Sistemi Operativi

ESERCITAZIONE

STALLO

ESEMPIO DI DOMANDA

(vale 3 punti su un totale di 33)

Un sistema adotta nei confronti dello stallo una politica di prevenzione basata sull'ordinamento delle risorse, che si suppongono seriali e non-prelazionabili. Tale politica permette a un processo di richiedere istanze della classe di risorsa R_k solo se $rank_k > rank_i$ per ogni classe di risorsa R_i di cui il processo ha già allocate alcune istanze.

Spiegare, eventualmente facendo ricorso ad un esempio, se si possono verificare deadlock nel caso in cui la condizione diventasse $rank_k \geq rank_i$.

Risposta

La modifica proposta implica che un processo possa chiedere ulteriori istanze di R_k anche se ne possiede già delle altre.
In tale situazione possono verificarsi sia la condizione di possesso e attesa che quella di attesa circolare, perciò, data la serialità e non-prelazionabilità delle risorse assunta per ipotesi, si verificano tutte e 4 le condizioni necessarie per il deadlock, che quindi può potenzialmente verificarsi.

Risposta

La modifica proposta implica che un processo possa chiedere ulteriori istanze di R_k anche se ne possiede già delle altre. In tale situazione possono verificarsi sia la condizione di possesso e attesa che quella di attesa circolare, perciò, data la serialità e non-prelazionabilità delle risorse assunta per ipotesi, si verificano tutte e 4 le condizioni necessarie per il deadlock, che quindi può potenzialmente verificarsi.

Infatti, si supponga di avere un sistema con $n = n1 + n2$ istanze di risorse di classe R_k, e che i processi $P1$ e $P2$ abbiano allocate, rispettivamente, $n1$ e $n2$ istanze di R_k. Una successiva richiesta di $P1$ e $P2$ per ottenere ulteriori istanze di R_k, sarebbe del tutto lecita data la politica di prevenzione adottata ma, dato che le risorse aggiuntive non potrebbero essere concesse, porterebbe a una condizione di possesso e attesa, nonché a una condizione di attesa circolare (perché $P1$ si metterebbe in attesa di $P2$ e viceversa) con conseguente comparsa del deadlock, dal momento che nessun altro processo potrebbe rilasciare risorse di classe R_k.

ESEMPI DI ESERCIZI

Esercizio 1 (vale 4 punti)

Un sistema con 4 processi *P1*, *P2*, *P3* e *P4* e 3 tipi di risorse seriali e non-prelazionabili *R1*, *R2* e *R3*, rispettivamente con 5, 8 e 16 istanze, adotta nei confronti dello stallo una politica mirata ad evitare che esso si presenti. Lo stato corrente del sistema è caratterizzato dalle seguenti matrici:

Max

4	1	4
3	1	4
5	7	13
1	1	6

Allocation

0	1	4
2	0	1
1	2	1
1	0	3

Esercizio 1 (vale 4 punti)

Si determini:

a) se il sistema è attualmente in uno stato sicuro;
b) se la richiesta del processo *P3* di assegnazione di 6 istanze di *R3* può essere soddisfatta;
c) se la richiesta del processo *P1* di assegnazione di 1 istanza di *R1* può essere soddisfatta.

Si motivino le risposte mostrando i passi principali degli algoritmi applicati (per esempio, il valore assunto dal vettore *Work* durante l'esecuzione).

Soluzione (a)

Per stabilire se il sistema è attualmente in uno stato sicuro dobbiamo innanzitutto determinare i valori della matrice *Need* e del vettore *Available*

Soluzione (a)

Per stabilire se il sistema è attualmente in uno stato sicuro dobbiamo innanzitutto determinare i valori della matrice *Need* e del vettore *Available*

Questi possono essere ottenuti considerando le seguenti relazioni:

$$Need = Max - Allocation$$

$$Available = \text{Risorse complessive} - \sum_i Allocation_i$$

Il vettore *Available*, delle risorse attualmente disponibili, è ricavabile sottraendo alle risorse complessive del sistema quelle già allocate (somma delle righe della matrice *Allocation*)

Soluzione (a)

Risorse complessive

5	8	16

Max

P1	4	1	4
P2	3	1	4
P3	5	7	13
P4	1	1	6

Allocation

P1	0	1	4
P2	2	0	1
P3	1	2	1
P4	1	0	3

Soluzione (a)

Risorse complessive

5	8	16

Max

P1	4	1	4
P2	3	1	4
P3	5	7	13
P4	1	1	6

Allocation

P1	0	1	4
P2	2	0	1
P3	1	2	1
P4	1	0	3

Quindi, la matrice *Need* ed il vettore *Available* sono:

Need

P1	4	0	0
P2	1	1	3
P3	4	5	12
P4	0	1	3

Available

1	5	7

Soluzione (a)

Applicando l'algoritmo per verificare se uno stato è sicuro, dopo aver inizializzato *Work* con *Available*, abbiamo:

	Need		
P1	4	0	0
P2	1	1	3
P3	4	5	12
P4	0	1	3

Allocation		
0	1	4
2	0	1
1	2	1
1	0	3

Work		
1	5	7

Soluzione (a)

Applicando l'algoritmo per verificare se uno stato è sicuro, dopo aver inizializzato *Work* con *Available*, abbiamo:

1. al primo ciclo il processo *P2* può terminare perché la necessità [1,1,3] è inferiore alla disponibilità [1,5,7]; quindi, recuperando le risorse di *P2*, *Work* diventa [3,5,8];

	Need		
P1	4	0	0
P2	1	1	3
P3	4	5	12
P4	0	1	3

Allocation		
0	1	4
2	0	1
1	2	1
1	0	3

Work		
1	5	7

Soluzione (a)

Applicando l'algoritmo per verificare se uno stato è sicuro, dopo aver inizializzato *Work* con *Available*, abbiamo:

1. al primo ciclo il processo *P2* può terminare perché la necessità [1,1,3] è inferiore alla disponibilità [1,5,7]; quindi, recuperando le risorse di *P2*, *Work* diventa [3,5,8];
2. al secondo ciclo il processo *P4* può terminare perché la necessità [0,1,3] è inferiore alla disponibilità [3,5,8]; quindi, recuperando le risorse di *P4*, *Work* diventa: [4,5,11];

	Need		
P1	4	0	0
P2	~~1~~	~~1~~	~~3~~
P3	4	5	12
P4	0	1	3

Allocation		
0	1	4
2	0	1
1	2	1
1	0	3

Work		
3	5	8

Soluzione (a)

Applicando l'algoritmo per verificare se uno stato è sicuro, dopo aver inizializzato *Work* con *Available*, abbiamo:

1. al primo ciclo il processo *P2* può terminare perché la necessità [1,1,3] è inferiore alla disponibilità [1,5,7]; quindi, recuperando le risorse di *P2*, *Work* diventa [3,5,8];
2. al secondo ciclo il processo *P4* può terminare perché la necessità [0,1,3] è inferiore alla disponibilità [3,5,8]; quindi, recuperando le risorse di *P4*, *Work* diventa: [4,5,11];
3. al terzo ciclo il processo *P1* può terminare perché la necessità [4,0,0] è inferiore alla disponibilità [4,5,11]; quindi, recuperando le risorse di *P1*, *Work* diventa: [4,6,15];

P1	4	0	0
P2	~~1~~	~~1~~	~~3~~
P3	4	5	12
P4	~~0~~	~~1~~	~~3~~

0	1	4
2	0	1
1	2	1
1	0	3

4	5	11

Soluzione (a)

Applicando l'algoritmo per verificare se uno stato è sicuro, dopo aver inizializzato *Work* con *Available*, abbiamo:

1. al primo ciclo il processo *P2* può terminare perché la necessità [1,1,3] è inferiore alla disponibilità [1,5,7]; quindi, recuperando le risorse di *P2*, *Work* diventa: [3,5,8];
2. al secondo ciclo il processo *P1* può terminare perché la necessità [4,0,0] è inferiore alla disponibilità [3,5,8]; quindi, recuperando le risorse di *P1*, *Work* diventa: [4,5,11];
3. al terzo ciclo il processo *P4* può terminare perché la necessità [4,0,0] è inferiore alla disponibilità [4,5,11]; quindi, recuperando le risorse di *P1*, *Work* diventa: [4,6,15];

	Need			Allocation			Work		
P1	4	0	0	0	1	4	4	6	15
P2	1	1	3	2	0	1			
P3	4	5	12	1	2	1			
P4	0	1	3	1	0	3			

4. infine, al quarto ciclo il processo *P3* può terminare perché la necessità [4,5,12] è inferiore alla disponibilità [4,6,15]; alla fine, recuperando le risorse di *P3*, *Work* diventa: [5,8,16].

Soluzione (a)

Applicando l'algoritmo per verificare se uno stato è sicuro, dopo aver inizializzato *Work* con *Available*, abbiamo:

1. al primo ciclo il processo *P2* può terminare perché la necessità [1,1,3] è inferiore alla disponibilità [1,5,7]; quindi, recuperando le risorse di *P2*, *Work* diventa [3,5,8];
2. al secondo ciclo il processo *P4* può terminare perché la necessità [0,1,3] è inferiore alla disponibilità [3,5,8]; quindi, recuperando le risorse di *P4*, *Work* diventa: [4,5,11];
3. al terzo ciclo il processo *P1* può terminare perché la necessità [4,0,0] è inferiore alla disponibilità [4,5,11]; quindi, recuperando le risorse di *P1*, *Work* diventa: [4,6,15];
4. infine, al quarto ciclo il processo *P3* può terminare perché la necessità [4,5,12] è inferiore alla disponibilità [4,6,15]; alla fine, recuperando le risorse di *P3*, *Work* diventa: [5,8,16].

Quindi, la sequenza di processi ⟨*P2*,*P4*,*P1*,*P3*⟩ è una sequenza sicura e perciò lo stato è sicuro.

Soluzione (a)

Applicando l'algoritmo per verificare se uno stato è sicuro, dopo aver inizializzato *Work* con *Available*, abbiamo:

1. al primo ciclo il processo *P2* può terminare perché la necessità [1,1,3] è inferiore alla disponibilità [1,5,7]; quindi, recuperando le risorse di *P2*, *Work* diventa [3,5,8];
2. al secondo ciclo il processo *P4* può terminare perché la necessità [0,1,3] è inferiore alla disponibilità [3,5,8]; quindi, recuperando le risorse di *P4*, *Work* diventa: [4,5,11];
3. al terzo ciclo il processo *P1* può terminare perché la necessità [4,0,0] è inferiore alla disponibilità [4,5,11]; quindi, recuperando le risorse di *P1*, *Work* diventa: [4,6,15];
4. infine, al quarto ciclo il processo *P3* può terminare perché la necessità [4,5,12] è inferiore alla disponibilità [4,6,15]; alla fine, recuperando le risorse di *P3*, *Work* diventa: [5,8,16].

Quindi, la sequenza di processi ⟨*P2*,*P4*,*P1*,*P3*⟩ è una sequenza sicura e perciò lo stato è sicuro.

Esercizio: verificare che un'altra sequenza sicura è ⟨P4,P2,P1,P3⟩

Soluzione (b)

È possibile soddisfare la richiesta [0,0,6] da parte di *P*3?
Applichiamo l'algoritmo del banchiere.

Soluzione (b)

È possibile soddisfare la richiesta [0,0,6] da parte di *P*3?
Applichiamo l'algoritmo del banchiere.
Poiché [0,0,6] ≤ $Need_3$ e [0,0,6] ≤ *Available*, in linea di principio la richiesta potrebbe essere soddisfatta.

Need

4	0	0
1	1	3
4	5	12
0	1	3

Available

1	5	7

Soluzione (b)

È possibile soddisfare la richiesta [0,0,6] da parte di *P*3?
Applichiamo l'algoritmo del banchiere.
Poiché [0,0,6] ≤ $Need_3$ e [0,0,6] ≤ *Available*, in linea di principio la richiesta potrebbe essere soddisfatta.
Se la richiesta fosse accordata, avremmo

Need

4	0	0
1	1	3
4	5	6
0	1	3

Available

1	5	1

Soluzione (b)

Applichiamo l'algoritmo per verificare se lo stato è sicuro

- Dobbiamo trovare una riga della matrice *Need* che, elemento per elemento, è minore o uguale al vettore *Work (= Available)*.

Need

P1	4	0	0
P2	1	1	3
P3	4	5	6
P4	0	1	3

Available

1	5	1

Soluzione (b)

Troviamo però che non esiste alcun processo le cui necessità residue possono essere soddisfatte dall'attuale vettore *Work (= Available)*.

Need

P1	4	0	0
P2	1	1	3
P3	4	5	6
P4	0	1	3

Available

1	5	1

Soluzione (b)

Troviamo però che non esiste alcun processo le cui necessità residue possono essere soddisfatte dall'attuale vettore *Work (= Available)*.
Quindi, lo stato non è sicuro e la richiesta non può essere soddisfatta.

Need

P1	4	0	0
P2	1	1	3
P3	4	5	6
P4	0	1	3

Available

1	5	1

Soluzione (c)

È possibile soddisfare la richiesta [1,0,0] da parte di *P1?*
Applichiamo nuovamente l'algoritmo del banchiere.

Soluzione (c)

È possibile soddisfare la richiesta [1,0,0] da parte di *P1?*
Applichiamo l'algoritmo del banchiere.
Poiché [1,0,0] ≤ $Need_1$ e [1,0,0] ≤ *Available*, in linea di principio la richiesta potrebbe essere soddisfatta.

Need

P1	4	0	0
P2	1	1	3
P3	4	5	12
P4	0	1	3

Available

1	5	7

Soluzione (c)

È possibile soddisfare la richiesta [1,0,0] da parte di *P1?*
Applichiamo l'algoritmo del banchiere.
Poiché [1,0,0] ≤ $Need_1$ e [1,0,0] ≤ *Available*, in linea di principio la richiesta potrebbe essere soddisfatta.
Se lo fosse, avremmo

Need

P1	3	0	0
P2	1	1	3
P3	4	5	12
P4	0	1	3

Available

0	5	7

Soluzione (c)

In tale stato, eseguendo l'algoritmo di verifica dello stato sicuro (fatelo come esercizio), si trova la sequenza sicura di processi ⟨*P*4,*P*2,*P*1,*P*3⟩ (che produce i vettori *Work* [1,5,10], [3,5,11], [4,6,15], [5,8,16]).
Quindi, lo stato è sicuro e la richiesta può essere soddisfatta.

Need

P1	3	0	0
P2	1	1	3
P3	4	5	12
P4	0	1	3

Available

0	5	7

Esercizio 2 (vale 3 punti)

Un sistema con 7 processi *P1*-*P7* e 6 risorse seriali e non-prelazionabili *R1*, *R2*, *R3*, *R4*, *R5* e *R6*, ciascuna di tipo diverso, adotta nei confronti dello stallo una politica di rilevamento e ripristino.

La situazione del sistema è la seguente:

- *P1* possiede *R1* e richiede *R2*;
- *P2* non possiede risorse e richiede *R3*;
- *P3* non possiede risorse e richiede *R2*;
- *P4* possiede *R4* e richiede sia *R2* che *R3*;
- *P5* possiede *R3* e richiede *R5*;
- *P6* possiede *R6* e richiede *R2*;
- *P7* possiede *R5* e richiede *R4*.

Esercizio 2 (vale 3 punti)

Un sistema con 7 processi *P1*-*P7* e 6 risorse seriali e non-prelazionabili *R1*, *R2*, *R3*, *R4*, *R5* e *R6*, ciascuna di tipo diverso, adotta nei confronti dello stallo una politica di rilevamento e ripristino.

La situazione del sistema è la seguente:

- *P1* possiede *R1* e richiede *R2*;
- *P2* non possiede risorse e richiede *R3*;
- *P3* non possiede risorse e richiede *R2*;
- *P4* possiede *R4* e richiede sia *R2* che *R3*;
- *P5* possiede *R3* e richiede *R5*;
- *P6* possiede *R6* e richiede *R2*;
- *P7* possiede *R5* e richiede *R4*.

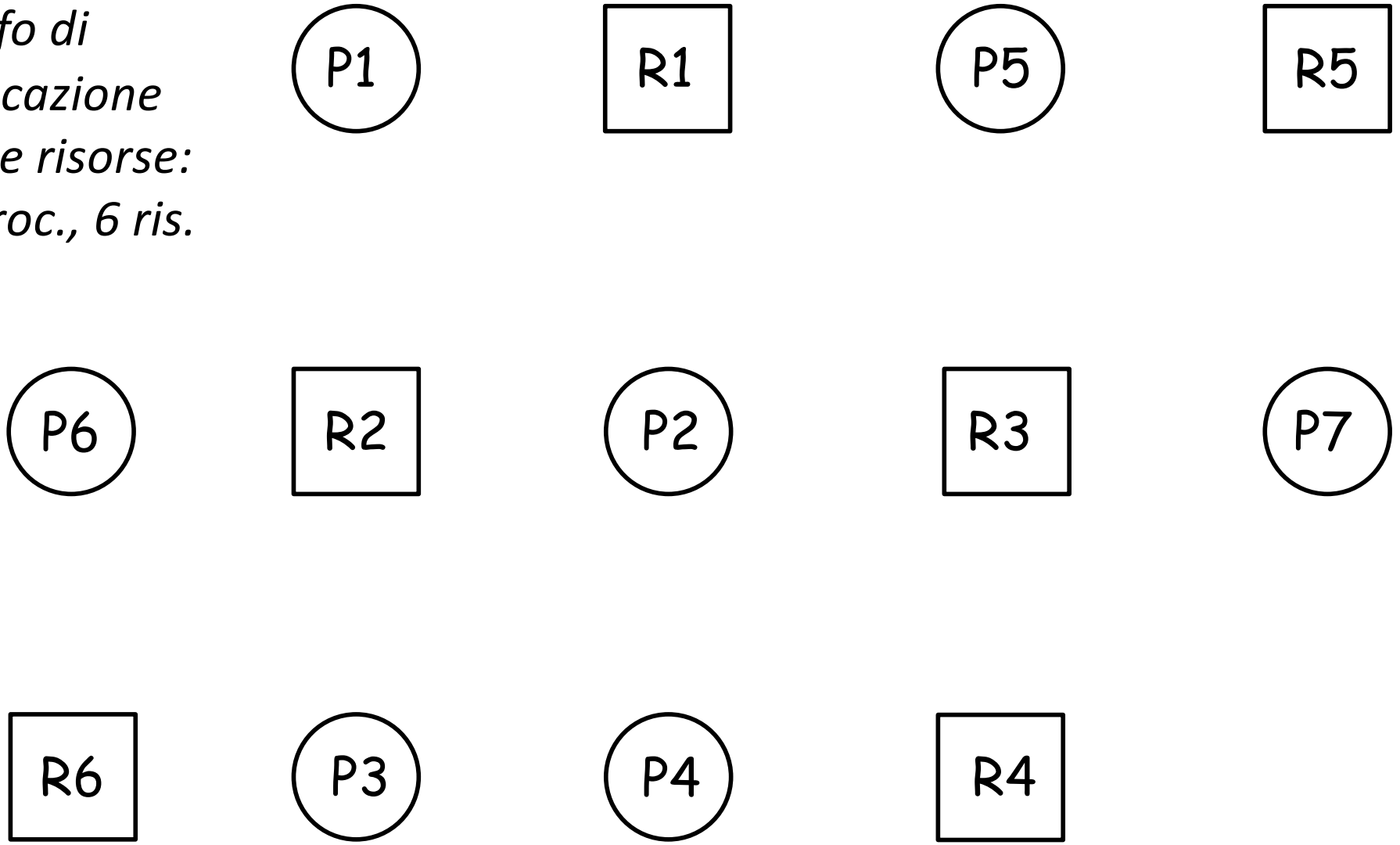
Si disegni il grafo di allocazione delle risorse e si determini se il sistema è in stallo e, in caso affermativo, quali sono i processi e le risorse coinvolti.

Soluzione

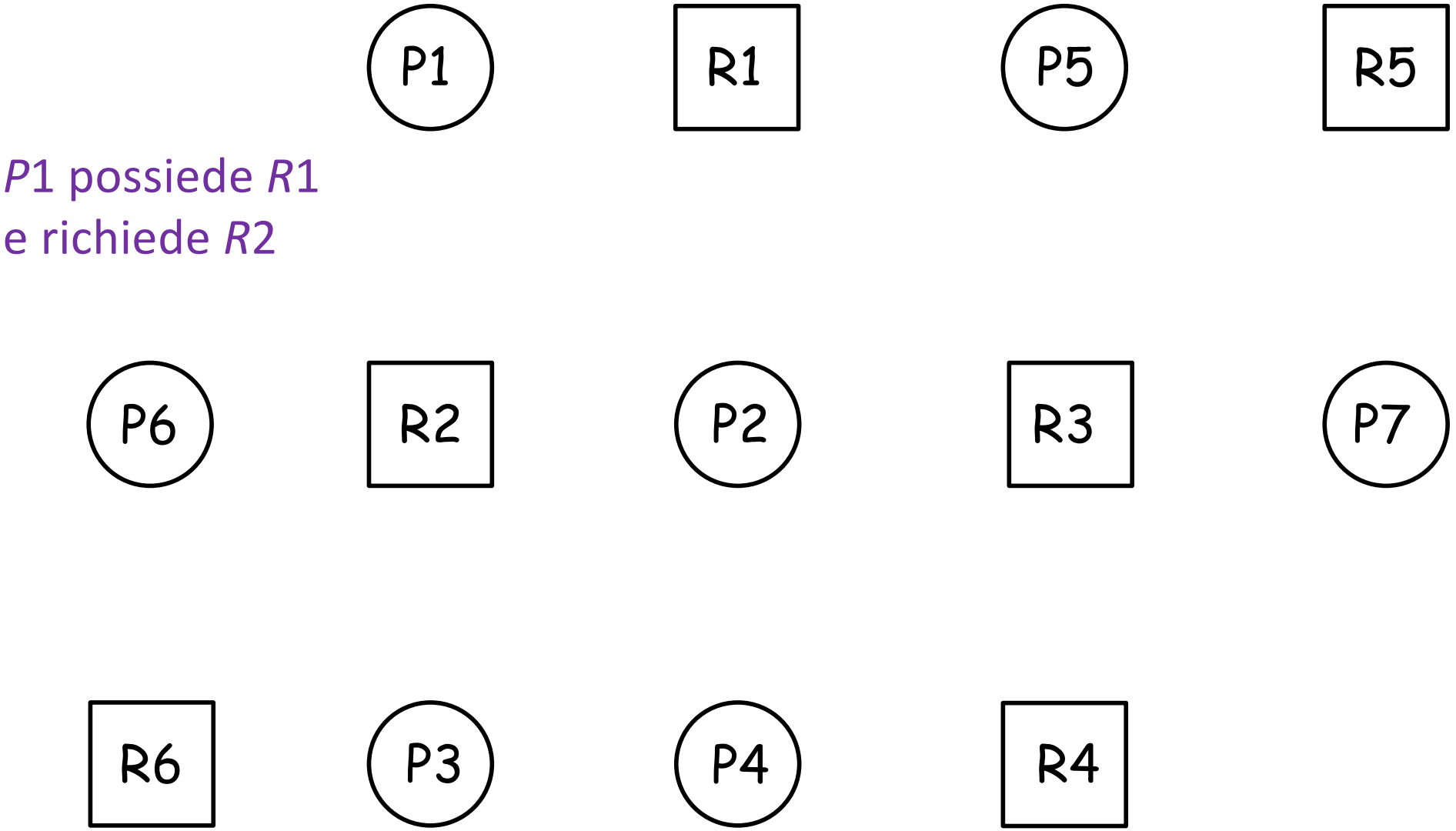
Disegniamo il grafo di allocazione delle risorse: 7 proc., 6 ris.

Soluzione

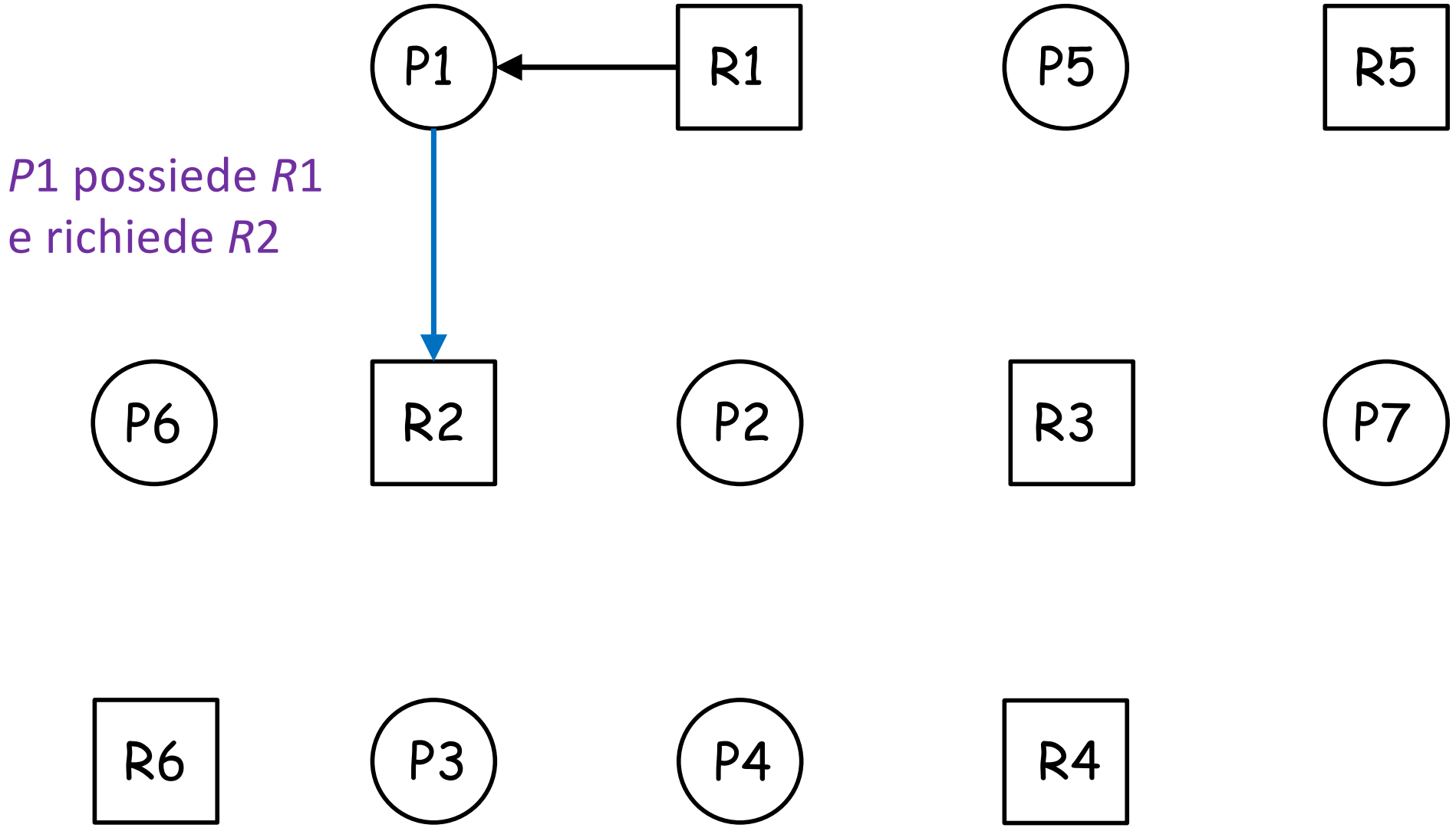
Disegniamo il grafo di allocazione delle risorse: 7 proc., 6 ris.



Soluzione

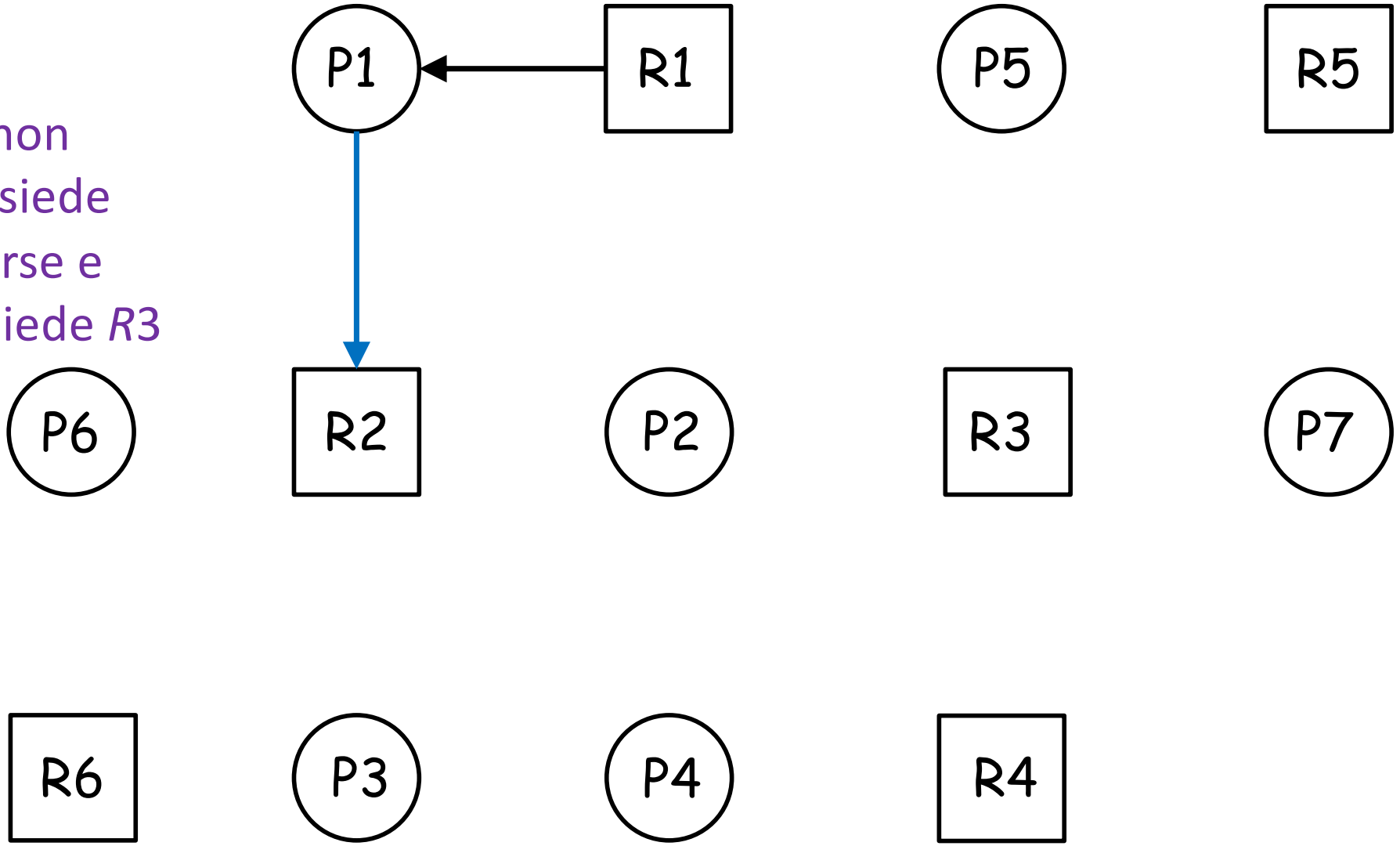


Soluzione



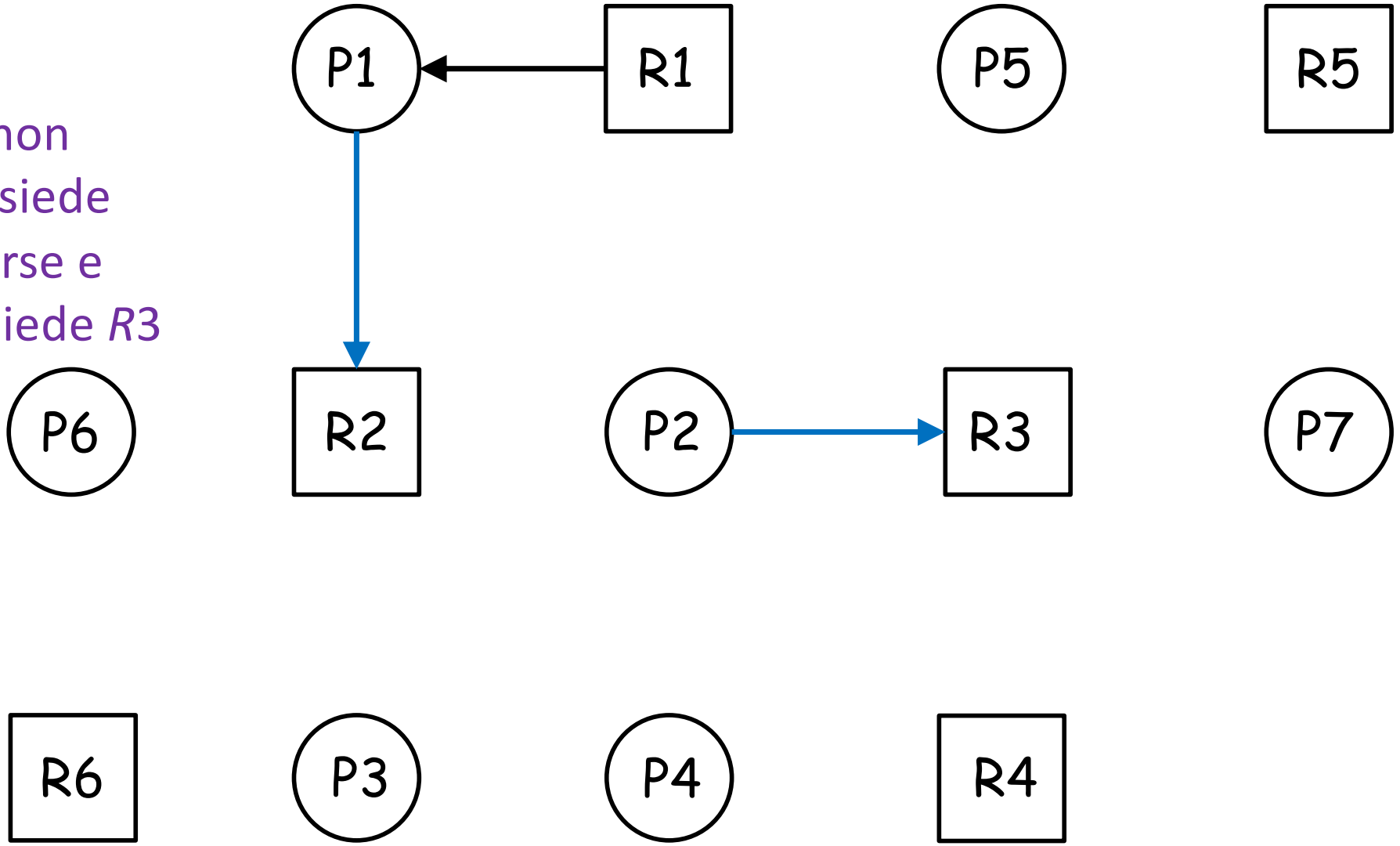
Soluzione

*P*2 non possiede risorse e richiede *R*3



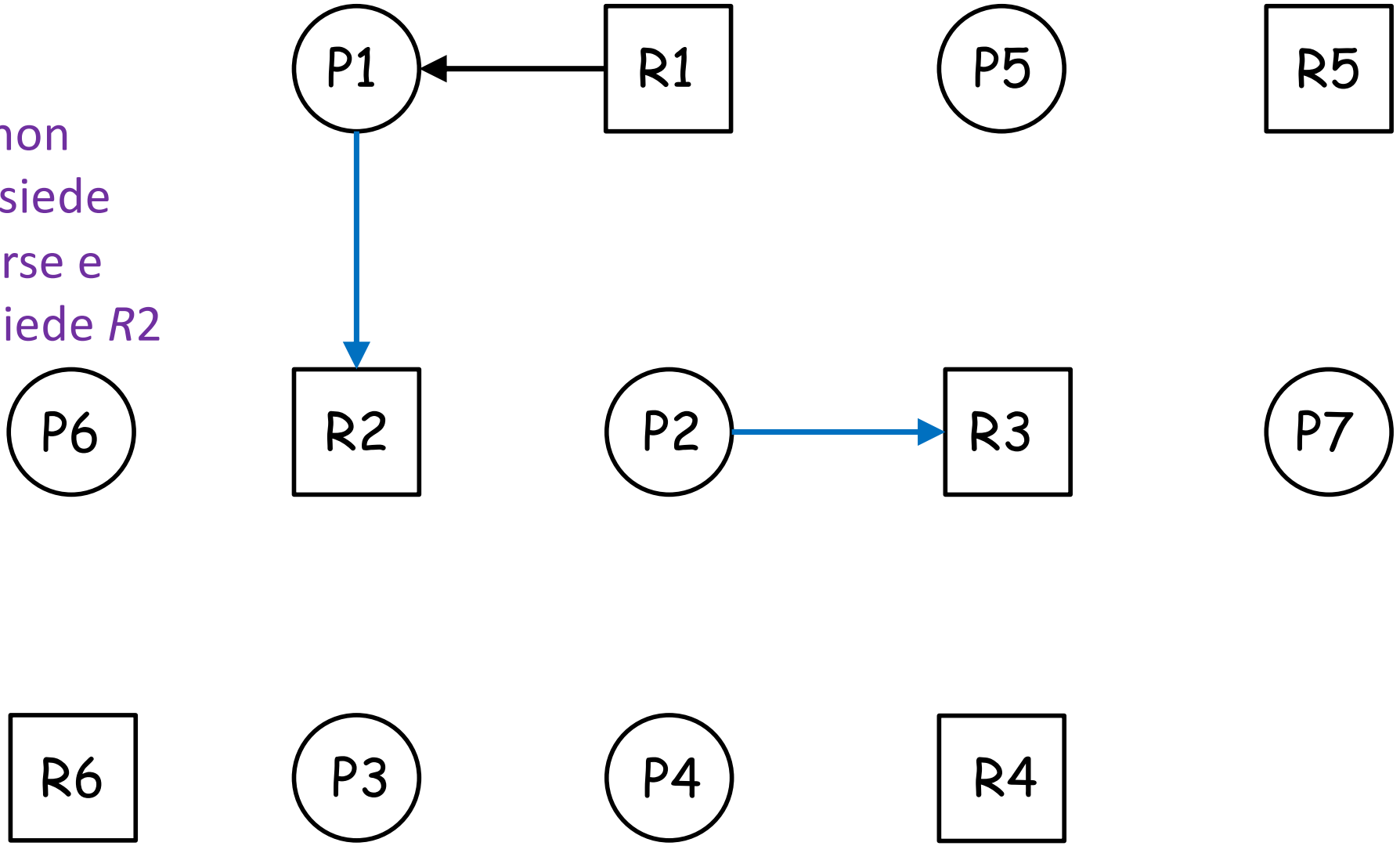
Soluzione

*P*2 non possiede risorse e richiede *R*3



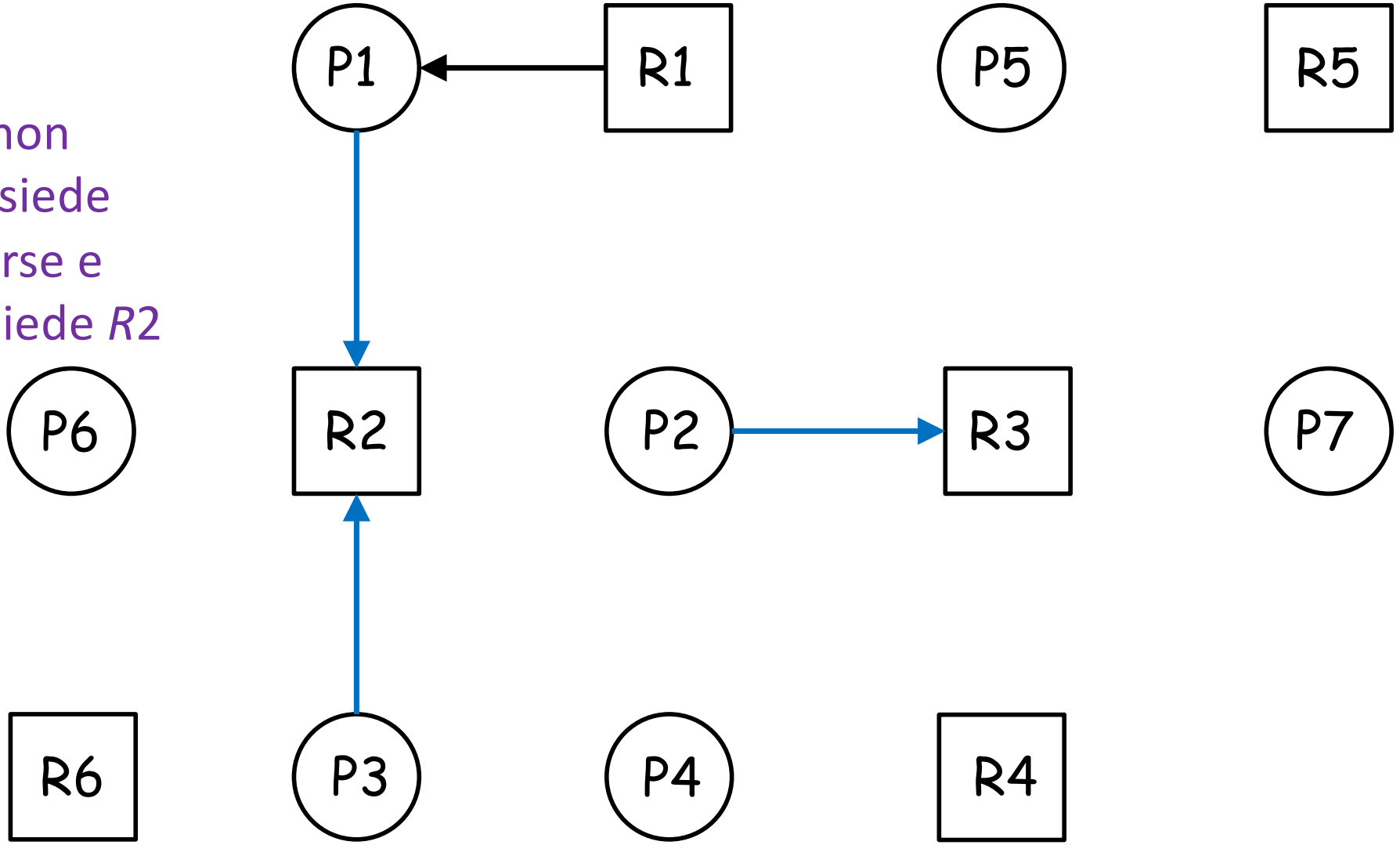
Soluzione

*P*3 non possiede risorse e richiede *R*2



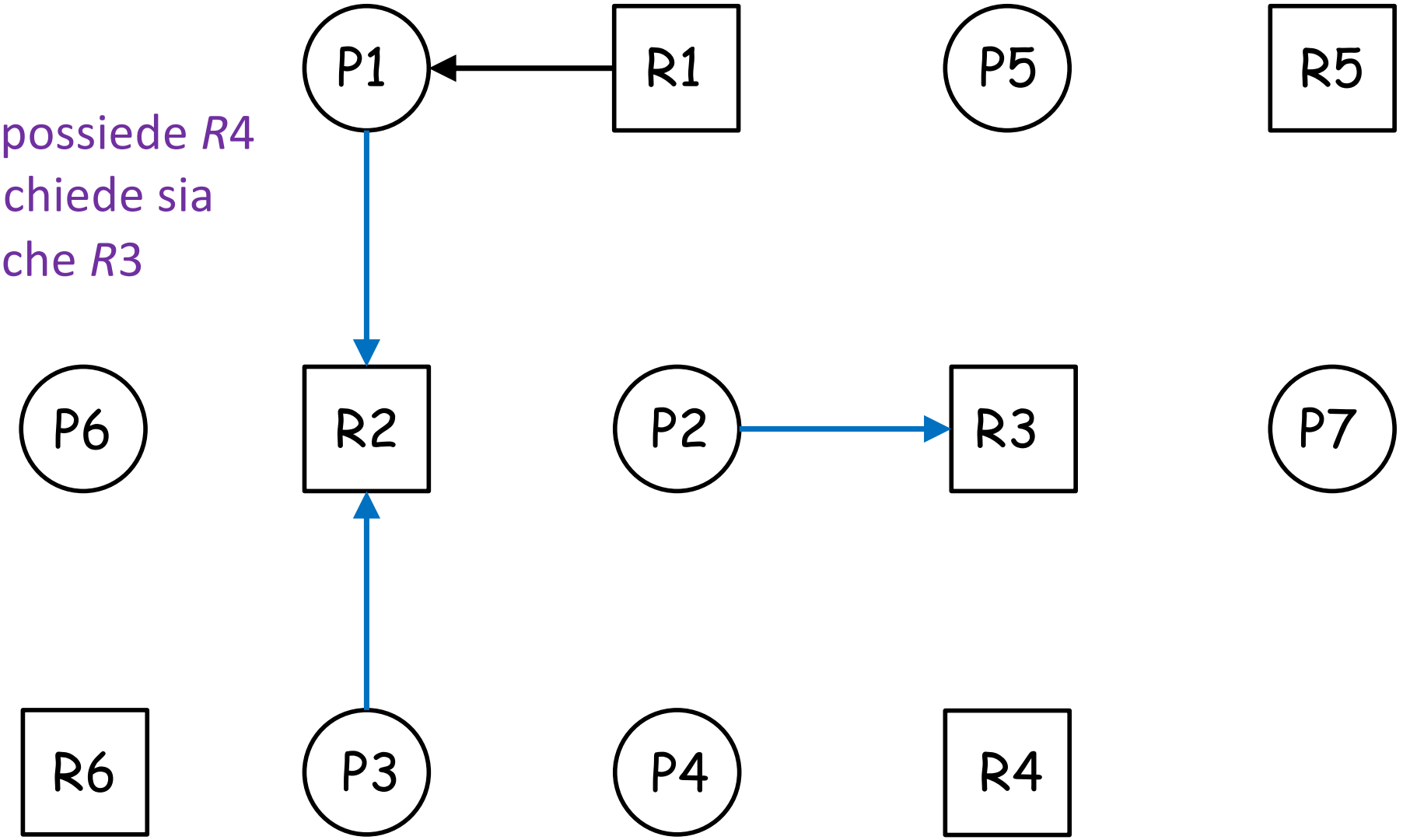
Soluzione

*P*3 non possiede risorse e richiede *R*2



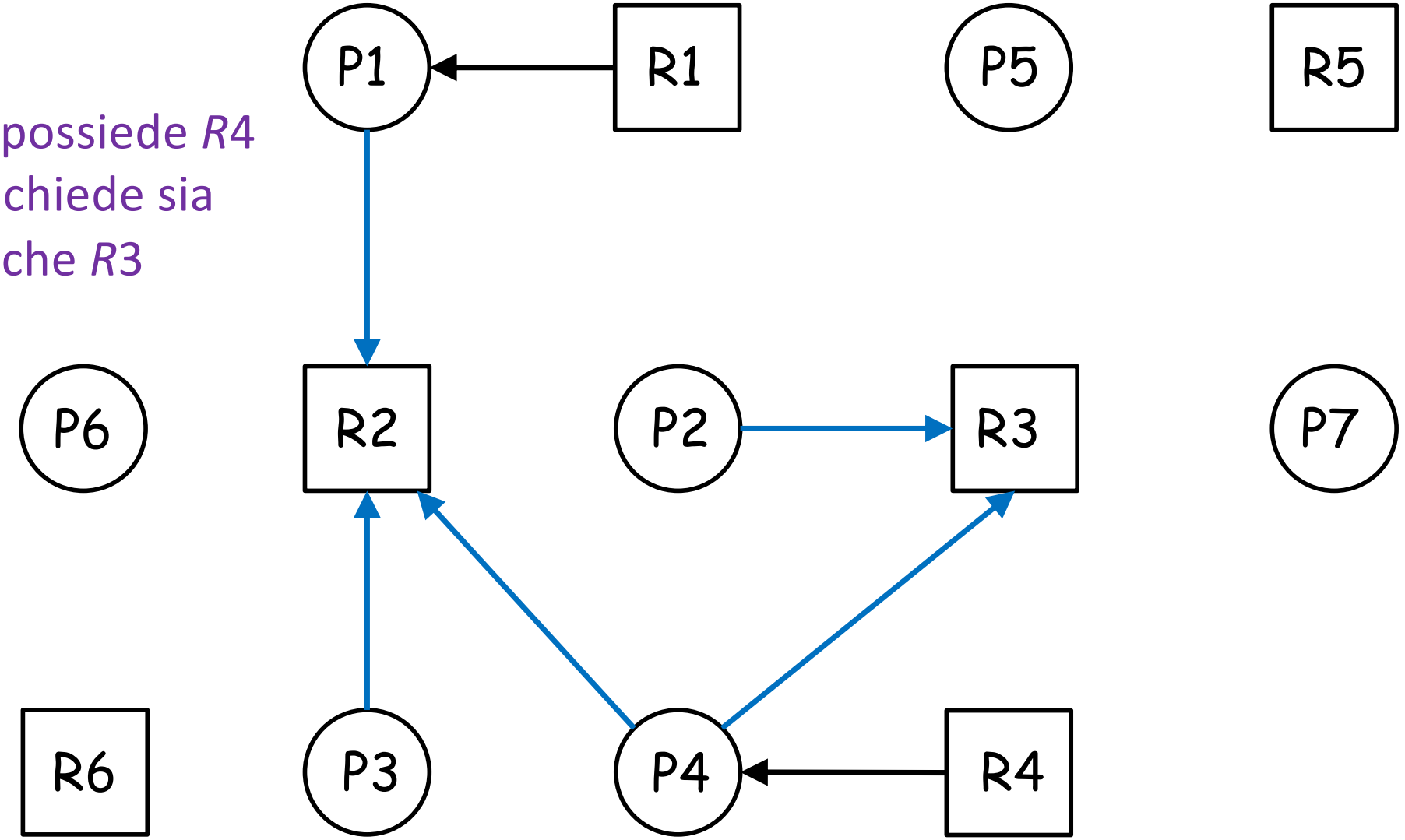
Soluzione

*P*4 possiede *R*4
e richiede sia
*R*2 che *R*3

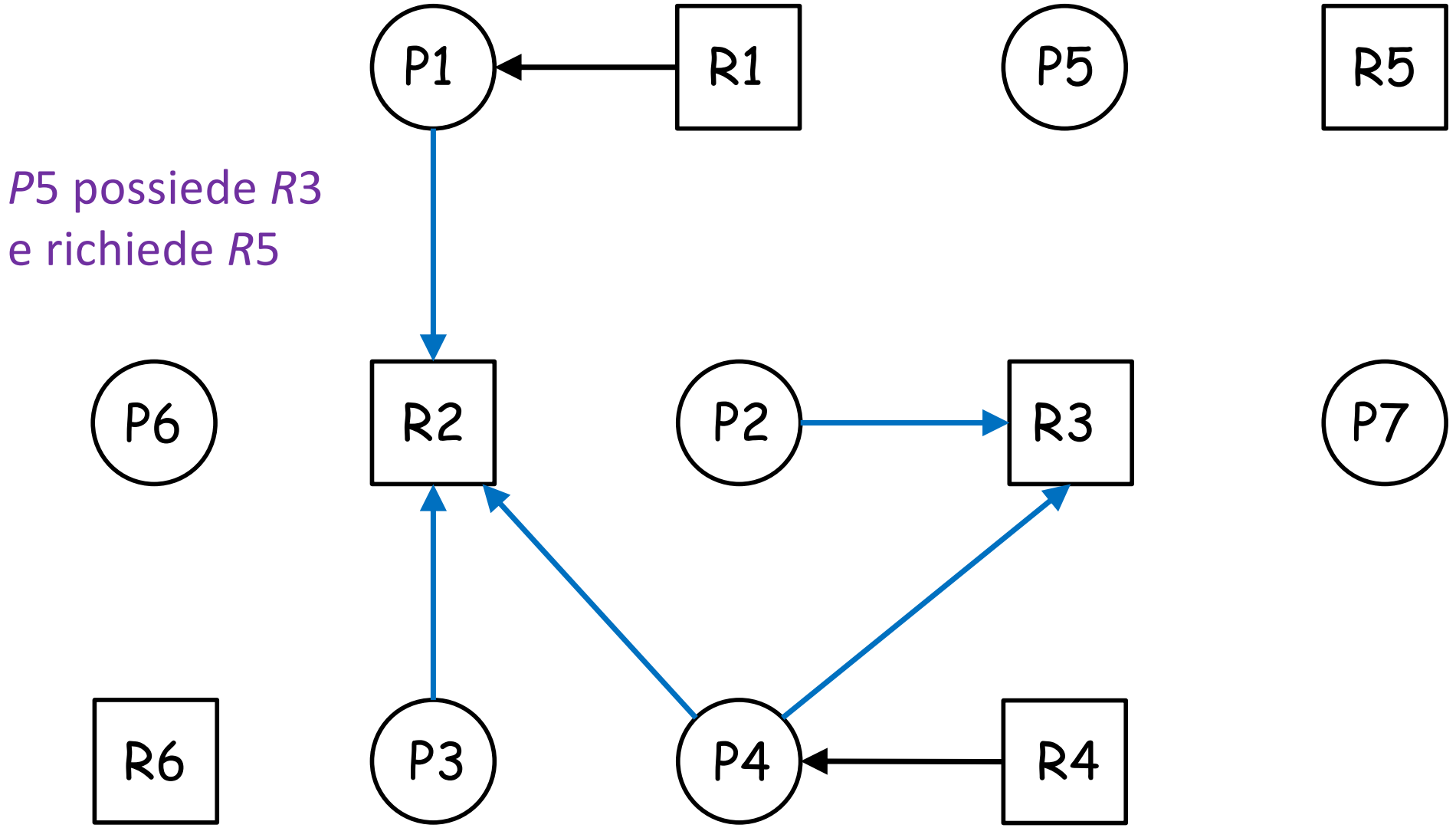


Soluzione

*P*4 possiede *R*4
e richiede sia
*R*2 che *R*3

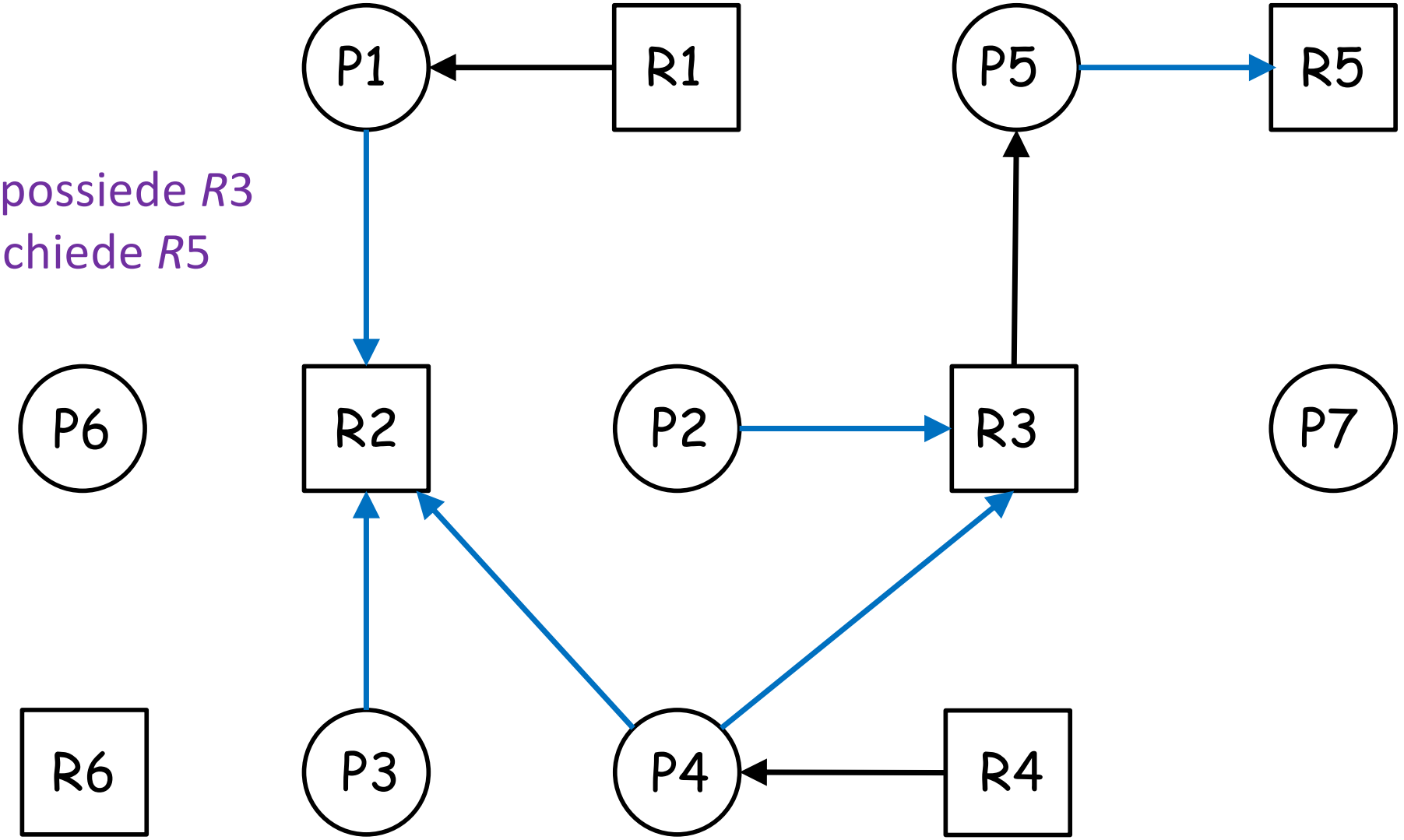


Soluzione



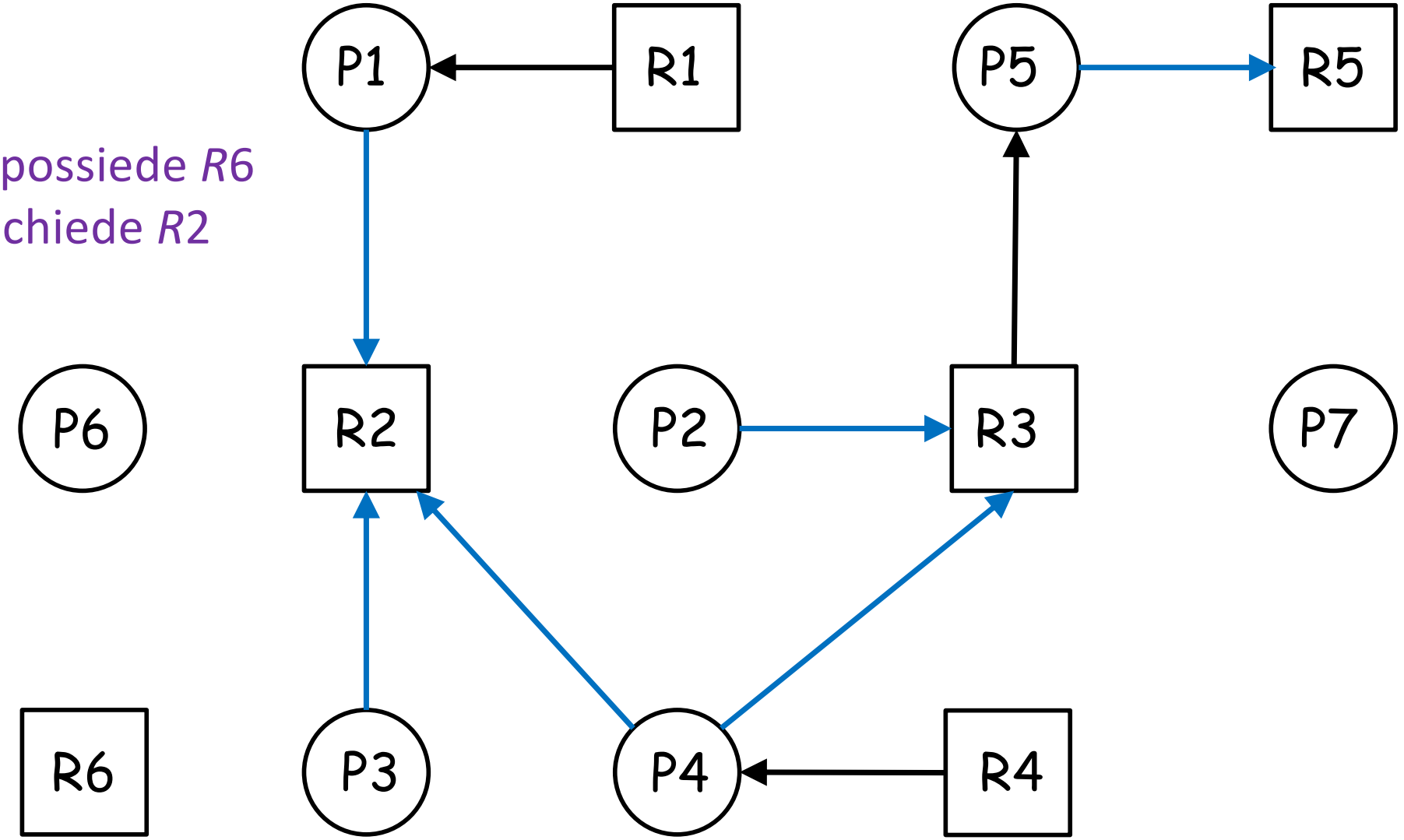
Soluzione

*P*5 possiede *R*3
e richiede *R*5



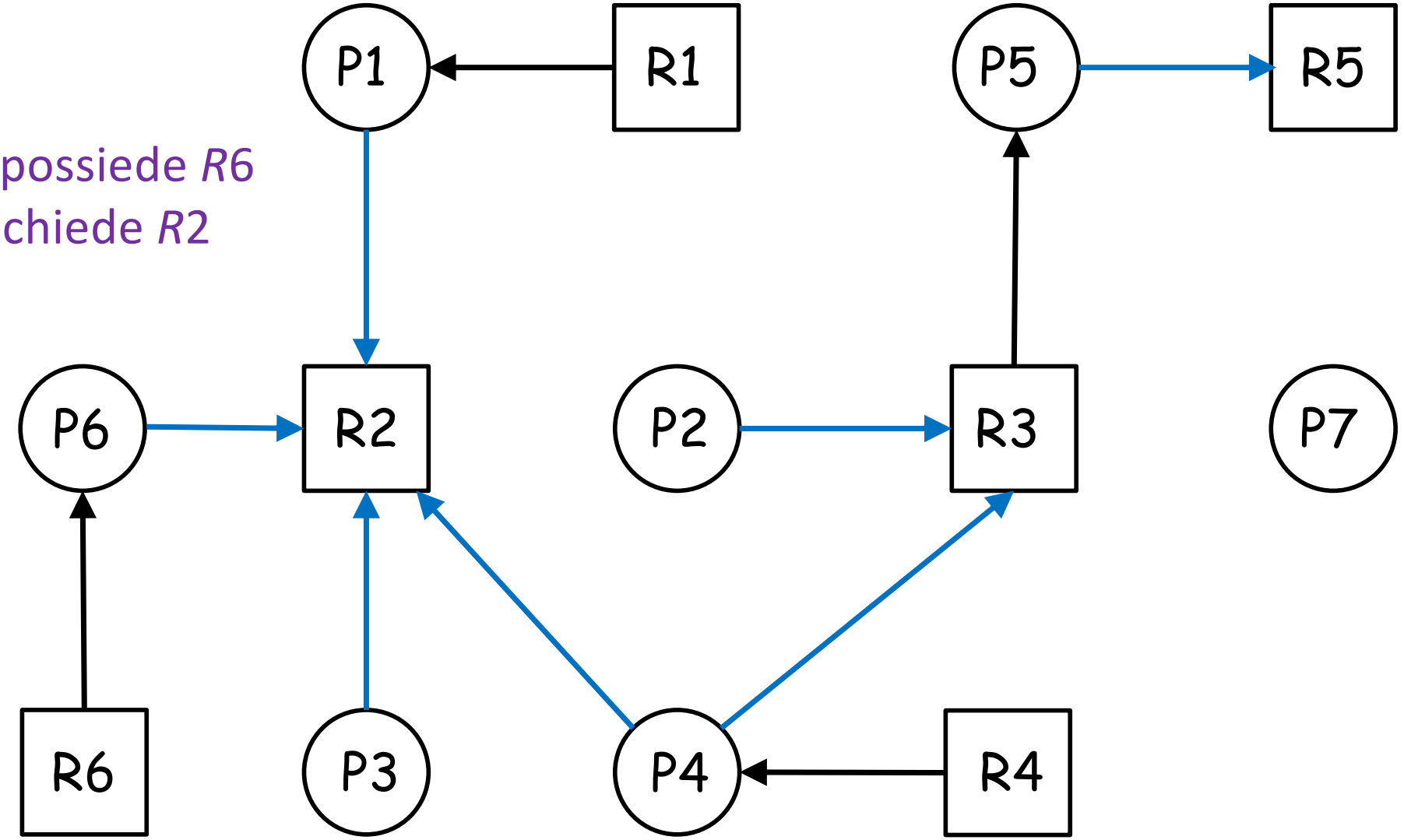
Soluzione

*P*6 possiede *R*6
e richiede *R*2

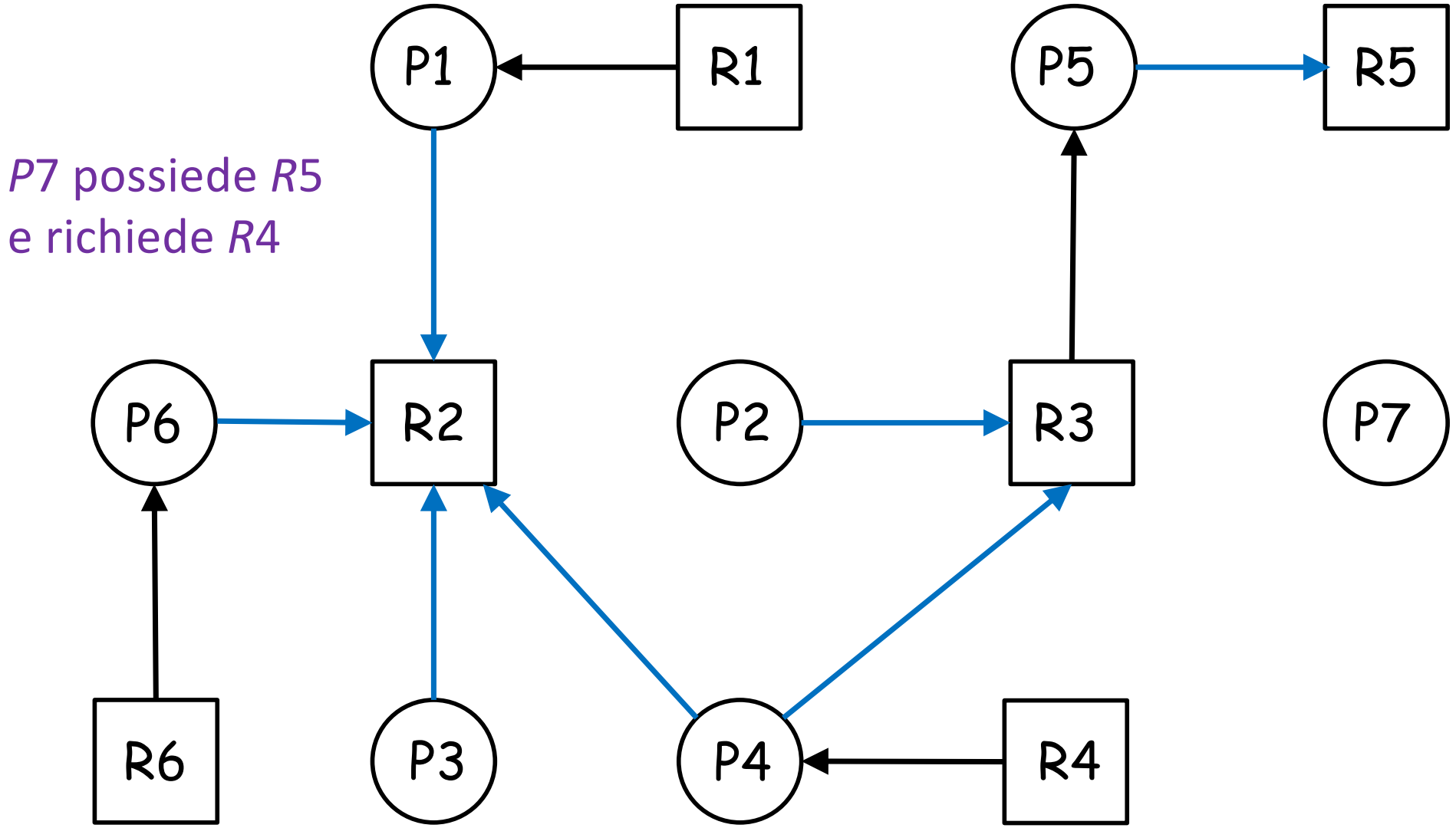


Soluzione

*P*6 possiede *R*6
e richiede *R*2

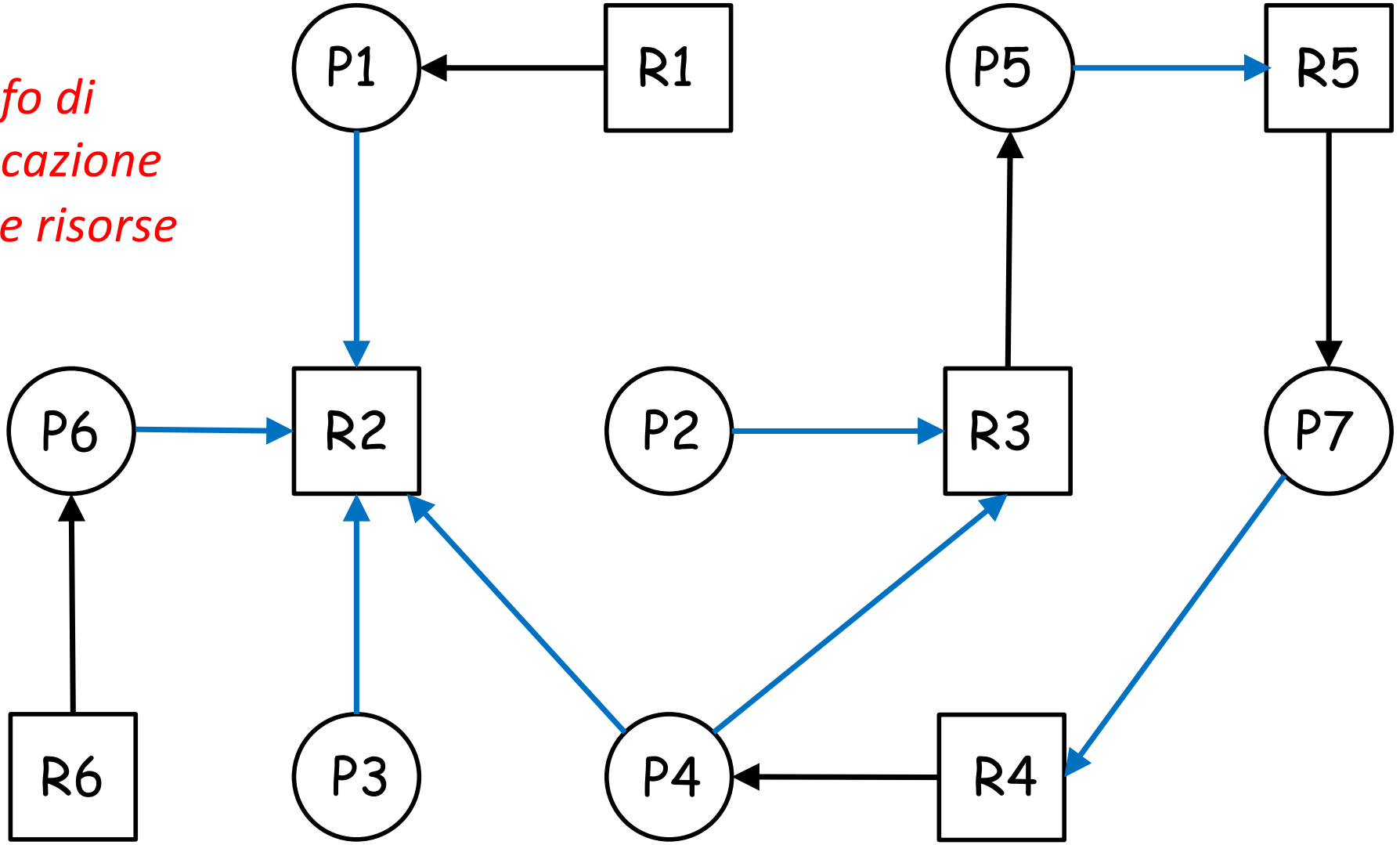


Soluzione



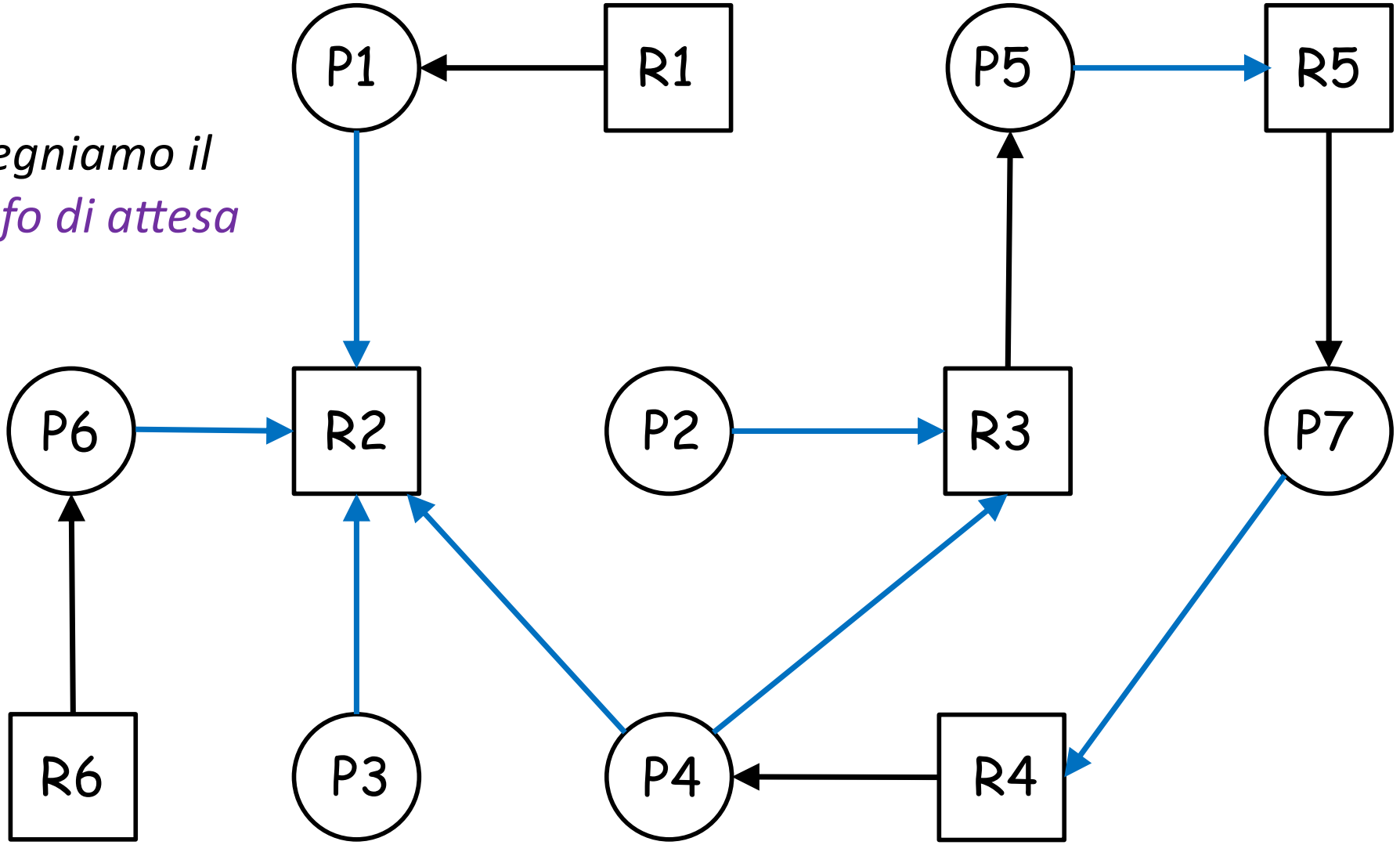
Soluzione

Grafo di allocazione delle risorse

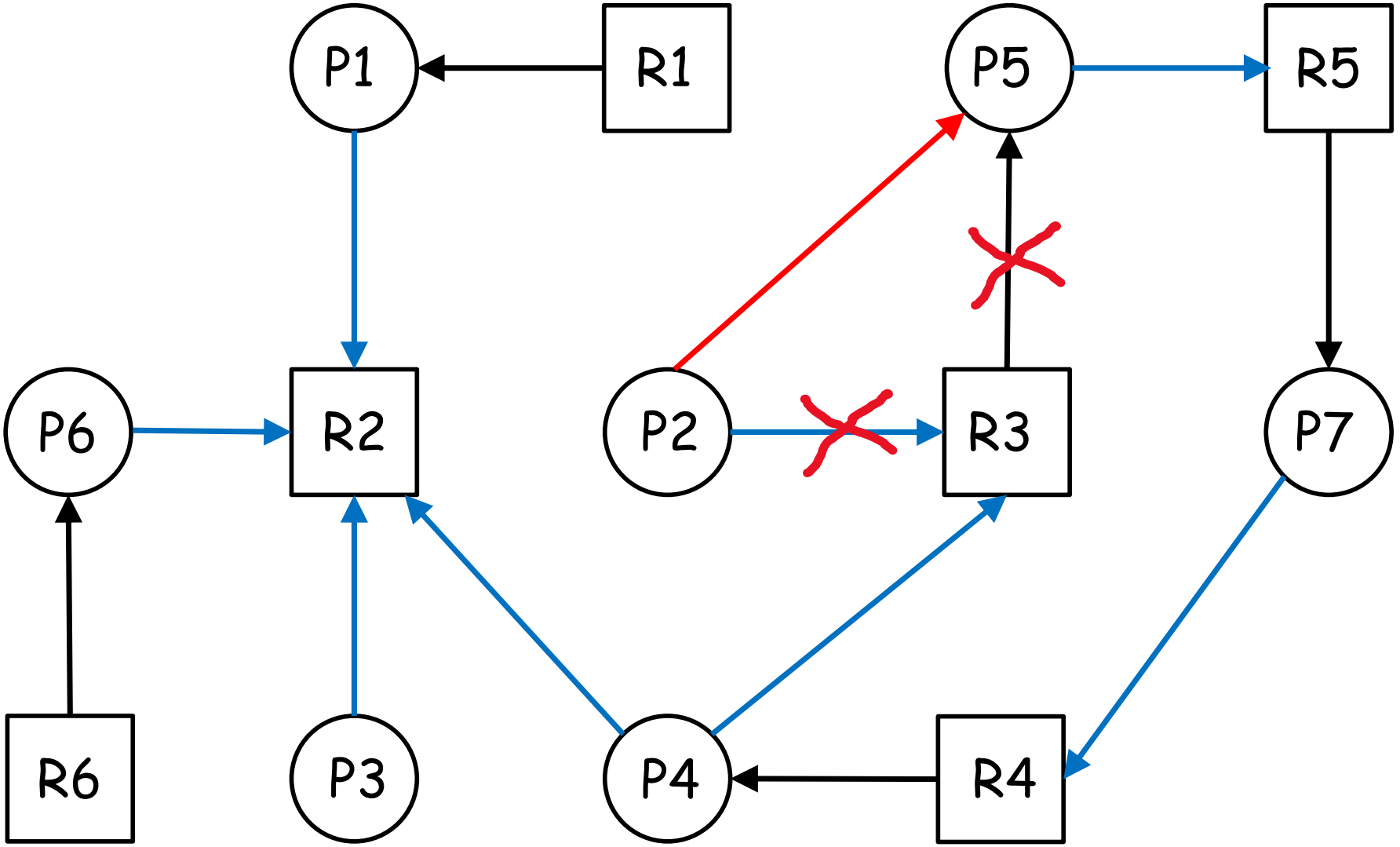


Soluzione

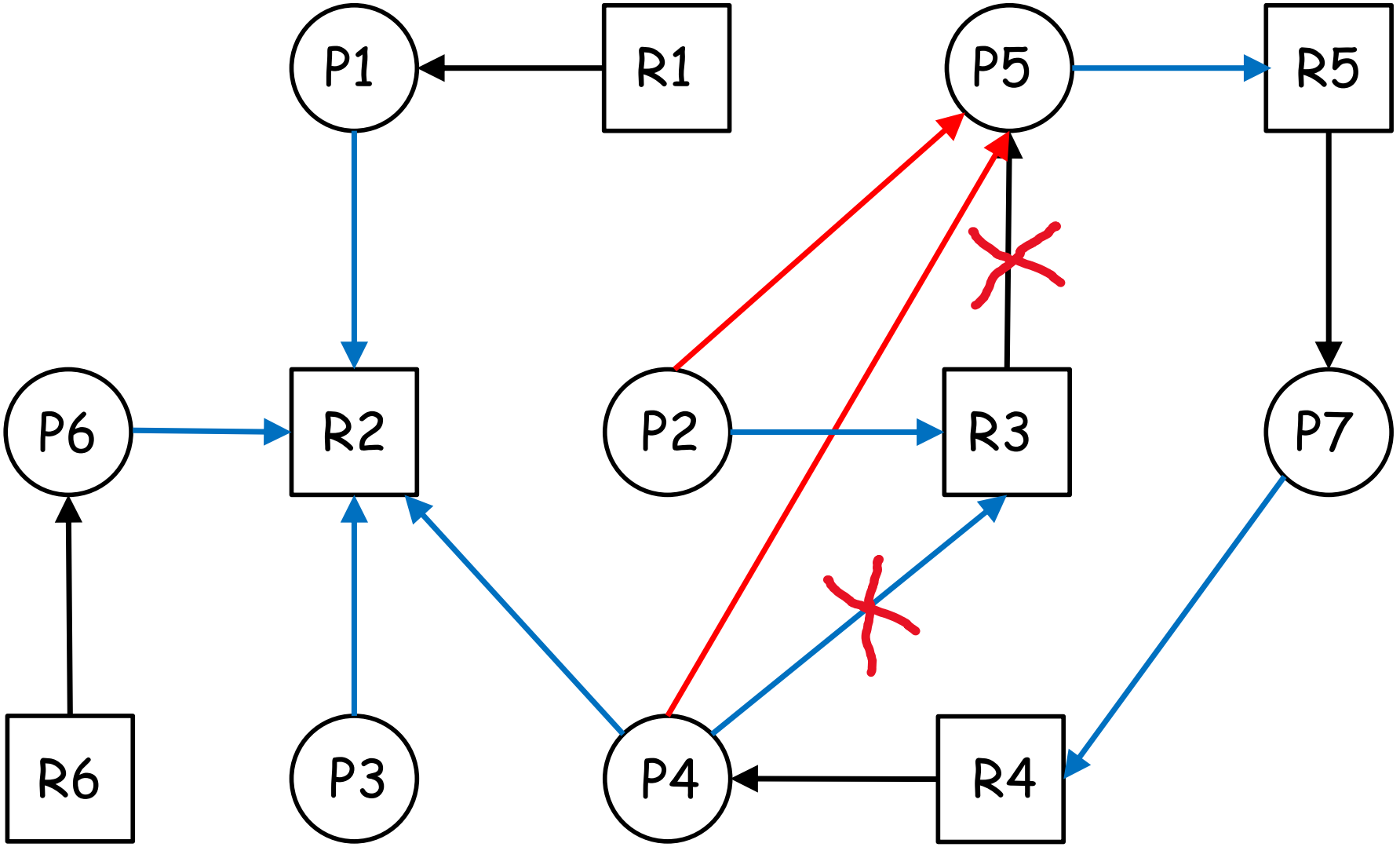
Disegniamo il
Grafo di attesa



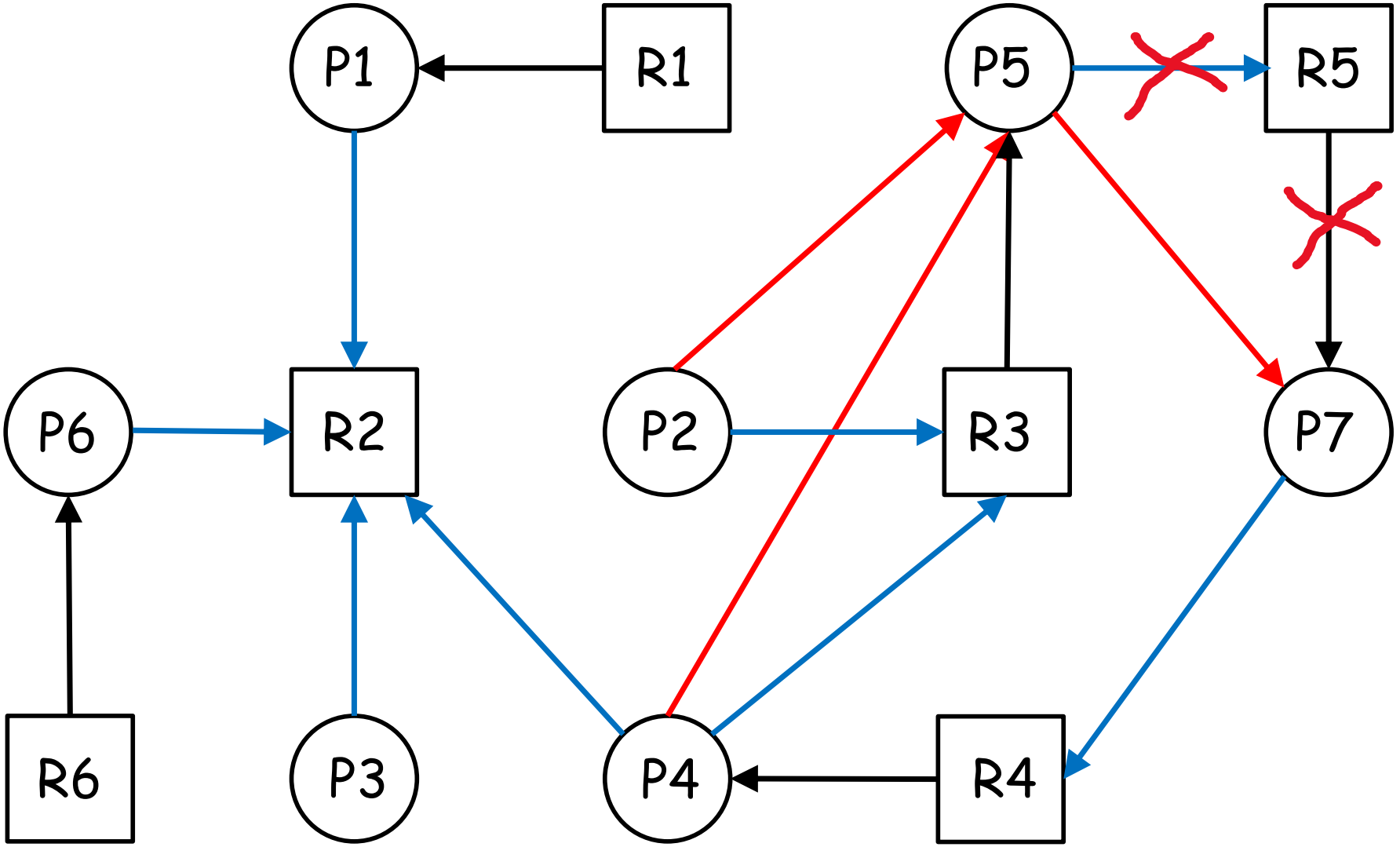
Soluzione



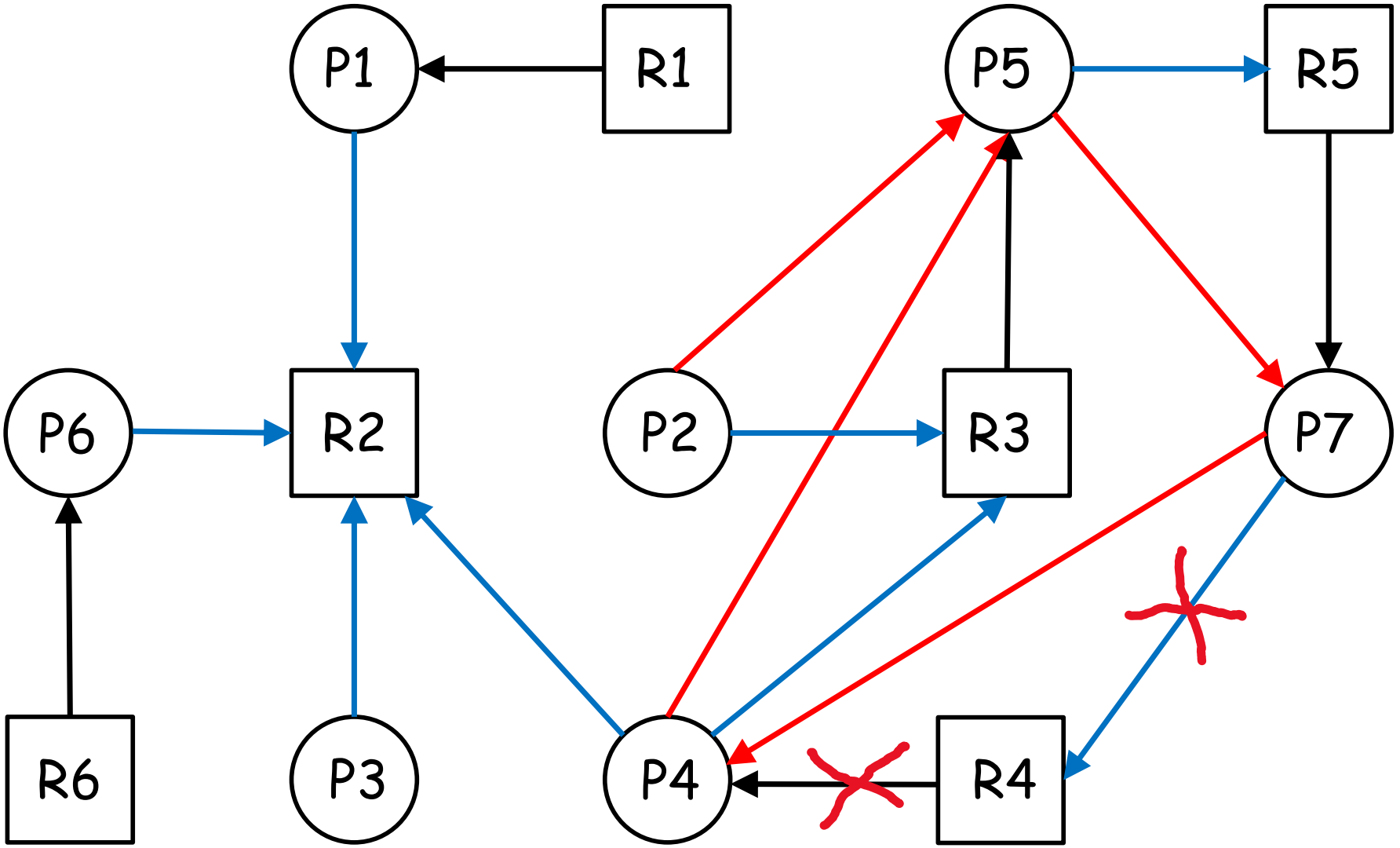
Soluzione



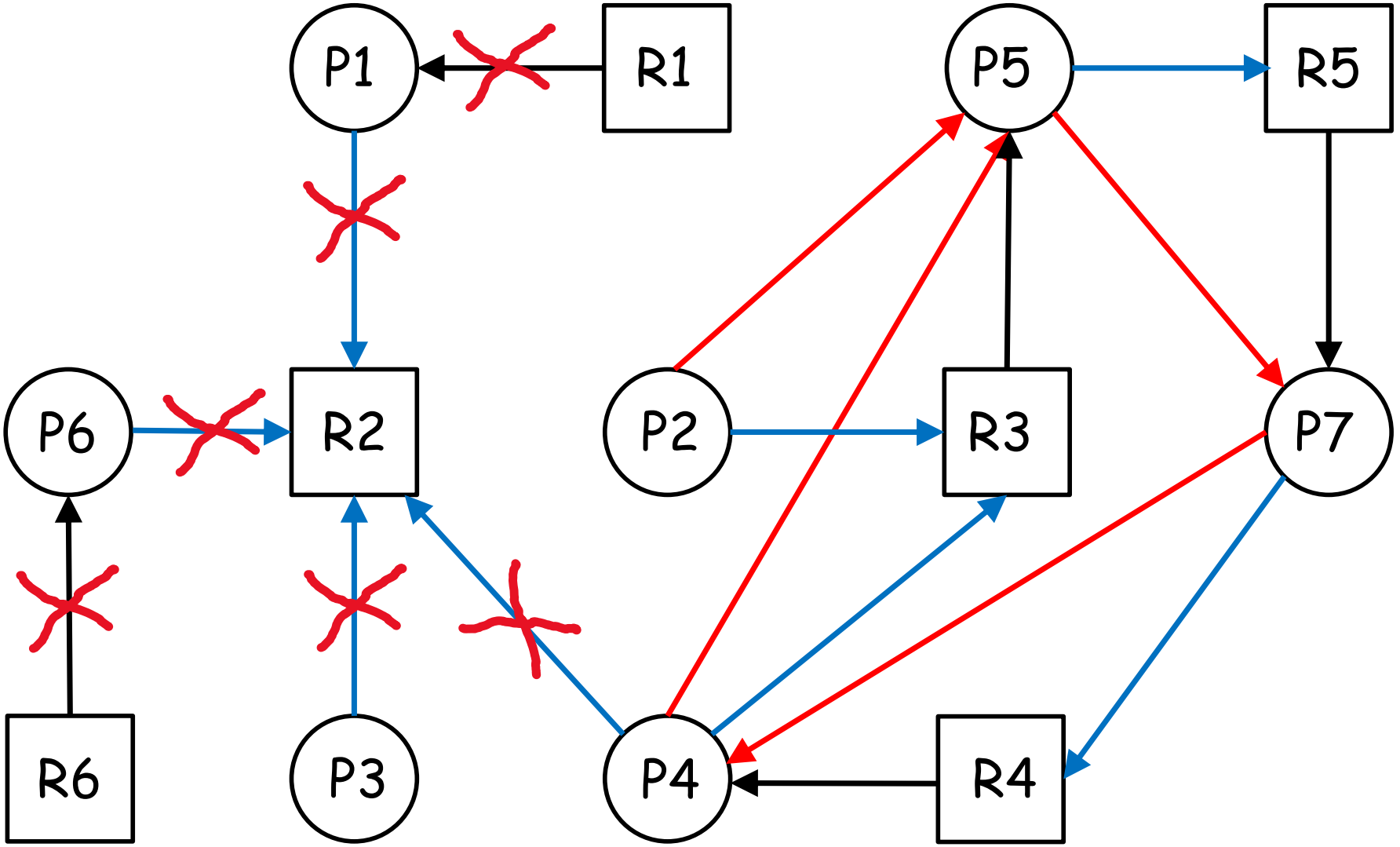
Soluzione



Soluzione

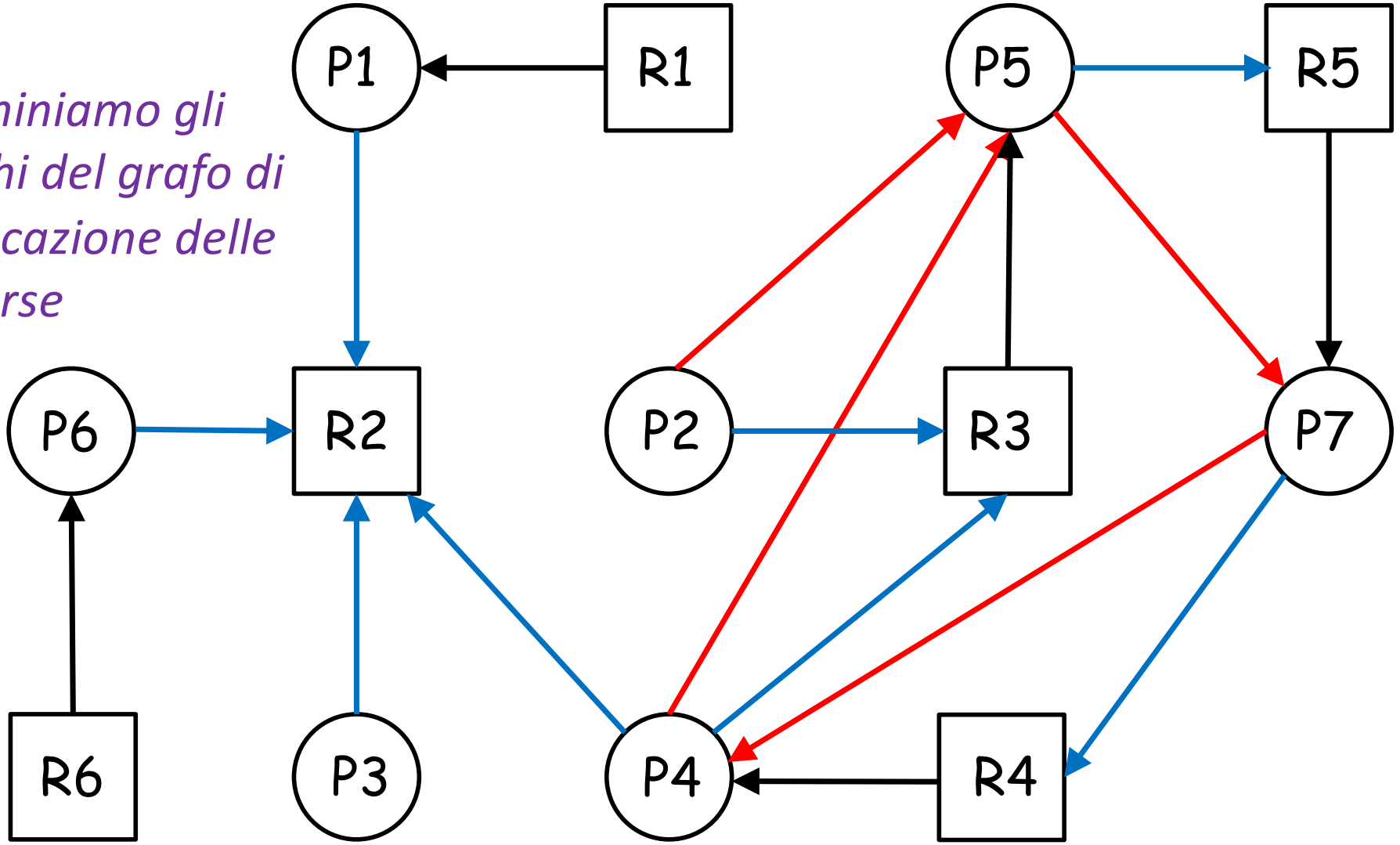


Soluzione

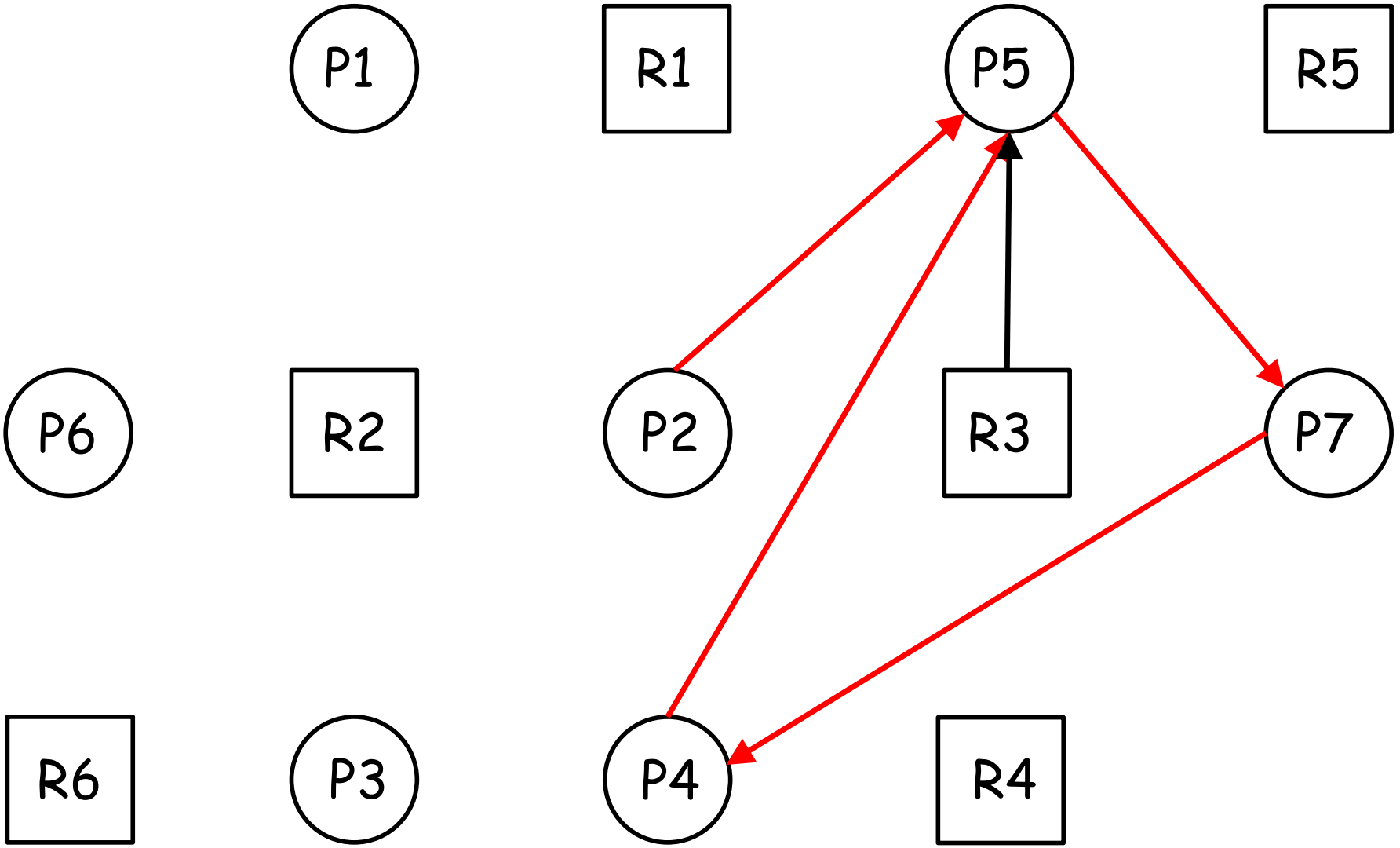


Soluzione

Eliminiamo gli archi del grafo di allocazione delle risorse

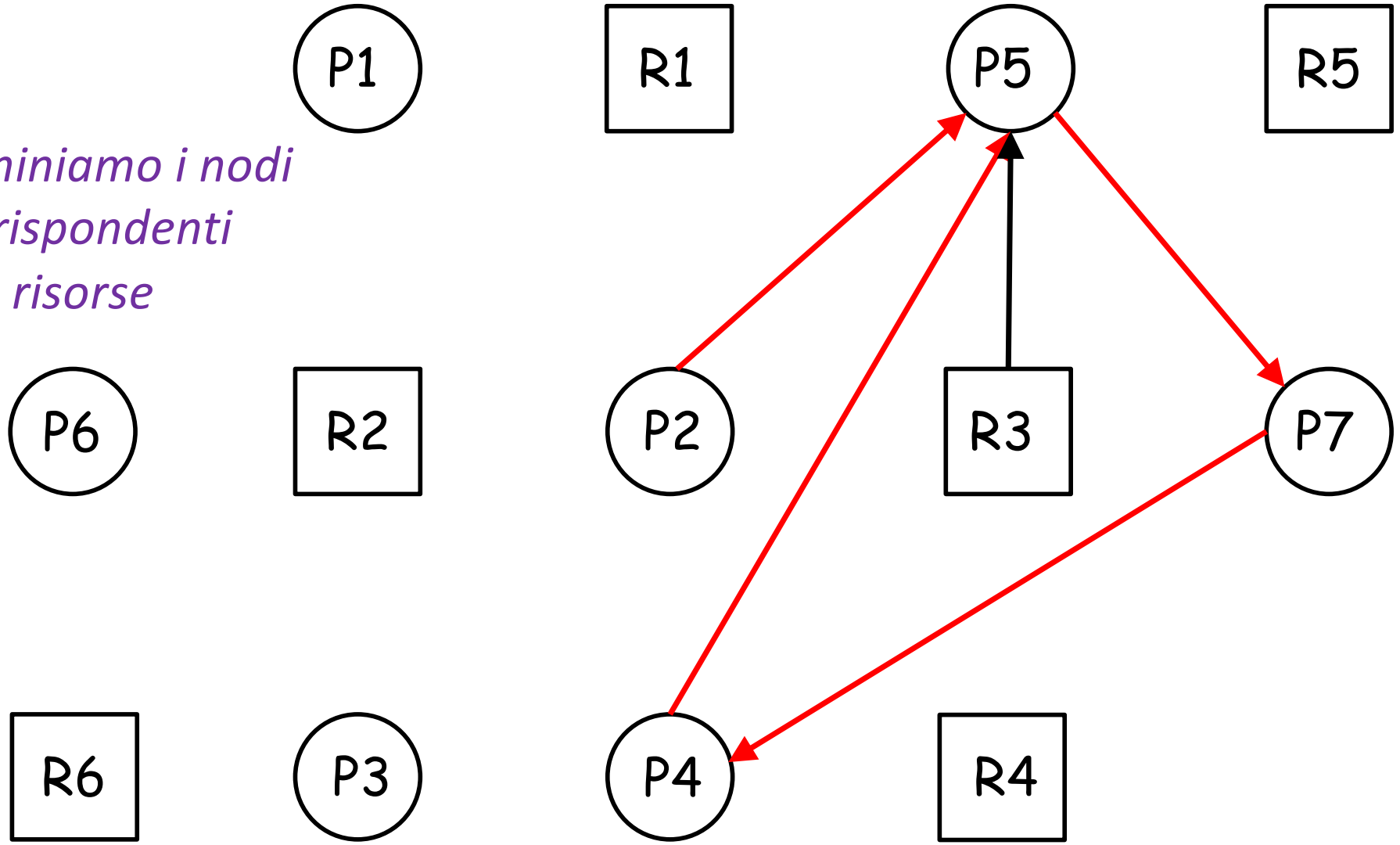


Soluzione



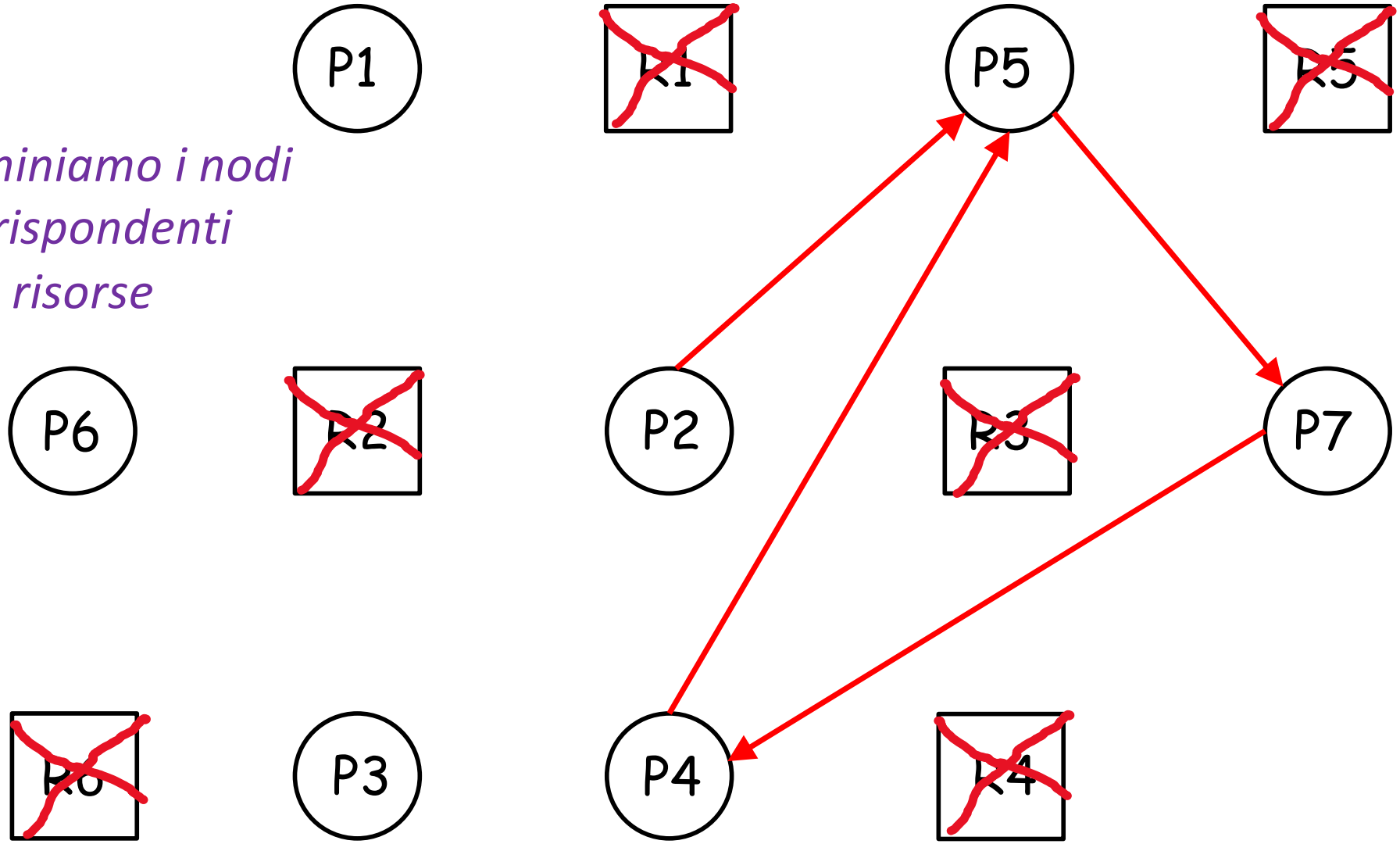
Soluzione

Eliminiamo i nodi corrispondenti alle risorse



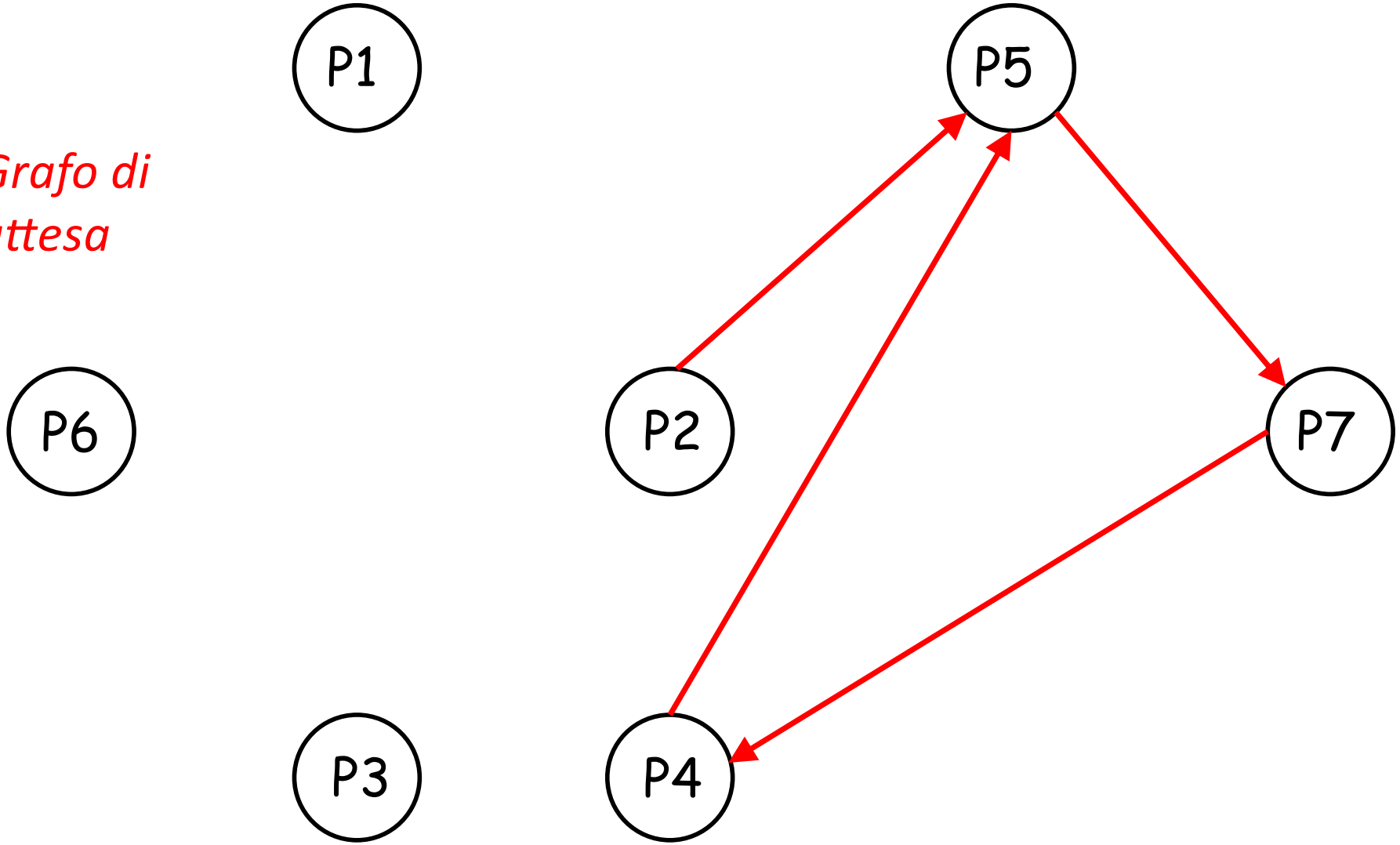
Soluzione

Eliminiamo i nodi corrispondenti alle risorse



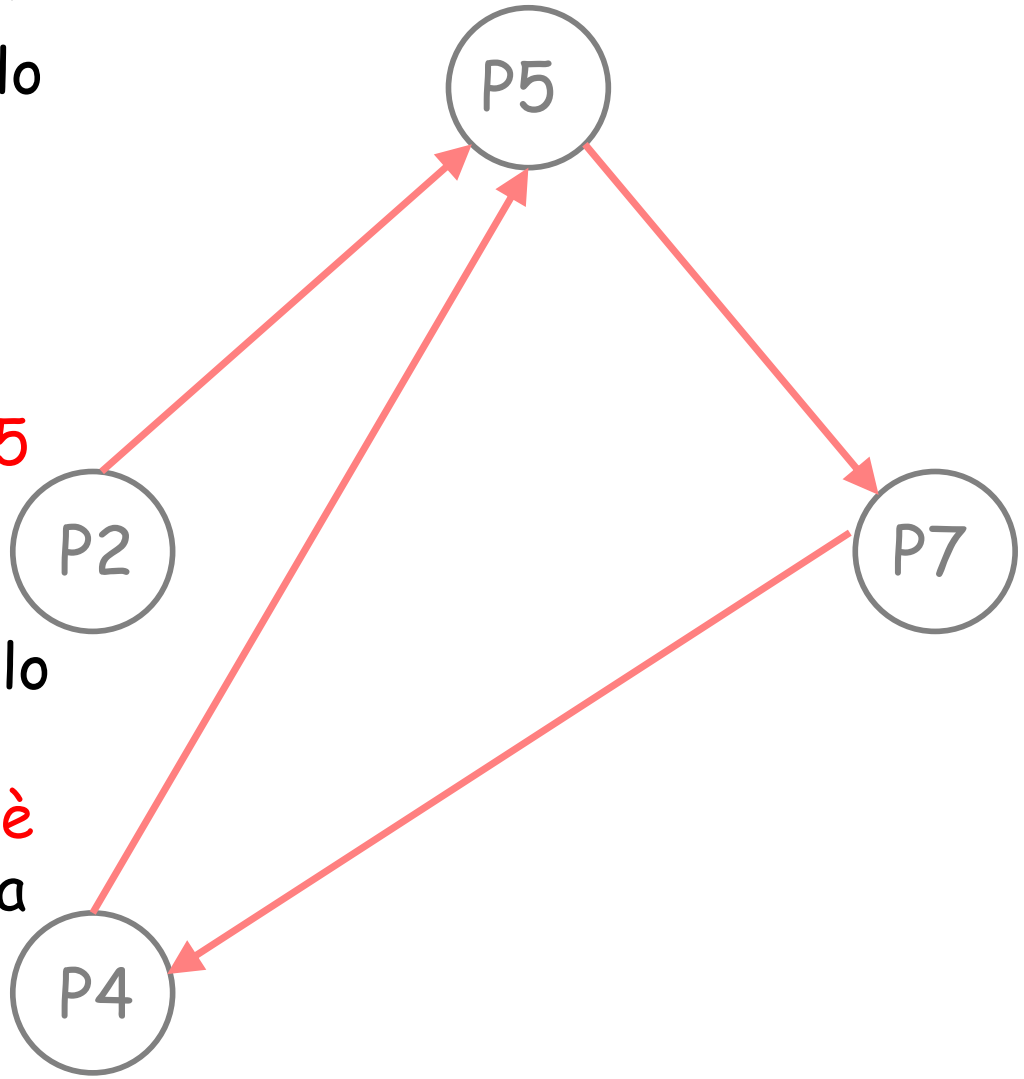
Soluzione

Grafo di attesa



Soluzione

Poiché c'è una sola risorsa per ogni tipo, l'esistenza di un ciclo nel grafo di attesa implica l'esistenza di uno stallo.
Nel caso specifico, il ciclo, e quindi lo stallo, coinvolge direttamente i processi *P4*, *P5* e *P7*. Le risorse coinvolte si ricavano dal grafo di allocazione delle risorse e nello specifico sono *R3*, *R4* e *R5*.
Inoltre, anche il processo *P2* è in stallo perché in attesa della risorsa *R3* assegnata al processo *P5* che è in stallo (e quindi mai rilasciata).



Esercizio 3 (vale 4 punti)

Un sistema con 3 processi P1, P2, e P3, e 4 tipi di risorse seriali e non-prelazionabili R1, R2, R3 e R4, rispettivamente con 2, 4, 3 e 4 istanze, adotta nei confronti dello stallo una politica di rilevamento e ripristino.

Inizialmente i processi non possiedono risorse.

Si consideri la seguente sequenza di richieste:

1) P1 richiede 1 unità di R1,
2) P2 richiede 2 unità di R2,
3) P3 richiede 2 unità di R3,
4) P1 richiede 2 unità di R4,
5) P2 richiede 3 unità di R4,
6) P1 richiede 3 unità di R2,
7) P3 richiede 1 unità di R1.

Esercizio 3 (vale 4 punti)

Mostrare come evolve il sistema compilando la tabella sottostante (una riga per ogni richiesta).

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init										
(1)										
(2)										
(3)										
(4)										
(5)										
(6)										
(7)										

Esercizio 3 (vale 4 punti)

Mostrare come evolve il sistema compilando la tabella sottostante (una riga per ogni richiesta).

…

Quindi, rispondere alle seguenti domande:

a) si supponga che, dopo l'ultima richiesta, P3 richieda 2 ulteriori unità di R3; questa sequenza porta al deadlock?

b) si supponga che, sempre dopo l'ultima richiesta, P3 termini rilasciando le risorse possedute; si ha deadlock in questo caso?

Argomentare le risposte.

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init										
(1)										
(2)										
(3)										
(4)										
(5)										
(6)										
(7)										

Init: la situazione iniziale ricavata dai dati del problema

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init										
(1)										
(										
(										
(										
(										
(										
(										

Un sistema con 3 processi P1, P2, e P3, e 4 tipi di risorse seriali e non-prelazionabili R1, R2, R3 e R4, rispettivamente con 2, 4, 3 e 4 istanze, adotta nei confronti dello stallo una politica di rilevamento e ripristino.
Inizialmente i processi non possiedono risorse.

Init: la situazione iniziale ricavata dai dati del problema

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)										
(										
(										
(										
(										
(										
(										

Un sistema con 3 processi P1, P2, e P3, e 4 tipi di risorse seriali e non-prelazionabili R1, R2, R3 e R4, rispettivamente con 2, 4, 3 e 4 istanze, adotta nei confronti dello stallo una politica di rilevamento e ripristino.
Inizialmente i processi non possiedono risorse.

Init: la situazione iniziale ricavata dai dati del problema

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)										
(2)										
(3)										
(4)										
(5)										
(6)										
(7)										

(1): P1 richiede 1 unità di R1

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)										
(3)										
(4)										
(5)										
(6)										
(7)										

(1): P1 richiede 1 unità di R1

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)										
(3)										
(4)										
(5)										
(6)										
(7)										

(2): P2 richiede 2 unità di R2

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)										
(4)										
(5)										
(6)										
(7)										

(2): P2 richiede 2 unità di R2

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)										
(4)										
(5)										
(6)										
(7)										

(3): P3 richiede 2 unità di R3

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)										
(5)										
(6)										
(7)										

(3): P3 richiede 2 unità di R3

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)										
(5)										
(6)										
(7)										

(4): P1 richiede 2 unità di R4

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)										
(6)										
(7)										

(4): P1 richiede 2 unità di R4

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)										
(6)										
(7)										

(5): P2 richiede 3 unità di R4

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)										
(7)										

(5): P2 richiede 3 unità di R4

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)										
(7)										

(6): P1 richiede 3 unità di R2

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)	1	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	0020
(7)										

(6): P1 richiede 3 unità di R2

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)	1	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	0020
(7)										

(7): P3 richiede 1 unità di R1

Soluzione

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)	1	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	0020
(7)	0	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	1020

(7): P3 richiede 1 unità di R1

Soluzione (a)

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)	1	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	0020
(7)	0	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	1020

a) si supponga che, dopo l'ultima richiesta, P3 richieda 2 ulteriori unità di R3; questa sequenza porta al deadlock?

Soluzione (a)

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)	1	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	0020
(7)	0	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	1020

a) Se P3 richiede 2 ulteriori unità di R3, dato che queste non sono disponibili, si verifica un deadlock che coinvolge tutti e tre i processi.

Soluzione (a)

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)	1	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	0020
(7)	0	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	1020

P1 e P2 sono in deadlock perché P1 aspetta che P2 rilasci le risorse R2 e P2 aspetta che P1 rilasci le risorse R4. P3 è in deadlock poiché fa una richiesta (2 unità di R3) che non può essere soddisfatta con le risorse attualmente disponibili.

Soluzione (b)

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)	1	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	0020
(7)	0	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	1020

b) si supponga che, sempre dopo l'ultima richiesta, P3 termini rilasciando le risorse possedute; si ha deadlock in questo caso?

Soluzione (b)

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)	1	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	0020
(7)	0	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	1020

b) Anche se P3 termina rilasciando le risorse possedute si verifica un deadlock, ma stavolta sono coinvolti solo i processi P1 e P2.

Soluzione (b)

richiesta	Disponibilità				Processo P1		Processo P2		Processo P3	
	R1	R2	R3	R4	Sospeso?	Risorse assegnate	Sospeso ?	Risorse assegnate	Sospeso ?	Risorse assegnate
init	2	4	3	4	NO	0000	NO	0000	NO	0000
(1)	1	4	3	4	NO	1000	NO	0000	NO	0000
(2)	1	2	3	4	NO	1000	NO	0200	NO	0000
(3)	1	2	1	4	NO	1000	NO	0200	NO	0020
(4)	1	2	1	2	NO	1002	NO	0200	NO	0020
(5)	1	2	1	2	NO	1002	SI R4(P1)	0200	NO	0020
(6)	1	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	0020
(7)	0	2	1	2	SI R2(P2)	1002	SI R4(P1)	0200	NO	1020

Infatti, benché il processo P3 possa terminare l'esecuzione e rilasciare sia 1 unità di R1 che 2 unità di R3, P1 e P2 restano comunque in deadlock perché P1 aspetta che P2 rilasci le risorse R2 e P2 aspetta che P1 rilasci le risorse R4.

Esercizio 4 (vale 4 punti)

Un sistema con 4 processi *P1*-*P4* e altrettanti tipi di risorse seriali e non-prelazionabili *R1*, *R2*, *R3* e *R4*, rispettivamente di molteplicità 3, 5, 5 e 4, adotta nei confronti dello stallo una politica di rilevamento e ripristino.

Ad un dato istante di tempo tutti i processi sono bloccati e si è raggiunto lo stato di stallo caratterizzato dalla seguenti matrici

Allocation

P1	0	1	1	0
P2	1	1	2	0
P3	0	1	1	2
P4	2	1	1	0

Request

1	1	0	2
0	2	1	1
2	2	0	0
0	0	2	3

Esercizio 4 (vale 4 punti)

Per la sua eliminazione si considerano le seguenti azioni alternative:

a) soppressione del processo *P1*;
b) sottrazione di 2 istanze di tipo *R1* al processo *P4*.

Si analizzi ciascuna di queste alternative e si verifichi se consente di eliminare lo stallo. Giustificare le risposte mostrando i dettagli dei ragionamenti effettuati.

Soluzione

Il vettore *Available*, delle risorse attualmente disponibili, è ricavabile sottraendo alle risorse complessive del sistema quelle già allocate; abbiamo così

$$Available = [3,5,5,4] - \sum_i Allocation_i$$
$$= [3,5,5,4] - [3,4,5,2]$$
$$= [0,1,0,2]$$

Il sistema è effettivamente in uno stato di stallo perché, non essendoci nello stato corrente risorse disponibili di tipo *R*1 e *R*3, le richieste dei quattro processi non sono soddisfacibili e fanno sì che tutti i processi siano sospesi.

Soluzione (a)

Cominciamo a esaminare l'effetto della prima azione: *soppressione del processo P1* con conseguente recupero delle risorse ad esso allocate.

Stato iniziale

Allocation

0	1	1	0
1	1	2	0
0	1	1	2
2	1	1	0

Request

1	1	0	2
0	2	1	1
2	2	0	0
0	0	2	3

Available

0	1	0	2

Soluzione (a)

Cominciamo a esaminare l'effetto della prima azione: *soppressione del processo P1* con conseguente recupero delle risorse ad esso allocate.
Lo stato ottenuto dopo tale azione è il seguente:

Allocation

1	1	2	0
0	1	1	2
2	1	1	0

Request

0	2	1	1
2	2	0	0
0	0	2	3

Available

0	2	1	2

Soluzione (a)

Si tratta a questo punto di verificare se lo stallo è stato eliminato.

A tale scopo, applichiamo l'algoritmo per il rilevamento dello stallo.

Dopo aver inizializzato *Work* con *Available* (= [0,2,1,2]), abbiamo che

a) al primo ciclo, il processo P2 può terminare perché la sua richiesta [0,2,1,1] è inferiore alla disponibilità; quindi, recuperando le risorse di *P2*, *Work* diventa [1,3,3,2];

Allocation

1	1	2	0
0	1	1	2
2	1	1	0

Request

0	2	1	1
2	2	0	0
0	0	2	3

Work

0	2	1	2

Soluzione (a)

Si tratta a questo punto di verificare se lo stallo è stato eliminato.

A tal fine applichiamo l'algoritmo per il rilevamento dello stallo.

Dopo aver inizializzato *Work* e *Finish* (= [0,2,1,2]), abbiamo:

a) al primo ciclo, il processo P2 può terminare perché la sua richiesta [0,2,1,1] è inferiore alla disponibilità; quindi, recuperando le risorse di *P2*, *Work* diventa [1,3,3,2];

Allocation

1	1	2	0
0	1	1	2
2	1	1	0

Request

~~0~~	~~2~~	~~1~~	~~1~~
2	2	0	0
0	0	2	3

Work

1	3	3	2

b) al secondo ciclo, però, né il processo P3 né il processo P4 possono terminare perché le loro richieste, [2,2,0,0] e [0,0,2,3] rispettivamente, non sono soddisfacibili con le risorse all'attualmente disponibili nel sistema.

Quindi, l'azione considerata non elimina lo stallo.

Soluzione (b)

Passiamo ora ad esaminare l'effetto della seconda azione: *sottrazione di 2 istanze di tipo R1 al processo P4* con conseguente recupero delle risorse prelazionate.

Stato iniziale

Allocation

0	1	1	0
1	1	2	0
0	1	1	2
2	1	1	0

Request

1	1	0	2
0	2	1	1
2	2	0	0
0	0	2	3

Available

0	1	0	2

Soluzione (b)

Passiamo ora ad esaminare l'effetto della seconda azione: *sottrazione di 2 istanze di tipo R1 al processo P4* con conseguente recupero delle risorse prelazionate.
Lo stato ottenuto dopo tale azione è il seguente:

Allocation

0	1	1	0
1	1	2	0
0	1	1	2
0	1	1	0

Request

1	1	0	2
0	2	1	1
2	2	0	0
2	0	2	3

Available

2	1	0	2

Soluzione (b)

Applichiamo nuovamente l'algoritmo per il rilevamento dello stallo. Dopo aver inizializzato *Work* con *Available* (= [2,1,0,2]), abbiamo che

a) al primo ciclo, il processo P1 può terminare perché la sua richiesta [1,1,0,2] è inferiore a *Work* = [2,1,0,2]; quindi, recuperando le risorse di *P1*, *Work* diventa [2,2,1,2];

Allocation

0	1	1	0
1	1	2	0
0	1	1	2
0	1	1	0

Request

1	1	0	2
0	2	1	1
2	2	0	0
2	0	2	3

Work

2	1	0	2

Soluzione (b)

Applichiamo nuovamente l'algoritmo per il rilevamento dello stallo. Dopo aver inizializzato *Work* con *Available* (= [2,1,0,2]), abbiamo che

a) al primo ciclo, il processo P1 può terminare perché la sua richiesta [1,1,0,2] è inferiore a *Work* = [2,1,0,2]; quindi, recuperando le risorse di *P1*, *Work* diventa [2,2,1,2];
b) al secondo ciclo, il processo P2 può terminare perché [0,2,1,1] ≤ [2,2,1,2]; *Work* diventa: [3,3,3,2];

Allocation

~~0~~	~~1~~	~~1~~	~~0~~
1	1	2	0
0	1	1	2
0	1	1	0

Request

~~1~~	~~1~~	~~0~~	~~2~~
0	2	1	1
2	2	0	0
2	0	2	3

Work

2	2	1	2

Soluzione (b)

Applichiamo nuovamente l'algoritmo per il rilevamento dello stallo. Dopo aver inizializzato *Work* con *Available* (= [2,1,0,2]), abbiamo che

a) al primo ciclo, il processo P1 può terminare perché la sua richiesta [1,1,0,2] è inferiore a *Work* = [2,1,0,2]; quindi, recuperando le risorse di *P1*, *Work* diventa [2,2,1,2];
b) al secondo ciclo, il processo P2 può terminare perché [0,2,1,1] ≤ [2,2,1,2]; *Work* diventa: [3,3,3,2];
c) al terzo ciclo, il processo P3 può terminare perché [2,2,0,0] ≤ [3,3,3,2]; *Work* diventa: [3,4,4,4];

Allocation

~~0~~	~~1~~	~~1~~	~~0~~
~~1~~	~~1~~	~~2~~	~~0~~
0	1	1	2
0	1	1	0

Request

~~1~~	~~1~~	~~0~~	~~2~~
~~0~~	~~2~~	~~1~~	~~1~~
2	2	0	0
2	0	2	3

Work

3	3	3	2

Soluzione (b)

Applichiamo nuovamente l'algoritmo per il rilevamento dello stallo. Dopo aver inizializzato *Work* con *Available* (= [2,1,0,2]), abbiamo che

Allocation				Request				Work			
~~0~~	~~1~~	~~1~~	~~0~~	~~1~~	~~1~~	~~0~~	~~2~~	3	4	4	4
~~1~~	~~1~~	~~2~~	~~0~~	~~0~~	~~2~~	~~1~~	~~1~~				
~~0~~	~~1~~	~~1~~	~~2~~	~~2~~	~~2~~	~~0~~	~~0~~				
0	1	1	0	2	0	2	3				

d) infine, al quarto ciclo, il processo P4 può terminare perché [2,0,2,3] ≤ [3,4,4,4]; *Work* diventa: [3,5,5,4].

Soluzione (b)

Applichiamo nuovamente l'algoritmo per il rilevamento dello stallo. Dopo aver inizializzato *Work* con *Available* (= [2,1,0,2]), abbiamo che

a) al primo ciclo, il processo P1 può terminare perché la sua richiesta [1,1,0,2] è inferiore a *Work* = [2,1,0,2]; quindi, recuperando le risorse di *P1*, *Work* diventa [2,2,1,2];
b) al secondo ciclo, il processo P2 può terminare perché [0,2,1,1] ≤ [2,2,1,2]; *Work* diventa: [3,3,3,2];
c) al terzo ciclo, il processo P3 può terminare perché [2,2,0,0] ≤ [3,3,3,2]; *Work* diventa: [3,4,4,4];
d) infine, al quarto ciclo, il processo P4 può terminare perché [2,0,2,3] ≤ [3,4,4,4]; *Work* diventa: [3,5,5,4].

Concludendo, nello stato ottenuto sottraendo 2 risorse di tipo *R1* al processo *P4*, le richieste correnti dei processi possono essere soddisfatte nell'ordine ⟨*P1*,*P2*,*P3*,*P4*⟩.

Quindi, l'azione considerata elimina lo stallo.

Esercizio 5 (vale 4 punti)

Dati tre processi A, B e C e tre risorse singole Q, R e S, utilizzabili in mutua esclusione e senza possibilità di prerilascio, supponiamo che il SO assegni le risorse al processo richiedente alla sola condizione che la risorsa richiesta sia disponibile.

Inizialmente tutte le risorse sono disponibili.

Si consideri la seguente sequenza di richieste e rilasci:

1. A richiede Q;
2. C richiede S;
3. C richiede Q;
4. B richiede R;
5. B richiede S;
6. A rilascia Q;
7. A richiede S;
8. C richiede R;
9. A richiede S;
10. C rilascia Q;
11. B rilascia R.

Esercizio 5 (vale 4 punti)

Mostrare come evolve il sistema compilando la tabella sottostante (una riga per ogni evento)

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1							
2							
3							
4							
•							
•							
•							
•							

Esercizio 5 (vale 4 punti)

Mostrare come evolve il sistema compilando la tabella sottostante (una riga per ogni evento)

e, quindi, rispondere alle seguenti domande:

a) la sequenza può essere interamente eseguita?
b) eventualmente, quali azioni non possono essere eseguite e perché?
c) si raggiunge uno stallo?
d) eventualmente, con quale azione si raggiunge lo stallo?

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							

1. A richiede Q;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							

1. A richiede Q;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							

2. C richiede S;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3							
4							
5							
6							
7							
8							
9							
10							
11							

2. C richiede S;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3							
4							
5							
6							
7							
8							
9							
10							
11							

3. C richiede Q;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4							
5							
6							
7							
8							
9							
10							
11							

3. C richiede Q;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4							
5							
6							
7							
8							
9							
10							
11							

4. B richiede R;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5							
6							
7							
8							
9							
10							
11							

4. B richiede R;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5							
6							
7							
8							
9							
10							
11							

5. B richiede S;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6							
7							
8							
9							
10							
11							

5. B richiede S;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6							
7							
8							
9							
10							
11							

6. A rilascia Q;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7							
8							
9							
10							
11							

6. A rilascia Q;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7							
8							
9							
10							
11							

7. A richiede S;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8							
9							
10							
11							

7. A richiede S;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8							
9							
10							
11							

8. C richiede R;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9							
10							
11							

8. C richiede R;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9							
10							
11							

9. A richiede S;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10							
11							

9. A richiede S;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10							
11							

10. C rilascia Q;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11							

10. C rilascia Q;

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11							

11. B rilascia R.

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q

11. B rilascia R.

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q

Tabella finale

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q

a) la sequenza può essere interamente eseguita?

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q

a) No, la sequenza non può essere interamente eseguita.

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q

b) eventualmente, quali azioni non possono essere eseguite e perché?

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q

b) 9), 10) e 11), perché i processi che dovrebbero eseguirle sono sospesi.

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q

c) si raggiunge uno stallo?

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q

c) Si, la sequenza provoca uno stallo.

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q

d) eventualmente, con quale azione si raggiunge lo stallo?

Soluzione

azione	Eseguibile?	Processo A		Processo B		Processo C	
		Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate	Sospeso?	Risorse assegnate
1	SI	NO	Q	NO	Ø	NO	Ø
2	SI	NO	Q	NO	Ø	NO	S
3	SI	NO	Q	NO	Ø	SI, Q(A)	S
4	SI	NO	Q	NO	R	SI, Q(A)	S
5	SI	NO	Q	SI, S(C)	R	SI, Q(A)	S
6	SI	NO	Ø	SI, S(C)	R	NO	S, Q
7	SI	SI, S(C)	Ø	SI, S(C)	R	NO	S, Q
8	SI	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
9	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
10	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q
11	NO	SI, S(C)	Ø	SI, S(C)	R	SI, R(B)	S, Q

d) l'azione 8) crea una situazione di attesa circolare

Soluzione

In conclusione:

a) la sequenza non può essere interamente eseguita;
b) le azioni 9), 10) e 11) non possono essere eseguite, perché i processi che dovrebbero eseguirle sono sospesi;
c) la sequenza provoca uno stallo;
d) lo stallo si raggiunge per effetto dell'azione 8), con la quale si crea una situazione di attesa circolare dal momento che il processo B aspetta la risorsa S e quindi il processo C, che a sua volta aspetta la risorsa R e quindi il processo B. Anche il processo A è bloccato perché attende il processo C.