
Simulazione distribuita

Lookahead

Deadlock Detection e Recovery

Sommario

- Messaggi “null”, forma standard, variazione
- Lookahead
 - Cos'è e perché è così importante
 - Si può modificare il suo valore?
- Deadlock Detection e algoritmo di recovery (Chandy/Misra)

Algoritmo basato su messaggi “null”

WHILE (simulation is not over)

wait until each FIFO contains at least one message

remove smallest time stamped event from its FIFO

process that event

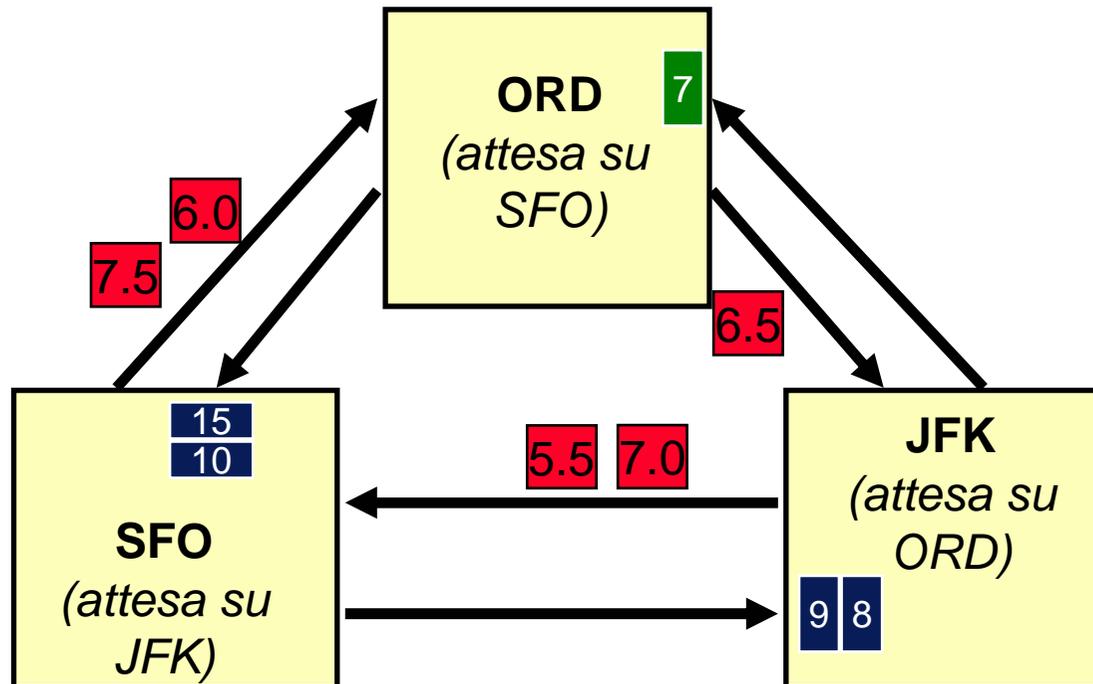
send null messages to neighboring LPs with time stamp indicating a lower bound on future messages sent to that LP (current time plus minimum transit time between airports)

END-LOOP

Variazione: gli LP richiedono esplicitamente un messaggio “null” quando una delle code diventa vuota

- meno messaggi da spedire e processare

Problemi dell'algoritmo



Messaggi "null":

JFK: timestamp = 5.5

SFO: timestamp = 6.0

ORD: timestamp = 6.5

JFK: timestamp = 7.0

SFO: timestamp = 7.5

ORD: processa il messaggio con TS 7

Sono necessari 5 messaggi "null" per un solo evento

0.5

Ritardo minimo tra gli aeroporti:

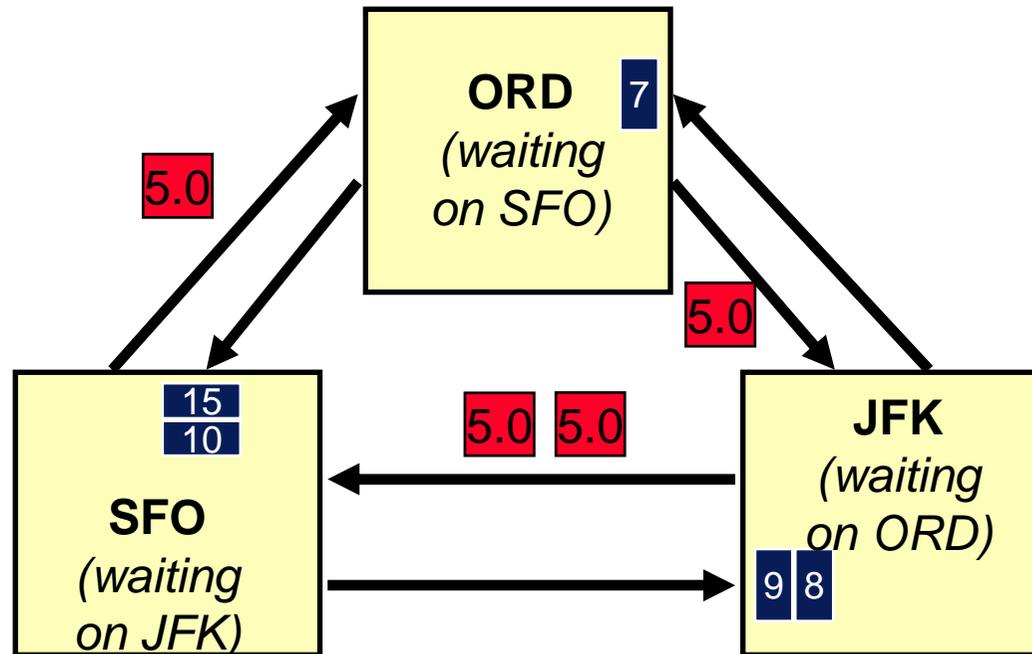
~~3~~ unità di tempo

JFK ha inizialmente tempo 5

Il numero dei messaggi "null" cresce con al ridursi del tempo minimo di volo.

Invece di Deadlock: Livelock

Cosa succederebbe se il ritardo minimo fosse ZERO?



Livelock: ciclo senza fine di messaggi “null” attraverso i quali nessun LP è in grado di avanzare il proprio tempo simulato.

Nel sistema non possono esistere cicli con valore di lookahead pari a zero (zero lookahead cycle).

Lookahead

L'algoritmo basato su messaggi "null" è fondato sulla capacità di predire la distanza del prossimo evento.

Questa distanza temporale è detta **lookahead**.

- ORD è a tempo simulato 5, il tempo minimo di comunicazione tra due aeroporti è fissato a 3, quindi il prossimo messaggio proveniente da ORD deve avere tempo minimo 8

Approfondimento:

- **Link Lookahead** (associato ad ogni link uscente)
- **LP Lookahead** (associato al Logical Process)

Link Lookahead e LP Lookahead sono equivalenti qualora il lookahead su tutti i link uscenti sia uguale

Lookahead e modello simulato

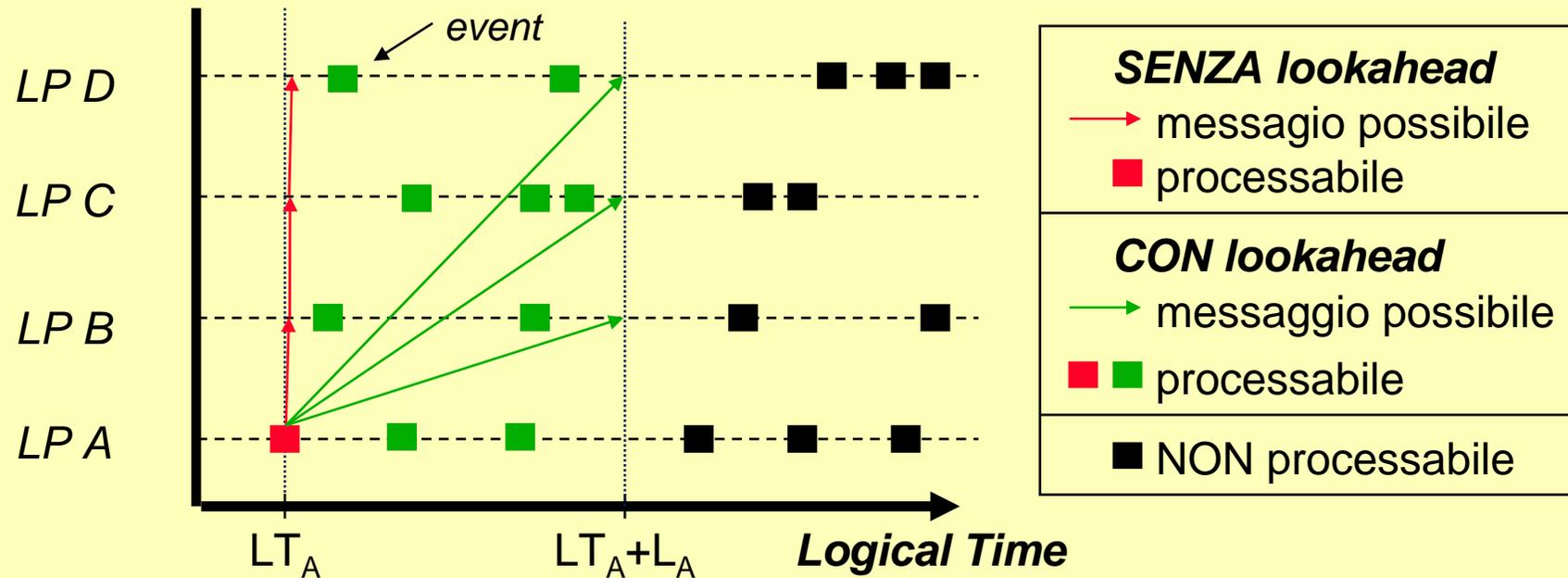
Il Lookahead è dipendente dal modello simulato:

- può derivare da vincoli fisici del sistema simulato, tempo minimo perchè un'entità possa modificare lo stato di un'altra (es. tempo di volo di un aereo, velocità di un proiettile ecc.)
- può derivare da caratteristiche delle entità simulate (es., nella simulazione non esiste nessun evento che possa modificare il comportamento di una certa entità per un determinato lasso di tempo futuro)
- può derivare da un margine di tolleranza (es., l'utente non è in grado di percepire differenze temporali di entità inferiore ad un certo valore)
- la simulazione può essere in grado di precalcolare la prossima interazione che avverrà, la conoscenza all'interno del sistema è sufficiente per avere un controllo preciso su quanto deve ancora avvenire (es., si seguono delle tracce prefissate di comportamento).

Le simulazioni time-stepped hanno un uso implicito del lookahead; gli eventi a “tempo attuale” sono considerati come indipendenti (e quindi concorrenti), i nuovi eventi generati si riferiscono a time-step futuri.

Importanza del Lookahead

Ogni LP deve processare gli eventi in ordine non decrescente



Ogni Logical Process A, dichiara un valore di lookahead L_A : il time stamp di ogni evento generato da LP deve essere $\geq LT_A + L_A$

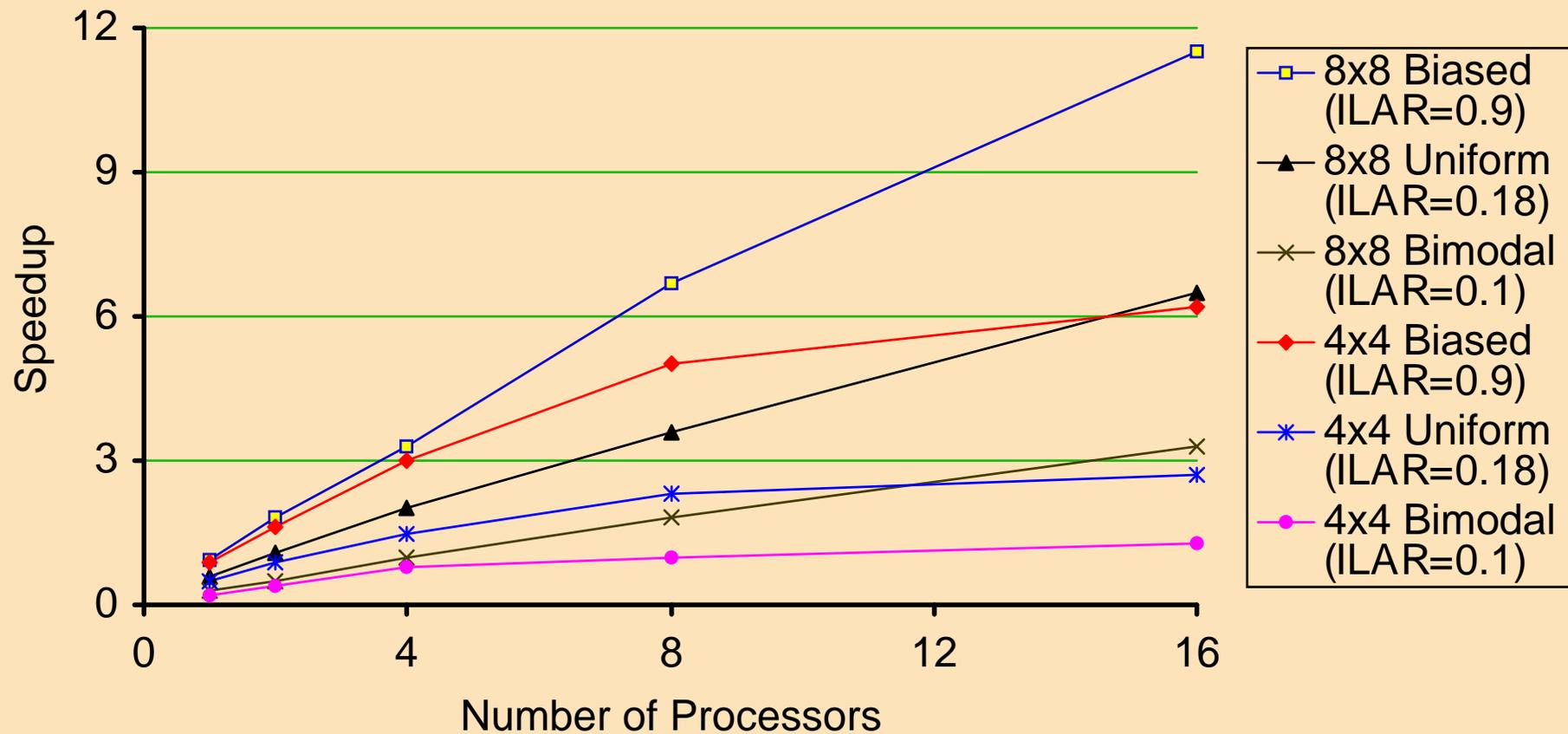
- Il lookahead è presente in modo “più o meno evidente” in tutti gli algoritmi di sincronizzazione pessimistici (detti anche conservativi)

Il lookahead è necessario per l'elaborazione concorrente dei messaggi con time-stamp diverso: grosso impatto sulle prestazioni!

Esempio di simulazione

confronto dello speed-up in presenza di un numero crescente di processori e variazione di ILAR e densità

$ILAR = \text{lookahead} / \text{average time stamp increment}$



Il lookahead ha un grosso impatto sulle prestazioni

Sommario

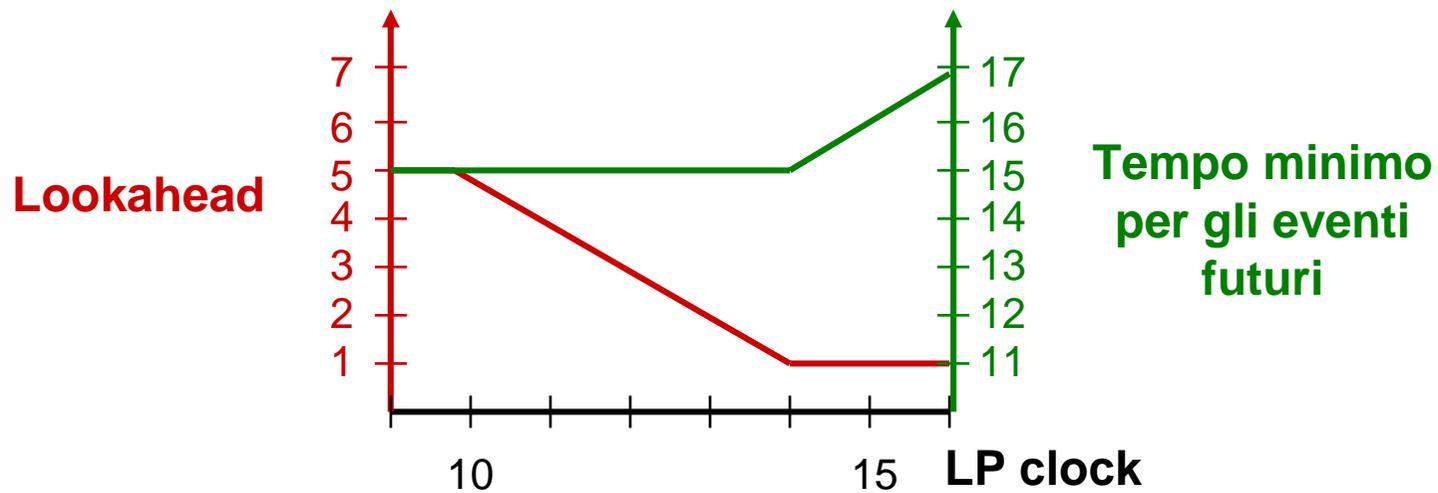
- Messaggi “null”, forma standard, variazione
- Lookahead
 - Cos'è e perché è così importante
 - Si può modificare il suo valore?
- Deadlock Detection e algoritmo di recovery (Chandy/Misra)

Modificare il valore di lookahead

- Incremento
 - Nessun problema, la modifica può essere anche immediata
- Decremento
 - I diritti acquisiti vanno garantiti
 - Il decremento non può essere immediato
 - Se un LP si trova a tempo simulato 10, implica che ha garantito che il suo messaggio successivo avrà tempo minimo 15
 - Se il lookahead fosse immediatamente decrementato ad 1, potrebbe generare un evento con time stamp 11
 - Il lookahead può essere decrementato di K unità di tempo solo dopo che per l'LP è trascorso un tempo simulato di lunghezza minima pari a K

Esempio

- SFO: tempo simulato = 10, lookahead = 5
- Tempo minimo per gli eventi futuri ≥ 15
- SFO: richiede una riduzione del lookahead ad 1



Sommario

- Messaggi “null”, forma standard, variazione
- Lookahead
 - Cos'è e perché è così importante
 - Si può modificare il suo valore?
- Deadlock Detection e algoritmo di recovery (Chandy/Misra)

Deadlock Detection & Recovery

Algoritmo eseguito in maniera indipendente da ogni LP

WHILE (simulation is not over)

wait until each FIFO contains at least one message

remove smallest time stamped event from its FIFO

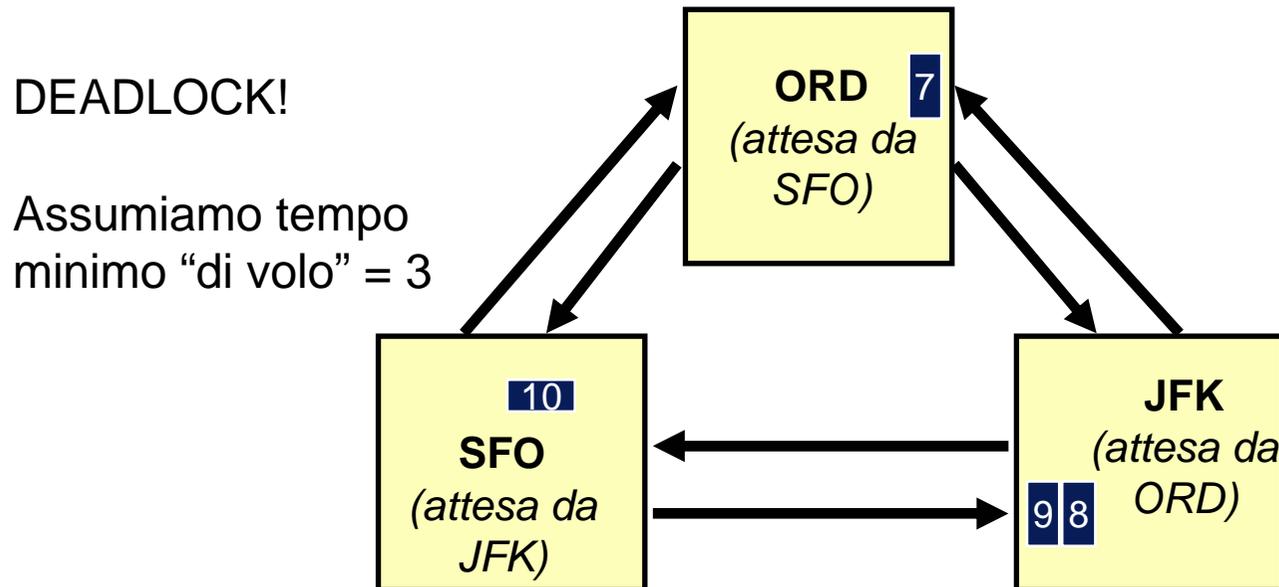
process that event

END-LOOP

- Non ci sono messaggi “null”
- La simulazione procede fino al deadlock
- Meccanismo per **rilevare** il deadlock
- Meccanismo per il **recovery** dal deadlock

Deadlock Recovery

Deadlock recovery: indentificare un evento “safe” (che può essere processato senza violare la causalità del sistema)



Quali eventi sono “safe”?

- Time stamp 7: evento “minimo” presente nel sistema
- Time stamp 8, 9: sono “safe” grazie al lookahead
- Time stamp 10: è “safe” anche questo evento (nota che ha lo stesso time stamp generabile da ORD)

Sommario

- Messaggi “null”, forma standard, variazione
- Lookahead
 - Cos'è e perché è così importante
 - Si può modificare il suo valore?
- Deadlock Detection e algoritmo di recovery (Chandy/Misra)