



Sistemi Distribuiti


Corso di Laurea in Ingegneria

Prof. Paolo Nesi
Parte 4b: Clock e ordinamenti
Department of Systems and Informatics
University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-4796523, fax: +39-055-4796363

Lab: DISIT, Sistemi Distribuiti e Tecnologie Internet
nesi@dsi.unifi.it, paolo.nesi@unifi.it
<http://www.disit.dsi.unifi.it>




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 1



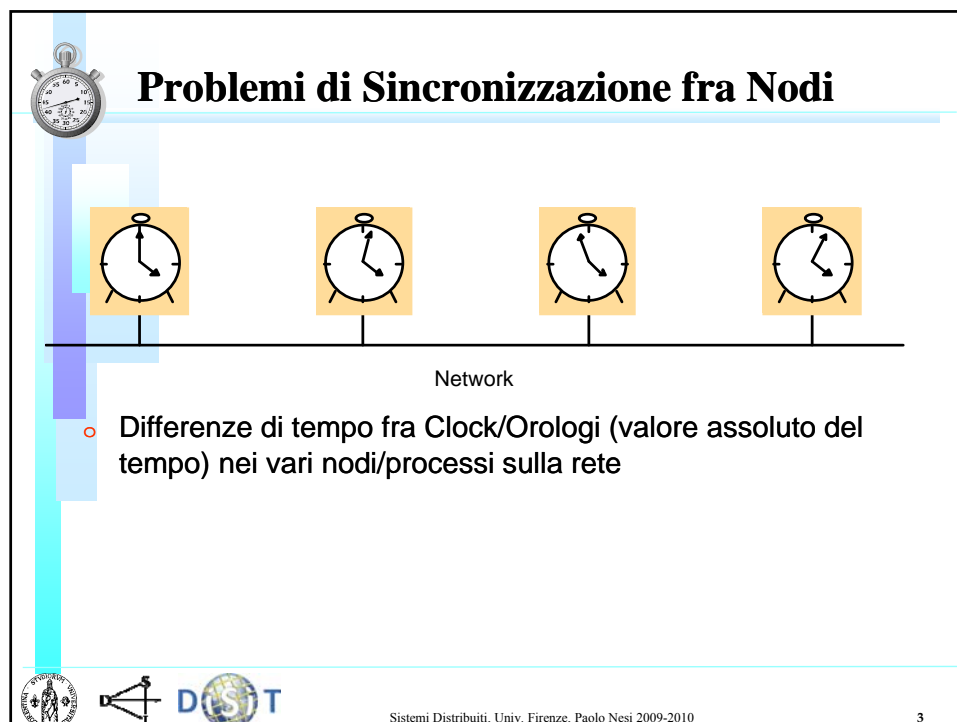
Clock e ordinamenti

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

- Per riferimenti si veda il libro.




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 2



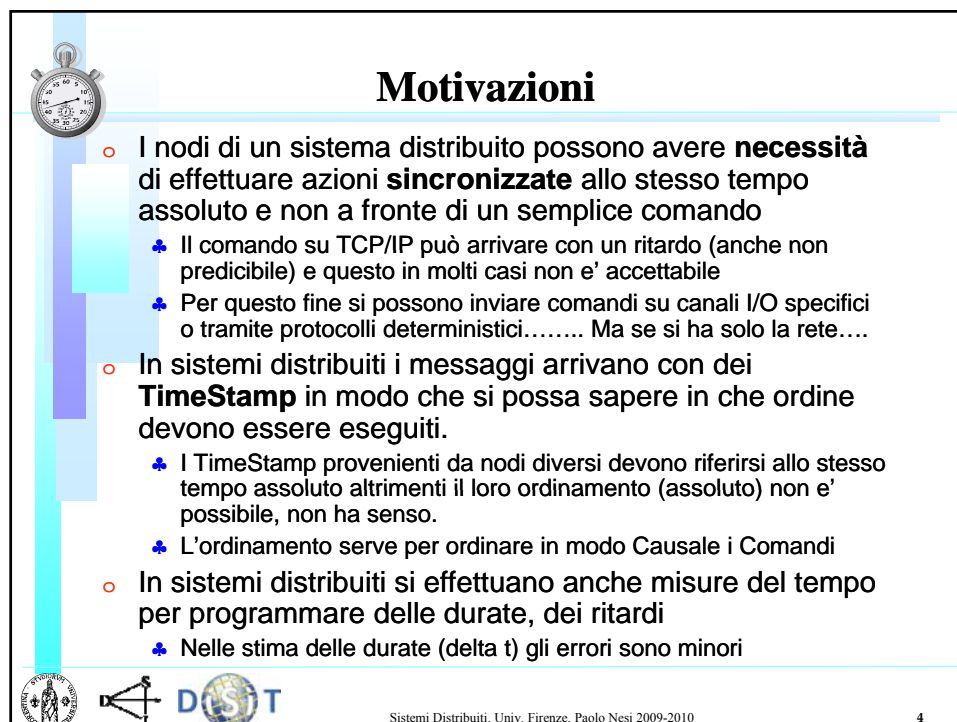
Problemi di Sincronizzazione fra Nodi

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

Network




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 3




Motivazioni

- I nodi di un sistema distribuito possono avere **necessità** di effettuare azioni **sincronizzate** allo stesso tempo assoluto e non a fronte di un semplice comando
 - ♣ 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.
 - ♣ I TimeStamp provenienti da nodi diversi devono riferirsi allo stesso tempo assoluto altrimenti il loro ordinamento (assoluto) non e' possibile, non ha senso.
 - ♣ 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




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 4




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 umano non addestrato sente differenze di 10 ms
- Questi problemi producono uno SKEW fra clock
 - ♣ differenza di tempo assoluto o comunque di tempo fra clock sincronizzati

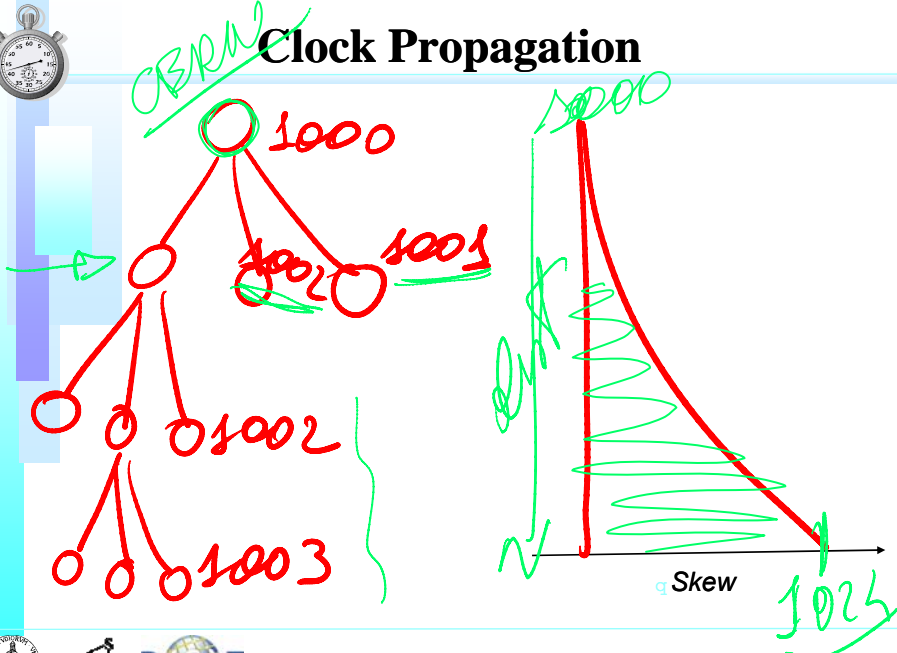


Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 5




Clock Propagation

CBRN



Skew



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 6



Problemi di Sincronizzazione fra Nodi

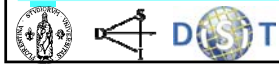
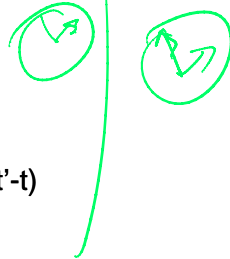
- o I singoli processi possono avere un proprio clock
- o certamente usare quello del sistema che e' HW e' meglio poiche' qualsiasi orologio SW dipende da eventuali ritardi di accesso al processo del clock alla CPU
- o Se due HW clock hanno una deriva:

$$(1-d)(t'-t) \leq H(t')-H(t) \leq (1+d)(t'-t)$$

ΔT

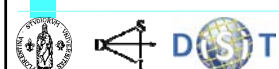
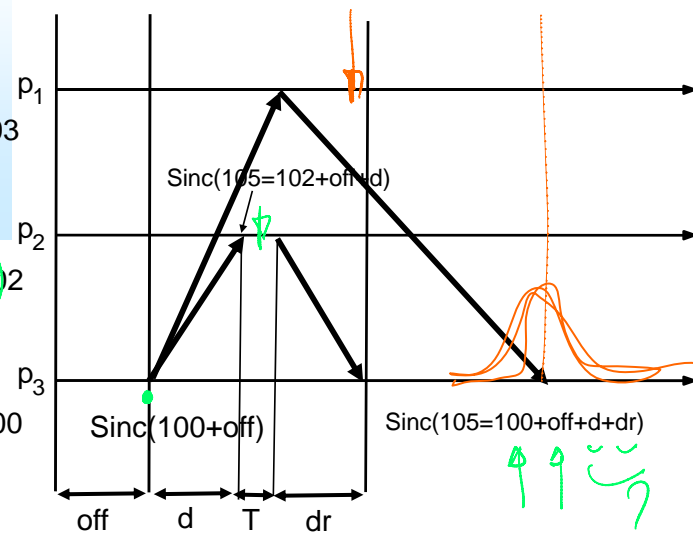
Dove:


- ♣ 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


- $T=103$
- $T=102$
- $T=100$






Sincronizzazione fra Nodi, MODO A

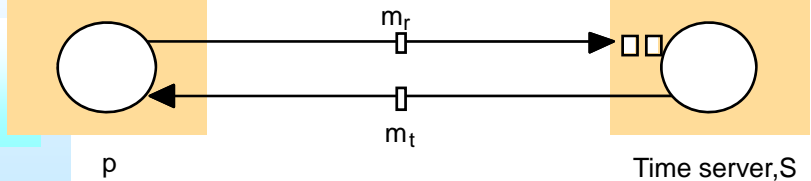
- Da un nodo Server si sincronizza gli altri nodi con dei messaggi
- I messaggi arrivano ai nodi con **tempi non predicibili**: $d(i)$, dove i è il nodo
- Le risposte arrivano al Server dai nodi con **tempi non predicibili**: $dr(i)$, dove i è il nodo
- Tramite iterazioni varie è possibile stimare per ogni nodo
 - ♣ valori medi, minimi e massimi per: $d(i)$ e $dr(i)$, certamente per $d(i)+T+dr(i)$. Dove T è il tempo di risposta del nodo, non noto.
 - ♣ Alla fine è possibile considerare anche:
 - ritardo(i) = $\lceil \text{mean}(d(i)+T+dr(i)) \rceil / 2$
 - ♣ Come valore medio di ritardo, che può essere comunicato al momento della sincronizzazione dal Server:
 - Sinc(i ,ritardo(i))
- Quando si comunica Sinc() non è detto che si abbia un tempo effettivo simile a quello medio, da questo nascono degli errori.




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 9



MODO B, Cristian (1989)



- È il client P che chiede al Server S il tempo assoluto T
- P misura il tempo che passa da quando manda m_r a quando riceve m_t : detto *Tround* (tempo per fare il giro)
- P impone/setta il suo tempo assoluto pari a $(T+\text{Tround}/2)$
- Errore pari a $\pm(\text{Tround}/2 - T_{min})$
 - ♣ T_{min} è il minimo dei Tround possibili
 - ♣ Pertanto in questo modo è possibile capire se l'errore è accettabile per il tipo di applicazioni in cui si vuole utilizzare il metodo



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 10

Sincronizzazione

roundtrip-time = $T \rightarrow$ ritardo di rete = $T/2$

Se $|d_2 - d_1| < \delta$:
fine

Se $|d_2 - d_1| > \delta$:
nuova iterazione

Se $T_3 + d_1 > T_3 \rightarrow$ Nuovo tempo di Peer2 = $T_3 + d_1$

- 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_i).
- δ si rilassa per raggiungere la convergenza.
- Necessaria periodica ri-sincronizzazione a causa della deriva dei clock fisici.

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 11

Risultati: Sincronizzazione


0.23ms

- 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**

Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010 12

Altri Metodi

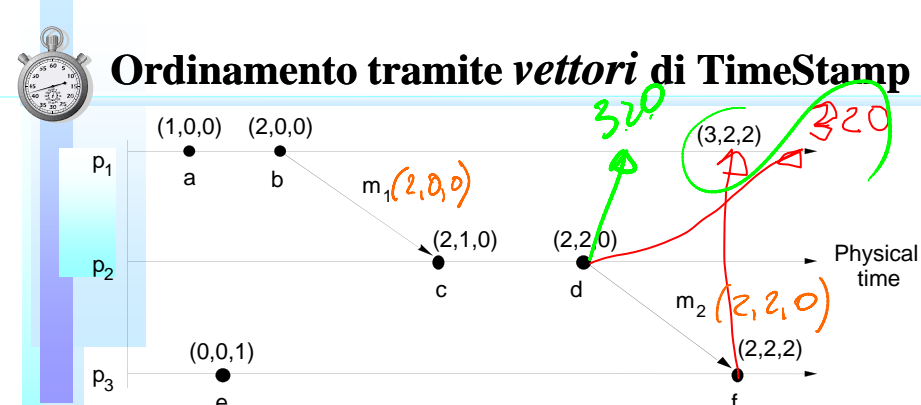
- Modo B non porta a risultati soddisfacenti quando i tempi di trasmissione sulla rete (Tround) sono elevati, valido per Intranet
- In reti geografiche sono necessarie altre soluzioni
 - ✦ le soluzioni iterative permettono di ridurre l'errore tramite approssimazioni successive con incrocio di messaggi fra i vari nodi della rete
- In generale la precisione assoluta non e' possibile
- L'ordinamento di eventi e' pero' 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




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

13

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 collegati ed etichettati
- I comandi sono ordinabili (2,2,0) e' successivo a (2,1,0)
- E' possibile in questo modo creare degli ordinamenti




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

14




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




Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

15



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 incerto 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 trova in posizioni diverse ☺
 - ♣ se si parla di dati si dovrebbe garantire l'UNDO di ogni comando, in certi casi fisici non e' possibile



Sistemi Distribuiti, Univ. Firenze, Paolo Nesi 2009-2010

16