Mutua esclusione distribuita

Sincronizzazione del clock

- Il clock di CPU distribuite non é sincronizzato
- Clock fisico (difficile) / Clock logico (semplice)
- In molti casi basta sincronizzare il clock logico

  Sincronizzazione del clock logico: cosa vuol dire?
- In molti casi non é necessario che il clock sia riferito al clock reale
- Sincronizzazione del Clock logico: verifica delle condizioni **avvenuto-prima**
- Se **a** e **b** sono eventi, allora **a** $\rightarrow$ **b** significa che l'evento **a** avviene prima di **b**
- Per esempio, se **a** é l'invio di un messaggio, e **b** la sua ricezione, allora **a** $\rightarrow$ **b**
- La relazione **avvenuto-prima** é transitiva
- Il tempo in cui avvengono gli eventi di processi é legato allo scambio di informazioni tra i processi
- Se due eventi **x** e **y** avvengono in processi che non si scambiano informazioni, gli eventi sono detti **concorrenti**
- Nella sincronizzazione logica, un evento **a** avviene in un istante C(a) che non deve essere attinente al tempo fisico, ma rispettare le condizioni **avvenuto-prima**

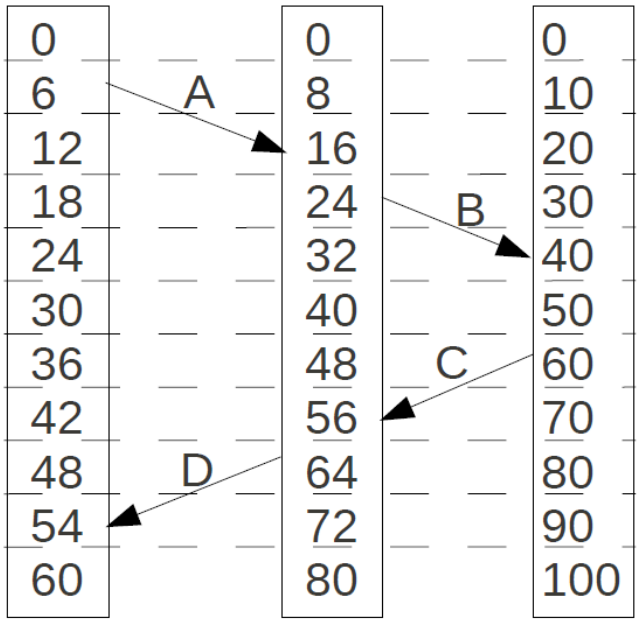
Algoritmo di LAMPORT

- Ipotesi: **a** é l'invio di un messaggio, e **b** la sua ricezione
- Ogni volta che arriva un messaggio viene incluso il *timestamp* dell'istante di invio.
- Quando il messaggio arriva al ricevitore, il clock fisico del ricevitore viene settato a : C(**a**)=timestamp + 1
- Sincronizza il clock logico in modo tale che siano verificate le seguenti condizioni:

1. Se **a** avviene prima di **b** nello stesso processo, $C(\mathbf{a}) < C(\mathbf{b})$
2. Se **a** e **b** sono l'ivio e la ricezione di un messaggio, $C(\mathbf{a}) < C(\mathbf{b})$
3. Per tutti gli eventi **a** e **b**, $C(\mathbf{a}) < C(\mathbf{b})$, $C(\mathbf{a}) \neq C(\mathbf{b})$

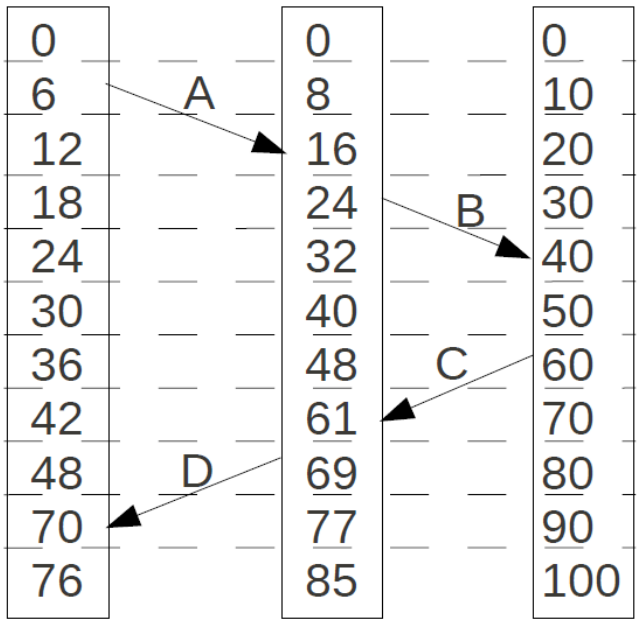
Algoritmo di Lamport (esempio)

Esempio: tre processi, ciascuno col suo clock.



Algoritmo di Lamport: soluzione

Soluzione di Lamport.



Algoritmo centralizzato

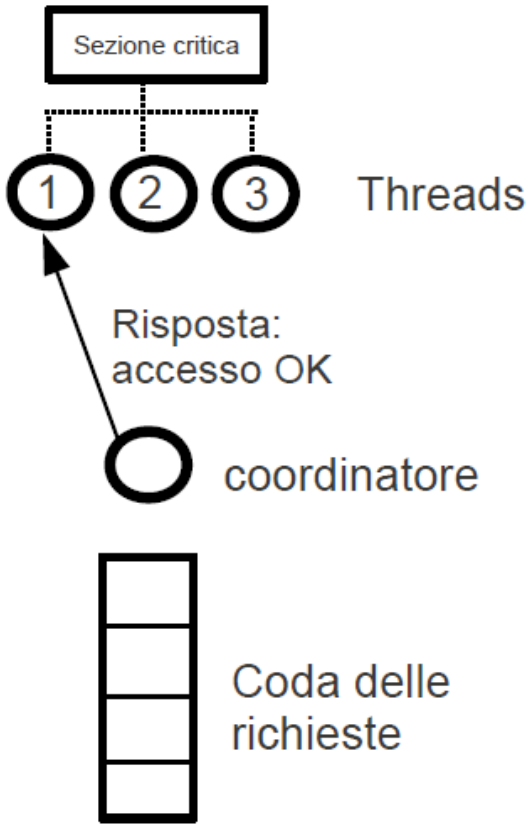
- Un processo viene eletto come coordinatore
- Un processo vuole entrare nella Sezione Critica $\rightarrow$ invia un messaggio di **Richiesta** al coordinatore
- Se nessun altro processo é nella Sezione Critica, il coordinatore risponde con un messaggio di **OK**
- Quando arriva la risposta il processo entra nella Sezione Critica
- Se non arriva la risposta, il processo conclude che non pu'øentrare nella Sezione Critica
- Se un altro processo é nella Sezione Critica, il coordinatore accoda la richiesta
- Quando un processo esce da una Sezione Critica, invia un messaggio di **Rilascio**. Il coordinatore, vede se ci sono richieste in coda e le soddisfa una alla volta

Vantaggi: semplice, richiede solo tre messaggi: **Richiesta**, **OK**, **Rilascio** Svantaggi: poco affidabile, collo di bottiglia, un processo non puó distibuere una condizione di 'Coordinatore terminato' o 'Richiesta rifiutata'

Algoritmo centralizzato

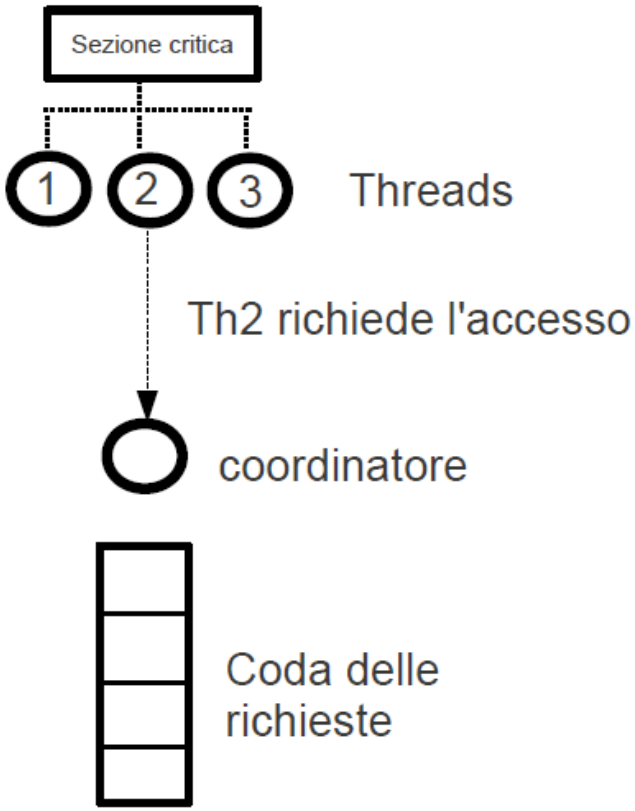
- il processo 1 manda un messaggio al coordinatore per chiedere accesso alla sezione critica

Algoritmo centralizzato



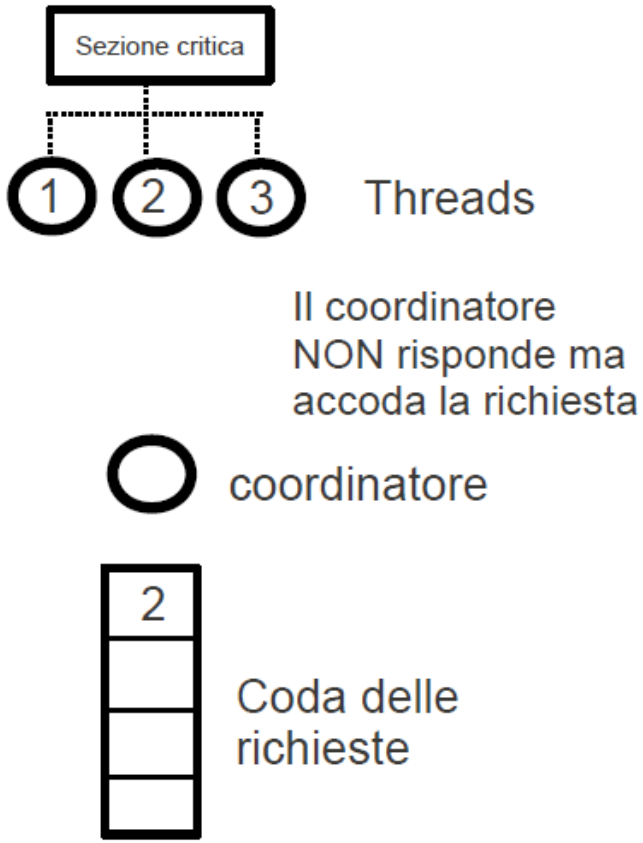
- la sezione critica non é utilizzata e puó quindi essere impegnata da Th1
- Th1 entra nella sezione critica appena riceve un messaggio di consenso dal coordinatore

Algoritmo centralizzato



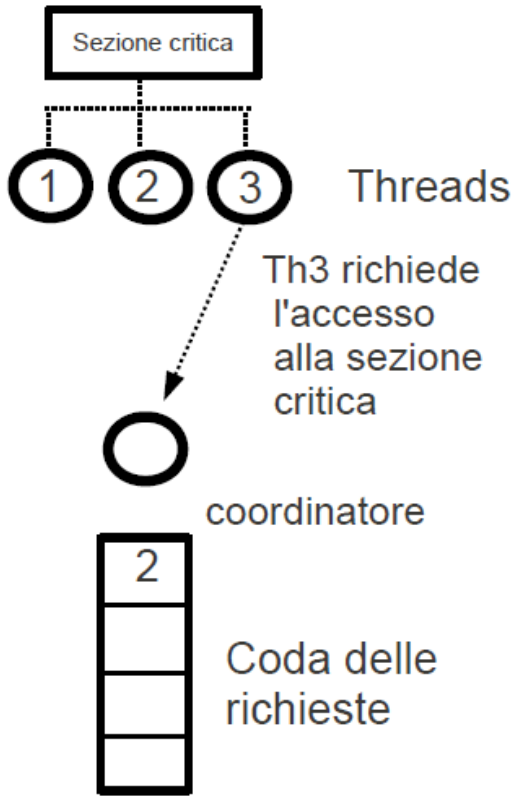
- mentre Th1 é nella sezione critica Th2 chiede di utilizzarla

Algoritmo centralizzato



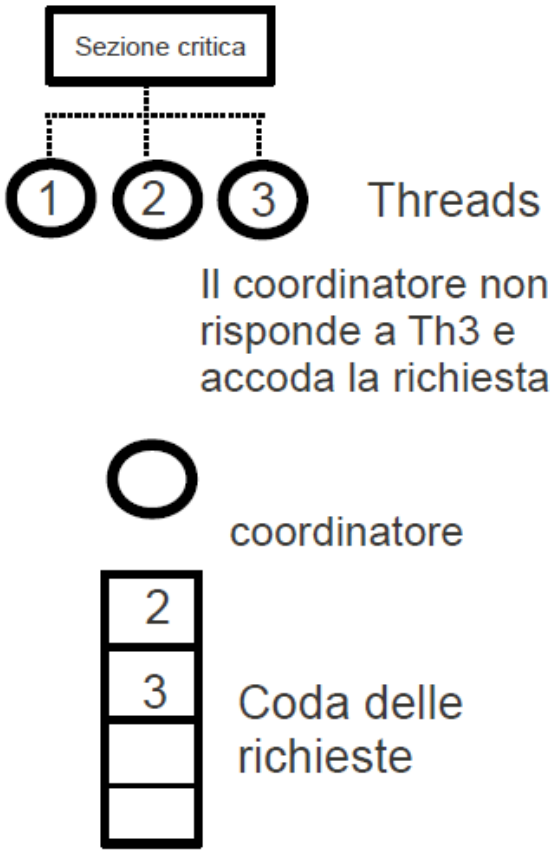
- il coordinatore non risponde e mette Th2 in coda (Th1 é ancora in SC)
- attenzione: i thread fanno attesa attiva aspettando un messaggio dal coordinatore

Algoritmo centralizzato



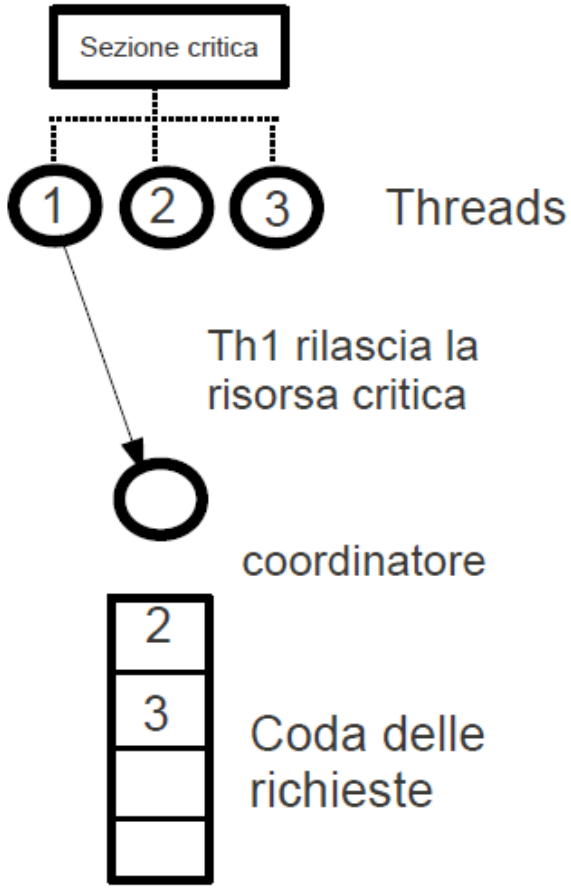
- il thread 3 chiede accesso alla sezione critica

Algoritmo centralizzato



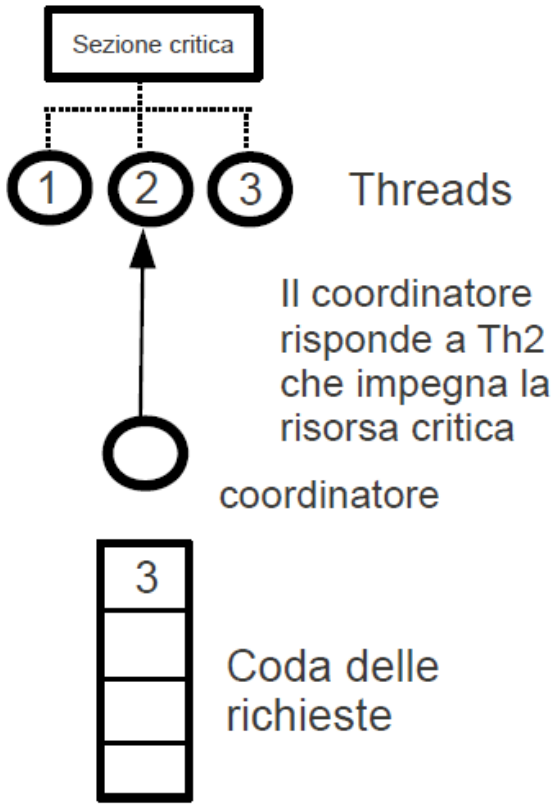
- il coordinatore non risponde (Th3 aspetta in attesa attiva)

Algoritmo centralizzato



- il thread 1 esce dalla sezione critica
- th1 avvisa il coordinatiore conun messaggio di rilascio

Algoritmo centralizzato



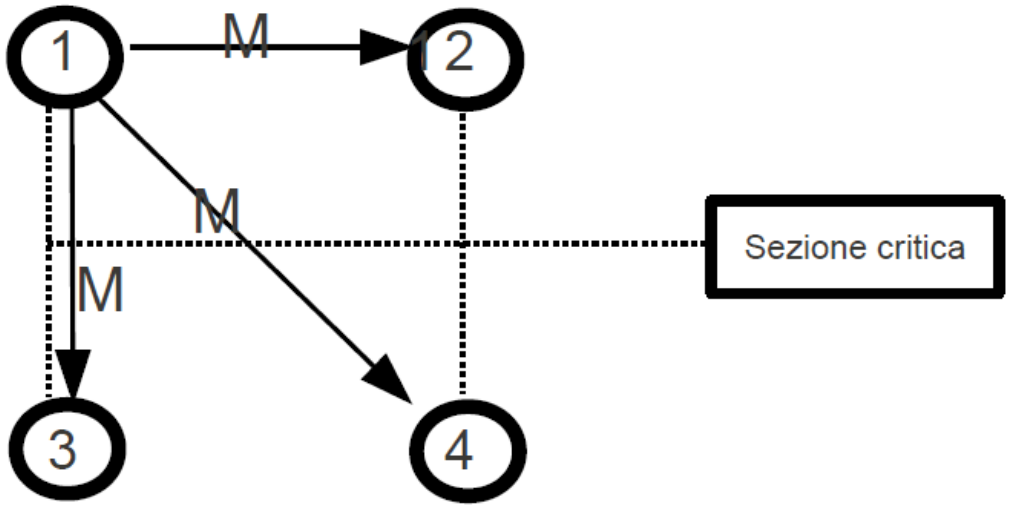
- appena il coordinatore riceve un messaggio di rilascio risponde al primo thread in coda
- il primo thread puó entrare nella SC

Approccio distribuito

- Un processo vuole entrare nella Sezione Critica → invia un messaggio di **Richiesta** a tutti gli altri ptocessi
- Il messaggio contiene il nome della Sezione Critica, il numero del processo e il timestamp (tempo corrente)
- Quando un processo riceve un messaggio di **Richiesta**, allora:
  1. se il processo non é nella Sezione Critica, e non ci vuole entrare, risponde con un messaggio di **OK**
  2. se il processo é giá nella Sezione Critca, nonrisponde e accoda la richiesta
  3. se il processo non é nella Sezione Critica ma ci vuole entrare, seleziona il processo con il minore numero..
  4. se il processo che ha inviato la richiesta ha il numero minore, il ricevente risponde OK. Se il ricevente ha il numero minore, accoda la richiesta.
- Quando un processo esce da una Sezione Critica, invia un messaggio di **OK** a tutti i processi accodati.

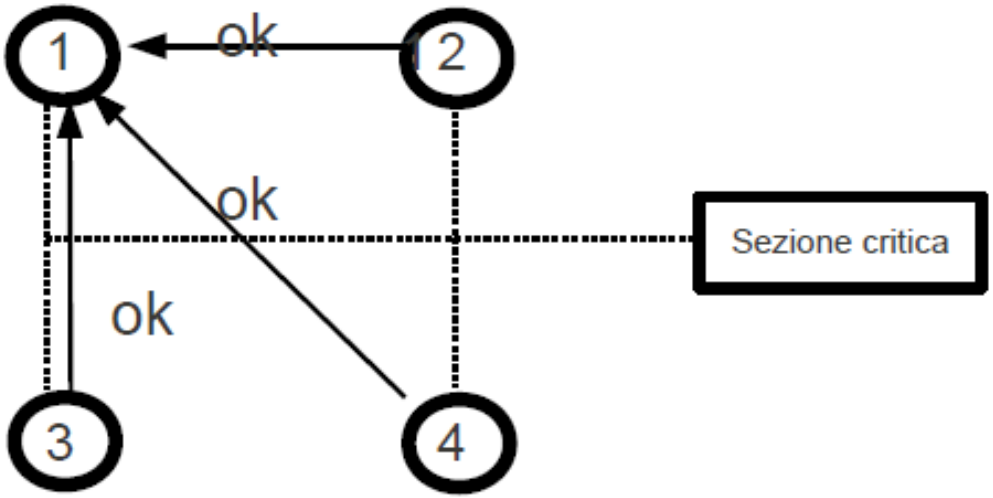
Svantaggio: l'algoritmo é piú costoso e meno robusto dell'approccio centralizzato

Approccio distribuito



- quando un processo vuole entrare nella sezione critica manda un messaggio M a tutti i processi remoti
- M = nome della regione critica — ID del processo — clock

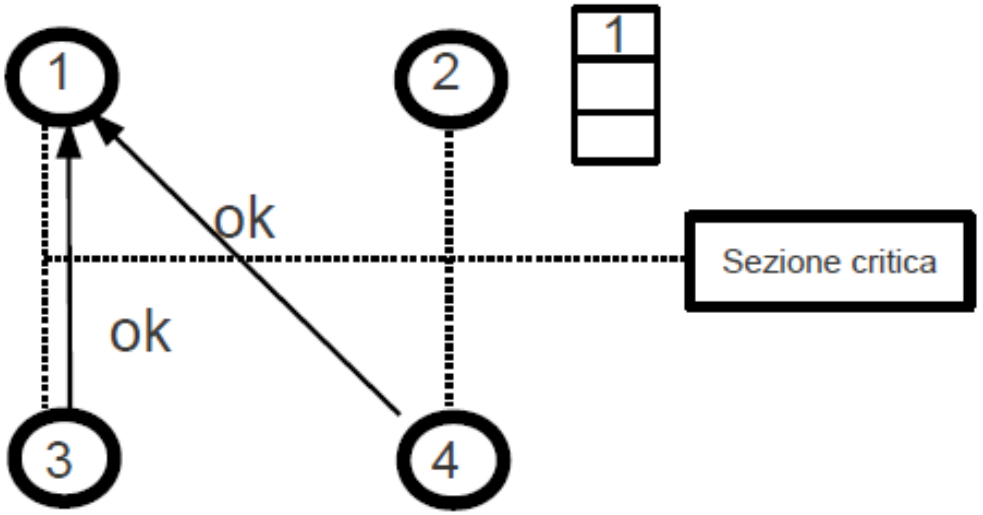
Approccio distribuito



Caso 1

- quando un processo remoto riceve M e non é nella Sezione Critica, risponde 'OK'
- se il processo che vuole entrare nella SC riceve OK da TUTTI i processi remoti, entra nella SC
- il processo nella SC riceve M dal processo k, lo mette in coda

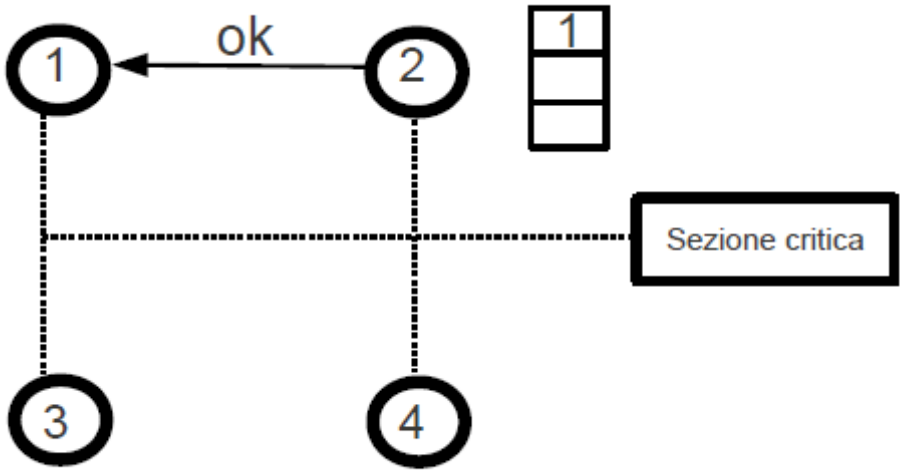
Approccio distribuito



Caso 2

- il processo nella SC riceve M dal processo k, lo mette in coda

Approccio distribuito



Caso 2

- quando il processo nella SC esce dalla sezione critica, manda 'OK' al processo in coda

Approccio distribuito: token ring logico

Algoritmo basato sui token (token ring logico)

- Quando l'anello viene inizializzato, il processo 0 invia un token che circola nell'anello
- Quando un processo acquisisce il toke, vede se vuole entrare nella Sezione Critica, e trattiene il token
- Quando finisce la Sezione Critica, rilascia il token

Vantaggi: assicura la mutua esclusione Svantaggi: affidabilitá

Confronto:

	nr. messaggi	ritardo (tempo messaggi)	problemi
centralizz.	3	2	crash coordinatore
distribuito	2(n-1)	2(n-1)	crash processi
token ring	da 1 a infinito	da 0 a n-1	perdita di token

Algoritmi di elezione del Coordinatore

Stabiliscono il processo coordinatore in ambiente distribuito
Ipotesi:

1. ogni processo ha un numero di processo (tipicamente indirizzo IP)
2. ogni processo conosce gli indirizzi degli altri
3. tutti i processi possono diventare coordinatori
4. per semplicitá, si assume 1 processo per macchina

Algoritmo dello Spaccone

- Quando un processo P nota che il coordinatore non risponde, inizia una elezione
- P invia un messaggio di **Elezione** a tutti i processi di numero maggiore
- Se nessuno risponde, P diventa il coordinatore
- Quando un processo con numero maggiore risponde, diventa il coordinatore
- Quando un processo diventa coordinatore, invia un messaggio di **Vittoria**

Elezione coordinatore con Algoritmo ad Anello

- Si assume che i processi siano collegati ad anello (logicamente o fisicamente)
- Quando un processo P nota che il coordinatore non risponde, inizia una elezione
- P invia un messaggio di **Elezione** a tutti i processi che seguono nell'anello
- Se un processo non risponde, si passa al seguente
- Ad ogni invio del messaggio, il processo aggiunge il suo numero alla lista processi
- Quando il messaggio di **Elezione** ritorna al processo P. Il processo P invia un messaggio di **Coordinatore** a tutti gli altri, per informare che il nuovo Coordinatore é quello con numero maggiore della lista
- Quando il processo P riceve il messaggio di **Coordinatore**, lo rimuove dall'anello