

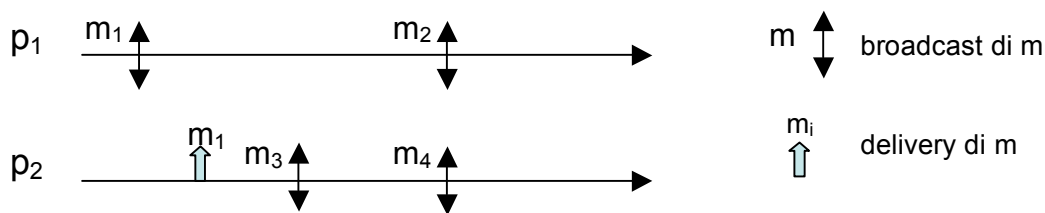
**Sistemi distribuiti 19/04/2006**  
**Corso di Laurea Specialistica in Ingegneria Informatica**

Cognome \_\_\_\_\_ Nome \_\_\_\_\_

**Domanda 1.**

Discutere in maniera comparativa la capacità di catturare relazioni tra eventi in un sistema distribuito rispettivamente dei logical clock e dei vector clock.

**Domanda 2**



Definire tutte le possibili sequenze di messaggi che rispettano il causal order.

**Domanda 3**

Si consideri un sistema di  $n$  processi  $\{p_1, p_2, \dots, p_N\}$ . Tali processi possono comunicare attraverso  $n$  registri regular  $(1, N)$ ,  $\{r_1, r_2, \dots, r_N\}$ . Ogni registro è inizializzato a 0. Il processo  $p_i$  è il writer del registro  $r_i$  e reader di tutti i registri.

Si progettino un algoritmo che risolva il seguente problema:

- I processi devono decidere una sequenza di due interi diversi da 0. I processi devono decidere un valore alla volta.
- Ogni processo propone un valore intero diverso da 0. Il secondo valore è proposto dopo la decisione del primo
- Ogni valore deciso deve essere stato precedentemente proposto. In particolare il secondo valore deciso deve essere scelto SOLO tra i valori proposti dopo la decisione del primo valore.

**Domanda 4**

Discutere il problema della starvation nell'algoritmo di Dijkstra per la mutua esclusione.

**Domanda 5**

L'algoritmo in Figura implementa un Total Order Probabilistico (PTO), il che significa che non sempre sono rispettate le proprietà di agreement e order del Total Order, ma solo in alcuni casi. L'algoritmo procede in una sequenza di round. Ogni processo inizia a round 0 e può inviare al più un solo messaggio per round, se vuole inviare un altro messaggio deve aspettare di cambiare round. Inoltre ogni processo può consegnare (in PTO) un messaggio al round  $r+1$  solo se ha concluso il round  $r$ .

Durante un round, i processi votano per i messaggi inviati. Ogni processo vota per un solo messaggio per ogni round. Ogni processo mantiene una lista di messaggi e voti relativi,

