Gestione del deadlock (seconda parte)

Algoritmi per evitare il deadlock

- In presenza di **una sola istanza** per ciascun tipo di risorsa:

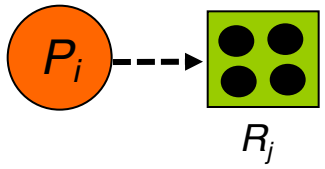
Usare il grafo di allocazione delle risorse

- In presenza di **più istanze** per tipo di risorsa:

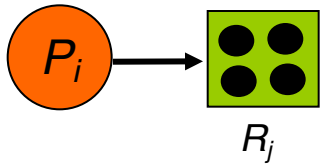
Algoritmo del banchiere (Banker's algorithm)

Uso del grafo di allocazione delle risorse per evitare il deadlock

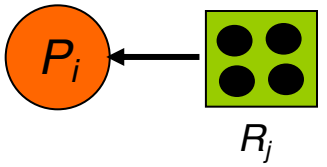
- Un ***arco di reclamo*** (*claim edge*) $P_i \rightarrow R_j$, rappresentato con una linea tratteggiata, indica che il processo P_i può richiedere la risorsa R_j in un qualsiasi momento futuro



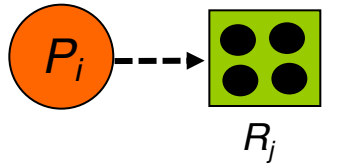
- Quando il processo richiede la risorsa, l'arco di reclamo è convertito in un ***arco di richiesta***



- Un arco di richiesta è convertito in un ***arco di assegnazione*** quando la risorsa è allocata al processo



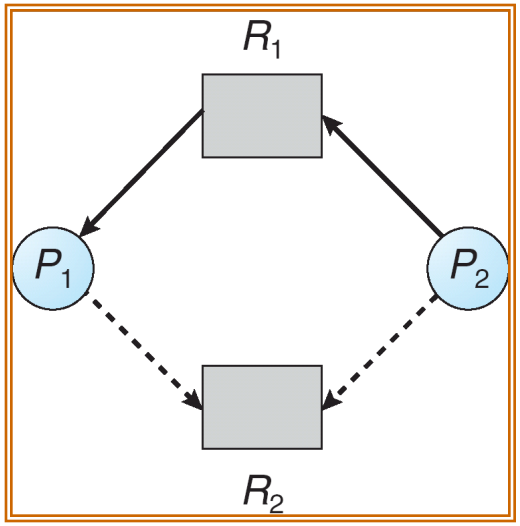
- Quando la risorsa è rilasciata dal processo, l'arco di assegnazione è convertito di nuovo in un ***arco di reclamo***.

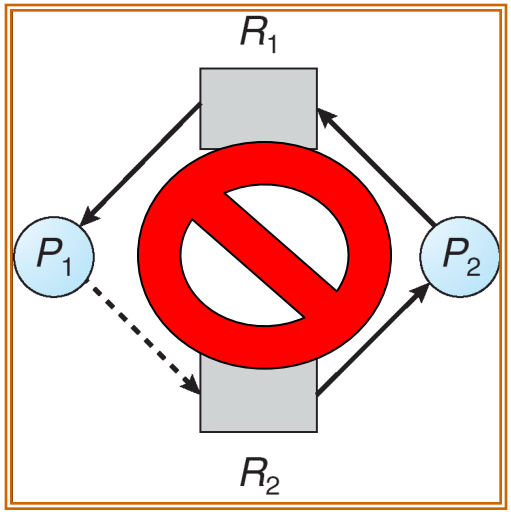


Le risorse devono essere richieste *a priori* nel sistema

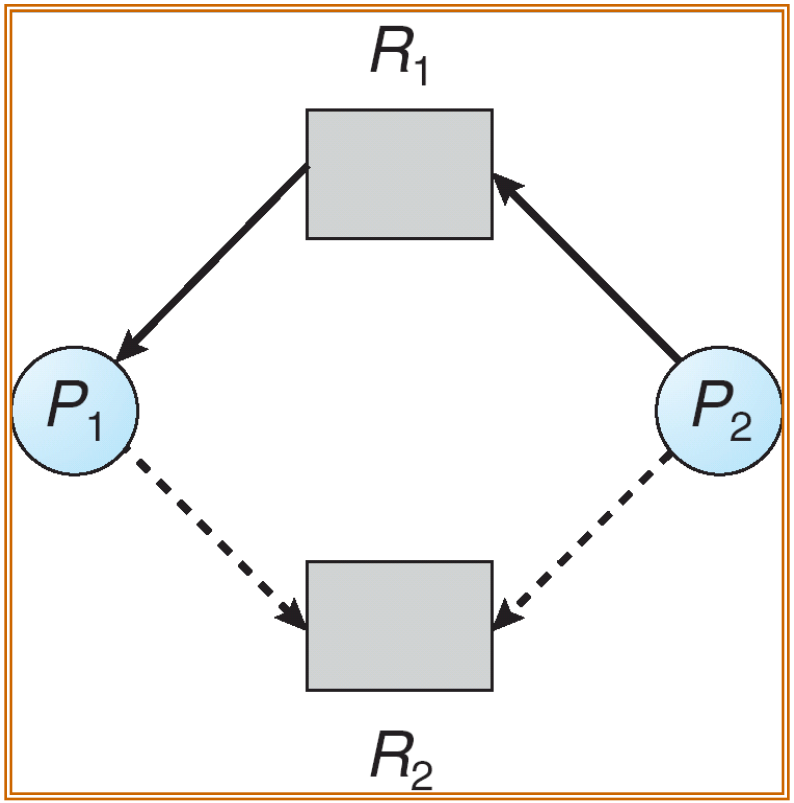
Uso del grafo di allocazione delle risorse per evitare il deadlock

- Si supponga che il processo P_i richieda una risorsa R_j
- La richiesta **può essere soddisfatta** solo se la conversione dell'arco di richiesta $P_i \rightarrow R_j$ nell'arco di assegnazione $R_j \rightarrow P_i$ **non causa la formazione di un ciclo** nel grafo di allocazione delle risorse

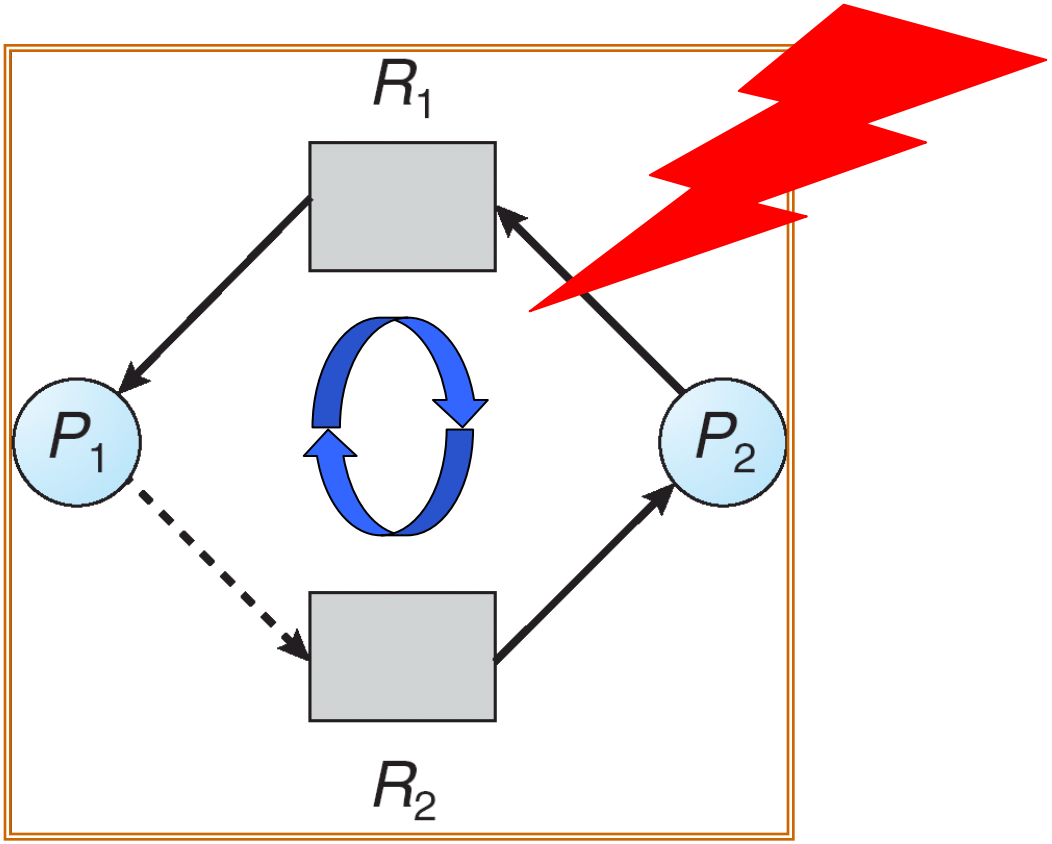




Uso del grafo di allocazione delle risorse per evitare il deadlock



Stato non sicuro in un grafo di assegnazione delle risorse



Algoritmo del banchiere

- Permette di gestire più istanze per tipo di risorsa nella rilevazione del deadlock.
- **Ogni processo deve dichiarare a priori il numero massimo di istanze di ciascun tipo di risorse di cui necessita per eseguire le sue operazioni.**
- Quando un processo richiede una risorsa potrebbe dover aspettare per ottenerla.
- Quando un processo ottiene tutte le tue risorse, le può usare e le deve restituirle in un tempo finito.

Strutture dati usate dall'algoritmo del banchiere

Sia ***n*** = numero di processi, ed ***m*** = numero di tipi di risorse.

- **Available**: *vettore* di lunghezza *m*. Se Available[*j*] = *k*, significa che sono disponibili *k* istanze del tipo di risorsa R_j
- **Max**: *matrice n x m*. Se *Max*[*i,j*] = *k*, allora il processo P_i può richiedere al massimo *k* istanze del tipo di risorsa R_j
- **Allocation**: *matrice n* x *m*. Se Allocation[*i,j*] = *k*, allora al processo P_i sono attualmente assegnate *k* istanze del tipo di risorsa R_j
- **Need**: *matrice n* x *m*. Se *Need*[*i,j*] = *k*, significa che il processo P_i per completare il suo compito può avere bisogno di altre *k* istanze del tipo di risorsa R_j

$$\boldsymbol{Need\,[i,j] = Max[i,j] - Allocation\,[i,j]}$$

Strutture dati usate dall'algoritmo del banchiere

Sia ***n*** = numero di processi, e

m = numero di tipi di risorse.

- **Available [1..m]** # risorse disponibili per tipo

- **Max [1..n, 1..m]** # max richiesta per tipo di ogni processo

- **Allocation [1..n, 1..m]** # di risorse attualmente assegnate

- **Need [1..n, 1..m]** # risorse da richiedere

$$Need\,[i,j] = Max[i,j] - Allocation\,[i,j]$$

Alg. del banchiere: assegnazione delle risorse

Request$_i$ = vettore delle richieste del processo P_i.

Se $Request_i[j] = k$, allora P_i vuole k istanze di R_j

1. Se $Request_i \leq Need_i$ salta al passo 2. Altrimenti riporta una condizione d'errore, in quanto il processo ha superato il numero massimo di richieste;
2. Se $Request_i \leq Available$, salta al passo 3. **Altrimenti P_i deve aspettare**, in quanto le risorse non sono disponibili
3. Simula l'assegnazione al processo P_i delle risorse richieste, modificando come segue lo stato di assegnazione delle risorse:

$$Available = Available - Request_i;$$

$$Allocation_i = Allocation_i + Request_i;$$

$$Need_i = Need_i - Request_i;$$

Verifica Stato sicuro (slide successiva)

   A. Se **Stato sicuro** $\Rightarrow$ le risorse sono assegnate a P_i
   B. **Altrimenti** $\Rightarrow$ P_i deve attendere e si ripristina il vecchio stato di allocazione delle risorse

Alg. del banchiere: verifica dello stato sicuro

1. Siano ***Work*** e ***Finish*** due vettori di lunghezza rispettivamente *m* ed *n*, inizializzati come segue:

   Work = *Available;*

   Finish [*i*] = *false* per *i* = 0, 1, …, *n*-1; ** indica i processi che si possono completare **/

2. Cerca un indice *i* tale che valgano entrambe le condizioni seguenti:

   (a) *Finish* [*i*] = *false;*

   (b) $Need_i \leq Work;$

   Se tale *i* viene trovato, esegui il passo 3

   Altrimenti, se tale *i* non esiste, salta al passo 4

3. *Work* = *Work* + $Allocation_i$; */* si aggiungono a Work le risorse assegnate a Pi*/*

   Finish[*i*] = *true*

   Torna al passo 2

4. **Se *Finish* [*i*] == *true* per ogni *i***, allora **il sistema è in uno stato sicuro**

   Altrimenti il sistema **non è in uno stato sicuro**.

Algoritmo del banchiere: un esempio

- 5 processi: $P_0 \ldots P_4$;

  3 tipi di risorse:

  ***A* (10 istanze)**, ***B* (5 istanze)**, e ***C* (7 istanze)**

Istantanea all'istante T_0:

	Allocation	***Max***	***Available***
	A B C	***A B C***	***A B C***
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Esempio (cont.)

- Il contenuto della matrice ***Need*** è definito come ***Max – Allocation***

	Allocation	***Max***	***Available***	***Need***
	A B C	***A B C***	***A B C***	***A B C***
P_0	0 1 0	7 5 3	3 3 2	7 4 3
P_1	2 0 0	3 2 2		1 2 2
P_2	3 0 2	9 0 2		6 0 0
P_3	2 1 1	2 2 2		0 1 1
P_4	0 0 2	4 3 3		4 3 1

Esempio (cont.)

- Il contenuto della matrice ***Need*** è definito come ***Max – Allocation***

	Allocation	***Max***	***Available***	***Need***	
	A B C	***A B C***	***A B C***	***A B C***	
P_0	0 1 0	7 5 3	3 3 2	7 4 3	**5**
P_1	2 0 0	3 2 2		1 2 2	**1**
P_2	3 0 2	9 0 2		6 0 0	**4**
P_3	2 1 1	2 2 2		0 1 1	**2**
P_4	0 0 2	4 3 3		4 3 1	**3**

- Il sistema è in uno stato sicuro in quanto la sequenza

  < P_1, P_3, P_4, P_2, P_0>

  soddisfa i criteri dello stato sicuro.

Esempio (cont.): P_1 richiede (1,0,2)

- Verifica che
  - **Request$_1$** $\leq$ **Need**, vale a dire che (1,0,2) $\leq$ (1,2,2)
  - **Request$_1$** $\leq$ **Available**, vale a dire che (1,0,2) $\leq$ (3,3,2)

	Allocation	***Need***	***Available***
	A B C	***A B C***	***A B C***
P_0	0 1 0	7 4 3	2 3 0
P_1	3 0 2	0 2 0	
P_2	3 0 2	6 0 0	
P_3	2 1 1	0 1 1	
P_4	0 0 2	4 3 1	

- Eseguendo l'algoritmo che verifica lo stato sicuro, risulta che la sequenza **< P_1, P_3, P_4, P_0, P_2>** soddisfa i criteri di stato sicuro, quindi si può soddisfare immediatamente la richiesta di P_1

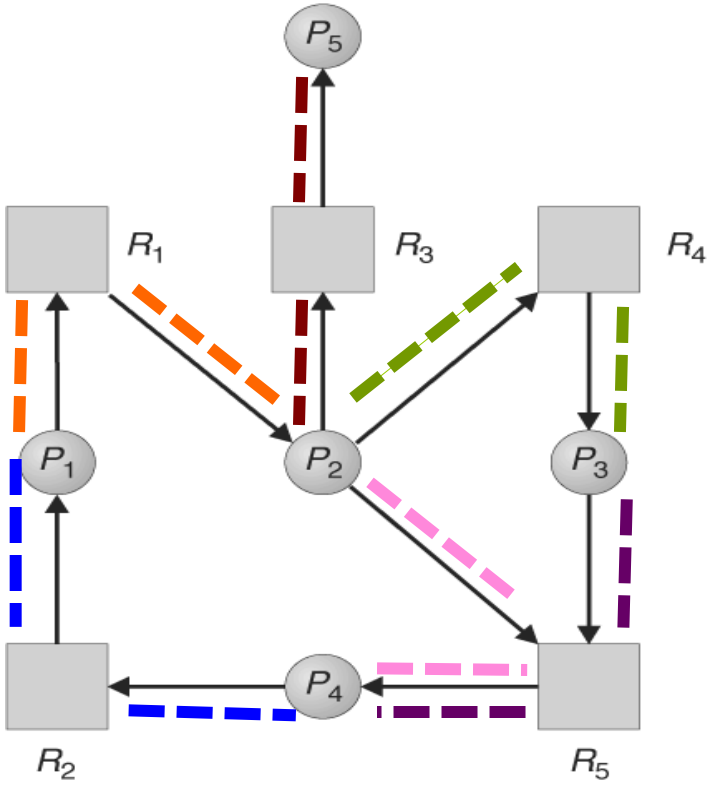
Rilevamento del deadlock

- Permettere al sistema di entrare in uno stato di deadlock
- Algoritmo di rilevamento (detection algorithm)
- Algoritmo di ripristino (recovery algorithm)

Istanza singola per ciascun tipo di risorsa

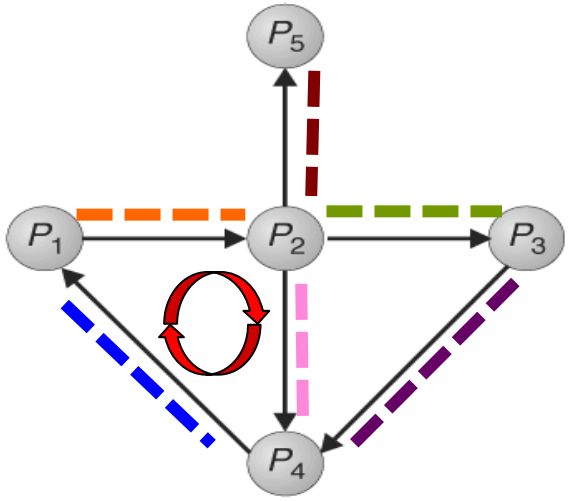
- Utilizzare un grafo d'attesa (*wait-for* graph)
  - I nodi sono processi
  - Un arco $P_i \rightarrow P_j$ indica the P_i attende che P_j rilasci una risorsa di cui ha bisogno P_i
- Invocare periodicamente un algoritmo che cerca cicli nel grafo. **La presenza di un ciclo indica una situazione di deadlock.**
- Un algoritmo per rilevare un ciclo in un grafo di n nodi richiede **O(n^2)** operazioni

Dal grafo di allocazione delle risorse al corrispondente grafo d'attesa



(a)

Grafo di allocazione delle risorse



(b)

Grafo d'attesa corrispondente

Uso degli algoritmi di rilevamento

- Quando, e quanto spesso, invocarli dipende da:
  - Frequenza con la quale si verifica un deadlock
  - Numero di processi che sarebbero influenzati dal deadlock
- **Si usano in sistemi che hanno esigenze particolari.**

Ripristino dal deadlock: terminazione di processi

1. **Terminazione** di **tutti i processi** in **deadlock**

   oppure

2. **Terminare un processo per volta** fino all'eliminazione del deadlock

- Nel secondo caso, la scelta del processo da terminare è influenzata da diversi fattori, tra i quali:
  - La priorità dei processi
  - L'elaborazione già effettuata dal processo, e quella ancora da effettuare per completare l'esecuzione
  - Le risorse usate dal processo
  - Le risorse necessarie al processo per completare l'esecuzione
  - Il tipo dei processi: interattivi o batch

Ripristino dal deadlock: preemption su risorse

- Si sottraggono risorse in successione ad alcuni processi e si assegnano ad altri finchè si ottiene l'interruzione del ciclo che ha determinato il deadlock

- Problemi:
  - **Selezione della vittima** – minimizzare i costi (ad esempio, considerare il tempo d'esecuzione già trascorso, prima di rimuovere una risorsa ad un processo)
  - **Rollback** – ritornare ad uno stato sicuro precedente, e riavviare il processo da quello stato
  - **Starvation** – bisognerebbe garantire che le risorse non siano sottratte sempre allo stesso processo