5. Timpul

5. Timpul

5.1. Măsurarea timpului fizic

5.2. Compensarea *drift*-ului

5.3. Sincronizarea ceasurilor în sisteme distribuite

5.4. Ceasuri logice

5.1. Măsurarea timpului fizic

Oscilator ➔ pulsuri perfect sincronizare
drift ➔ 1 sec./11,6 zile
Soluția ➔ sincronizarea prin mesaje
mesaje ➔ întârzieri impredictibile
UTC = Universal Time Coordination

5.2. Compensarea drift-ului

Probleme implicate de drift-ul ceasurilor:

- declanșarea evenimentelor la momente de timp specificate
  - săritură înainte
  - săritură înapoi
- întârzieri specificate
- actualizarea copiilor ➔ ***compensarea continuă*** a actualizărilor ceasului

Notații:

- S – timpul dat de aplicații (prin citirea ceasurilor software)
- H – timpul dat de hardware
- δ – factor de compensare

Formula compensării drift-ului:

$S(t)=H(t) + \delta(t)$

$\delta(t)=a \cdot H(t)+b$

a=? ; b=?

T_s - valoarea timpului ceasului software când H=h,

T_r - timpul fizic real

Pentru ca S să dea timpul fizic real după *n* tic-uri:

$T_s=(1+a)\cdot h + b$

$T_r + n=(1+a)(h+n)+b$

$a=(T_r - T_s)/n$

$b=T_s - (1+a)\cdot h= [T_s(n+h)-(T_r+n)\cdot h]/n$

5.3. Sincronizarea ceasurilor în sisteme distribuite

precizie?

Cristian's method (clock synchronization)

S – server de timp

C – client

S este sincronizat cu UTC

C trimite un request pentru a afla timpul

S răspunde cu timpul curent al ceasului t_s.

T_{tr} – durata transmisiei:

$$T_{tr} = T_{min} + T_x$$

T_{min} – durata minimă a transmisiei (măsurată sau estimată ?)

T_x – durata necunoscută (>0)

t_c – momentul când clientul trimite request-ul

t_r – timpul când C primește mesajul de la S

➔ $T_{change} = t_r - t_c$

Clientul setează timpul său curent t:

$t = t_s + T_{change}/2$

Precizia (acurateţea)

T_{CS} notează durata transmisiei C➔S:

$T_{CS} = T_{min} + T_x$,

T_{SC} notează durata transmisiei S➔C:

$T_{SC} = T_{min} + T_y$

$T_{change} = T_{CS} + T_{SC}$

t’ – momentul de timp când C primeşte mesajul de la S :

$t' = t_s + T_{min} + T_y$

C cunoaşte T_{min}, dar nu poate evalua T_y.

C primeşte mesajul de la s S (time t’) în intervalul:

$[t_s + T_{min}, t_s + T_{change} - T_{min}]$

Lungimea intervalului = acurateţea (precizia) sincronizării:

$(T_{change} - 2 \cdot T_{min}) = T_x + T_y$

Acurateţea timpului de trimitere a mesajului: $(T_{change}/2 - T_{min})$.

T_{change} mai mic ➔ precizie mai mare

***Dezavantaj*:** S poate să se defecteze

Soluţie: mai multe servere de timp

5.4. Ceasuri logice

Lamport time stamp ➔ ceasuri logice

Timpul logic şi ceasuri logice

Dacă două evenimente au loc în acelaşi interval de timp, ordinea lor este dată de ordinea observării.

Un proces trimite un mesaj: evenimentul *send(m)* ***este înaintea*** evenimentului *receive(m)*.

Ordine cauzală:

$$x \xrightarrow{P} y$$

x, y evenimente, P este un proces în cadrul căruia au loc evenimente.

Relaţii:

1) dacă există un proces P: $x \xrightarrow{P} y$ implică $x \rightarrow y$

2) Pentru fiecare mesaj *m* sent trimis de la un proces la altul

$$send(m) \rightarrow receive(m)$$

3) Dacă x,y şi z sunt evenimente astfel încât

$x \rightarrow y$ şi $y \rightarrow z$ $\rightarrow$ $x \rightarrow z$ (tranzitivitate)

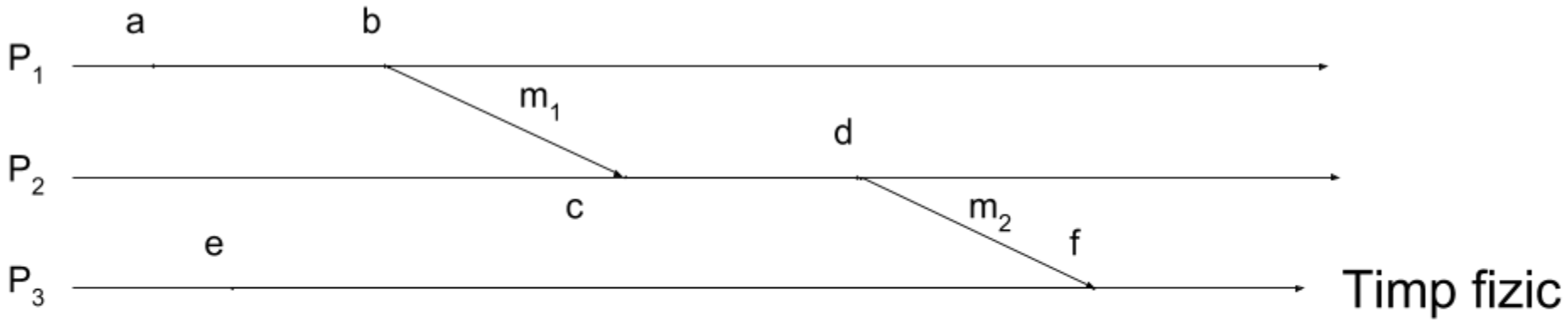


Diagrama spațiu-timp

$a \to b, c \to d, b \to c, d \to f, e \to f$ ➔ a → f .

dar NU $a \to e$ sau $e \to a$

➔ a și e sunt concurente ***a||e.***

Relația $\longrightarrow$ cauzalitate potențială.

Conclusion: Două evenimente pot fi legate prin relaţia ***happened before*** chiar dacă nu există o relaţie directă.

Ceas logic

Fiecare proces P are propriul său ceas logic C_p. ➔ marcaj de timp

$C_p(a)$*:* timpul când un eveniment *a* are loc în P.

Implementarea relaţiei $\longrightarrow$ ***:***

Procesele își actualizează ceasurile lor logice și își transmit valorile:

1) C_p este incrementat înainte de fiecare eveniment care are loc în P: $C_p = C_p + 1$

2) Când P trimite un mesaj m, îl încarcă cu valoarea $t = C_p$.

3) Cțnd un proces Q primește un mesaj (m,t), Q calculează valoarea ceasului digital:

$C_q = max(C_q,t)$

și apoi actualizează marcajul de timp al evenimentelor.

Dacă $\mathrm{a} \rightarrow \mathrm{b}$ implică $\mathrm{C(a)} < \mathrm{C(b)}$.

Reciproca NU ESTE ADEVĂRATĂ: $C(a) < C(b)$ NU ➔ $a \rightarrow b$.

Ceasuri logice ordonate total

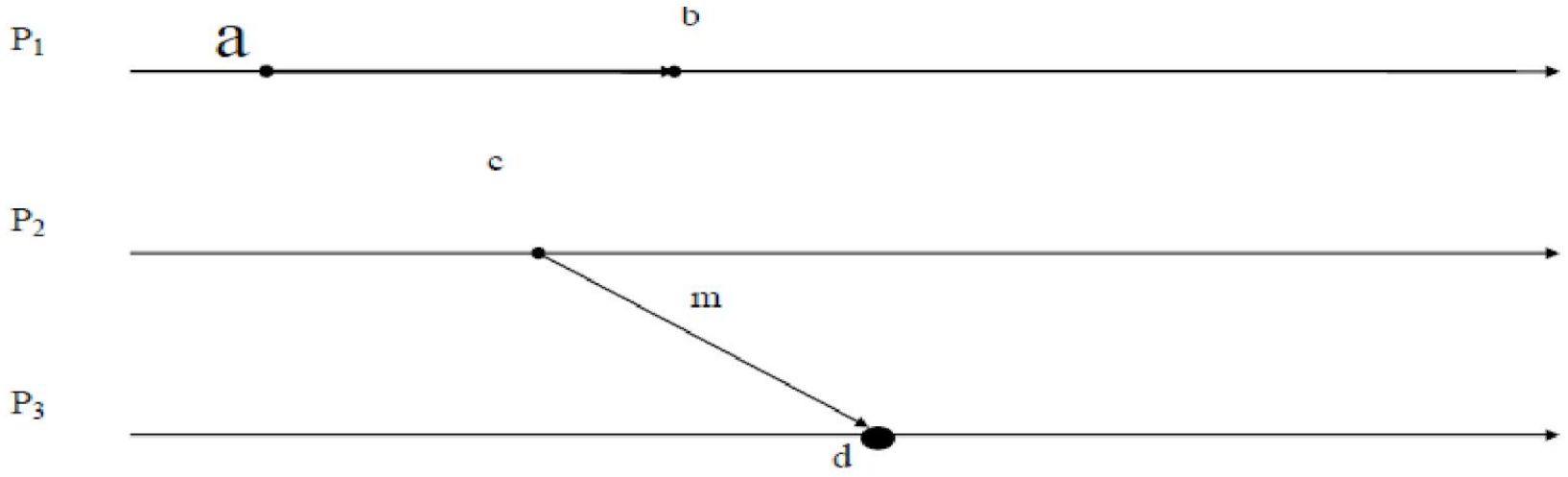
Folosesc identificatorii proceselor ca ordine.

Evenimentul *a* are loc în P_a cu marcajul de timp T_a.

Evenimentul *b* are loc în P_b cu marcajul de timp T_b.

Timp logic global: (T_a,P_a) și (T_b,P_b).

Este definit $(T_a,P_a)<(T_b,P_b)$ dacă și numai dacă $T_a<T_b$ or $T_a=T_b$ și $P_a<P_b$.



Ordinea totală

→ $(T_a,P_1)<(T_d,P_3)$ fără $a \rightarrow d$.

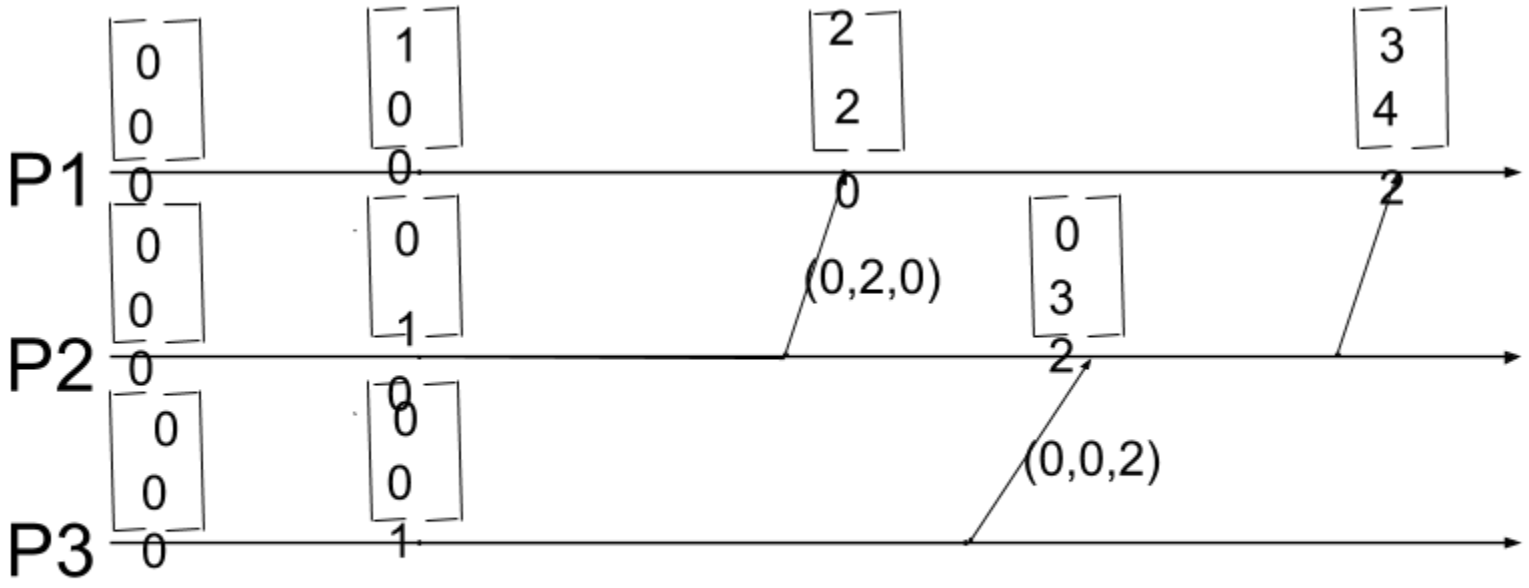
Ceasuri logice vectoriale

N procese ➔ un vector cu ceasurile lor logice

Fiecare proces are o copie a ceasurilor

Când trimite un mesaj, fiecare proces include şi ceasul său vectorial.

Când primeşte un mesaj, fiecare proces îşi actualizează ceasul său vectorial.



Ceasuri logice vectoriale

$VC(e_i)$ e_i.

Update rules:

$VC(e_i)[i] = VC[i] + 1$, Dacă e_i este un eveniment intern sau *send*.

$VC(e_i) = \max\{VC, TS(m)\}$ și $VC(e_i)[i] = VC[i] + 1$, dacă e_i este *receive*.

Pentru $\forall j \neq i$ componenta j a ceasului vectorial din procesul p_i are interpretarea:

$VC(e_i)[j]$ - numărul de evenimente în p_j care sunt precedente cauzal evenimentului e_i din p_i.

$VC(e_i)[i]$ - numărul de evenimente din p_i care au fost executate până la e_i inclusiv acesta (adică poziția în ordine canonică a evenimentului e_i din procesul p_i.

Given orders:

VC=VC' dacă şi numai dacă *VC[i]=VC'[i]* pentru orice *i*

VC≤VC' dacă şi numai dacă *VC[i] ≤VC'[i]* pentru orice *i*

VC<VC' dacă şi numai dacă *VC[i] ≤VC'[i]* pentru orice *i şi VC≠VC'*.

Ordinea cauzală a evenimentelor:

e→e' ➔ *VC(e)<VC(e')*

VC(e)<VC(e') ➔ *e→e'*

concurenţă *e||e' dacă nu VC(e)≤VC(e')* şi *VC(e')≤VC(e).*
Problema toleranţei sincronizării activitaţilor în diferite calculatoare

Soluţia: Dacă două activități executate în calculatoare diferite nu pot fi sincronizate cu precizie acceptabilă, ele trebuie implementate în același calculator.

*

***

****

*** Sfârşit ***

****

***

*