

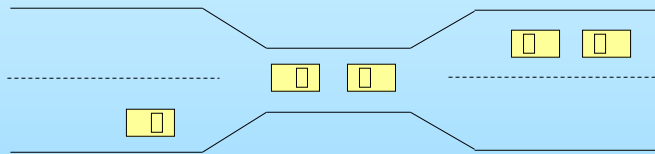
# Gestione dello Stallo

## Il problema dello stallo

- Un insieme di processi bloccati ognuno in possesso di una risorsa ed in attesa di acquisire una risorsa posseduta da un altro processo (nell'insieme).
- *Esempio*
  - Il sistema ha due unità nastro.
  - $P_1$  e  $P_2$  ognuno in possesso di una unità e ognuno ha bisogno di una seconda unità.
- *Esempio*
  - semafori  $A$  e  $B$ , inizializzati a 1

$P_0$	$P_1$
<code>wait (A);</code>	<code>wait(B)</code>
<code>wait (B);</code>	<code>wait(A)</code>

## Esempio del ponte



- Traffico solo in una direzione.
- Ogni sezione del ponte vista come risorsa.
- Se accade uno stallo, può essere risolto se una macchina torna indietro.
- Molte macchine potrebbero dover tornare indietro in caso di stallo.
- E' possibile avere attesa indefinita.

Sistemi Operativi A.A 2015/2016

## Modello del sistema

- Tipi di risorse  $R_1, R_2, \dots, R_m$   
*es: cicli CPU, spazio memoria, dispositivi I/O*
- Ogni tipo di risorsa  $R_i$  ha  $W_i$  istanze.
- Ogni processo usa una risorsa come segue:
  - richiesta
  - uso
  - rilascio

Sistemi Operativi A.A 2015/2016

## Caratterizzazione dello stallo

lo stallo può avvenire se si hanno 4 condizioni contemporaneamente (condizioni necessarie):

- **Mutua esclusione:** solo un processo alla volta può usare una risorsa.
- **Possesso e attesa:** un processo in possesso di una o più risorse è in attesa di acquisire altre risorse possedute da altri processi.
- **No prelazione:** una risorsa può essere rilasciata solo volontariamente dal processo che la possiede quando ha finito di usarla.
- **Attesa circolare:** esiste una sequenza  $\{P_0, P_1, \dots, P_n\}$  di processi in attesa tale che  $P_0$  è in attesa di una risorsa posseduta da  $P_1$ ,  $P_1$  è in attesa risorsa posseduta da  $P_2$ , ...,  $P_{n-1}$  è in attesa di una risorsa posseduta da  $P_n$ , e  $P_n$  è in attesa di una risorsa posseduta da  $P_0$ .

Sistemi Operativi A.A 2015/2016

## Grafo di allocazione delle risorse

Un insieme di vertici  $V$  e un insieme di archi  $E$ .

- $V$  è partizionato in due insiemi:
  - $P = \{P_1, P_2, \dots, P_n\}$ , insieme di tutti i processi nel sistema.
  - $R = \{R_1, R_2, \dots, R_m\}$ , insieme di tutti i tipi di risorsa nel sistema.
- **arco di richiesta:** l'arco orientato  $P_i \rightarrow R_j$  indica che il processo  $P_i$  sta richiedendo una risorsa del tipo  $R_j$
- **arco di assegnazione:** l'arco orientato  $R_j \rightarrow P_i$  indica che una istanza di  $R_j$  è assegnata a  $P_i$

Sistemi Operativi A.A 2015/2016

## Grafo di allocazione delle risorse (Cont.)

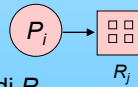
- Processo



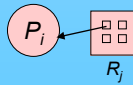
- Tipo di risorsa con 4 istanze



- $P_i$  richiede una istanza di  $R_j$

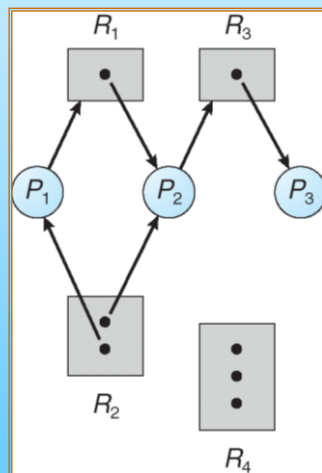


- $P_i$  possiede una istanza di  $R_j$



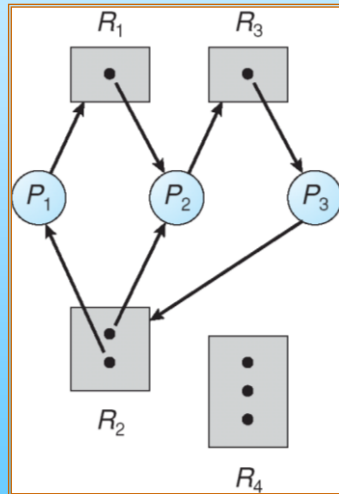
Sistemi Operativi A.A. 2015/2016

## Esempio di un grafo di allocazione risorse



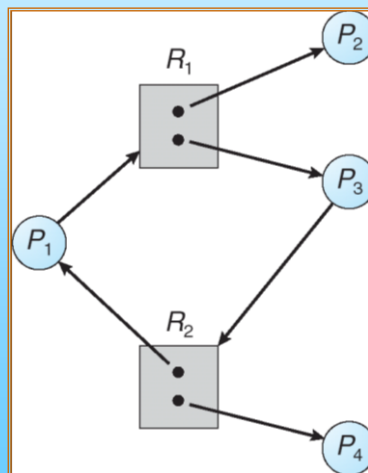
Sistemi Operativi A.A. 2015/2016

## Un grafo di allocazione risorse con stallo



Sistemi Operativi A.A 2015/2016

## Grafo allocazione risorse con ciclo ma non in stallo



Sistemi Operativi A.A 2015/2016

## Fatti

- Se il grafo non ha cicli  $\Rightarrow$  no stallo.
- Se il grafo contiene un ciclo  $\Rightarrow$ 
  - se solo una istanza per tipo di risorsa, STALLO.
  - se più di una istanza per tipo risorsa, si ha possibilità di stallo.

Sistemi Operativi A.A. 2015/2016

## Metodi per la gestione dello stallo

- **Prevenire o evitare** situazioni di stallo in modo che il sistema non entri mai in stallo.
- **Permettere** che il sistema entri in stallo, individuarlo e poi ripristinarlo.
- **Ignorare** il problema fingendo che situazioni di stallo non possano mai accadere nel sistema. Questa è la 'soluzione' di molti sistemi operativi tra cui Linux e Windows.

Sistemi Operativi A.A. 2015/2016

## Prevenzione dello stallo

Si vincola la modalità con cui si fanno richieste, in modo da escludere almeno una condizione necessaria per lo stallo

- **Mutua esclusione** – non necessaria per risorse condivisibili; obbligatoria per risorse non condivisibili.
- **Possesso e attesa** – garantire che quando un processo richiede una risorsa non possieda altre risorse.
  - processo richiede e acquisisce tutte le risorse necessarie prima dell'esecuzione o richiede risorse solo quando non ne possiede.
  - bassa utilizzazione delle risorse, possibile attesa indefinita.

Sistemi Operativi A.A. 2015/2016

## Prevenzione dello stallo (Cont.)

- **No prelazione** –
  - se un processo in possesso di alcune risorse richiede un risorsa non disponibile allora tutte le risorse possedute vengono rilasciate.
  - le risorse prelazionate sono aggiunte alla lista delle risorse per cui il processo è in attesa.
  - il processo ripartirà solo quando potrà acquisire tutte le sue vecchie risorse più le nuove che sta richiedendo.
- **Attesa circolare** – imporre un ordinamento totale a tutti i tipi di risorsa e imporre che ogni processo richieda le risorse in ordine crescente.

Sistemi Operativi A.A. 2015/2016

## Evitare lo stallo

E' necessario che il sistema possieda delle informazioni apriori.

- Il modello più semplice prevede che ogni processo dichiari il *numero massimo* di risorse di ogni tipo di cui potrebbe avere bisogno.
- L'algoritmo per evitare lo stallo esamina dinamicamente lo stato di allocazione delle risorse per assicurare che non ci potrà mai essere una condizione di attesa circolare.
- Lo stato di allocazione delle risorse è definito dal numero di risorse disponibili e allocate e dalle richieste massime dei processi.

Sistemi Operativi A.A. 2015/2016

## Stato Sicuro

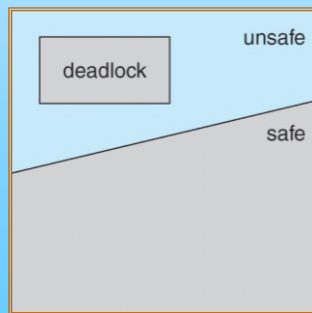
- Quando un processo richiede una risorsa disponibile il sistema deve decidere se l'allocazione immediata al processo lascia il sistema in uno stato sicuro.
- Il sistema è in uno **stato sicuro** se esiste una sequenza sicura di tutti i processi.
- La **sequenza**  $\langle P_1, P_2, \dots, P_n \rangle$  è **sicura** se per ogni  $P_i$ , le risorse che  $P_i$  può ancora richiedere possono essere soddisfatte con le risorse attualmente disponibili + le risorse possedute da tutti i processi  $P_j$ , con  $j < i$ .
  - Se le necessità di  $P_i$  non sono disponibili, allora  $P_i$  può aspettare fino a che tutti  $P_j$  abbiano finito.
  - Quando  $P_j$  hanno terminato,  $P_i$  può ottenere tutte le risorse necessarie, eseguire, ritornare le risorse allocate e terminare.
  - Quando  $P_i$  termina,  $P_{i+1}$  può ottenere le risorse necessarie e così via...

Sistemi Operativi A.A. 2015/2016



## Fatti

- Se un sistema è in uno **stato sicuro**  $\Rightarrow$  **no stallo**.
- Se un sistema è in uno **stato non sicuro**  $\Rightarrow$  **possibilità di stallo**.
- **Evitare stallo**  $\Rightarrow$  assicurare che un sistema non entrerà mai in uno **stato non sicuro**.



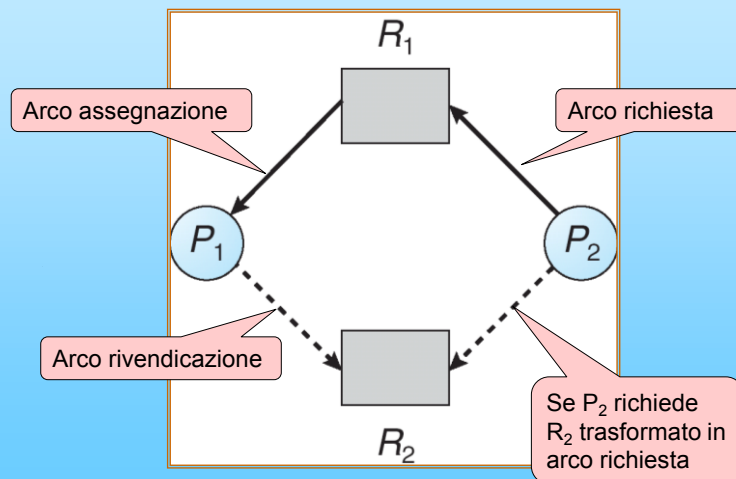
Sistemi Operativi A.A. 2015/2016

## Algoritmo con grafo allocazione risorse

- Si usa in caso di unica istanza per ogni tipo di risorsa
- *Arco di rivendicazione*:  $P_i \rightarrow R_j$  indica che il processo  $P_i$  potrebbe richiedere la risorsa  $R_j$ ; rappresentato da riga tratteggiata.
- *Arco di rivendicazione* convertito in un *arco di richiesta* quando un processo richiede la risorsa.
- Quando la risorsa viene rilasciata dal processo, *arco di assegnazione* convertito in *arco di rivendicazione*.
- Le risorse devono essere rivendicate *a priori* dal sistema.
- Algoritmo: una risorsa viene concessa se dopo sostituzione arco (rivendicazione  $\rightarrow$  assegnazione) non si forma un ciclo nel grafo.
- Costo:  $O(n^2)$

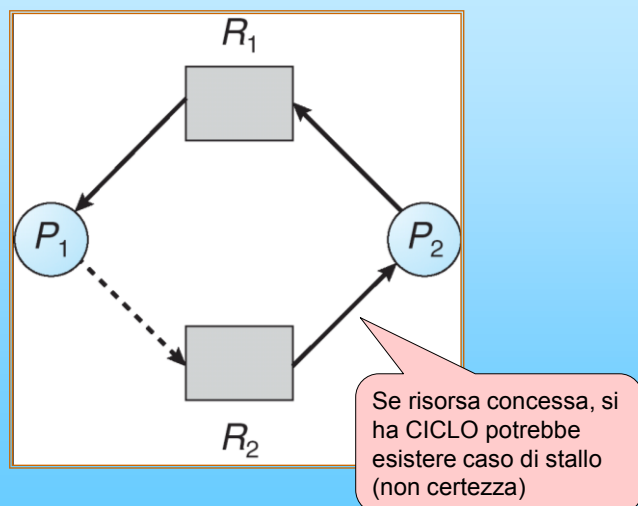
Sistemi Operativi A.A. 2015/2016

## Grafo allocazione risorse per evitare lo stallo



Sistemi Operativi A.A 2015/2016

## Stato non sicuro nel grafo assegnazione risorse



Sistemi Operativi A.A 2015/2016

## Algoritmo del Banchiere

- Si applica al caso di risorse con istanze multiple.
- Ogni processo deve dichiarare a priori il massimo uso per ogni tipo di risorsa.
- Quando un processo richiede una risorsa potrebbe dover attendere.
- Quando un processo prende tutte le risorse necessarie le deve ritornare in un tempo finito.

Sistemi Operativi A.A. 2015/2016

## Strutture dati per l'algoritmo del Banchiere

Sia  $n$  = numero di processi, e  $m$  = numero di tipi di risorse.

- *Available*: Vettore lungo  $m$ . Se  $Available[j] = k$ , ci sono  $k$  istanze della risorsa  $R_j$  disponibili.
- *Max*: matrice  $n \times m$ . Se  $Max[i,j] = k$ , allora il processo  $P_i$  potrebbe richiedere al massimo  $k$  istanze del tipo di risorsa  $R_j$ .
- *Allocation*: matrice  $n \times m$ . Se  $Allocation[i,j] = k$  allora  $P_i$  ha attualmente allocate  $k$  istanze di  $R_j$ .
- *Need*: matrice  $n \times m$ . Se  $Need[i,j] = k$ , allora  $P_i$  potrebbe richiedere altre  $k$  istanze di  $R_j$  per completare il task.

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

Sistemi Operativi A.A. 2015/2016

## Algoritmo verifica sicurezza

1. Sia *Work* e *Finish* vettori di lunghezza *m* e *n*.  
Inizializzati:  
 $Work = Available$   
 $Finish[i] = false$  per  $i = 1, 2, \dots, n$ .
2. Trova un valore di *i* tale che:  
(a)  $Finish[i] = false$   
(b)  $Need_i \leq Work$   
Se *i* non esiste, vai a passo 4.
3.  $Work = Work + Allocation_i$   
 $Finish[i] = true$   
vai a passo 2.
4. Se  $Finish[i] == true$  per tutti gli *i*, allora il sistema si trova in uno **stato sicuro**!

Sistemi Operativi A.A. 2015/2016

## Algoritmo di richiesta risorse per il processo $P_i$

$Request_i$  = vettore richieste per il processo  $P_i$ .

Se  $Request_i[j] = k$  allora il processo  $P_i$  vuole *k* istanze del tipo di risorsa  $R_j$ .

1. Se  $Request_i \leq Need_i$  vai a passo 2. Altrimenti, errore dal momento che il processo richiede più risorse di quelle dichiarate.
2. Se  $Request_i \leq Available$ , vai a passo 3. Altrimenti  $P_i$  deve aspettare, perché le risorse non sono disponibili.
3. Pretende di allocare le risorse richieste a  $P_i$  modificando lo stato come segue:

$$Available = Available - Request_i;$$

$$Allocation_i = Allocation_i + Request_i;$$

$$Need_i = Need_i - Request_i;$$

- ▶ Se dopo cambiamento **stato sicuro**  $\Rightarrow$  le risorse sono allocate a  $P_i$ .
- ▶ Se **stato unsafe**  $\Rightarrow P_i$  deve aspettare e il vecchio stato allocazione risorse deve essere ripristinato

Sistemi Operativi A.A. 2015/2016

## Esempio Algoritmo del Banchiere

- 5 processi  $P_0 - P_4$ ; 3 tipi risorsa: A (10 istanze), B (5 istanze), and C (7 istanze).
- Snapshot al tempo  $T_0$ :

	<u>Allocation</u>			<u>Max</u>			<u>Available</u>		
	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			

Sistemi Operativi A.A 2015/2016

## Example (Cont.)

- Calcolo matrice Need = Max – Allocation.

	<u>Allocation</u>			<u>Max</u>			<u>Need</u>			<u>Available</u>		
	A	B	C	A	B	C	A	B	C	A	B	C
$P_0$	0	1	0	7	5	3	7	4	3	3	3	2
$P_1$	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			

- Il sistema è in uno stato sicuro dato che la sequenza  $\langle P_1, P_3, P_4, P_2, P_0 \rangle$  soddisfa i criteri di sicurezza.

Sistemi Operativi A.A 2015/2016

## Esempio $P_1$ richiede (1,0,2) (Cont.)

- Controllo  $\text{Request} \leq \text{Available}$   $(1,0,2) \leq (3,3,2) \Rightarrow \text{true}$ .

	<u>Allocation</u>	<u>Need</u>	<u>Available</u>
	A B C	A B C	A B C
$P_0$	0 1 0	7 4 3	<b>2 3 0</b>
$P_1$	<b>3 0 2</b>	<b>0 2 0</b>	
$P_2$	3 0 1	6 0 0	
$P_3$	2 1 1	0 1 1	
$P_4$	0 0 2	4 3 1	

- Eseguendo alg. controllo sicurezza mostra che la sequenza  $\langle P_1, P_3, P_4, P_0, P_2 \rangle$  soddisfa il requisito.
- La richiesta di  $P_4$  (3,3,0) può essere concessa?
  - NO non disponibili
- La richiesta di  $P_0$  (0,2,0) può essere concessa?
  - NO stato non sicuro

Sistemi Operativi A.A 2015/2016

## Rilevamento dello Stallo

- Si permette al sistema di entrare in stallo
- Si usa algoritmo di rilevamento per rilevare condizione di stallo
- Si usa uno schema di ripristino

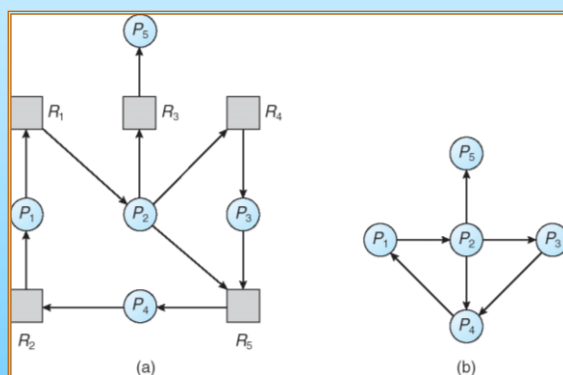
Sistemi Operativi A.A 2015/2016

## Singola istanza per tipo risorsa

- Mantenere grafo d'attesa
  - I vertici sono processi.
  - $P_i \rightarrow P_j$  se  $P_i$  è in attesa di  $P_j$ .
- Periodicamente si invoca algoritmo per cercare ciclo nel grafo.
- Un algoritmo per cercare ciclo nel grafo richiede  $O(n^2)$  operazioni, dove  $n$  è il numero di vertici del grafo.

Sistemi Operativi A.A. 2015/2016

## Grafo allocazione risorse e Grafo d'attesa



Grafo allocazione risorse    Corrispondente grafo d'attesa

Sistemi Operativi A.A. 2015/2016

## Molte istanze per tipo di risorsa

- *Available*: Un vettore di lunghezza  $m$  indica il numero di risorse disponibili per ogni tipo di risorsa.
- *Allocation*: Una matrice  $n \times m$  definisce il numero di risorse per ogni tipo allocate al processo.
- *Request*: Una matrice  $n \times m$  indica la richiesta corrente di ogni processo. Se  $Request[i][j] = k$ , allora il processo  $P_i$  sta richiedendo  $k$  risorse in più del tipo di risorsa  $R_j$ .

Sistemi Operativi A.A. 2015/2016

## Algoritmo rilevazione stallo

1. Sia *Work* e *Finish* due vettori di lunghezza  $m$  e  $n$ ,  
Inizializzare:
  - (a)  $Work = Available$
  - (b) Per  $i = 1, 2, \dots, n$ , se  $Allocation_i \neq 0$ , then  
 $Finish[i] = false$ ; altrimenti,  $Finish[i] = true$ .
2. Trovare un indice  $i$  tale che:
  - (a)  $Finish[i] == false$
  - (b)  $Request_i \leq Work$Se  $i$  non esiste, vai al passo 4.
3.  $Work = Work + Allocation_i$   
 $Finish[i] = true$   
vai al passo 2.
4. Se  $Finish[i] == false$ , per qualche  $i$ ,  $1 \leq i \leq n$ , allora il **sistema è in stallo**. Inoltre, se  $Finish[i] == false$ , allora  $P_i$  è in stallo.

L'Algoritmo richiede un ordine di  $O(m \times n^2)$  operazioni per rilevare la condizione di stallo.

Sistemi Operativi A.A. 2015/2016



## Esempio

- Cinque processi  $P_0 \dots P_4$ ; tre tipi di risorse A (7 istanze), B (2 istanze), and C (6 istanze).
- Snapshot a tempo  $T_0$ :

	<u>Allocation</u>			<u>Request</u>			<u>Available</u>		
	A	B	C	A	B	C	A	B	C
$P_0$	0	1	0	0	0	0	0	0	0
$P_1$	2	0	0	2	0	2			
$P_2$	3	0	3	0	0	0			
$P_3$	2	1	1	1	0	0			
$P_4$	0	0	2	0	0	2			

- La sequenza  $\langle P_0, P_2, P_3, P_1, P_4 \rangle$  porterà ad avere  $Finish[i] = \text{true}$  per ogni  $i$ .

Sistemi Operativi A.A 2015/2016

## Esempio (Cont.)

- $P_2$  richiede una istanza in più di C.

	<u>Allocation</u>			<u>Request</u>			<u>Available</u>		
	A	B	C	A	B	C	A	B	C
$P_0$	0	1	0	0	0	0	0	0	0
$P_1$	2	0	0	2	0	2			
$P_2$	3	0	3	0	0	1			
$P_3$	2	1	1	1	0	0			
$P_4$	0	0	2	0	0	2			

- Stato del sistema?
  - Può ottenere risorse possedute da  $P_0$ , ma insufficienti per soddisfare richieste degli altri processi.
  - Esiste uno stallo formato dai processi  $P_1, P_2, P_3$ , e  $P_4$ .

Sistemi Operativi A.A 2015/2016

## Uso dell'alg. di rilevamento

- Quando e ogni quanto invocarlo dipende da:
  - Ogni quanto lo stallo potrebbe verificarsi?
  - Quanti processi dovranno essere annullati (rolled back)?
    - ▶ uno per ogni ciclo disgiunto
- Se l'alg. di rilevamento è invocato arbitrariamente, possono formarsi molti cicli nel grafo delle risorse ed è difficile determinare quale dei processi in stallo ha "causato" lo stallo.
- Chiamarlo troppo spesso ha costo computazionale elevato
- Ripristino dallo stallo:
  - terminare processi
  - prelazione risorse

Sistemi Operativi A.A. 2015/2016

## Ripristino dallo stallo: Terminazione dei processi

- Terminare tutti i processi in stallo.
- Terminare un processo alla volta fino all'eliminazione del ciclo.
- In base a cosa scegliere il processo da terminare?
  - Priorità del processo
  - Per quanto tempo ha computato e quanto ancora rimane.
  - Risorse che il processo ha usato.
  - Risorse ancora necessarie al completamento.
  - Numero processi che devono essere terminati.
  - Il tipo di processo: interattivo o batch

Sistemi Operativi A.A. 2015/2016

## Ripristino da stallo: Prelazione risorse

- **Selezionare una vittima** – minimizzando il costo.  
(es. n. risorse possedute, quantità di tempo già spesa)
- **Fare Rollback** – ritornare ad uno stato sicuro e far ripartire il processo da questo stato.
- **Attesa indefinita** – uno stesso processo selezionato sempre come vittima → includere il numero di rollback nel fattore di costo.