

Sistemi Operativi II

Corso di Laurea in Ingegneria Informatica

Facolta' di Ingegneria, Universita' "La Sapienza"

Docente: Francesco Quaglia

Tecniche di gestione del deadlock:

1. Caratterizzazione del deadlock
2. Prevenzione del deadlock
3. Deadlock avoidance
4. Deadlock detection/recovery

Classificazione delle risorse

Riutilizzabili

- non può essere utilizzata contemporaneamente da più processi
- non viene distrutta dopo l'utilizzo (non scompare)
- **esempi**: CPU, canali di I/O, memoria centrale
- tipicamente è il sistema operativo a determinarne l'assegnazione ai processi

Non-riutilizzabili

- viene distrutta dopo l'utilizzo (scompare)
- tipicamente non c'è limite al numero di volte in cui essa può essere riprodotta
- **esempi**: interrupt, segnali, messaggi
- può essere “fornita” da processo a processo

Accesso alle risorse

Richiesta

- se la richiesta non può essere soddisfatta immediatamente, il processo richiedente deve attendere fino a che la risorsa non sia disponibile

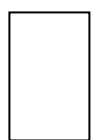
Assegnazione

- il processo può operare sulla risorsa per gli scopi specifici per cui ne ha richiesto l'uso

Rilascio

- il processo rilascia la risorsa, qualora essa sia riutilizzabile
-
- per le risorse riutilizzabili, il sistema (operativo) mantiene una tabella di assegnazione ai processi
 - per tutte le risorse, il sistema (operativo) mantiene una tabella delle richieste dei processi

Modello di richiesta/assegnazione



risorsa
riutilizzabile



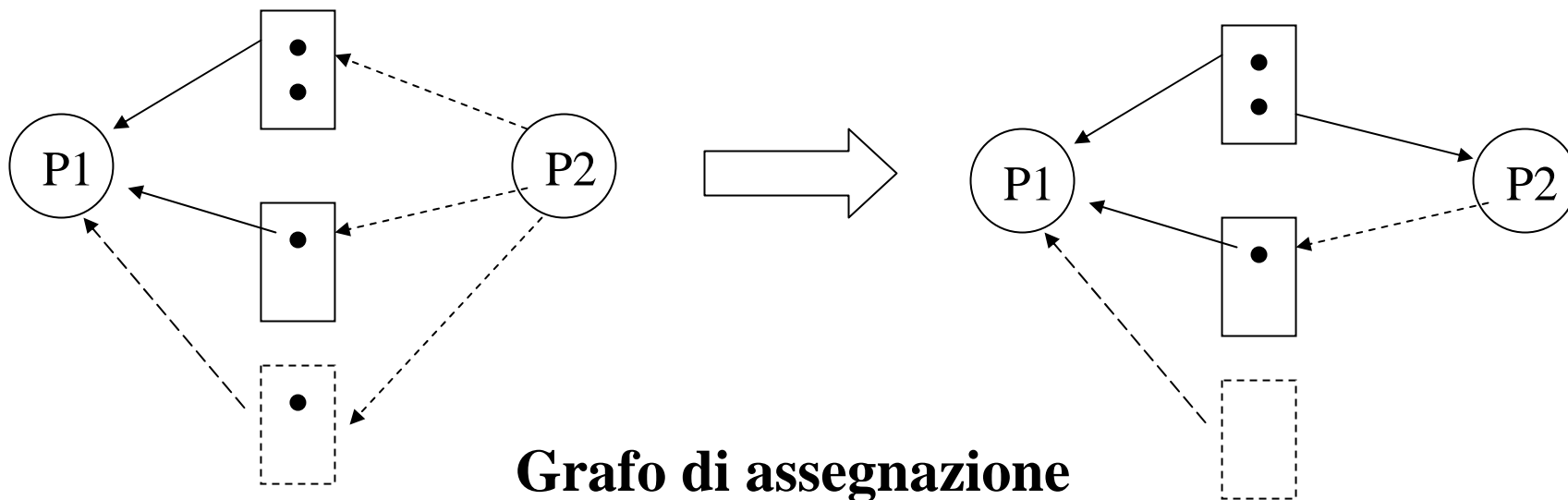
risorsa non
riutilizzabile

● istanza di una data
risorsa

—————> risorsa assegnata

-----> richiesta pendente

-----> risorsa può essere creata da



Grafo di assegnazione

Descrizione del deadlock

- un ciclo nel grafo di allocazione delle risorse può significare la presenza di un deadlock
- i processi eventualmente coinvolti sono quelli facenti parte del ciclo
- nel caso in cui il ciclo riguardi risorse riutilizzabili con una sola istanza allora siamo realmente in presenza di un deadlock
- nel caso in cui il ciclo riguardi risorse non riutilizzabili per le quali il processo nel ciclo è l'unico produttore, allora siamo realmente in presenza di un deadlock
- nel caso di risorse riutilizzabili con istanze multiple, o di risorse non riutilizzabili per le quali è possibile che esse siano prodotte da un processo esterno al ciclo, allora la presenza del ciclo è condizione necessaria ma non sufficiente per il deadlock

Condizioni necessarie: caso di risorse riutilizzabili

Mutua esclusione

- la risorsa non è condivisibile
- se occupata, un processo richiedente deve attendere fino a che essa non sia rilasciata

Possesso e attesa

- un processo può richiedere (attendere) risorse dopo averne acquisite già altre

Impossibilità di prelazione

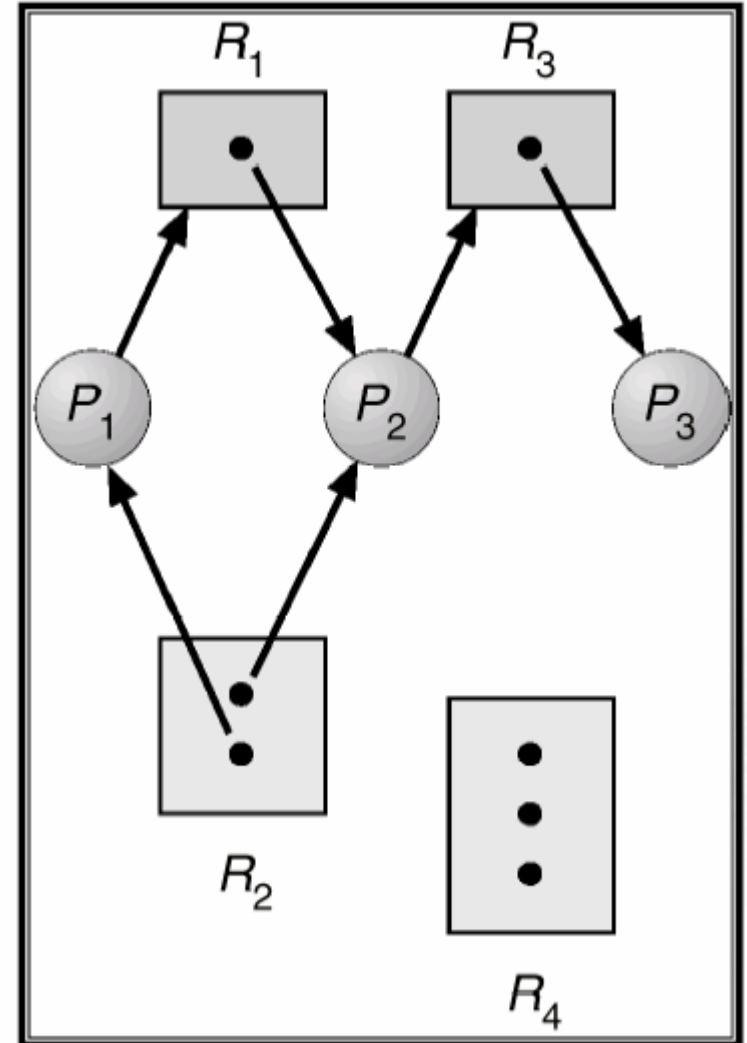
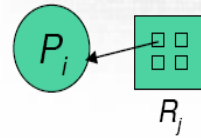
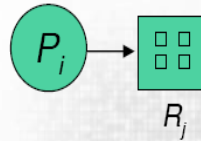
- una risorsa assegnata ad un processo può essere rilasciata dal processo solo spontaneamente
-

Attesa circolare

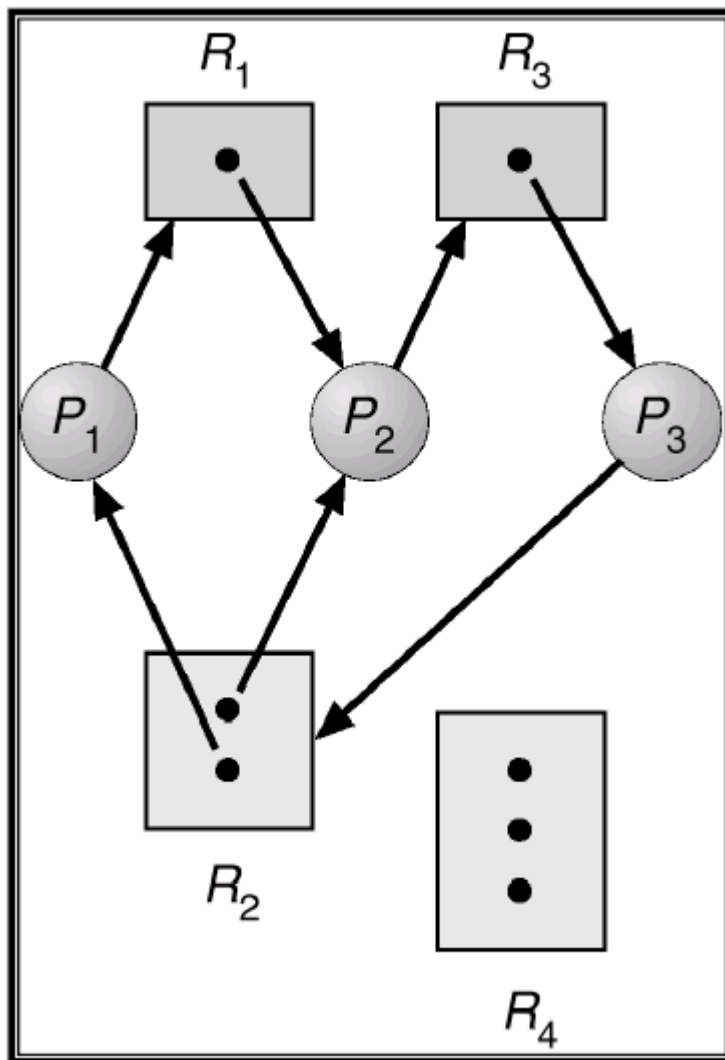
- deve esistere un gruppo di processi (P_1, \dots, P_n) tali che P_1 è in attesa di una risorsa di P_2 , P_2 è in attesa di una risorsa di P_3 , ... , P_{n-1} è in attesa di una risorsa di P_n , P_n è in attesa di una risorsa di P_1

Un esempio di grafo di assegnazione (resource allocation graph)

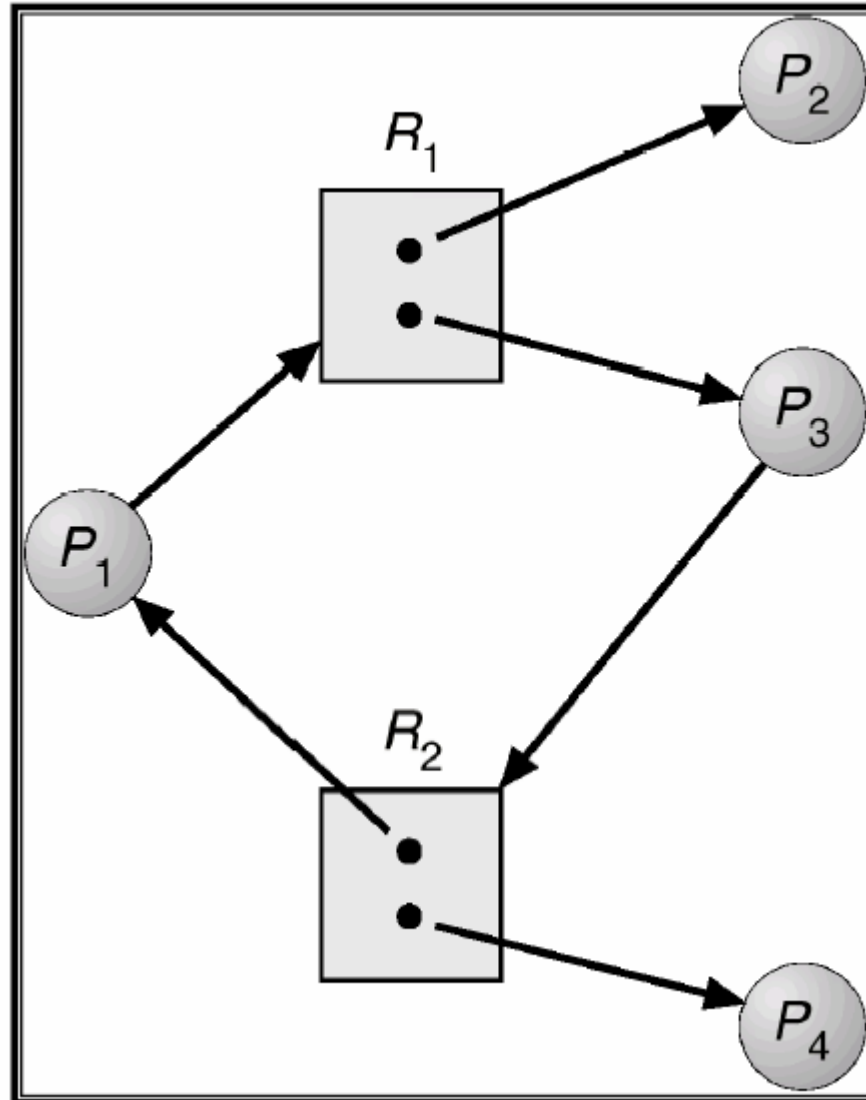
- Process
- Resource Type with 4 instances
- P_i requests instance of R_j
- P_i is holding an instance of R_j



Un esempio di grafo di assegnazione con deadlock



Un esempio di grafo di assegnazione con un ciclo ma non un deadlock



Gestione del deadlock

Prevenzione

- usare metodi atti a far sì che almeno una delle condizioni necessarie non si possa verificare

Avoidance

- si lascia la possibilità per le condizioni necessarie di verificarsi
- l'assegnazione delle risorse ai processi avviene secondo schemi tali che il deadlock non possa verificarsi

Detection/recovery

- le risorse vengono assegnate ai processi senza alcuno schema di protezione contro il deadlock
- se dei deadlock realmente si verificano, si cerca di individuarli e di “sbloccare” i processi coinvolti

Prevenzione del deadlock

Prevenzione del “possesso e attesa”

- si forza un processo a richiedere tutte le risorse di cui può necessitare all’inizio della sua esecuzione (possibilità di sottoutilizzo delle risorse e di starvation)
- si ammette che un processo che possiede risorse possa chiederne altre, a patto di rilasciare le prime

Prelazione sulle risorse

- se un processo entra in attesa su di una risorsa, allora gli vengono sottratte le risorse attualmente possedute
- queste ultime vengono reinserite nell’insieme di risorse per il quale il processo stesso è in attesa

Prevenzione di “attesa circolare”

- le risorse vengono ordinate in base ad un identificatore
- un processo può richiedere risorse solo secondo un ordine crescente dei loro identificatori

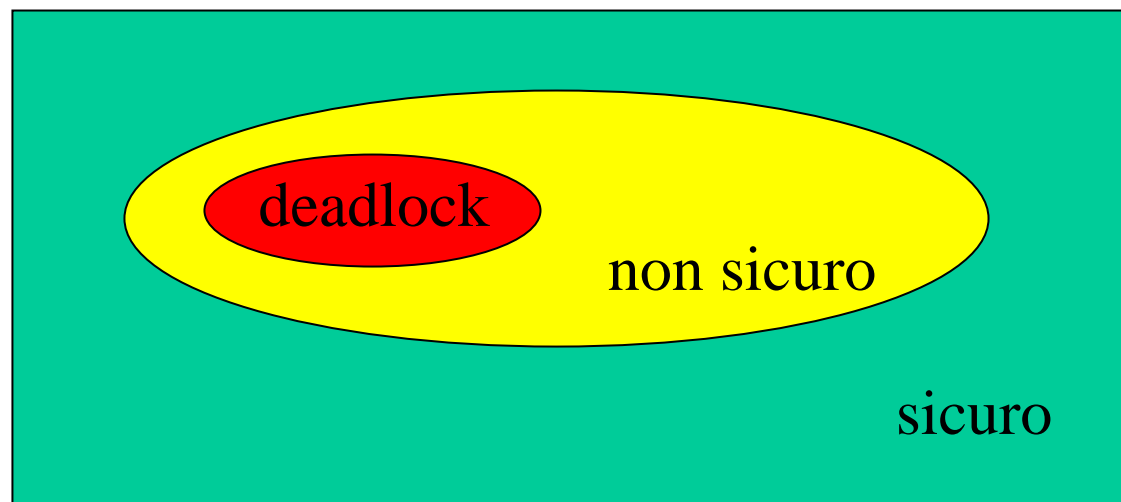
Deadlock avoidance

Sequenza sicura

- (P_1, \dots, P_n) è una sequenza di processi sicura se le risorse di cui necessita P_i sono libere o in possesso di processi P_j con $j < i$

Stato sicuro

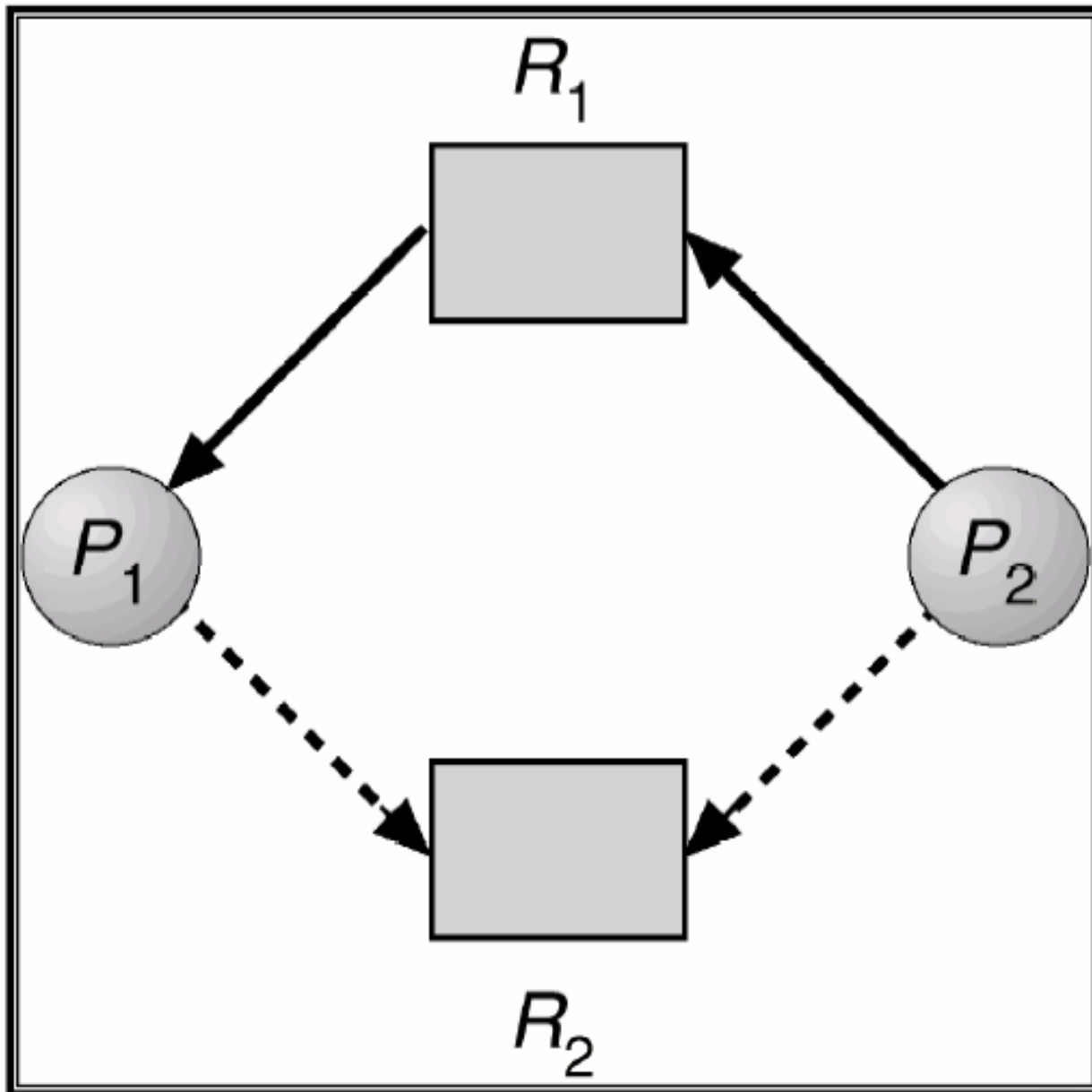
- il sistema è in uno stato sicuro di assegnazione di risorse se, quando assegna risorse ai processi, allora essi appartengono ad una sequenza sicura



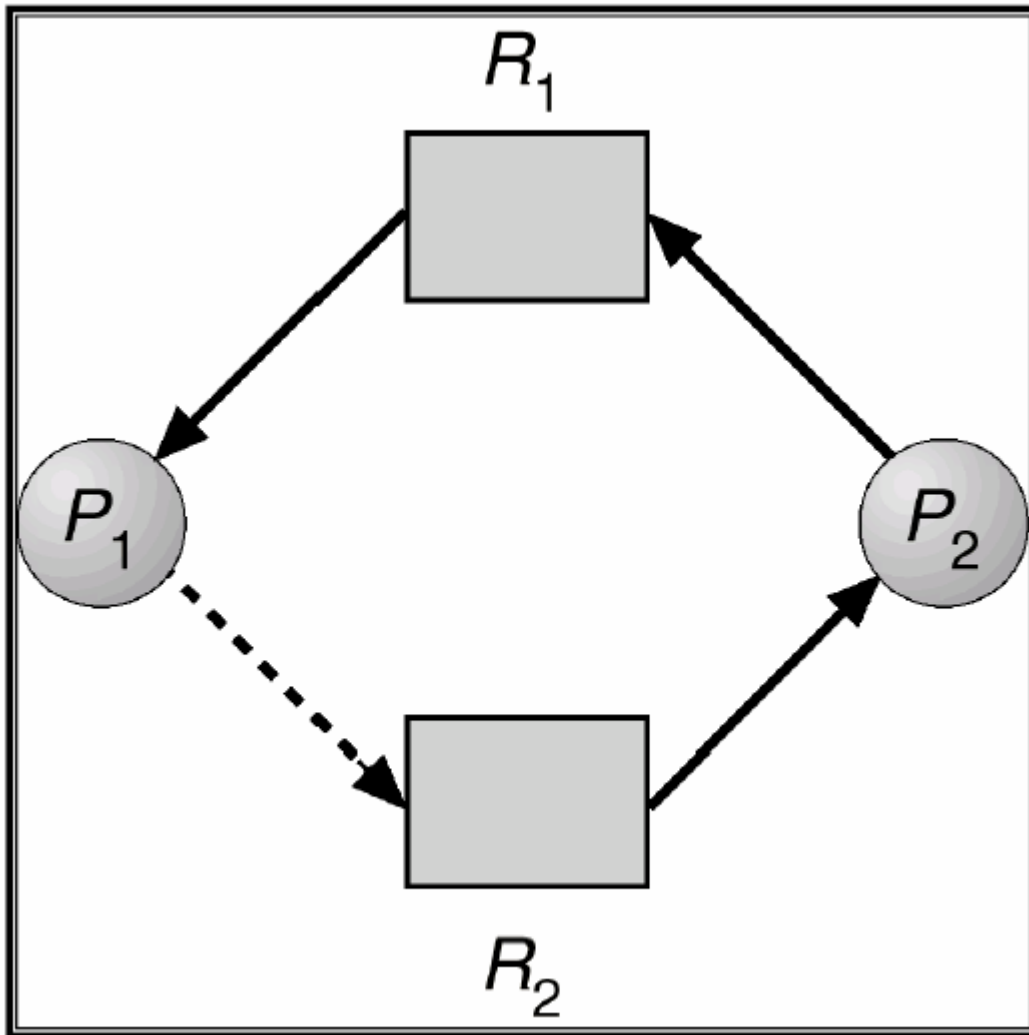
Algoritmo con grafo di allocazione

- valido per istanze singole di ogni risorsa
- si utilizza un nuovo tipo di arco nel grafo di assegnazione, detto arco di reclamo
- un arco di reclamo verso una risorsa implica che il processo può richiedere l'uso della risorsa durante la sua esecuzione
- gli archi di reclamo devono essere inseriti all'attivazione del processo (necessità di conoscere le risorse di cui il processo può far richiesta)
- un arco di reclamo viene trasformato in un arco di assegnazione se, e solo se, esso non è coinvolto in un ciclo
- c'è necessità di visitare il grafo di assegnazione ad ogni richiesta di risorsa da parte di un processo
- il costo è quindi $O(n^2)$ per ogni assegnazione da effettuare

Uso di archi di reclamo



Stato insicuro in caso di assegnazione



Algoritmo del banchiere

- valido per istanze multiple di ogni risorsa
- quando un processo è attivato, deve notificare il numero massimo di risorse di cui esso necessita
- strutture dati
 1. Vettore *Available*. $Available[j]$ indica il numero di istanze della j -esima risorsa
 2. Matrice *Max*. $Max[i,j]$ indica il massimo numero di istanze della risorsa j -esima che possono essere richieste dall' i -esimo processo
 3. Matrice *Allocation*. $Allocation[i,j]$ indica il numero di istanze della risorsa j -esima attualmente assegnate all' i -esimo processo
 4. Matrice *Need*. $Need[i,j]$ indica il numero di istanze della risorsa j -esima che potranno essere richieste in futuro all' i -esimo processo (necessità residua)

Verifica della sicurezza

- siano $Work$ e $Finish$ vettori di taglia m (numero di risorse) ed n (numero di processi)
- inizializzazione: $Work = Available$, $\forall i Finish [i] = FALSE$
- passi di computazione:
 1. Cerca un i tale che: $Finish[i] = FALSE$ **and** $\forall j Need[i,j] \leq Work [j]$
se tale i non esiste allora vai al passo 3
 2. $\forall j Work[j] = Work [j] + Allocation[i,j]$
 $Finish[i] = TRUE$
torna al passo 1
 3. Se $Finish[i] = TRUE$ per ogni valore di i , allora il sistema è in uno stato sicuro

Costo computazionale $O(m \times n^2)$

Decisione di assegnazione

- sia $Request_i$ un vettore di taglia m (numero di risorse) indicante l'attuale richiesta di risorse da parte del processo i -esimo
- passi di computazione:
 1. Se $Request_i \leq Need_i$ allora vai al passo 2, altrimenti esci con una condizione di errore
 2. Se $Request_i \leq Available$ allora vai al passo 3, altrimenti metti il processo richiedente in attesa
 3. Simulazione dell'allocazione di risorse
$$Available = Available - Request_i$$
$$Allocation_i = Allocation_i + Request_i$$
$$Need_i = Need_i - Request_i$$
 4. Verifica sulla sicurezza: se lo stato è sicuro l'allocazione di risorse è resa permanente

Un esempio

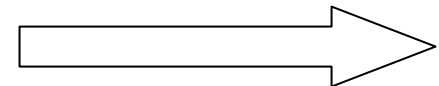
5 processi, 3 tipi di risorse

Stato attuale

	Allocation	Max	Available	Need (Max – Allocation)
P ₀	010	753	332	743
P ₁	200	322		122
P ₂	302	922		621
P ₃	211	222		011
P ₄	002	433		431

lo stato è sicuro perchè la sequenza $\langle P_1, P_3, P_4, P_2, P_0 \rangle$ soddisfa i criteri di sicurezza

continua



Supponiamo che il processo P_1 richieda un'altra istanza della risorsa di tipo A e due istanze della risorsa di tipo C

In tal caso $Request_1 = (1,0,2)$

Il nuovo stato sarebbe

	Allocation	Max	Available	Need
P_0	010	753	230	743
P_1	302	322		020
P_2	302	922		620
P_3	211	222		011
P_4	002	433		431

che è ancora sicuro

Se ora P_2 richiedesse risorse secondo $Request_2 = (0,2,0)$ il nuovo stato sarebbe non sicuro, pur essendo le risorse chieste correntemente disponibili

Deadlock detection

Istanze singole di ciascuna risorsa

- il grafo di allocazione può essere usato per determinare la presenza di cicli

Istanze multiple di risorsa

- inizializzazione:

$$Work = Available,$$

$$\forall i \text{ if } Allocation_i \neq 0 \text{ then } Finish[i] = FALSE \text{ else } Finish[i] = TRUE$$

- passi di computazione:

1. Cerca un i tale che: $Finish[i] = FALSE$ and $Request_i \leq Work$
se tale i non esiste allora vai al passo 3

2. $Work = Work + Allocation_i$

$$Finish[i] = TRUE$$

torna al passo 1

3. Se $Finish[i] = FALSE$ per qualche valore di i , allora il sistema è in deadlock (i processi in deadlock hanno il flag a FALSE)

Un esempio

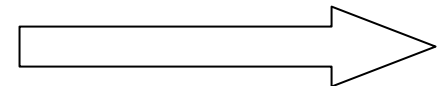
5 processi, 3 tipi di risorse (A,B,C), istanze relative (7,2,6)

Stato attuale

	Allocation	Request	Available
P ₀	010	000	000
P ₁	200	202	
P ₂	303	000	
P ₃	211	100	
P ₄	002	002	

lo stato è sicuro perchè la sequenza $\langle P_0, P_2, P_3, P_1, P_4 \rangle$ soddisfa i criteri di sicurezza

continua



Supponiamo che il processo P_2 richieda un'altra istanza della risorsa di tipo C

Il nuovo stato sarebbe

	Allocation	Request	Available
P_0	010	000	000
P_1	200	202	
P_2	303	001	
P_3	211	100	
P_4	002	002	

Il sistema entra in deadlock coinvolgendo i processi P_1, P_2, P_3 e P_4

Uso dell'algoritmo di rilevamento

- ad ogni richiesta di risorsa da parte di un processo
 1. Capacità di determinare la richiesta, e quindi il processo, che causa il deadlock
 2. Costo elevato
 3. Capacità di intervento tempestivo (non degrada eccessivamente l'utilizzo delle risorse)
- su base periodica
 1. Incapacità di determinare il processo, e quindi la richiesta, che causa il deadlock
 2. Costo ridotto
 3. Ridotta capacità di intervento tempestivo (maggiore degradazione dell'utilizzo delle risorse)

Deadlock recovery

Terminazione di processo

- approccio non incrementale: terminazione di tutti i processi coinvolti nel deadlock
- approccio incrementale: terminazione di un processo per volta fino all'eliminazione dei cicli di deadlock. Criteri di selezione:
 1. Priorità
 2. Tempo di calcolo
 3. Tipo di risorse occupate (per alcune il rilascio potrebbe essere più semplice che per altre)

Prelazione

- deassegnazione di risorse ad un processo coinvolto nel deadlock
- aspetti coinvolti:
 1. Selezione di una vittima
 2. Starvation

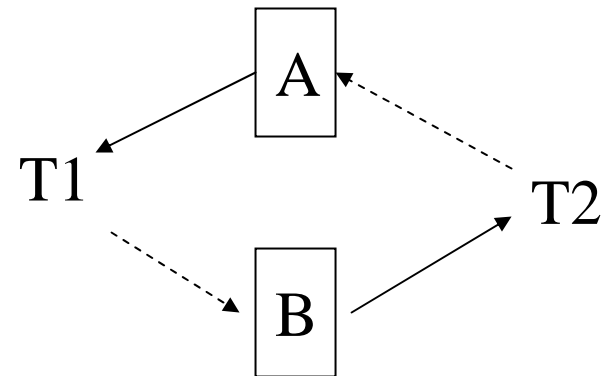
Deadlock in pratica

Sistemi transazionali (database) con controllo di concorrenza “pessimistico”:

- vengono assegnati dei lock sui dati
- i **Read-lock** possono essere condivisi
- i **Write-lock** sono esclusivi

T1: Read(A), Write(B)

T2: Read(B), Write(A)



ORACLE, DB2, Informix, MySQL