TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA
Macroarea di Ingegneria
Dipartimento di Ingegneria Civile e Ingegneria Informatica

Esercizi su sincronizzazione dei clock

Corso di Sistemi Distribuiti e Cloud Computing
A.A. 2025/26

Valeria Cardellini

Laurea Magistrale in Ingegneria Informatica

Esercizio 1

- Un orologio indica 10:27:54.0 (hr:min:sec) quando si scopre che è avanti di 4 s. Spiegare perché non è desiderabile reimpostarlo sull'ora corretta in quel momento e mostrare (numericamente) come dovrebbe essere regolato in modo da essere corretto dopo che sono trascorsi 8 s

a) Monotonicità del tempo per poter assegnare timestamp
b) $S = c(E - Tskew) + Tskew$
essendo E il clock erroneo, $Tskew$ = 10:27:54 e c l'incognita
Ma $S = Tskew+4$ quando $E = Tskew+8$
Quindi $Tskew+ 4 = c(Tskew + 8 - Tskew) + Tskew$
da cui c è pari a 0,5

Esercizio 2

Si supponga che un client usi l'algoritmo di Cristian e che i valori del round-trip time e del timestamp restituito dal time server siano:

Round-trip time (ms)	Time (hr:min:sec:msec)
22	10:54:23.674
25	10:54:25.450
20	10:54:28.342

Come imposta il client il suo clock in modo da massimizzare l'accuratezza e perché?

Il client imposta il suo clock fisico a

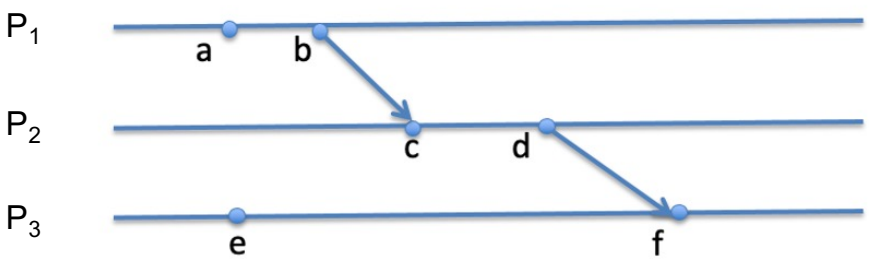
10:54:28.342 + 20 ms/2 = 10:54:28.352

poiché seleziona il valore del clock del time server corrispondente al minimo RTT, in modo da avere la migliore accuratezza essendo $\alpha = \pm (T_{round}/2 - min)$

Esercizio 3

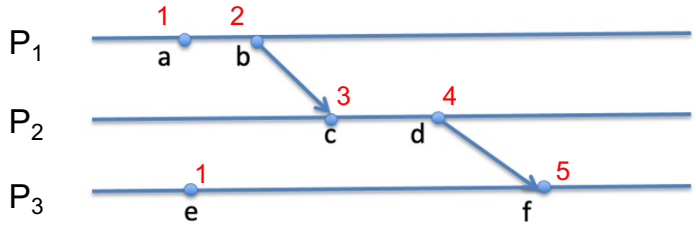
a) Con riferimento al diagramma sottostante, calcolare il clock logico scalare e vettoriale di tutti gli eventi da *a* ad *f*

b) Discutere se, dato il valore del clock sia scalare sia vettoriale determinato al punto a), si può affermare che $c||e$



Esercizio 3: soluzione

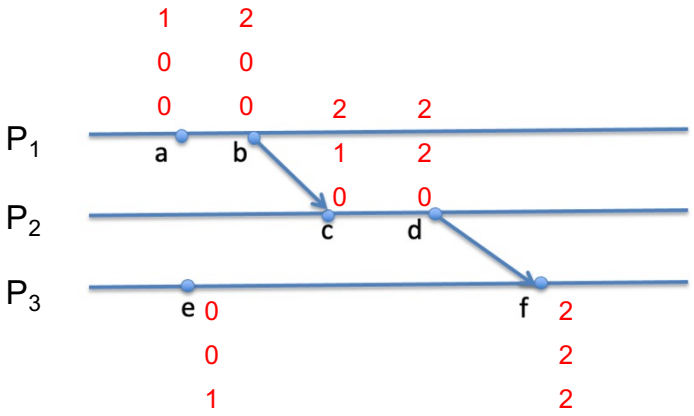
- Clock logico scalare:



Confrontando il valore del clock logico scalare degli eventi *c* ed *e*, si può solo concludere che $c \not\rightarrow e$ essendo 3 > 1 ma non si può concludere se $e \rightarrow c$ oppure $c \parallel e$

Esercizio 3: soluzione

- Clock logico vettoriale:

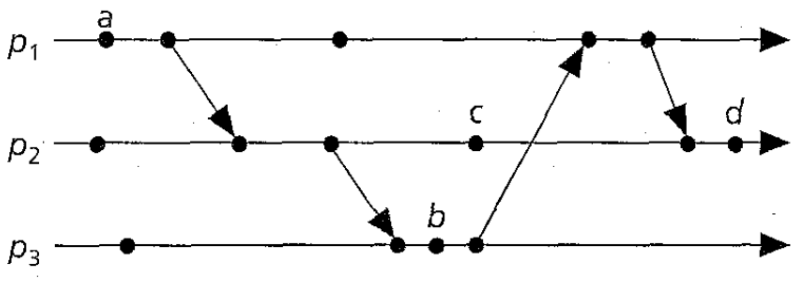


Confrontando il valore del clock logico vettoriale degli eventi *c* ed *e*, si può concludere che $c \parallel e$ essendo not((2 1 0) < (0 0 1)) and not((0 0 1) < (2 1 0))

Esercizio 4

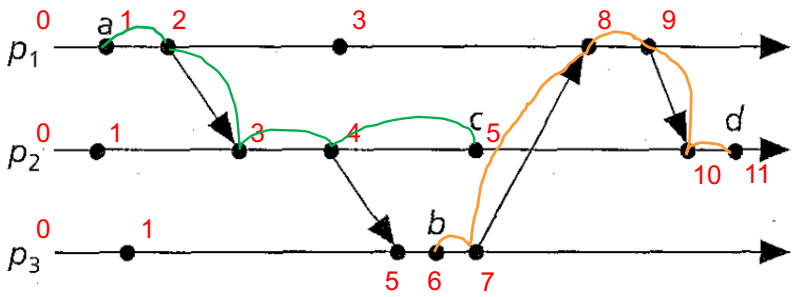
a) Con riferimento al diagramma sottostante, calcolare il clock logico scalare e vettoriale di tutti gli eventi

b) In base ai valori del clock scalare e di quello vettoriale, discutere se le coppie di eventi (*a*, *c*) (*b*, *c*) e (*b*, *d*) sono in relazione happened-before oppure no, motivando opportunamente la risposta



Esercizio 4: soluzione

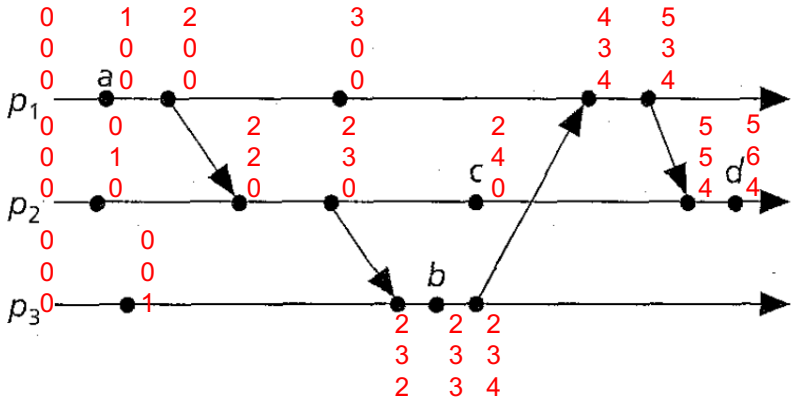
- Clock logico scalare:



  - Confrontando il valore del clock logico scalare degli eventi, si può solo concludere che:
    - $c \nrightarrow a$, $b \nrightarrow c$, $d \nrightarrow b$
  - Applicando le proprietà della relazione happened-before:
    - $a \rightarrow c$ (sequenza verde), $b \parallel c$, $b \rightarrow d$ (sequenza arancione)

Esercizio 4: soluzione

- Clock logico vettoriale:



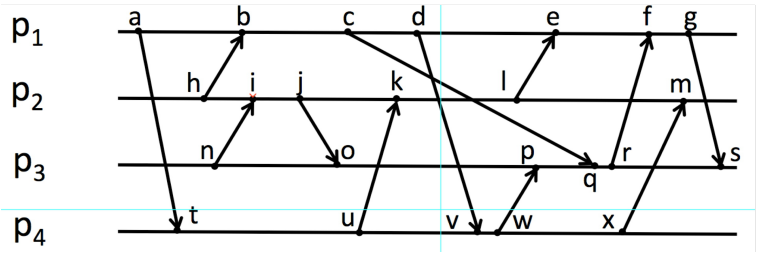
  - Confrontando il valore del clock logico vettoriale degli eventi, si può concludere che:
    - $a \rightarrow c$ essendo (1 0 0) < (2 4 0)
    - $b \parallel c$ essendo not((2 3 3) < (2 4 0)) and not((2 4 0) < (2 3 3))
    - $b \rightarrow d$ essendo (2 3 3) < (5 6 4)

Esercizio 5

a) Con riferimento al diagramma sottostante, calcolare il clock logico scalare di tutti gli eventi da *a* ad *x*

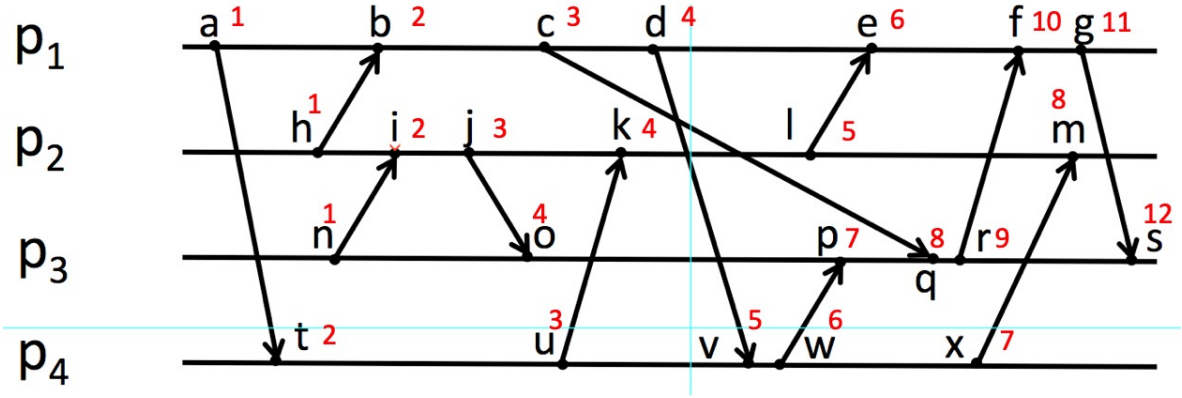
b) Calcolare il clock logico vettoriale di tutti gli eventi da *a* ad *x*

c) In base ai valori calcolati, si spieghi se può essere identificata una violazione della causalità, motivando la risposta



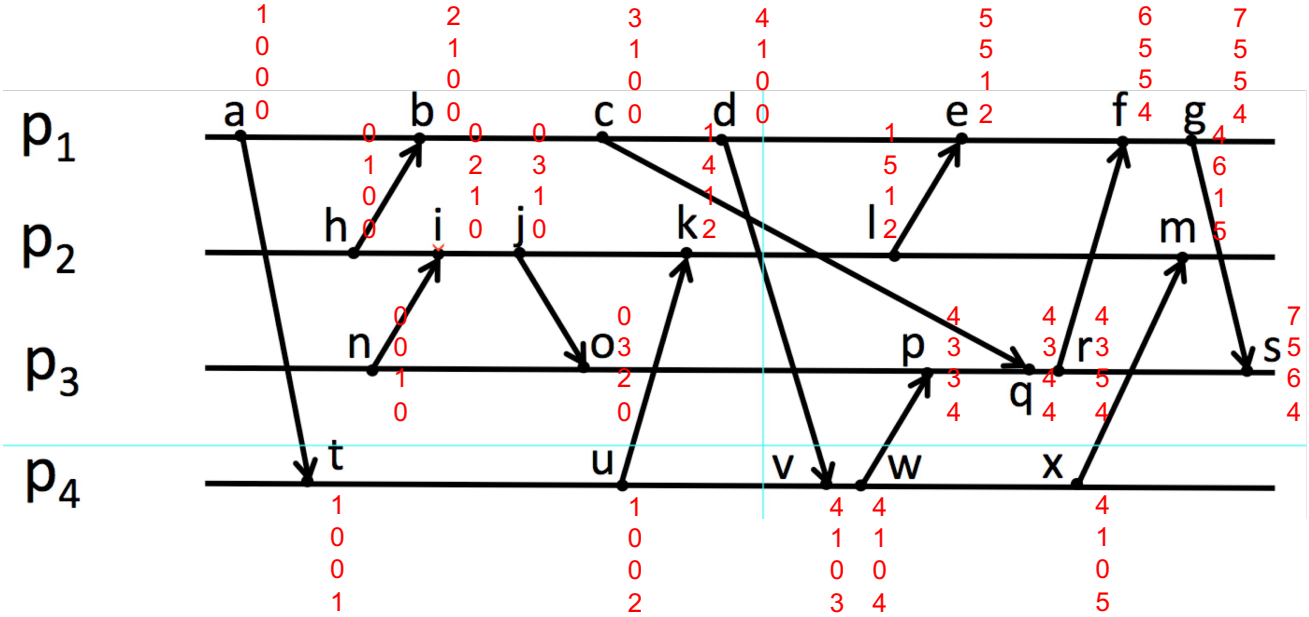
Esercizio 5: soluzione

- Clock logico scalare:



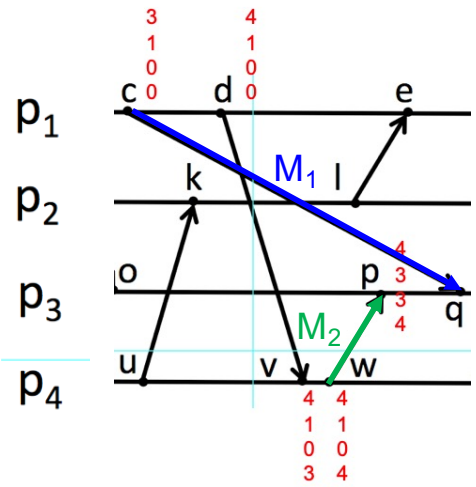
Esercizio 5: soluzione

- Clock logico vettoriale:



Esercizio 5: soluzione

- Violazione di causalità: gli eventi coinvolti sono *c, p* e *q* (p_3 riceve prima *p* e poi *q*)



  - p_3 può identificare la violazione di causalità tramite il clock vettoriale
  - M_1 inviato da p_1 ha timestamp (3 1 0 0)
  - M_2 inviato da p_4 ha timestamp (4 1 0 4)
  - (3 1 0 0) < (4 1 0 4): un processo che ha già visto *c* (ovvero p_4) ha inviato un messaggio a p_3 prima che p_3 abbia ricevuto M_1
  - Quindi p_3 si accorge di aver ricevuto prima l'effetto e poi la causa confrontando i due timestamp: quello di M_1 (ricevuto da p_3 dopo M_2) è minore del timestamp di M_2