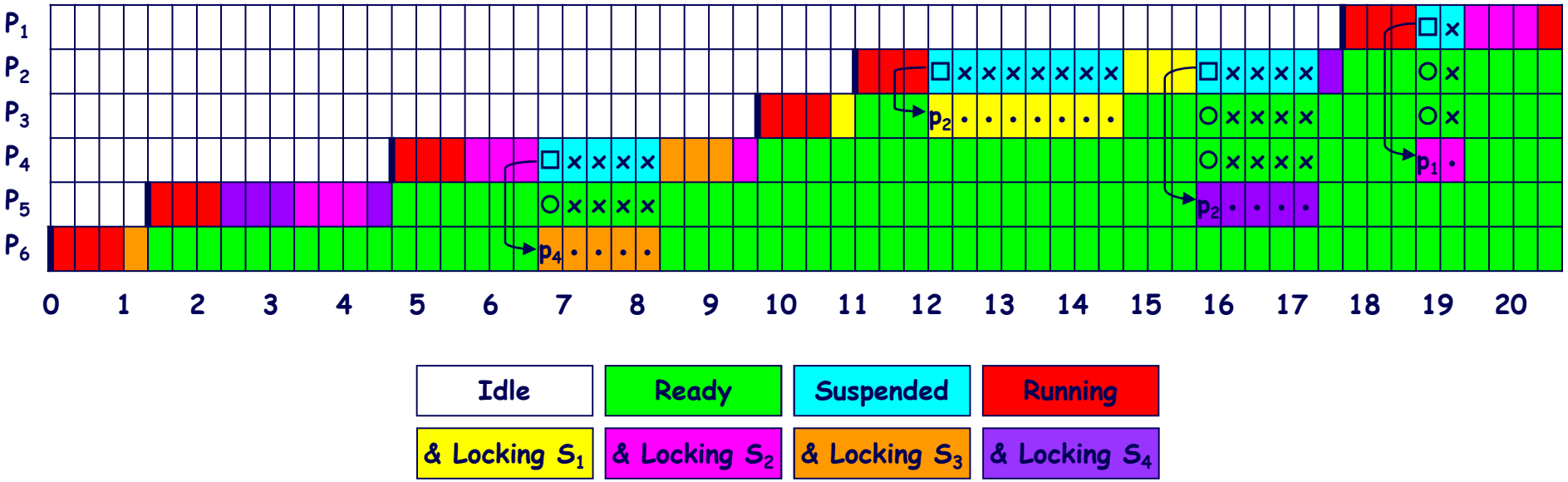
$a_4 = a_5 + 3 + \varepsilon,$

$a_3 = a_4 + 5,$

$a_2 = a_3 + 1 + \varepsilon,$

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

con $\varepsilon \to 0$.



… IL PROTOCOLLO PI …

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

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

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		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

$B_1 = 9 = \min \{5+3+1,\ 4+5\}$ ☹

$B_2 = 4 = \min \{3+1,\ 2+3\}$ ☹

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

$B_4 = 0$

… 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 simpleton 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 …

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:

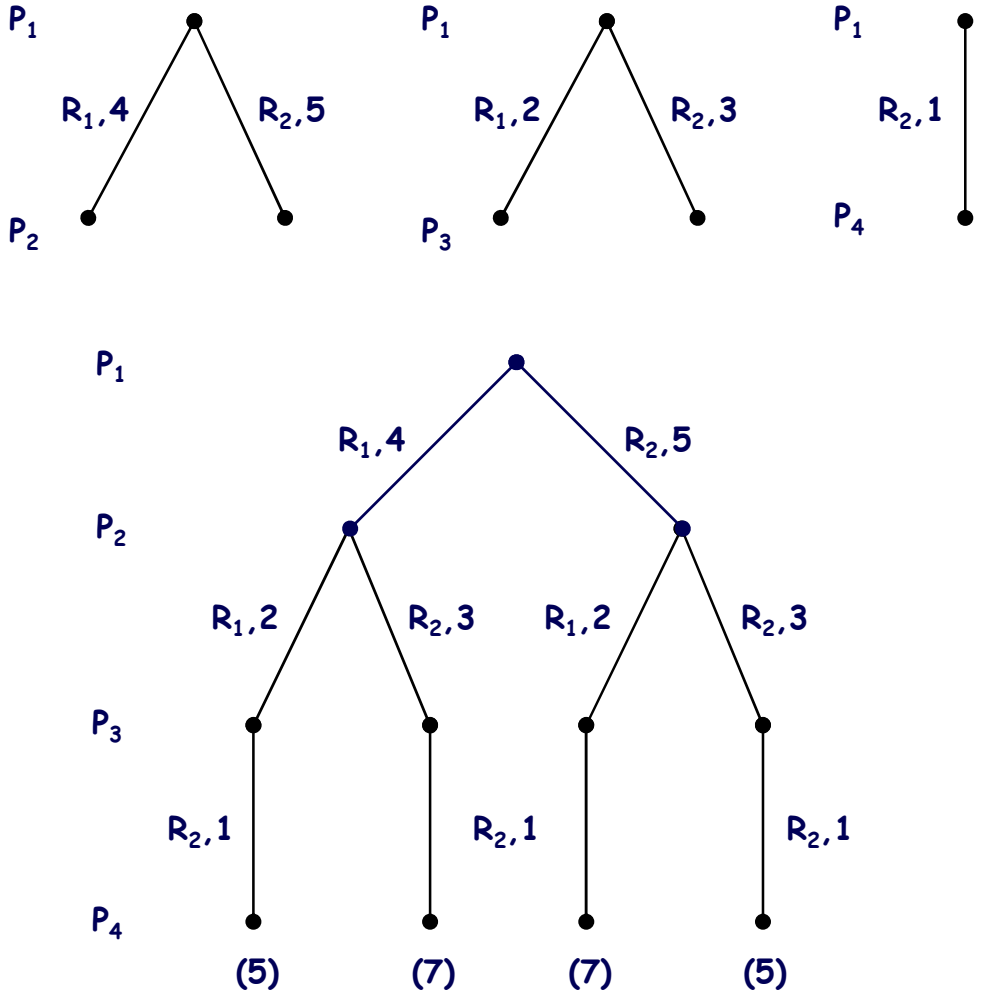
$B_1 = 7$

$B_2 = 3$

$B_3 = 1$

$B_4 = 0$

Calcolo di B_1



… IL PROTOCOLLO PI …

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

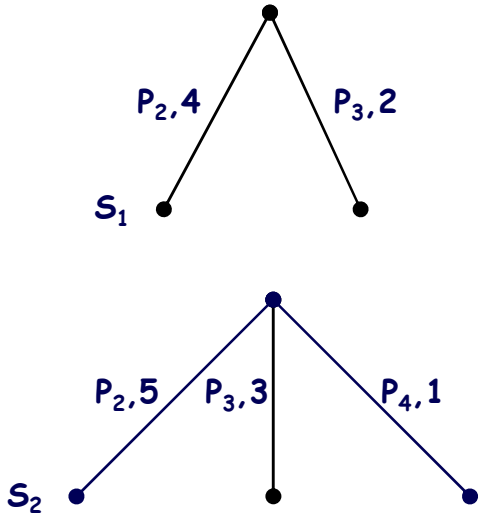
I simpleton sono associati ai semafori che possono bloccare P_i. I rami di un simpleton rappresentano i processi che accedono al corrispondente semaforo. Il peso del ramo corrispondente al processo P_j nell'albero associato al semaforo S_k coincide con il tempo di esecuzione Z_{jk} della più lunga sezione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

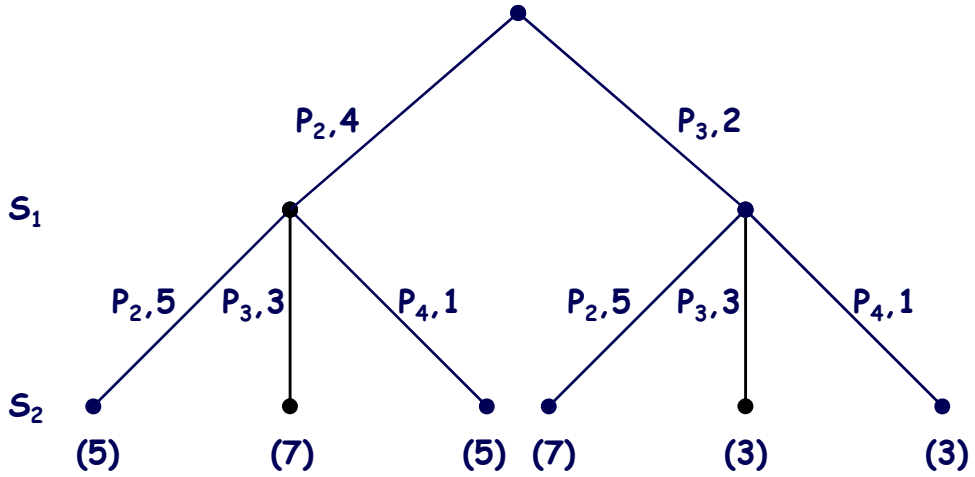
La costruzione dell'albero finale, così come l'attribuzione di un peso ad ogni percorso che conduce dalla radice ad una foglia, procede in maniera analoga. In particolare, se in un percorso sono presenti più rami corrispondenti ad uno stesso processo, si considera soltanto il contributo del ramo di peso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

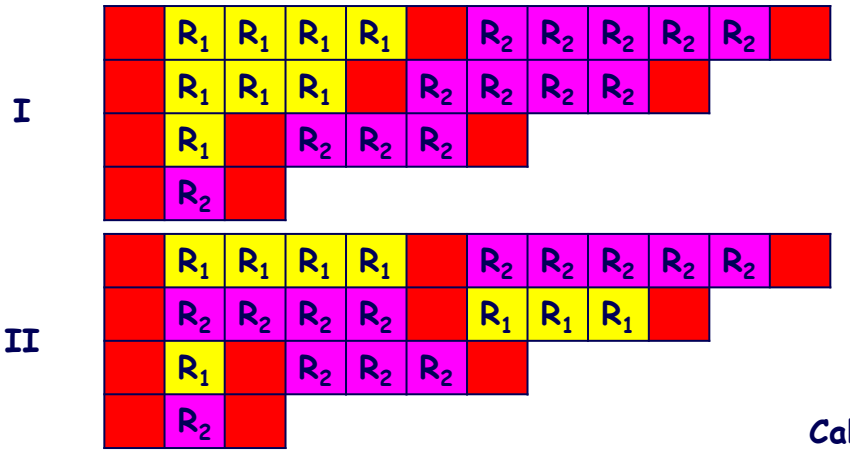
$B_1 = 7$





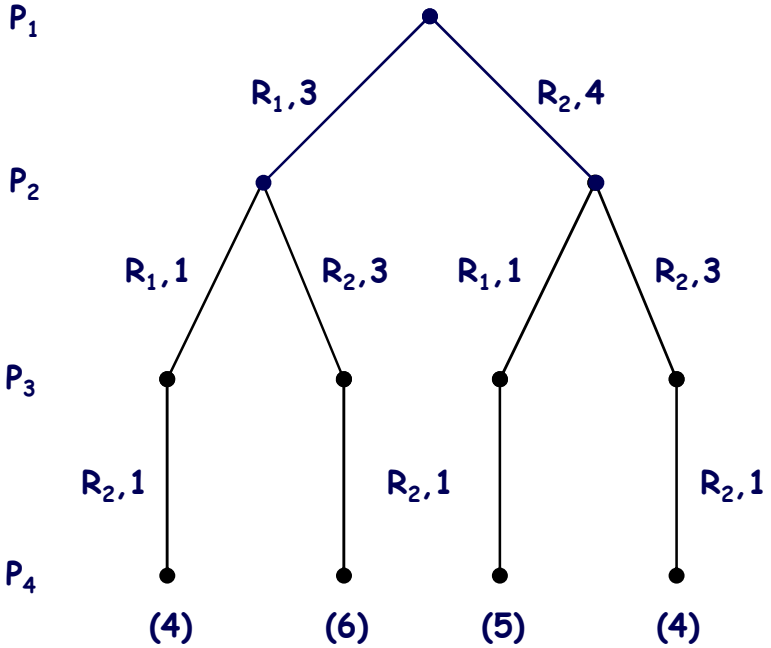
… IL PROTOCOLLO PI …

Rajkumar: still an upper bound

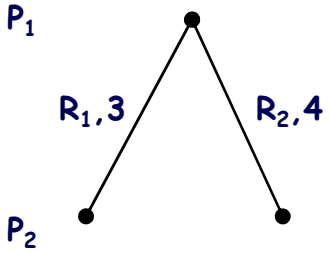


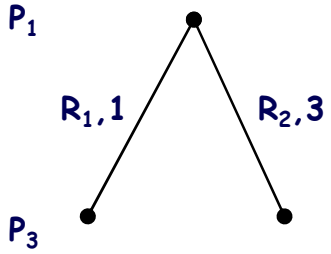
priorità		R_1	R_2
p_1 (max)	P_1	4	5
p_2	P_2	3	4
p_3	P_3	1	3
p_4 (min)	P_4		1

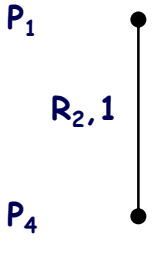
Calcolo di B_1



$B_1 = \min\{4+3+1,\ 3+4\} = 7$ ☹







I ☺ $B_1 = 6$

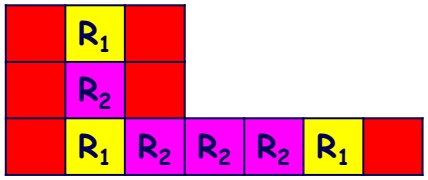
II 😐 $B_1 = 5$

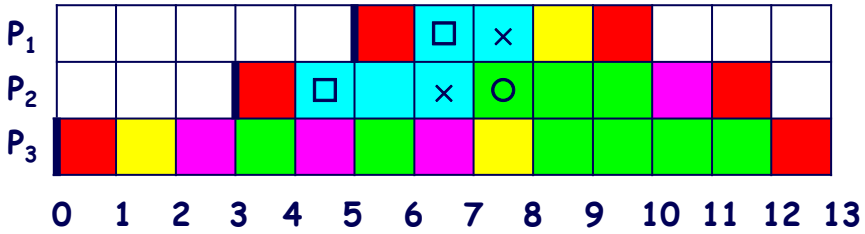
… IL PROTOCOLLO PI …

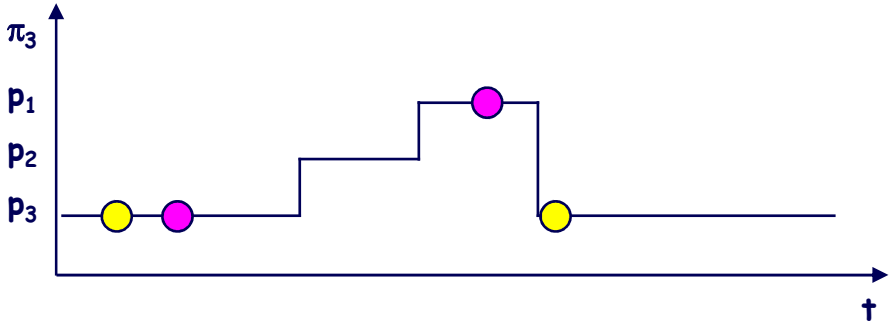
(2) Sezioni critiche annidate

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.

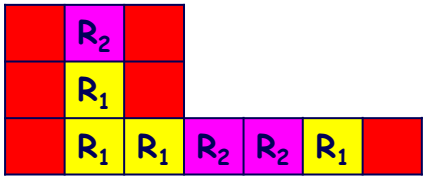
	a [t.u.]	C [t.u.]
P_1	5	3
P_2	3	3
P_3	0	7

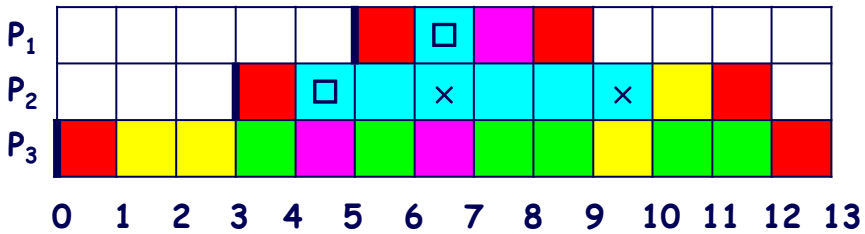


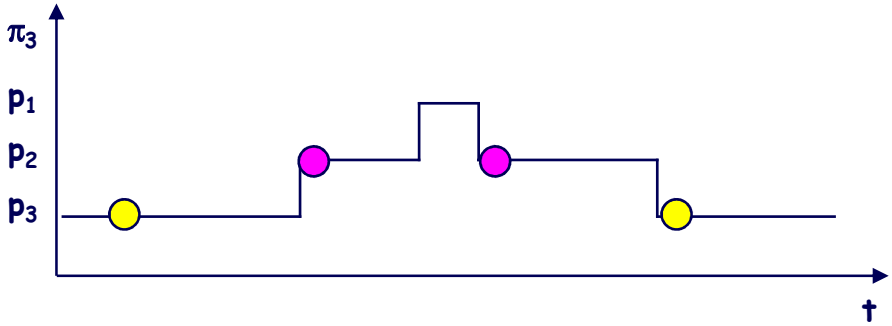




	a [t.u.]	C [t.u.]
P_1	5	3
P_2	3	3
P_3	0	7



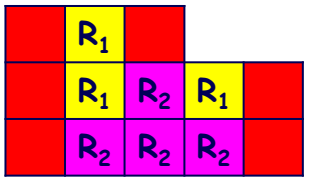


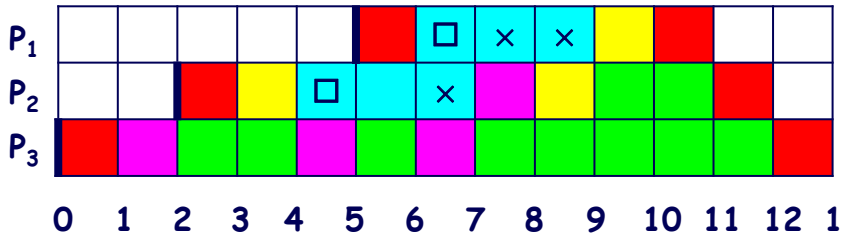


… IL PROTOCOLLO PI …

L'ereditarietà della priorità è transitiva.

	a [t.u.]	C [t.u.]
P_1	5	3
P_2	2	5
P_3	0	5

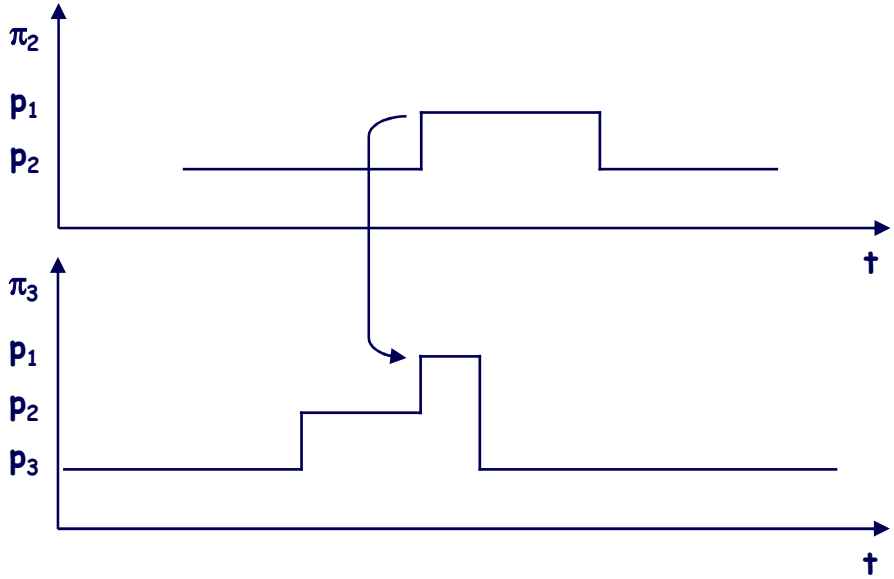




$PC_1 = p_1$

$PC_2 = p_2$

	R_1	R_2
P_1	1	
P_2	3	1
P_3		3



"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 P_1 subisce il massimo tempo di blocco $B_1 = 6$ t.u. se:

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

… IL PROTOCOLLO PI …

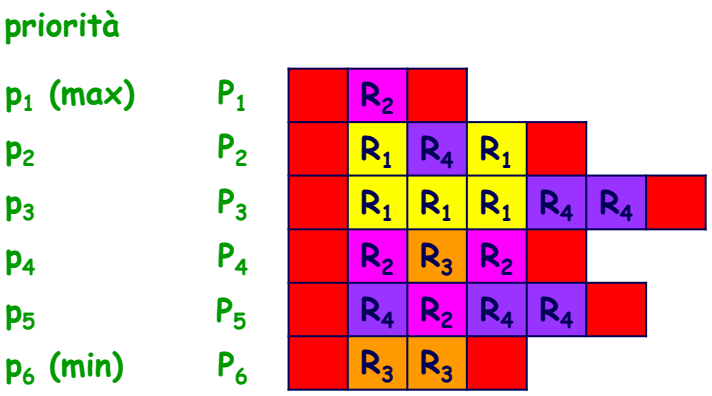
Un processo P_i di priorità p_i può essere bloccato da un processo P_j di priorità $p_j < p_i$ se P_j condivide una risorsa con un processo che ha o può ereditare una priorità $\geq p_i$.

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 ($i = 1, \ldots, N$) può essere identificato con il metodo di Rajkumar.

Con riferimento all'applicazione A_6'



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

$B_1 = 5$

$B_2 = 12$

$B_3 = 9$

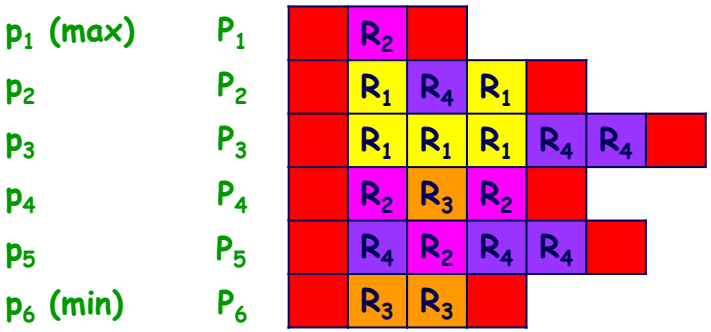
$B_4 = 6$

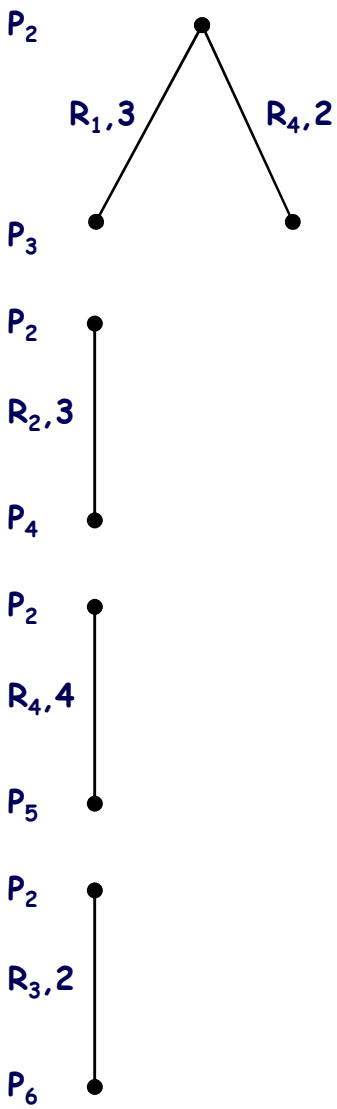
$B_5 = 2$

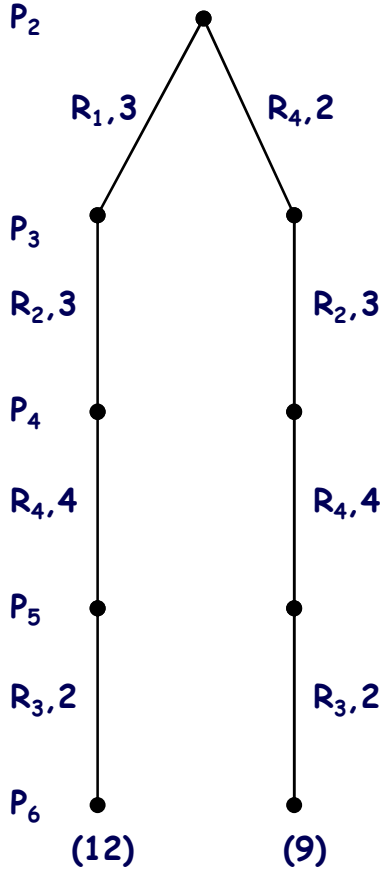
$B_6 = 0$

… IL PROTOCOLLO PI …

Applicazione A_6': calcolo di B_2



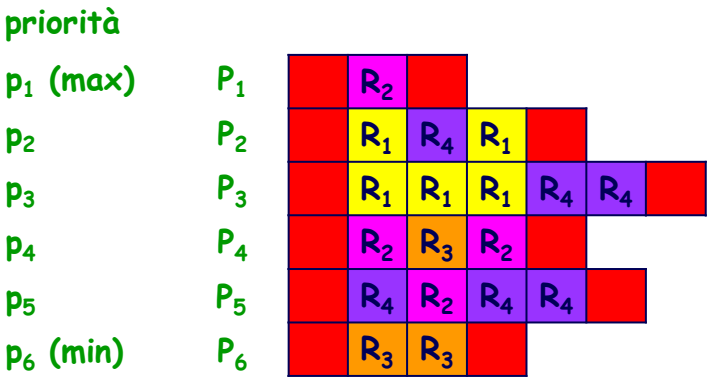


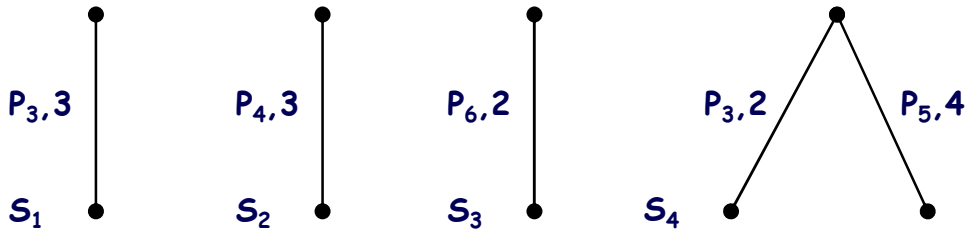


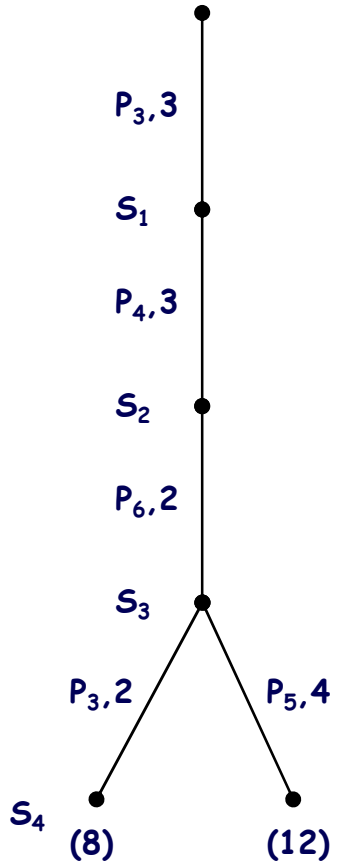
B_2 = 12 [t.u.]

… IL PROTOCOLLO PI …

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

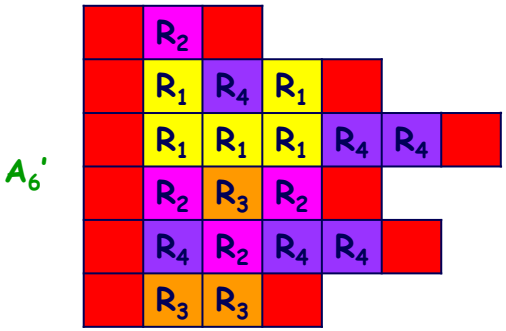






B_2 = 12 [t.u.]

… IL PROTOCOLLO PI



$B_1 = 5$

$B_2 = 12$

$B_3 = 9$

$B_4 = 6$

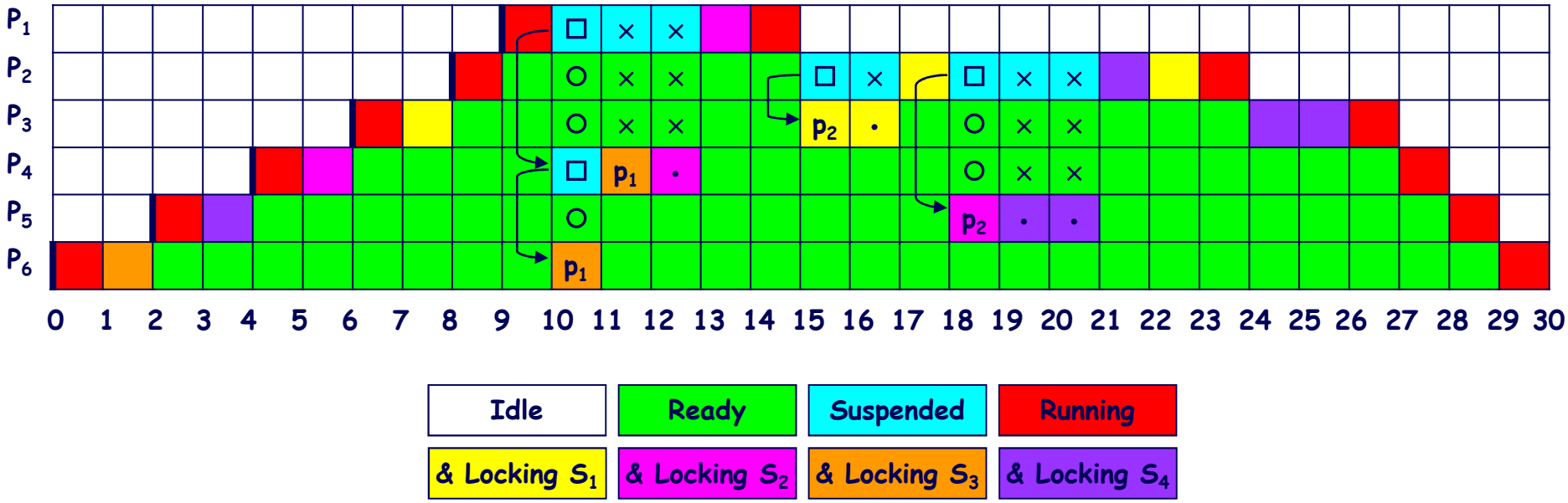
$B_5 = 2$

$B_6 = 0$

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 $\varepsilon \to 0$.



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

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

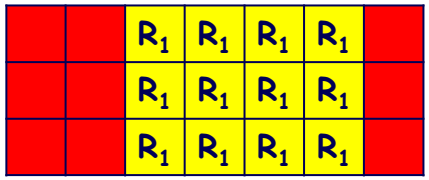
coincidente con il più elevato tetto di priorità delle risorse al momento allocate, a meno che tale processo detenga già la risorsa (le risorse) il cui tetto di priorità coincide con $\Pi\Gamma_S$.

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.

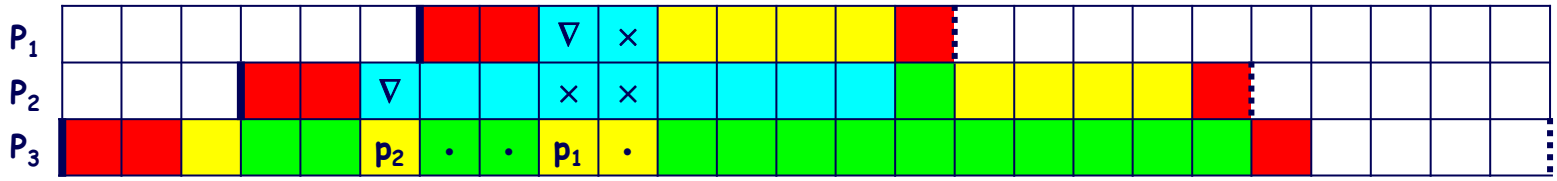
… IL PROTOCOLLO PC …

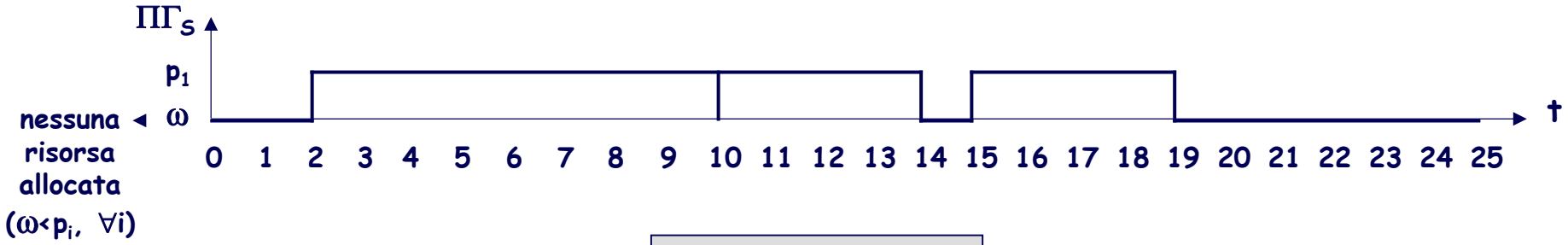
Implica la conoscenza a priori delle risorse utilizzate dai processi (onde poterne determinare il tetto di priorità).

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



$PC_1 = p_1$



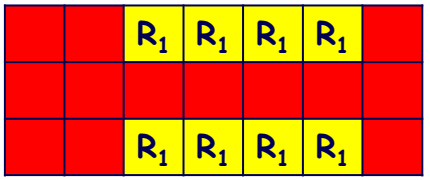


∇ Ceiling Blocking (CB)

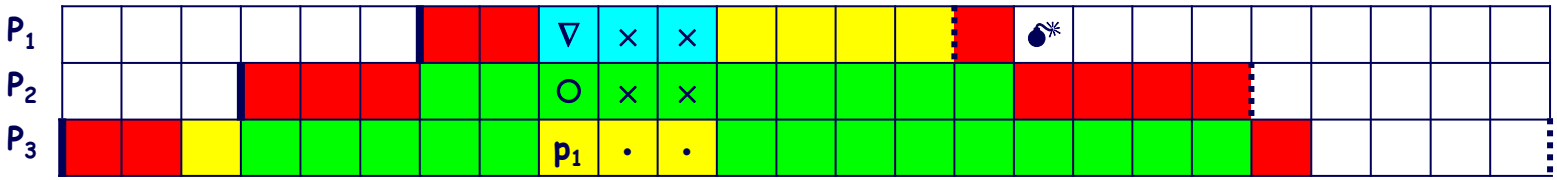
… IL PROTOCOLLO PC …

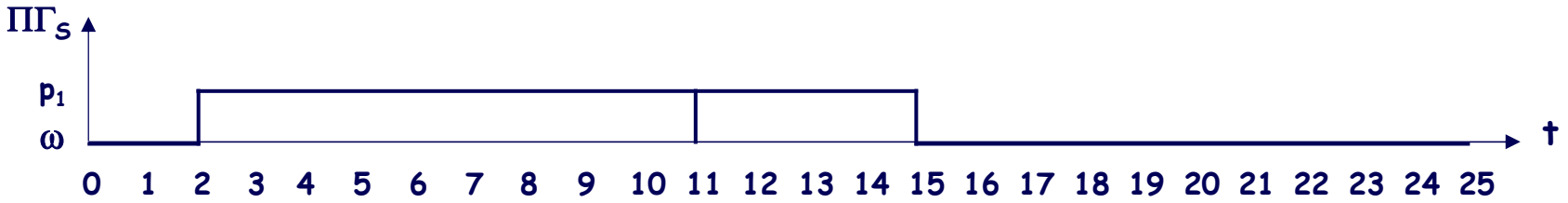
Previene inversioni di priorità incontrollate (grazie al meccanismo della ereditarietà della priorità).

		priorità	a [t.u.]	d [t.u.]	C [t.u.]
	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



$PC_1 = p_1$



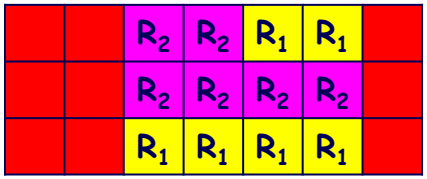


… IL PROTOCOLLO PC …

Previene la concatenazione di blocchi

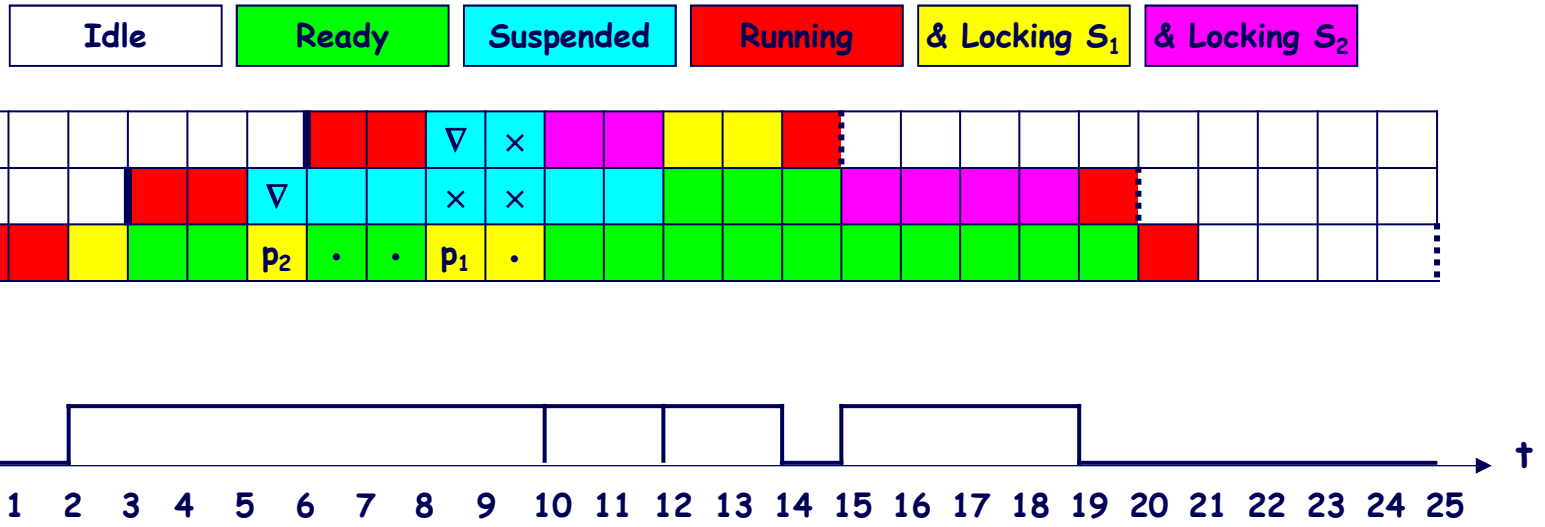
(un processo P_i può essere bloccato solo all'atto dell'accesso alla prima risorsa da esso utilizzata, e ciò se contestualmente risulta $p_i \leq \Pi\Gamma_S$. Ottenuto l'accesso allorché $p_i > \Pi\Gamma_S$, P_i può completare l'esecuzione senza subire ulteriori blocchi, poiché tutte le risorse di cui 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.]
	P_1	p_1 (max)	6	15	7
A_3	P_2	p_2	3	20	7
	P_3	p_3 (min)	0	25	7



$PC_1 = p_1$

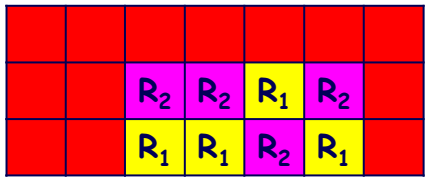
$PC_2 = p_1$



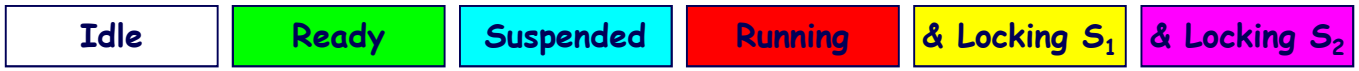
… IL PROTOCOLLO PC …

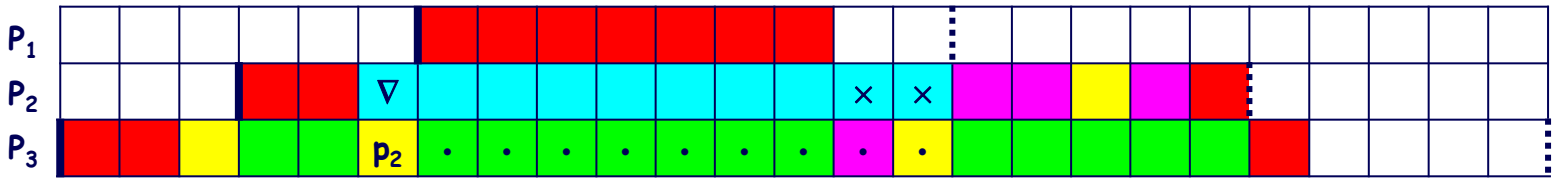
Previene situazioni di deadlock.

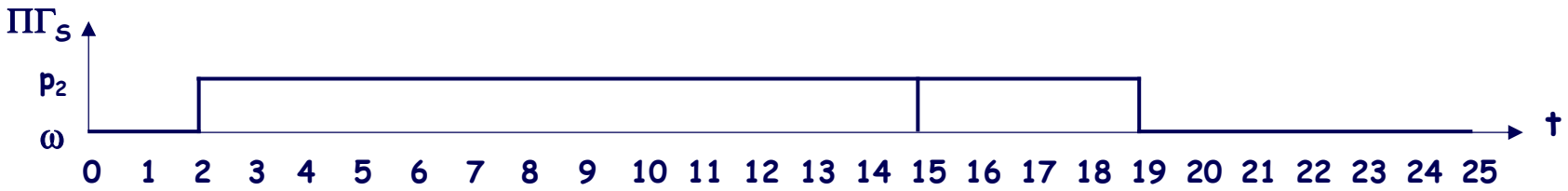
		priorità	a [t.u.]	d [t.u.]	C [t.u.]
	P_1	p_1 (max)	6	15	7
A_4	P_2	p_2	3	20	7
	P_3	p_3 (min)	0	25	7



!? $PC_1 = p_2$ $PC_2 = p_2$







… 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 $\geq p_i$:

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

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		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	

$B_1 = 4 = \max \{2, 0, 4\}$

$B_2 = 4 = \max \{0, 4\}$

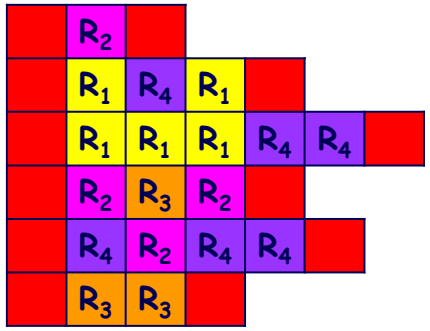
$B_3 = 4$

$B_4 = 0$

… 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])

$PC_1 = p_2$
$PC_2 = p_1$
$PC_3 = p_4$
$PC_4 = p_2$

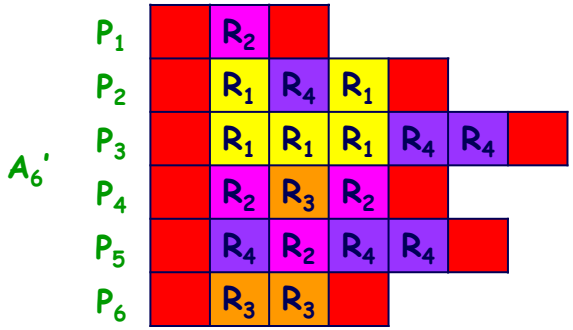
priorità		R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
p_1 (max)	P_1		1				1		
p_2	P_2	1			1	3			1
p_3	P_3	3			2	3			2
p_4	P_4		1	1			3	1	
p_5	P_5		1		2		1		4
p_6 (min)	P_6			2				2	
			A_6				A_6'		

B_1 = 1 / 3
B_2 = 3 / 4
B_3 = 2 / 4
B_4 = 2 / 4
B_5 = 2 / 2
B_6 = 0 / 0

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



$PC_1 = p_2$

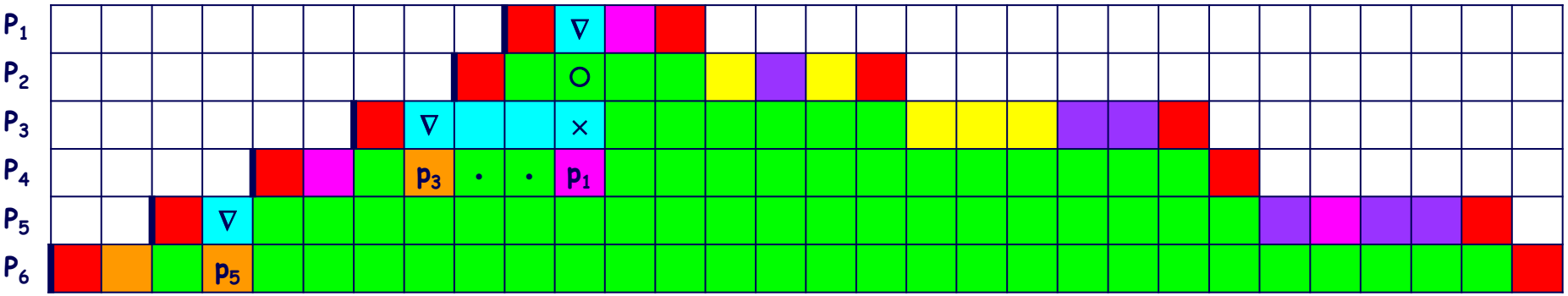
$PC_2 = p_1$

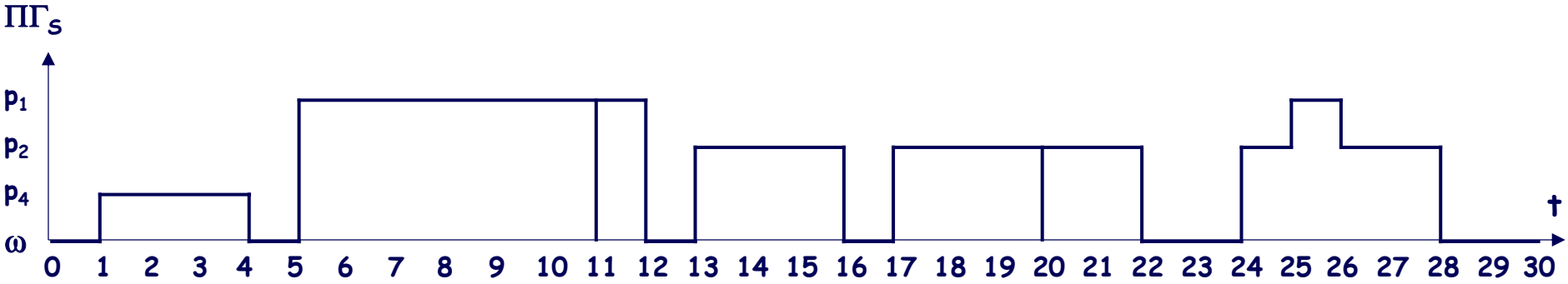
$PC_3 = p_4$

$PC_4 = p_2$

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.

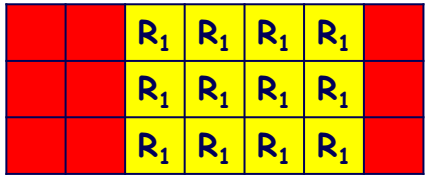
Una risorsa viene sempre allocata al processo che ne fa richiesta.

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 $p_k > p_i$.

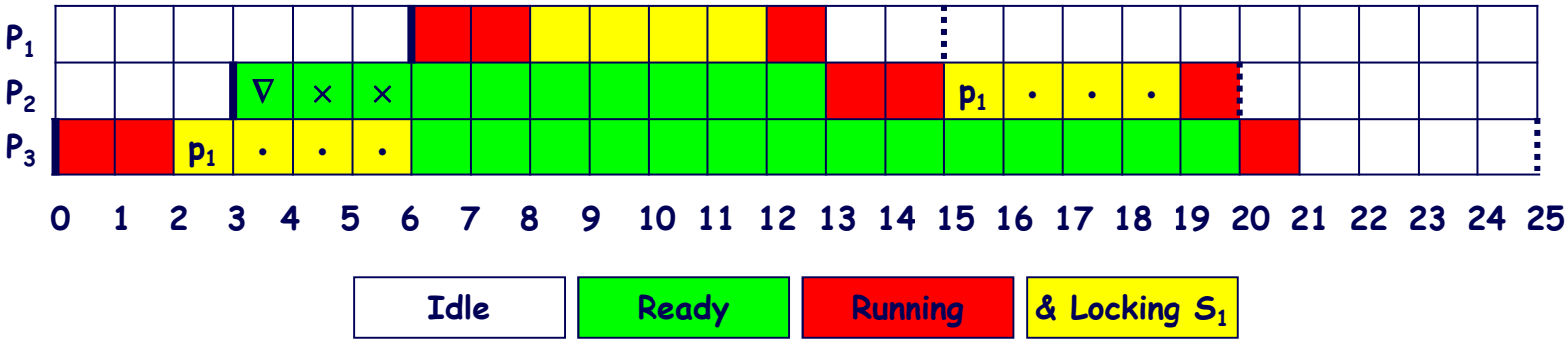
… IL PROTOCOLLO IPC …

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
	P_2	p_2	3	20	7
	P_3	p_3 (min)	0	25	7



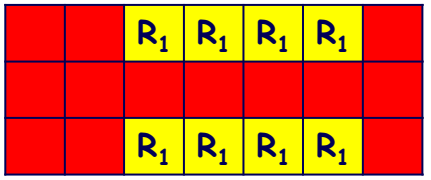
$PC_1 = p_1$



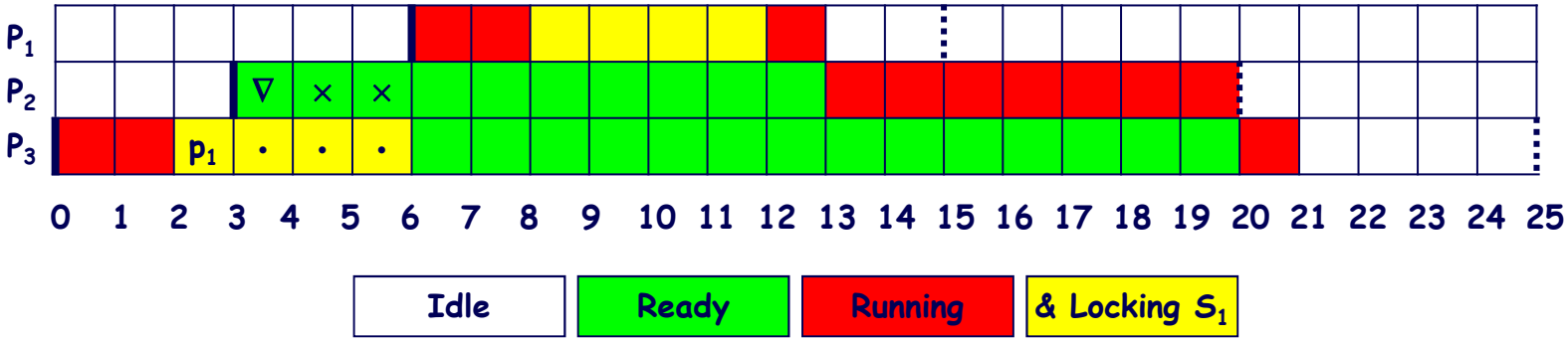
… IL PROTOCOLLO IPC …

Previene inversioni di priorità incontrollate.

		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

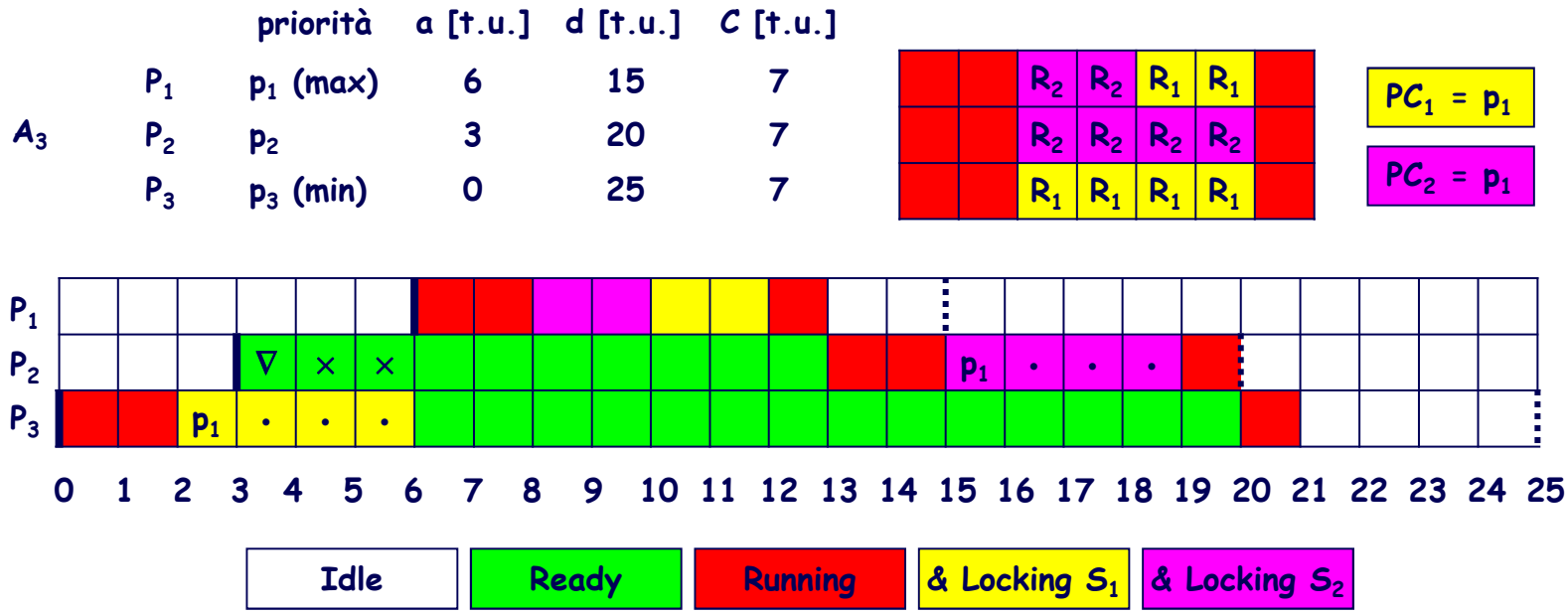


$PC_1 = p_1$



… IL PROTOCOLLO IPC …

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



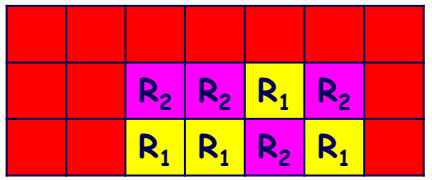
Come nel caso del protocollo PC, il massimo tempo di blocco B_i di un processo P_i di priorità p_i è uguale al tempo di esecuzione Z_{jk} della più lunga sezione critica relativa a qualunque risorsa R_k condivisa da un processo di priorità $p_j < p_i$ e da un processo di priorità $\geq p_i$:

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

… IL PROTOCOLLO IPC …

Previene situazioni di deadlock.

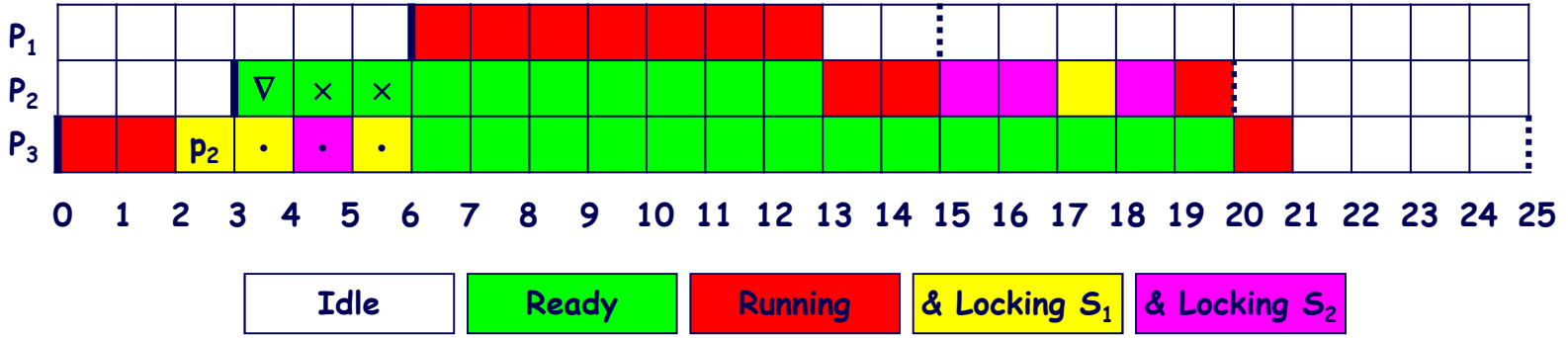
		priorità	a [t.u.]	d [t.u.]	C [t.u.]
	P_1	p_1 (max)	6	15	7
A_4	P_2	p_2	3	20	7
	P_3	p_3 (min)	0	25	7



!?

$PC_1 = p_2$

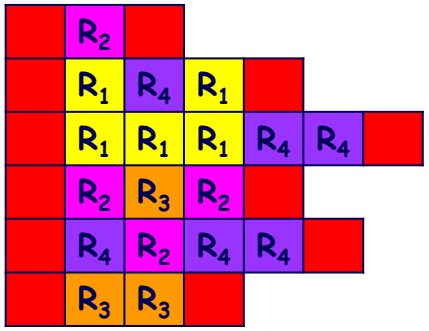
$PC_2 = p_2$



… 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

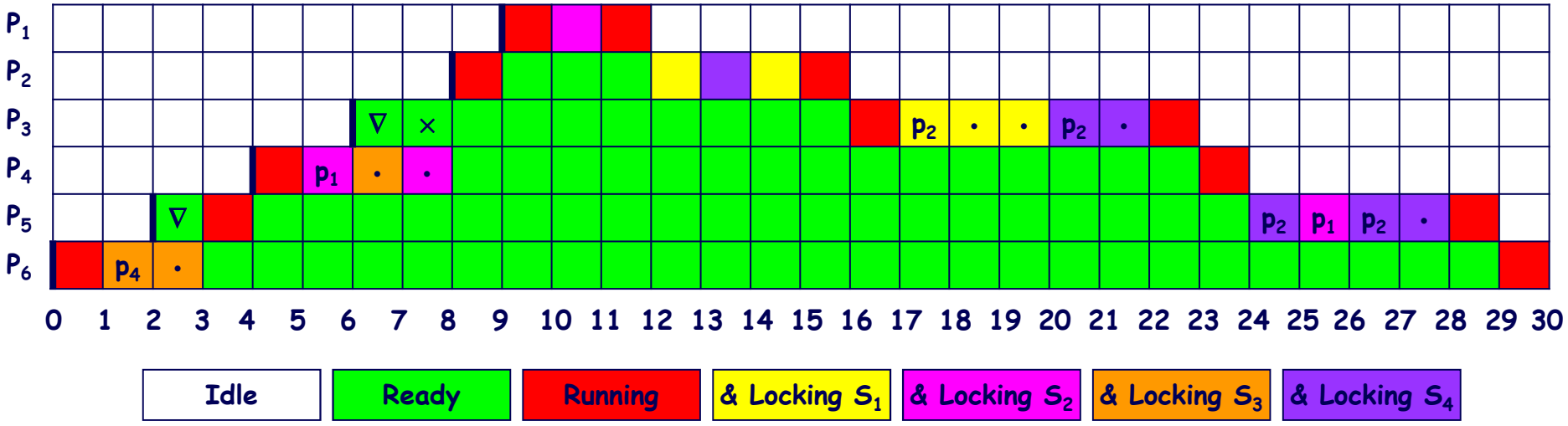
p_6 (min)

$PC_1 = p_2$

$PC_2 = p_1$

$PC_3 = p_4$

$PC_4 = p_2$



Il tempo di risposta di P_1 e P_2 migliora rispetto all'applicazione del protocollo PC (13 → 12, 17 → 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:

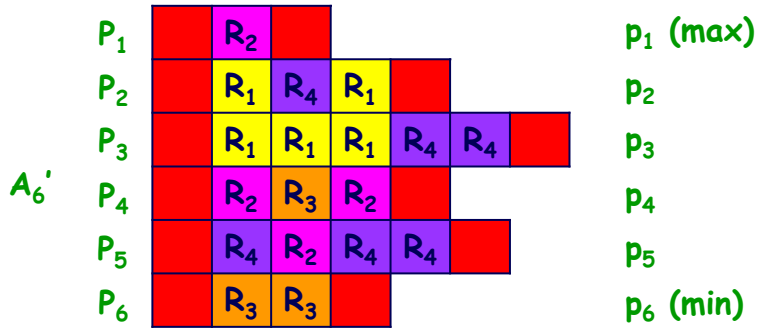
Il tetto di priorità del sistema $\Pi\Gamma_S$ è aggiornato ogni qual volta viene allocata o rilasciata una risorsa.

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 …

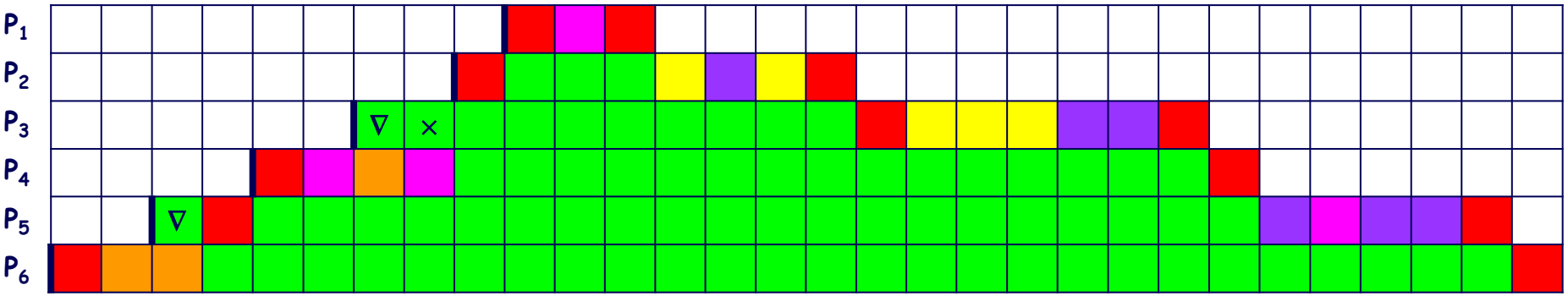


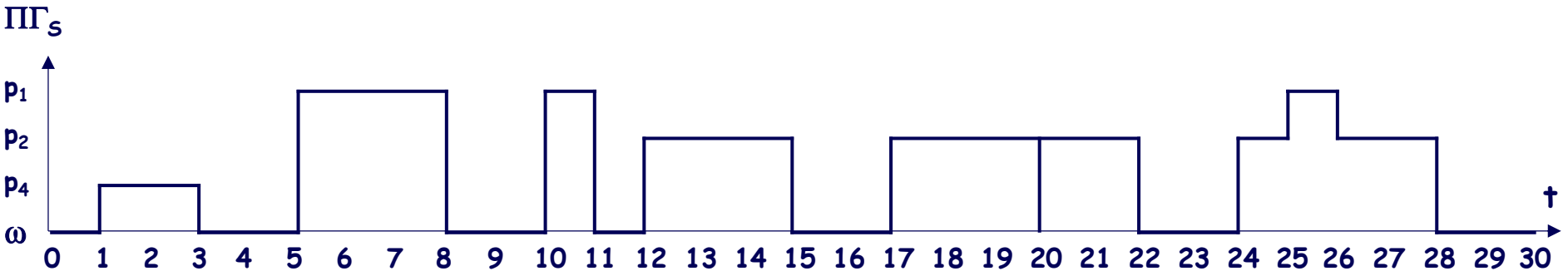
p1 (max)
p2
p3
p4
p5
p6 (min)

$PC_1 = p_2$ $PC_2 = p_1$ $PC_3 = p_4$ $PC_4 = p_2$

Idle Ready Running

& Locking S_1 & Locking S_2 & Locking S_3 & Locking S_4





… 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;

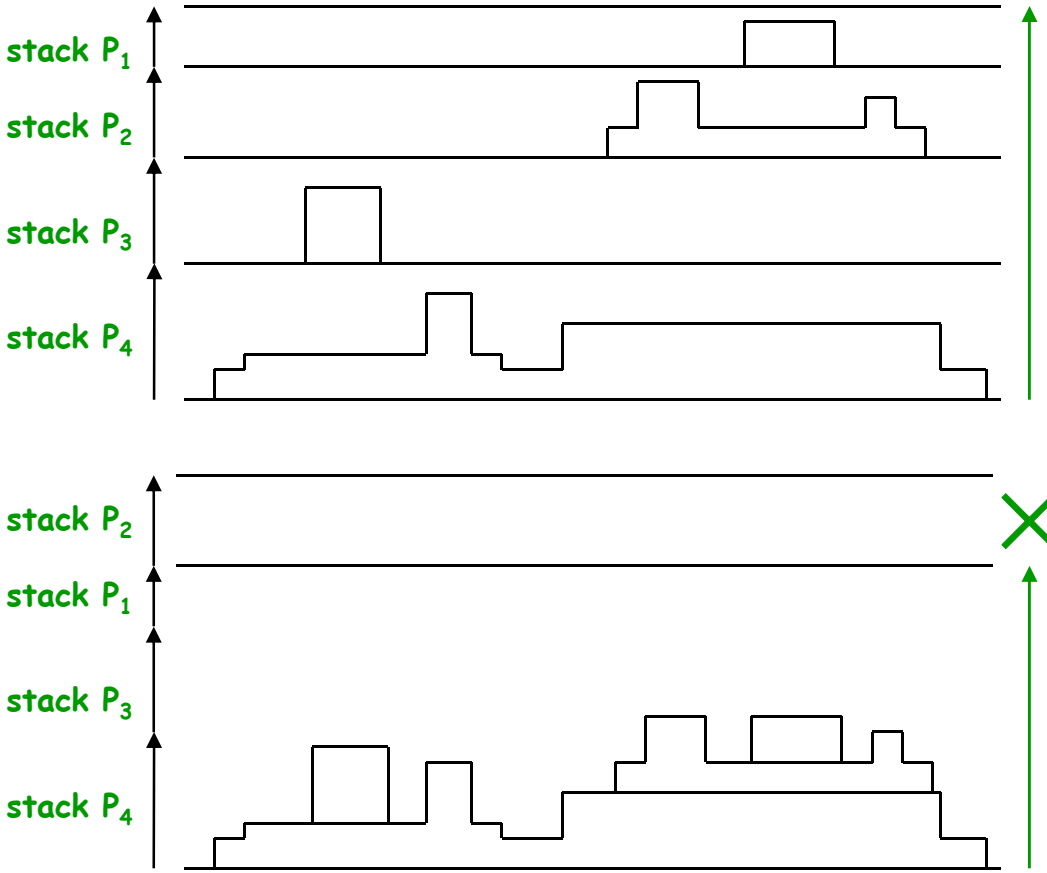
☺ semplicità realizzativa;

☺ 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 ($pl_i > pl_j$).

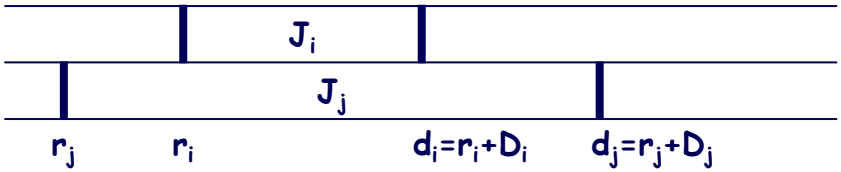
  Ricadono in questa categoria i sistemi che operano la schedulazione dei processi in accordo alla strategia EDF, con livelli di preemption definiti come segue:

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

… IL PROTOCOLLO PREEMPTION CEILING …

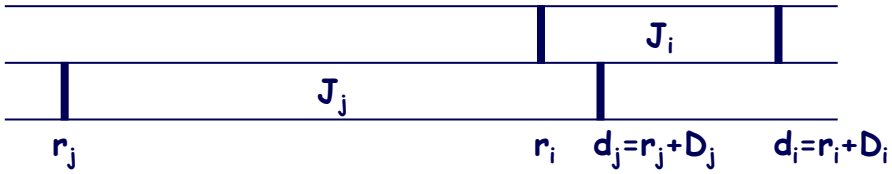
La condivisione di risorse tra due processi P_i e P_j aventi deadline relativa D_i e D_j, rispettivamente, non implica, qualora siano schedulati in accordo alla strategia EDF, che ciascuno di essi incorra, allorché in esecuzione con priorità massima, in una situazione di blocco imputabile all'altro.

$D_j > D_i \rightarrow pl_j < pl_i$



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

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



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

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:

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

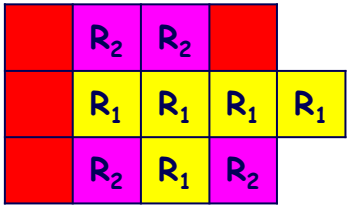
dove PL_k rappresenta il "preemption ceiling" associato staticamente alla risorsa R_k, coincidente con il massimo livello di preemption dei processi che ad essa possono accedere:

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

- 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 {
- 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]])
}



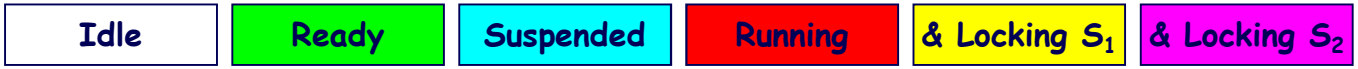
pl_1 (max)

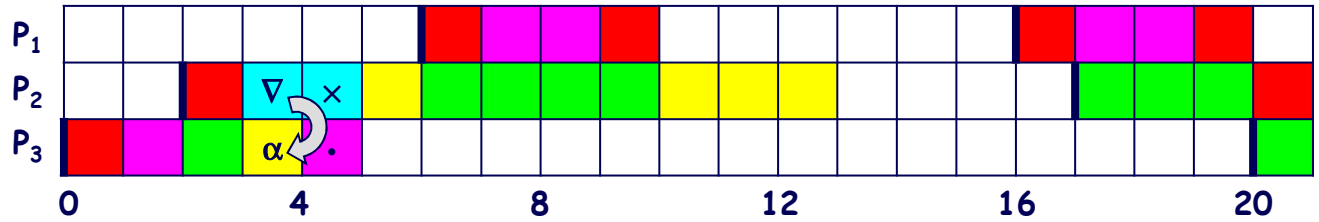
pl_2

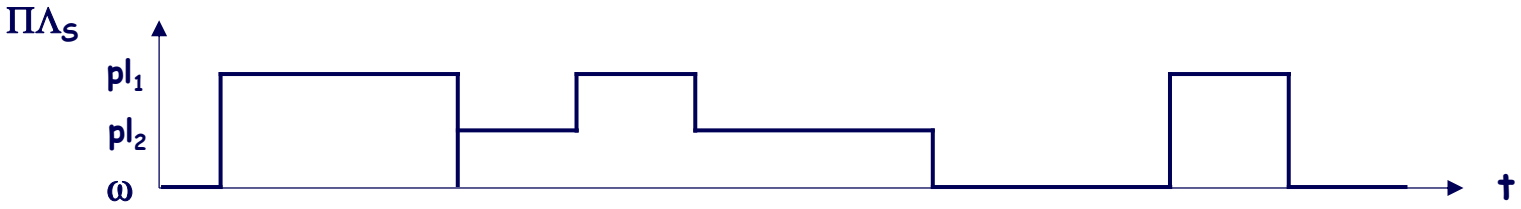
pl_3 (min)

$PL_1 = pl_2$

$PL_2 = pl_1$







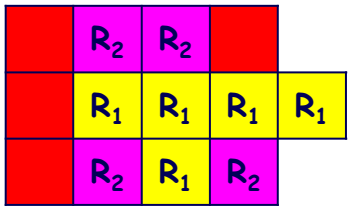
… IL PROTOCOLLO PREEMPTION CEILING …

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 $PL_k > pl_j$:

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

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

A_7 {
- 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]])
}



pl_1 (max)
pl_2
pl_3 (min)

$PL_1 = pl_2$

$PL_2 = pl_1$

	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 …

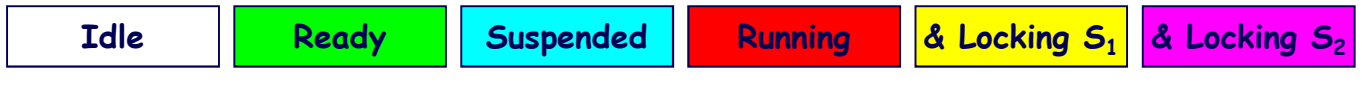
Con riferimento all'applicazione A_8, il massimo tempo di blocco ([t.u.]) di ciascun processo risulta:

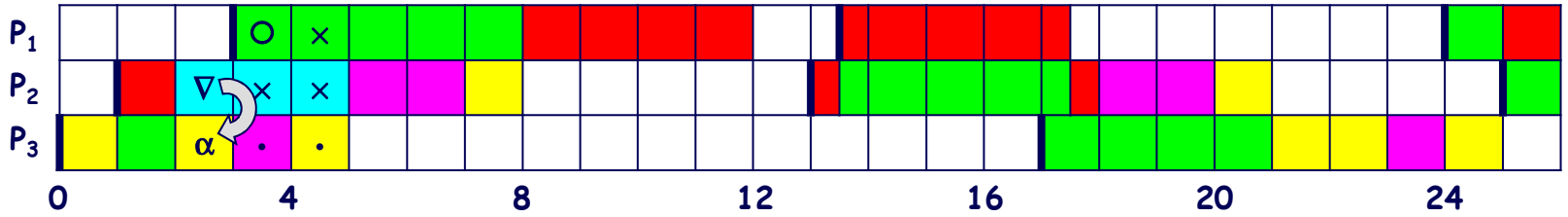
A_8 {
- P_1 (3, 10.5, 4, 10.5)
- P_2 (1, 12, 4, 12; [R_2;2] [R_1;1])
- P_3 (0, 17, 4, 17; [R_1;4 [R_2;1]])
}

pl_1 (max)
pl_2
pl_3 (min)

$PL_1 = pl_2$
$PL_2 = pl_2$

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



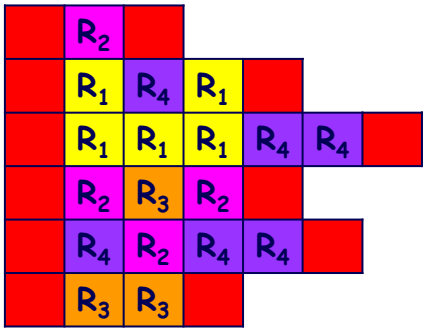


P_1 può essere bloccato da P_3 (in maniera indiretta) ma non da P_2, in quanto
P_2 può subire preemption solo da P_1 e P_1 non necessita delle risorse eventualmente detenute da P_2.

… 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])

$PL_1 = pl_2$
$PL_2 = pl_1$
$PL_3 = pl_4$
$PL_4 = pl_2$

	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
pl_1 (max) P_1		1				1		
pl_2 P_2	1			1	3			1
pl_3 P_3	3			2	3			2
pl_4 P_4		1	1			3	1	
pl_5 P_5		1		2		1		4
pl_6 (min) 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'

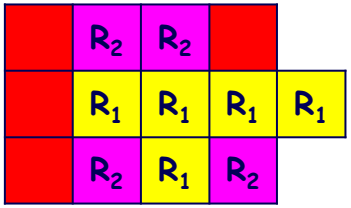
STACK RESOURCE POLICY [BAKER (91)] …

- 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.

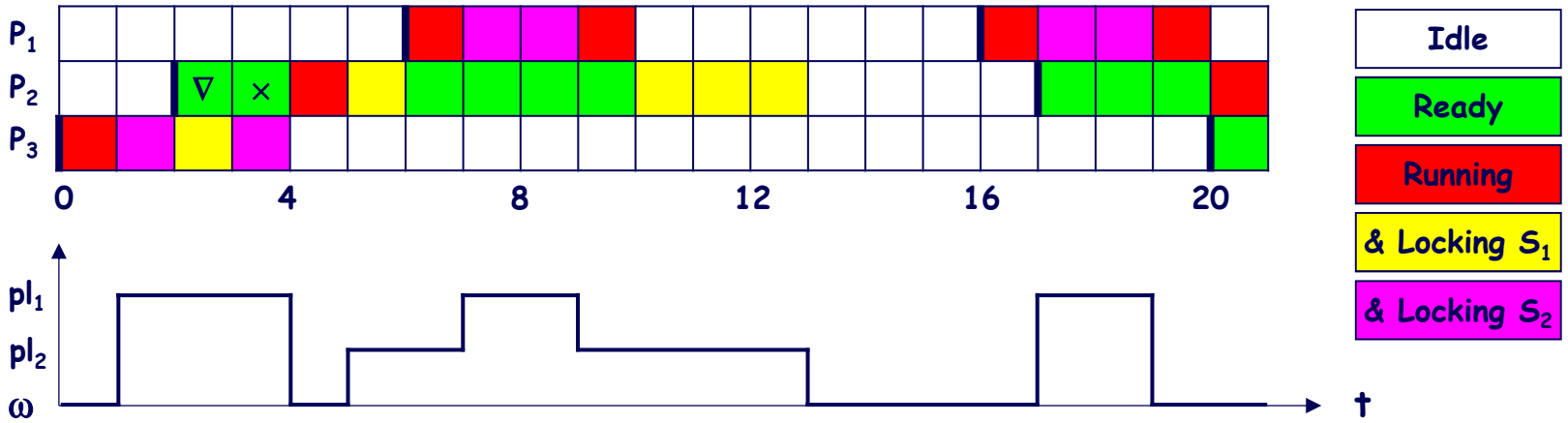
… STACK RESOURCE POLICY …

A_7 { 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]]) }



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:

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

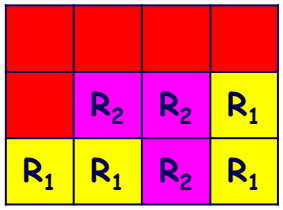
A_7:

	R_1	R_2	
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)
- P_2 (1, 12, 4, 12; [R_2;2] [R_1;1])
- P_3 (0, 17, 4, 17; [R_1;4 [R_2;1]])
}

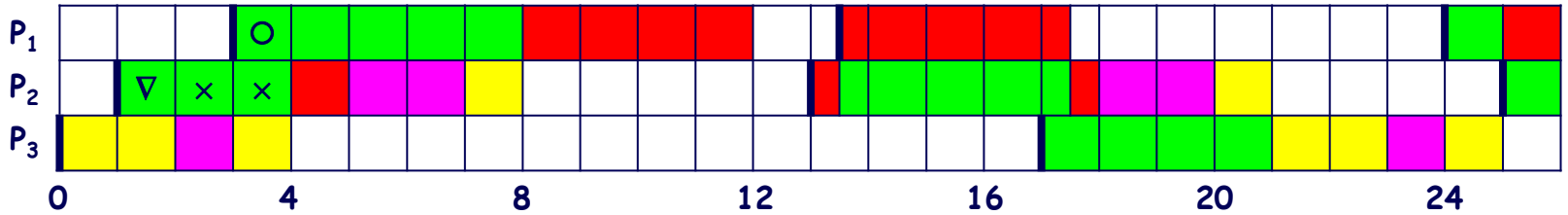


pl_1 (max)
pl_2
pl_3 (min)

$PL_1 = pl_2$
$PL_2 = pl_2$

	R_1	R_2
P_1		
P_2	1	2
P_3	4	1

$B_1 = 4$
$B_2 = 4$
$B_3 = 0$



… 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 ($k = 1, \ldots M$),
- n_{jk} il numero massimo di unità di R_k richiesto da ciascun processo P_j ($j = 1, \ldots, N$),
- υ_k ($k = 0, \ldots, u_k$) 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:

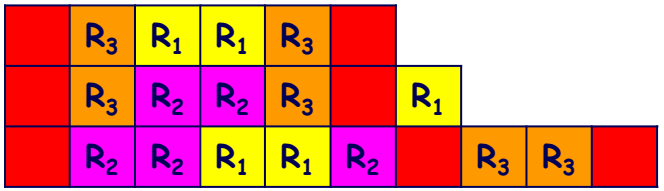
$$\Pi\Lambda_S = \max_k \{\Pi\Lambda_k(\upsilon_k)\}$$

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

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

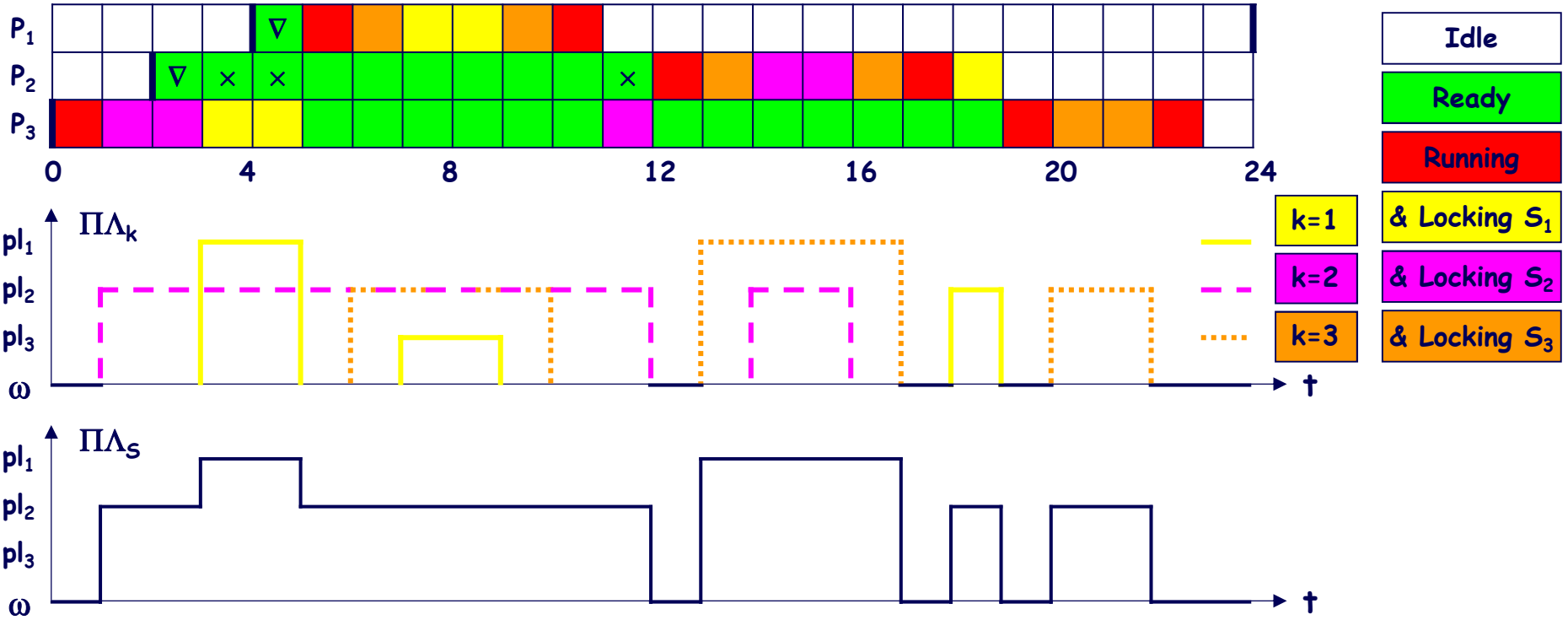
… STACK RESOURCE POLICY …

A_9 {
- 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])



pl_1 (max)
pl_2
pl_3 (min)

k	u_k	n_{1k}	n_{2k}	n_{3k}	$\Pi\Lambda_k(\upsilon_k=0)$	$\Pi\Lambda_k(\upsilon_k=1)$	$\Pi\Lambda_k(\upsilon_k=2)$	$\Pi\Lambda_k(\upsilon_k=3)$
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



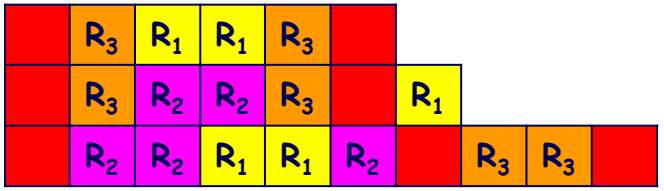
… STACK RESOURCE POLICY …

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 $\Pi\Lambda_k(0) > pl_j$:

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

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

A_9 {
- 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])



pl_1 (max)
pl_2
pl_3 (min)

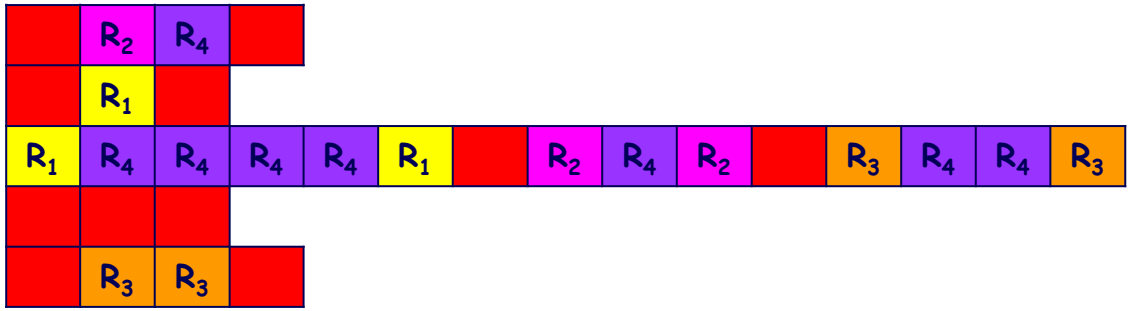
k	1	2	3
$\Pi\Lambda_k(0)$	pl_1	pl_2	pl_1

	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

… STACK RESOURCE POLICY …

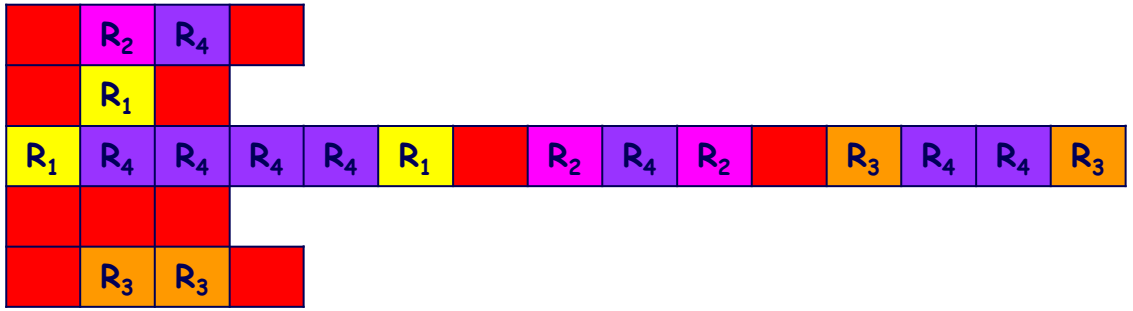
A_{10}

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)



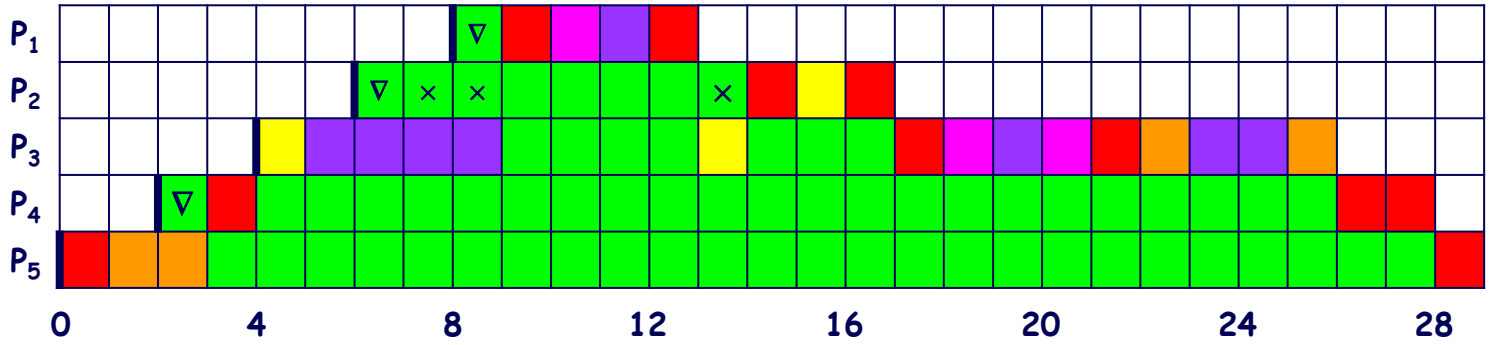
k	u_k	n_{1k}	n_{2k}	n_{3k}	n_{4k}	n_{5k}	$\Pi\Lambda_k(\upsilon_k=0)$	$\Pi\Lambda_k(\upsilon_k=1)$	$\Pi\Lambda_k(\upsilon_k=2)$
1	2		1	2			pl_2	pl_3	ω
2	2	1		2			pl_1	pl_3	ω
3	1			1		1	pl_3	ω	-
4	1	1		1			pl_1	ω	-

… STACK RESOURCE POLICY …

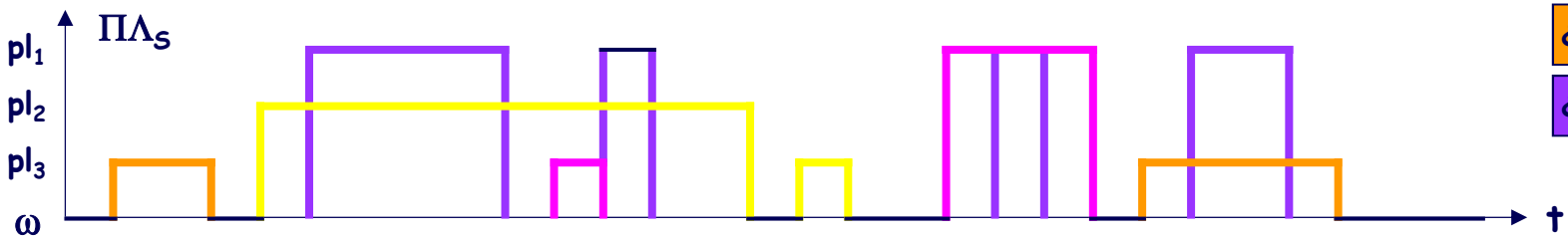


pl_1 (max)
pl_2
pl_3
pl_4
pl_5 (min)

k	u_k	n_{1k}	n_{2k}	n_{3k}	n_{4k}	n_{5k}	$\Pi\Lambda_k(\upsilon_k=0)$	$\Pi\Lambda_k(\upsilon_k=1)$	$\Pi\Lambda_k(\upsilon_k=2)$
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	-



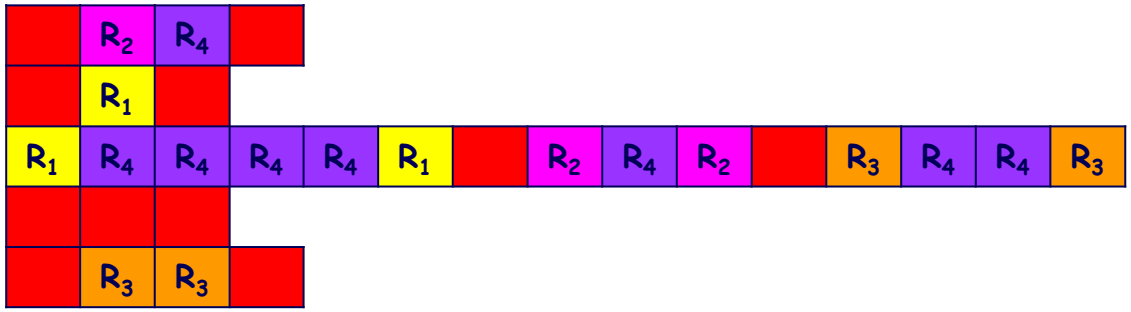
Idle
Ready
Running
& Locking S_1
& Locking S_2
& Locking S_3
& Locking S_4



… 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	$B_1 = 6$
P_2	1				$B_2 = 6$
P_3	6	3	4	4	$B_3 = 2$
P_4					$B_4 = 2$
P_5			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), …, P_N (T_N, C_N), ordinati per periodo crescente (priorità decrescente), sia schedulabile con RMPO è che:

$$\forall i,\ 1 \le i \le N,\ \sum_{k=1}^{i} \frac{C_k}{T_k} + \frac{B_i}{T_i} \le U_{RMPO}(i) = \begin{cases} 1 \\ i\left(2^{1/i} - 1\right) \end{cases}$$

a) se i periodi sono in relazione armonica

b) in caso contrario

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

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

… 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).

A_5 {

	priorità
P_1 (6, 15, 6, 15; [R_1;4] [R_2;2])	p_1 (max)
P_2 (4, 20, 4, 20; [R_2;2])	p_2
P_3 (2, 50, 4, 50)	p_3
P_4 (0, 100, 6, 100; [R_1;4])	p_4 (min)

}

	B_1	B_2	B_3	B_4
NPCSP	4	4	4	0
PIP	6	4	4	0
PCP	4	4	4	0
IPCP	4	4	4	0

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

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

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

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

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

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

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

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

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 A_5 soltanto per quanto concerne la deadline ([t.u.]) dei processi P_3 e P_4 (D_3=30 (anziché 50), D_4=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), …, P_N (T_N, C_N), ordinati per periodo crescente, sia schedulabile con EDF è che:

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

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

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

… 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).

A_7 {
- 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]])

	B_1	B_2	B_3
NPCSP	4	3	0
PCP	3	3	0
SRP	3	3	0

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

A_7: $C_1/T_1 + C_2/T_2 + B_1/T_1 = 4/10 + 5/15 + 4/10 = 1.13 > 1$ NO

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

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

P_3: $C_1/T_1 + C_2/T_2 + C_3/T_3 = 4/10 + 5/15 + 4/20 = 0.93 \leq 1$ 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), …, P_N (T_N, C_N, D_N), ordinati per deadline relativa crescente, sia schedulabile con EDF è che:

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

Esempio: la seguente applicazione,

A_7* {
- 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]])
}

	B_1	B_2	B_3
NPCSP	4	3	0
PCP	3	3	0
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: $C_1/D_1 + B_1/D_1 = 4/9 + 4(3)/9 = 0.889(0.778) \le 1$ OK

P_2: $C_1/D_1 + C_2/D_2 + B_2/D_2 = 4/9 + 5/15 + 3/15 = 0.978 \le 1$ OK

P_3: $C_1/D_1 + C_2/D_2 + C_3/D_3 = 4/9 + 5/15 + 4/16 = 1.028 > 1$ 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), …, P_N (T_N, C_N, D_N), ordinati per deadline relativa crescente, sia schedulabile con EDF è che (approccio "processor demand"):

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

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

Esempio: l'applicazione

A_7^* {
- 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]])

	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

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

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

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

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

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

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

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

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

	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