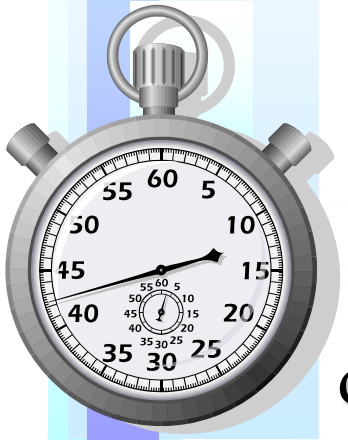


Parte: 4b – Clock e Ordinamenti Temporali

Corso di: Sistemi Distribuiti
Lauree in: Ingegneria Informatica,
delle Telecomunicazioni ed Informatica di Scienze

Prof. Paolo Nesi

Department of Systems and Informatics, University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-2758515, fax: +39-055-2758570

DISIT Lab, Sistemi Distribuiti e Tecnologie Internet

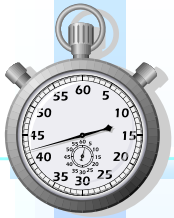
http://www.disit.dinfo.unifi.it/

paolo.nesi@unifi.it
http://www.disit.dinfo.unifi.it/nesi

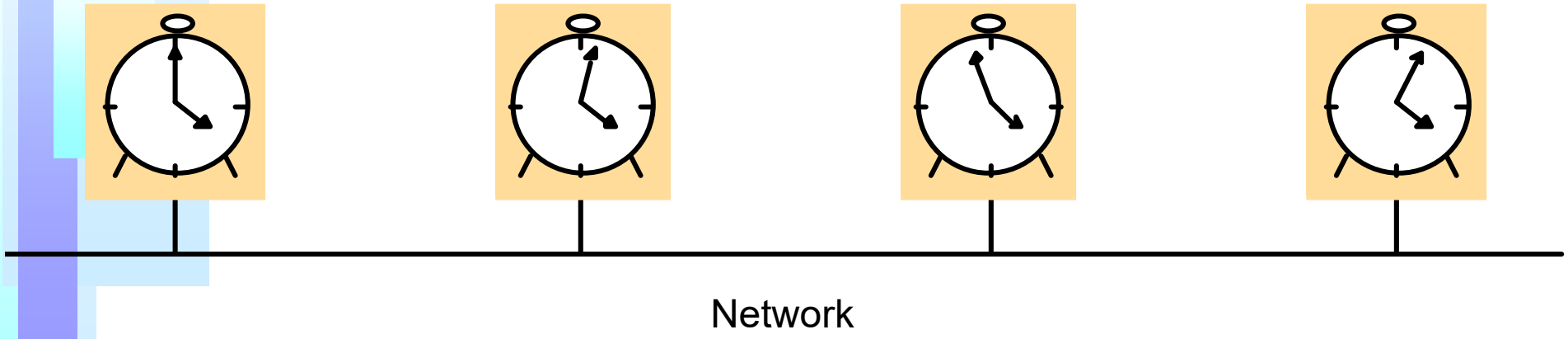

Clock e ordinamenti

- Motivazioni
- Problemi di sincronizzazione fra nodi
- Algoritmi di sincronizzazione
- Sincronizzazione di tempo assoluto fra nodi
- Ordinamento di eventi sui nodi

- Per riferimenti si veda il libro.



Problemi di Sincronizzazione fra Nodi



- Differenze di tempo fra Clock/Orologi (valore assoluto del tempo) nei vari nodi/processi sulla rete

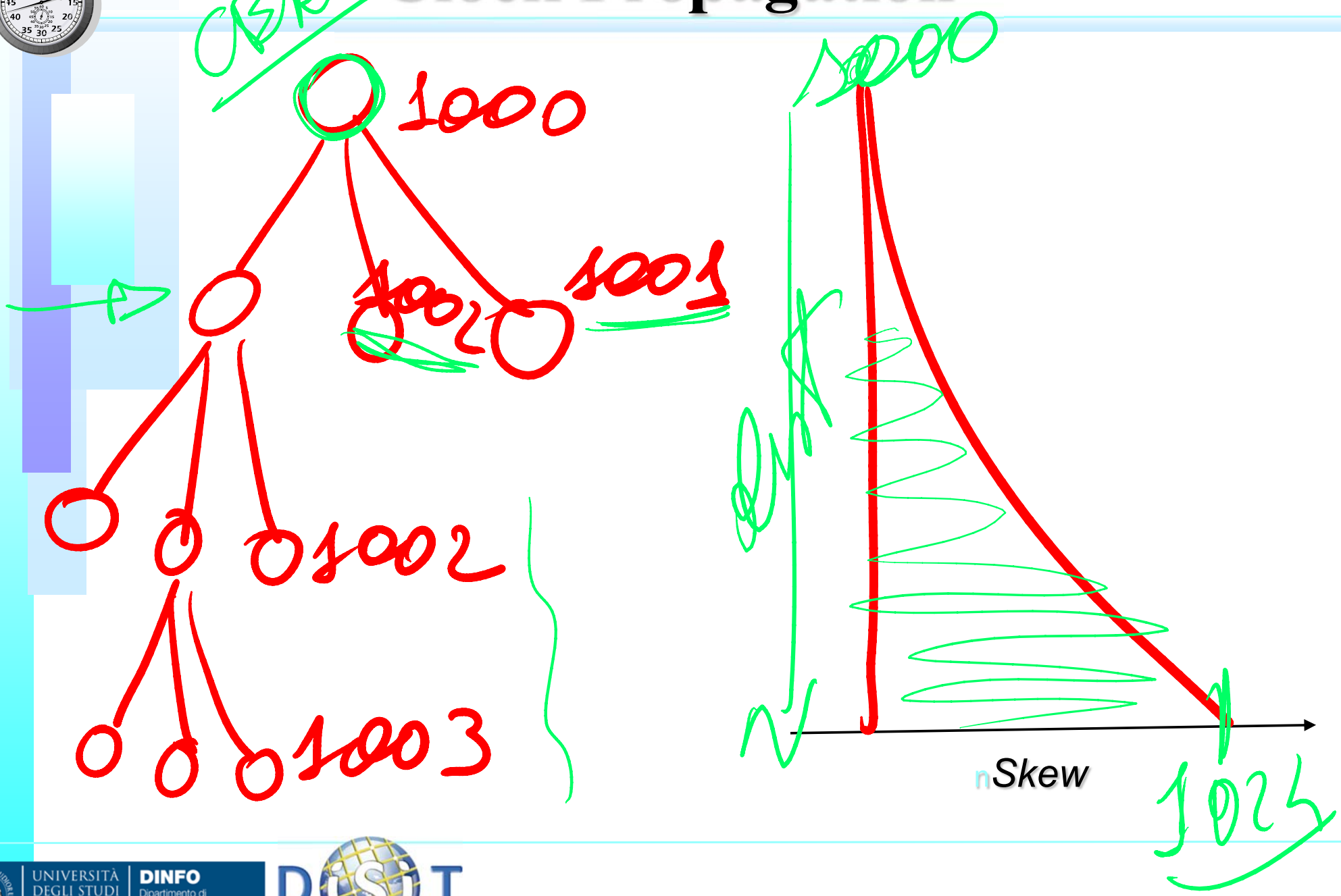
Motivazioni

- I nodi di un sistema distribuito possono avere **necessità** di effettuare azioni **sincronizzate** rispetto allo stesso tempo assoluto e non a fronte di un semplice comando d sinc.
  - Il comando su TCP/IP può arrivare con un ritardo (anche non predicibile) e questo in molti casi non e' accettabile
  - Per questo fine si possono inviare comandi su canali I/O specifici o tramite protocolli deterministici…….. ma se si ha solo la rete….
- In sistemi distribuiti i messaggi arrivano con dei **TimeStamp** in modo che si possa sapere in che ordine devono essere eseguiti (sono stati inviati)
  - I TimeStamp provenienti da nodi diversi devono riferirsi allo stesso tempo assoluto altrimenti il loro ordinamento (assoluto) non e' possibile, non ha senso, puo' essere errato.
  - L'ordinamento serve per ordinare in modo Causale i Comandi
- In sistemi distribuiti si effettuano anche misure del tempo per programmare delle durate, dei ritardi
  - Nelle stima delle durate (delta t) gli errori sono minori

Problemi di Sincronizzazione fra Nodi

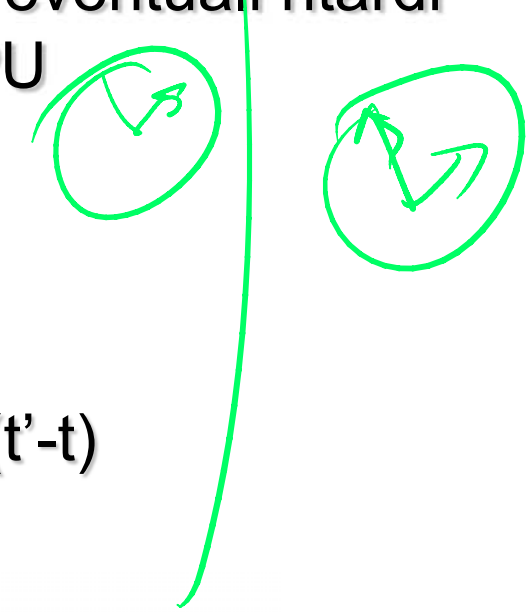
- La precisione degli orologi al quarzo e' dell'ordine di 10^{-6} , quelli molto precisi sono anche 10^{-7} o 10^{-8}
  - ***Drift o deriva***
- Un quarzo che ha precisione 10^{-6} puo' produrre uno scarto di un secondo ogni 11.6 giorni circa
  - Pertanto si ha circa un decimo di secondo ogni giorno
- In certe applicazioni real time una differenza di un decimo di secondo non e' accettabile,
  - per esempio quando devo far partire due audio insieme, un grande stadio, una stazione, etc.
  - un umano non addestrato sente differenze di 10 ms
- Questi problemi producono uno SKEW / deriva fra clock
  - differenza di tempo assoluto o comunque di tempo fra clock sincronizzati

Clock Propagation



Problemi di Sincronizzazione fra Nodi

- I singoli processi possono avere un proprio clock
- certamente usare quello del sistema che e' HW e' meglio poiche' qualsiasi orologio SW dipende da eventuali ritardi di **accesso al processo** del clock alla CPU



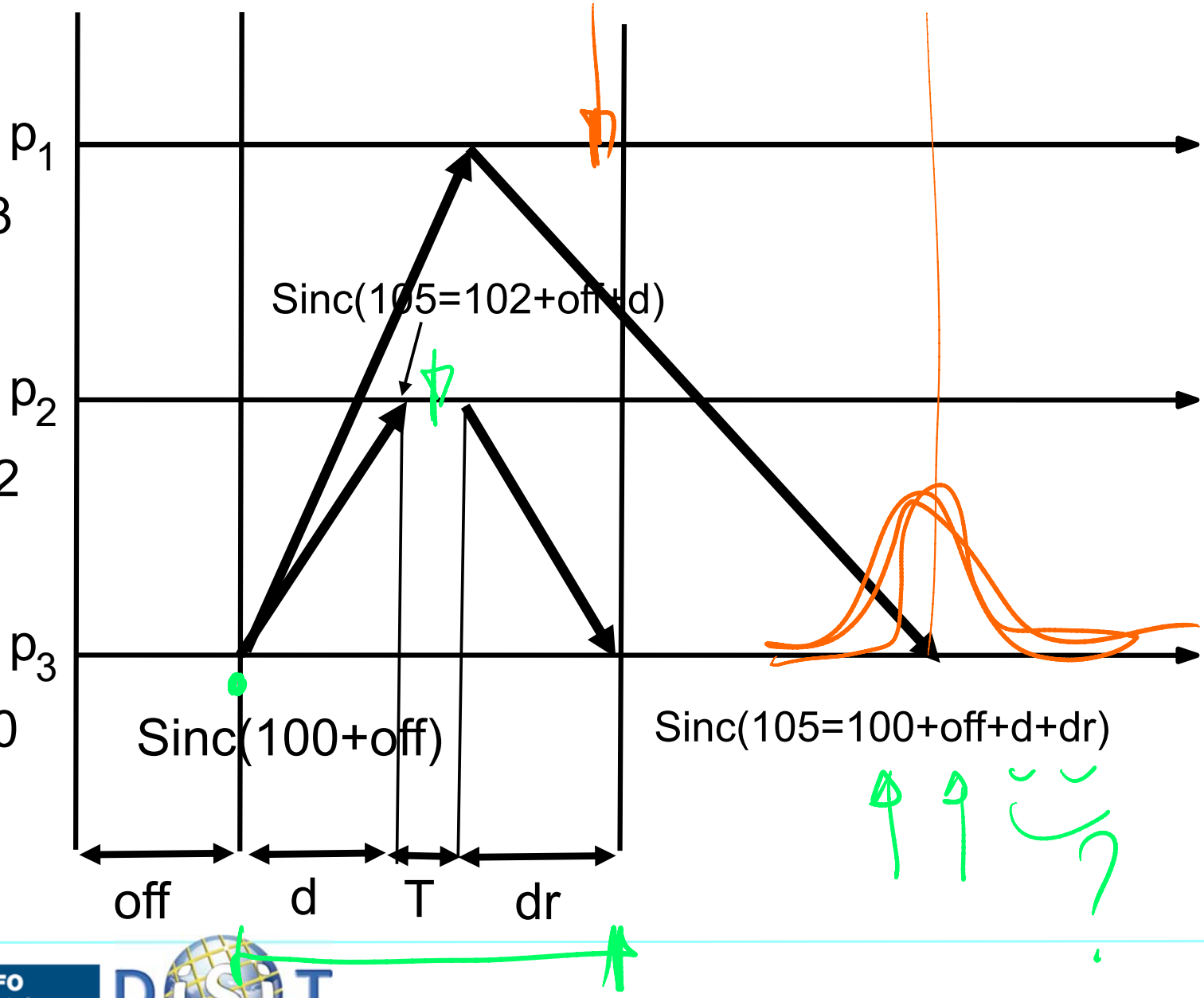
- Se due HW clock hanno una deriva:

$$(1-d)(t'-t) \;<=\; H(t')-H(t) \;<=\; (1+d)(t'-t)$$

Dove:

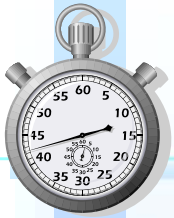
- H(t) e' il valore del clock del computer H al tempo t.
- d e' il valore della deriva per esempio 10^{-6}

Sincronizzazione fra Nodi, MODO A



- d: delay di andata
- dr: delay di ritorno
- T: tempo di esecuzione, reazione
- off: offset, scarto di sincronizzazione reale

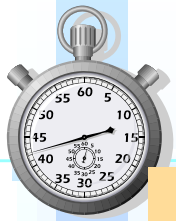
- Di Seguito vediamo modelli di correzione:
  - MODO A
  - MODO B
  - Modello C



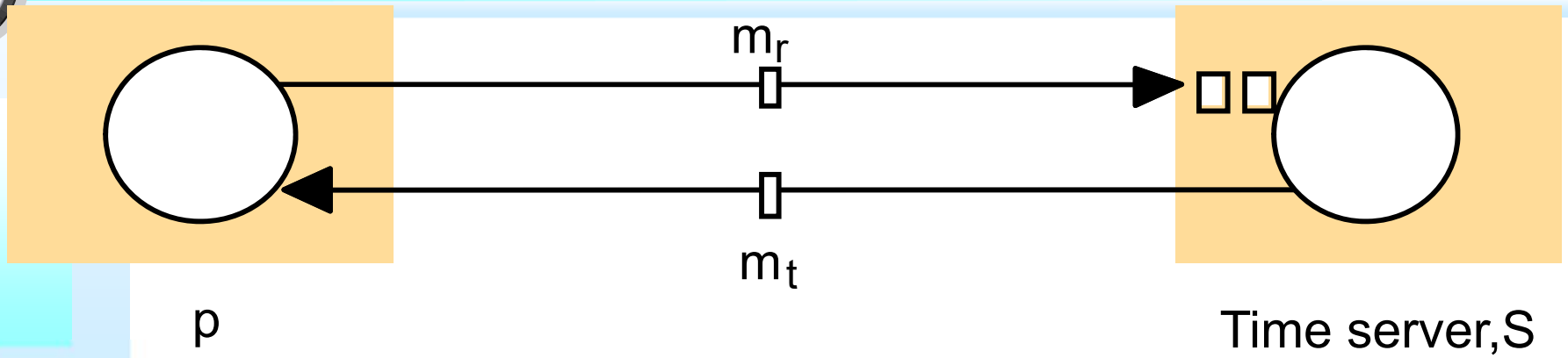
Sincronizzazione fra Nodi, MODO A

- Da un nodo Server di sincronizza gli altri nodi con dei messaggi
- I messaggi arrivano ai nodi con
  tempi non predicibili: d(i), dove i e' il nodo
- Le risposte arrivano al Server dai nodi con
  tempi non predicibili: dr(i), dove i e' il nodo

- Tramite varie iterazioni e' possibile stimare per ogni nodo
  - valori medi, minimi e massimi per: d(i) e dr(i), certamente per d(i)+T+dr(i). Dove T e' il tempo di risposta del nodo, non noto.
  - Alla fine e' possibile considerare anche:
    - ritardo(i) = [mean(d(i)+T+dr(i))] / 2
  - Come valore medio di ritardo, che può essere comunicato al momento della sincronizzazione dal Server:
    - Sinc(i,ritardo(i))
- Quando si comunica Sinc() non e' detto che si abbia un tempo effettivo simile a quello medio, da questo nascono degli errori.

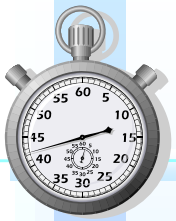


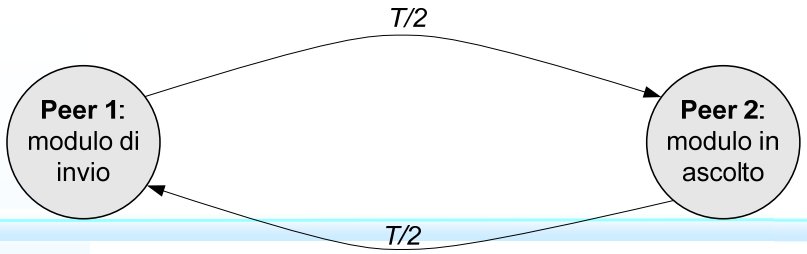
MODO B, Cristian (1989)



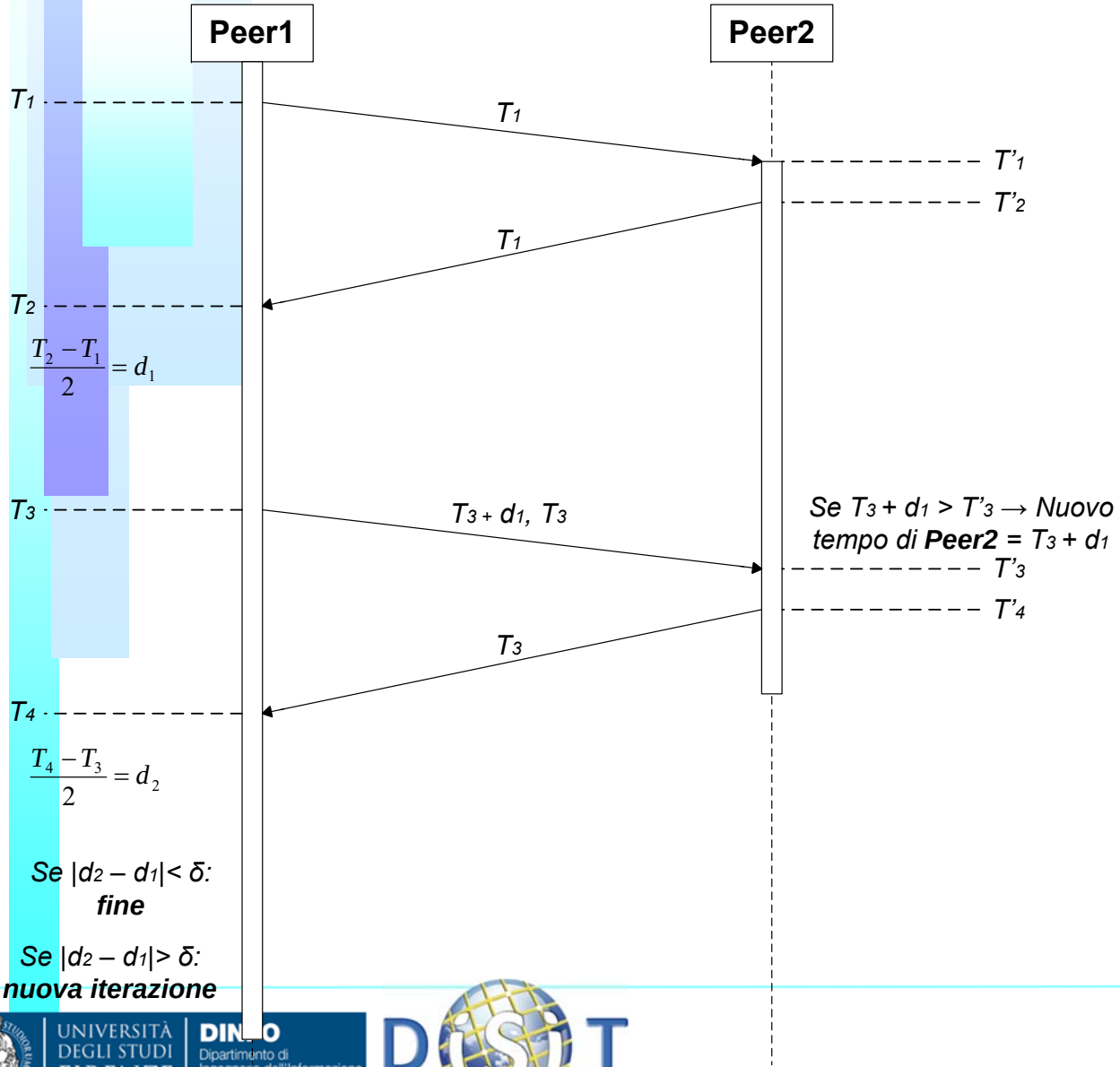
- E' il client P che chiede al *Server S* il tempo assoluto T
- P misura il tempo che passa da quando si manda *mr* a quando si riceve *mt*:
  - detto *Tround* (tempo per fare il giro)
- P impone/setta il suo tempo assoluto pari a (T+Tround/2)
- Errore pari a ±*(Tround/2 - Tmin)*
  - Tmin e' il minimo dei Tround possibili
  - Pertanto in questo modo e' possibile capire se l'errore e' accettabile per il tipo di applicazioni in cui si vuole utilizzare tale metodo

Sincronizzazione





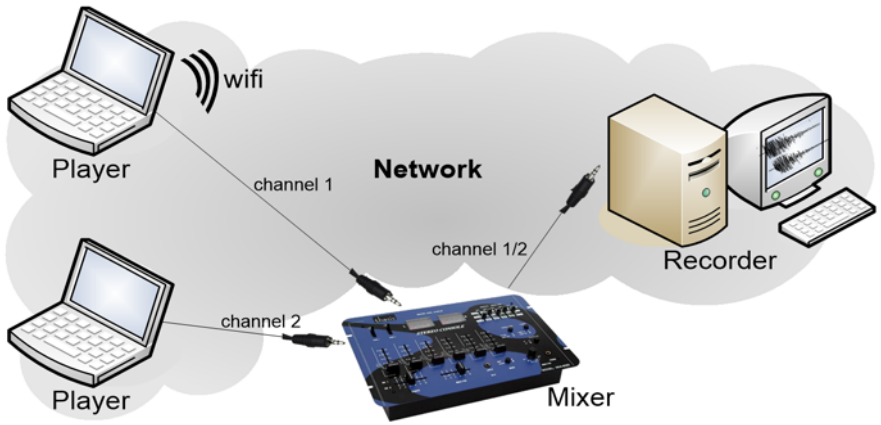
roundtrip-time = T → ritardo di rete = T/2



- Il peer con *Clock Logico* più elevato impone il tempo agli altri peer.
- Il nuovo tempo impostato non può mai essere inferiore al precedente (funzione monotona crescente).
- Alla connessione ogni peer imposta a 0 il suo clock logico e viene 'contattato' dagli altri peer attivi per essere sincronizzato.
- Si tenta di minimizzare l'errore nella comunicazione del ritardo di rete (d_1).
- δ si rilassa per raggiungere la convergenza.
- Necessaria periodica ri-sincronizzazione a causa della deriva dei clock fisici.

Risultati: Sincronizzazione

- Test di sincronizzazione tramite impulsi audio.
- Delay di esecuzione di 1 secondo.
- Mixer collegato a macchina in registrazione.
- Ingressi al mixer: altoparlanti di 2 pc (uno connesso tramite wifi, l'altro tramite cavo).
- Misura del ritardo nell'esecuzione dell'impulso.
- **0,2ms < Ritardo < 2,5ms**

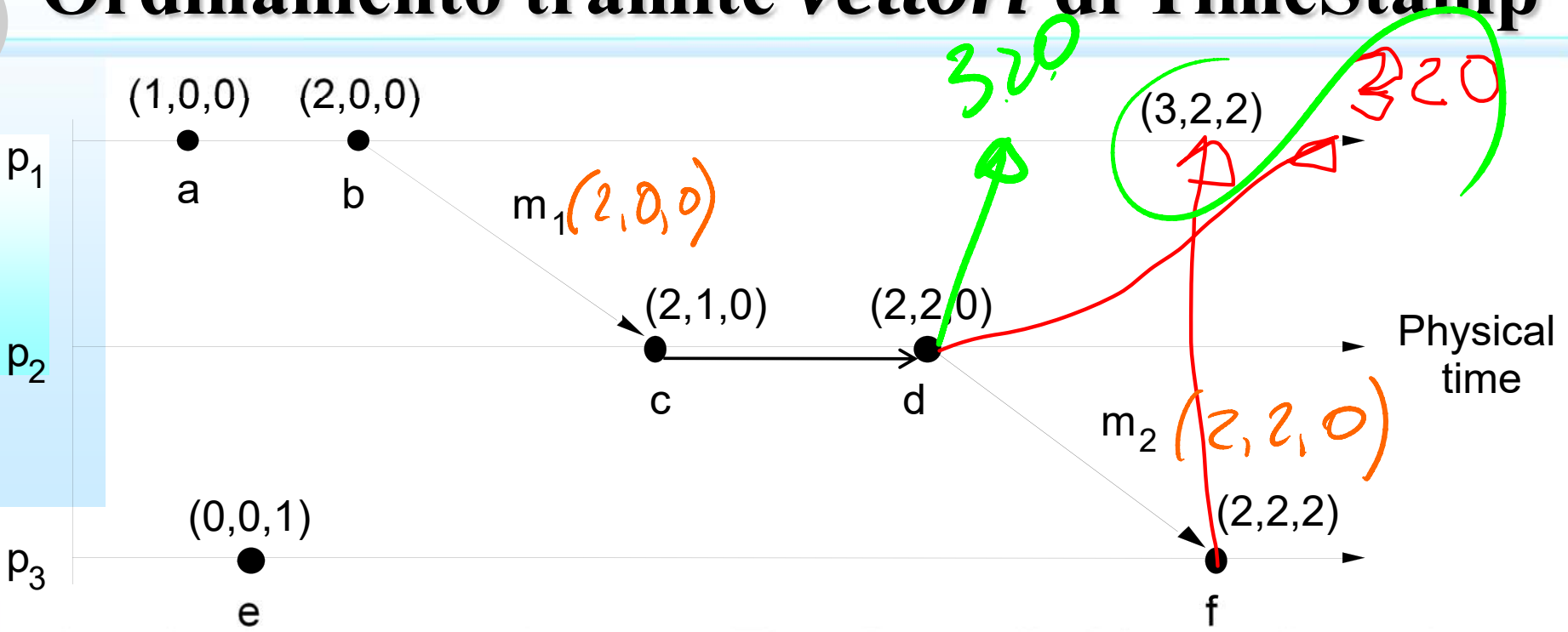


Altri Metodi

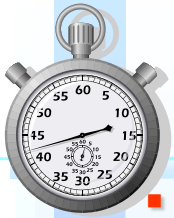
- **Modo B** non porta a risultati soddisfacenti quando i tempi di trasmissione sulla rete (Tround) sono elevati,
  - valido per Intranet
- In reti geografiche sono necessarie altre soluzioni
  - le soluzioni iterative permettono di ridurre l'errore tramite approssimazioni successive con l'incrocio di messaggi fra i vari nodi della rete

- In generale la precisione infinita/assoluta non e' possibile
- L'ordinamento di eventi e' però possibile per garantire la causalità fra comandi in sistemi di tempo reale stretto per CSCW o che altro
  - Queste cose sono impossibili da realizzare con una base dei tempi assoluta

Ordinamento tramite *vettori* di TimeStamp



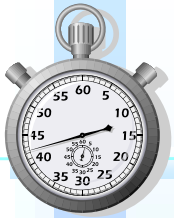
- Ogni nodo riceve eventi con un TimeStamp/Lable ereditato da altri eventi (P1, P2, P3)
- Ogni nodo tiene conto di tutti i TimeStamp/lable degli altri N nodi
- Si crea una catena di eventi gollegati ed etichettati
- I comandi sono ordinabili (2,2,0) e' successivo a (2,1,0)
- E' possibile in questo modo creare degli ordinamenti



Sequenza di eventi

- **Eventi:**
  - (1,0,0): evento a T=1 di P1
  - (2,0,0): evento a T=2 di spedizione di P1, succ a (1,0,0)
  - (2,1,0): evento succ a (2,0,0) di P2, ricezione, a T=1 di P2
  - (2,2,0): evento succ a (2,1,0) di P2, invio, a T=2 di P2
  - (2,2,2): evento a T=2 di P3, ricezione di (2,2,0), succ a (0,0,1) T=1 di P3
  - ……
  - (3,2,2) potrebbe arrivare prima di (3,2,0) su P1, ma se questo accade P1 puo' riordinare tali eventi

- Da un eventuale non corretto ordinamento e' possibile riportarsi all'ordine corretto
  - creare degli ordinamenti
  - Complessità o(N)
  - Ha senso solo per comandi dipendenti e collegati, dovrebbe essere evitato per comandi indipendenti



Ordinamento di comandi/eventi

- **Un sistema gode della proprieta' di convergenza** se:
  - si accorge di condizioni di disordine temporale e riporta con delle operazioni di ordinamenti in uncerto tempo ad uno stato ordinato uguale per tutti
  - (importante per sistemi cooperativi, vi sono degli esempi nelle slide dei sistemi collaborativi)

- Non e' detto che il sistema possa rimettere semplicemente in ordine i comandi quando se ne accorge,
  - il danno potrebbe essere gia' stato fatto, per esempio in un sistema di controllo per macchine obliteratrici
    - Muovi e poi buca e' diverso da buca e poi muovi
    - Il buco si trovera' in posizioni diverse ☺
  - se si parla di dati si dovrebbe garantire l'UNDO di ogni comando, in certi casi fisici non e' possibile, il buco e' fatto ☹