Blocul III. Sincronizare, coordonare și acord

Sisteme și servicii distribuite

Inginerie Telematică de gradul III

Cuprins

Introducere

- SID caracterizat prin utilizarea algoritmilor distribuiți:

  – Informații relevante distribuite între mai multe mașini

  – Luarea deciziilor numai pe baza informațiilor locale

  – Este necesar să se evite punctele de defecțiune

  – Nu există un ceas comun sau absolut

Introducere

De ce este timpul important într-un SID?

- Aplicații în care este necesară măsurarea precisă a timpului:
  - „Timp absolut”: în raport cu o referință externă. necesara sincronizare cu referința respectivă.
- Sincronizare cel puțin relativă, pentru a evita anumite probleme tipic sistemelor distribuite:
  - Evitați problemele de inconsecvență
  - Verificați autenticitatea cererilor către servere (Kerberos)
  - Ștergeți duplicatele etc.
- Cauzalitate: cunoașterea ordinii în care se petrec evenimentele în a S.I.D.
  - indiferent dacă este relativ la o referință de timp comună
- Vom studia două tipuri de mecanisme de sincronizare: ceasurile fizice și cele logice.

Cuprins

Model

– A - a colecție de n procese pi (i=1,2, ..n)

– if º starea lui pi : valoarea tuturor variabilelor unui proces.
Se schimbă, se transformă atunci când procesul este executat.

– Acțiune: fiecare proces execută o serie de acțiuni, care vor fi să trimită, să primească sau să efectueze o operațiune internă (care își schimbă starea)

– e º eveniment, eveniment al unei acțiuni pe care o realizează procesul in timpul alergarii

– și ® ¡ e' º Evenimentul e are loc înaintea evenimentului e' în interiorul pi

– Evenimentele din cadrul unui proces pot fi comandate total unic

– hi =<ei 0, ei 1, ei[noua],... > istoria procesului pi

Sincronizarea ceasului. Ceasuri fizice

Cum se măsoară de fapt timpul fizic?

- Einstein. Timpul nu este absolut: nu există un ceas global absolut în univers la care să ne putem întoarce
- Măsurare astronomică:
  - Eveniment de tranzit solar: când soarele atinge punctul cel mai înalt
  - Zi solară: interval dintre două tranzite solare
  - Al doilea solar: 1 zi solară/(24x3600)
  - Problemă: perioada de translație și rotație a Pământului în jurul Soarelui nu este constantă. Se stabilește media secundă solară
- Ceas atomic (în 1948)
  - Al doilea: timpul necesar atomului de cesiu 133 pentru a face ~9,2 miliarde de tranziții. Ea corespunde aproximativ cu secunda solară medie
  - 50 de laboratoare cu ceasuri Cesium 133 în lume.
  - Ora atomică internațională: media ceasurilor atomice
- Ora universală coordonată (UTC)
  - Introduceți o secundă bisecătoare pentru a corecta decalajul de timp astronomice și atomice
  - Stațiile terestre și sateliții îl transmit în mod regulat.
  - GPS oferă această oră cu o precizie de 1 microsecundă

Sincronizarea ceasului. Ceasuri fizice

- Ceasul fizic al unui computer:
  - Cristal de cuarț: oscilează cu o perioadă constantă (aproximativ)
    - Contor: scade cu o unitate cu fiecare oscilatie
    - La atingerea 0, are loc o întrerupere a ceasului și contorul ceasului este reîncărcat. jurnalul de întreținere.
    - H(t): valoarea ceasului de cuarț
    - C(t)=aH(t)+b Ceas software: măsoară aproximativ timpul fizic t
    - Evenimentele succesive au momente diferite de timp dacă apar într-un interval de timp mai mic decât rezoluția ceasului.
  - Distorsiunea ceasului: diferență instantanee între valorile timpului oricare două mașini
  - Clock drift: variația perioadei oscilatorului de cuarț (local)
  - Rate de deplasare a ceasului: schimbarea dintre o deplasare a ceasului și un ceas perfect. În jurul orei 10-6 – Sincronizare internă: diferențele dintre ceasurile C i 11,6 zile

unui sistem P este limitată, indiferent de relația acestora cu timpul real

  - Sincronizare externă: ceasurile nu trebuie să se abate de la timpul real mai mult de o anumita amploare

Cuprins

Ceasuri fizice. Sincronizare.

- Sincronizarea ceasurilor fizice. Având în vedere o sursă de timp UTC externă, S(t) și un interval de timp real I:
  - Sincronizare externă

$$|S(t) - C_i(t)| < \alpha \quad i = 1,2,..N \text{ } \forall t \in I$$

  - Ceasurile sunt precise în D
  - Sincronizare internă

$$|C_i(t) - C_j(t)| < \alpha \quad i, j = 1,2,..N \text{ și } t \in I$$

  - Ceasurile sunt de acord cu D

Sincronizare internă: Sistem sincron distribuit

- Sistem sincron distribuit => este mărginit următorul:
  - deriva ceasului
  - Întârzierea maximă de transmitere a unui mesaj: Tmax
  - Timpul de execuție al fiecărui pas al unui proces
- Un mesaj trimite ora locală t altuia într-un mesaj m
- Întârziere minimă de transmisie: Tmin
- Putem limita distorsiunea ceasului |Ck - Ci |?
- Cea mai bună estimare pentru două procese:

$$C_n(t) = t + \frac{T_{max} + T_{min}}{\text{Două}}$$

- Cu distorsiune maximă u/2 cu

$$u = T_{max} - T_{min}$$

- Pentru N procese distorsiuni maxime

$$\left(1 - \frac{\text{unu}}{N}\right)u$$

Sincronizare externă în sisteme asincrone

- Algoritmul lui Cristian
  - Avem un proces conectat la o sursă UTC externă (Time Server, ST)
  - Obiectiv: sincronizarea restului proceselor cu sursa externă prin mesaje din procesul conectat

1. Un proces solicită ora de la ST printr-un mesaj mr
2. Primiți răspunsul într-un mesaj mt care spune real time is tserver
3. Calculează timpul dus-întors TRTT și își setează ceasul C la tserver + TRTT /2

$$CT = \text{server} + \frac{T_{RTT}}{\text{Două}} \pm \left(\frac{Tç_{RTT}}{\text{Două}} - T_{min} \right)$$

și sau

Precizie sau distorsiune maximă

Ceasuri fizice. NTP.

- Network Time Protocol: definește o arhitectură pentru un serviciu de timp și un protocol pentru distribuirea informațiilor de timp pe Internet.
  - Oferă un serviciu care permite clienților să se sincronizeze cu acuratețe cu sursa UTC prin Internet (întârzieri nelimitate). • Tehnici statistice pentru filtrarea marcajelor de timp și

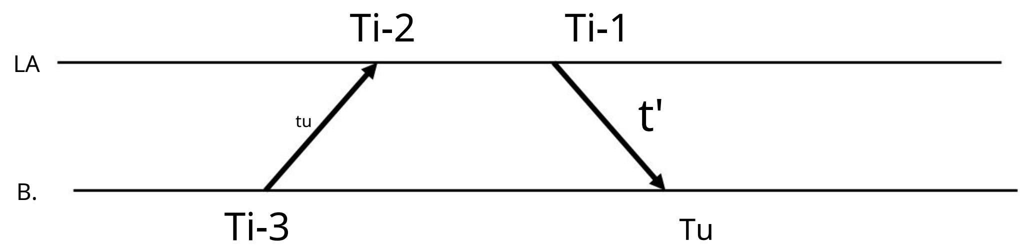
    discriminarea calitatii surselor.
  - Oferă un serviciu de încredere care poate depăși pierderile de conectivitate.
    - Servere și rute redundante.
  - Permite clienților să reconfigureze și să echilibreze frecvent problemele legate de deviația ceasului.
  - Asigurați protecție împotriva interferențelor în serviciul de timp cu origine rău intenționată.
    - Autentificare server.

NTP: operațiune

- Subrețea de sincronizare: servere primare și secundare
  - Fiecare nivel (numit strat) introduce o mică eroare.
  - Mașini utilizator la ultimul nivel.
  - Reconfigurabil dacă un server se defectează.
- Sincronizare între servere NTP prin trei moduri:
  - Multicast: utilizare în LAN de mare viteză. Precizie insuficientă în multe cazuri.
  - Remote Procedure Call (RPC): similar cu alg. din Creştin. Precizie mai bună decât multicast.
  - Simetric: Între servere la cel mai înalt nivel.
- Schimb de mesaje UDP:
  - În RPC și simetric: client și server schimbă perechi de postări
  - Fiecare mesaj include ora: • ora locală a orei în care ultimul mesaj dintre ambele procese a fost primit (Ti-2) și trimis (Ti-3)
    • și ora locală de transmitere a mesajului curent (Ti-1)

Ceasuri fizice. Sincronizare.

- Calculul offset-ului (distorsiunea ceasului) în NTP



offset s̊āū CC LA B. æ t' - t ç ö ÷ + sau i
și Două sau

estimarea offset s̊au i æ T T T T + eu 2 eu 3 i eu 1 ö ÷
și Două sau

întârziere o d = i Ti 2 Ti 3 + Ti Ti 1

((i sau - d i / Două) £ + ooo i d i / Două)

Filtrarea datelor este aplicată perechilor <di,oi> , care oferă o măsură statistică a calității estimării numită dispersie de filtru.

Cuprins

Ceasuri logice. Sincronizare.

- Ceas logic: instrument care permite definirea ordinii evenimentelor fără a măsura timpul fizic în care acestea apar.
- Nomenclatură
  - A - a colecție de n procese pi (i=1,2, ..n)
  - dacă starea pi
  - e º eveniment, eveniment al unei acțiuni pe care o realizează procesul in timpul alergarii
  - și ® ¡ e' º Evenimentul e are loc înaintea evenimentului e' în interiorul pi
  - hi =<ei 0, ei 1, ei next ,... > istoria procesului pi

Ceasuri logice: relații cauzale

- Un proces ð toate evenimentele sunt ordonate

Și într-un sistem distribuit?

- Noi stim aia:
  - Dacă două evenimente au loc în același proces, atunci ele au loc în ordinea în care pi le observă.
  - Ori de câte ori un mesaj este trimis între două procese, evenimentul de trimitere are loc înainte de evenimentul de primire.

Ceasuri logice. ordonarea cauzală

- Relația sa întâmplat înainte de Lampport „®”:

  (1) Dacă $ pi / ® e' Þ e i ® și'

  e (2) " mesajul m, trimiteți(m) ® primi (m)

  (3) Dacă e, e', e'' / ee', e' ® e'' Þ e ® și''

  Când două evenimente nu sunt legate prin „®”, ele sunt concurente

- Trebuie să asociem fiecare eveniment cu o valoare de timp asupra căreia toate procesele sunt de acord, astfel încât:

  (1) Dacă $ pi /® ¡e' Þ C(e)<C(e')

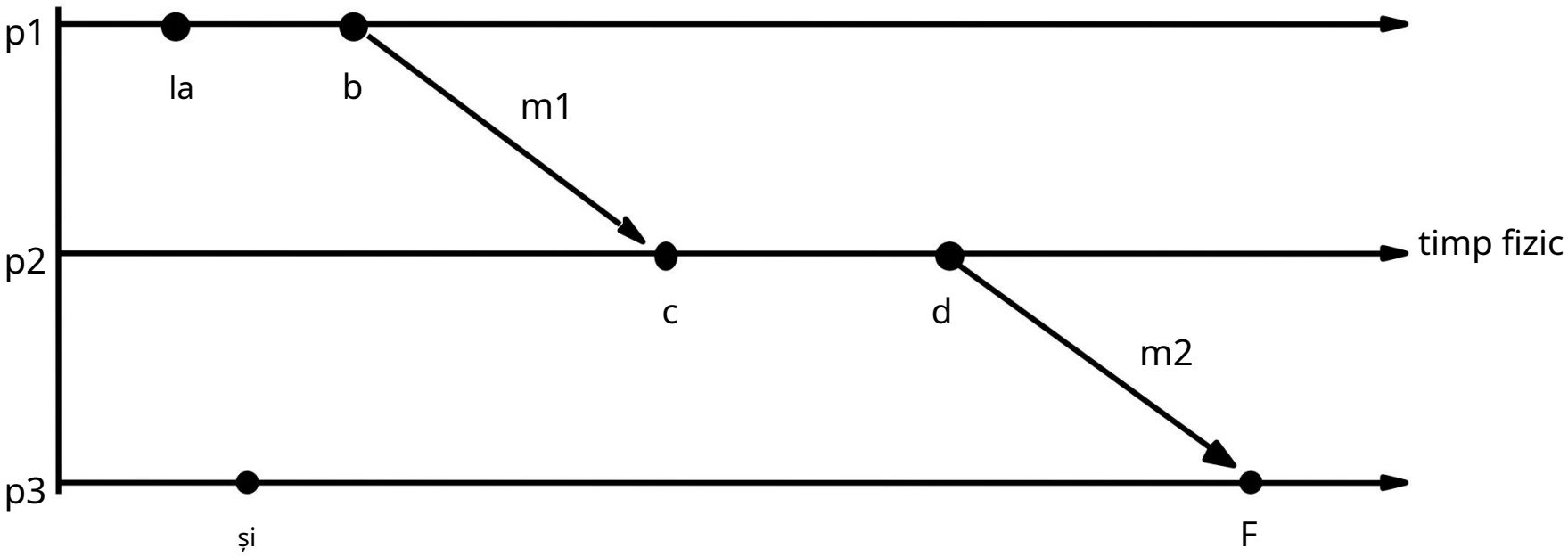
  e (2) " mesajul m, C(trimite(m)) ® C(primire(m))

  (3) Dacă e, e', e'' /®ee', e'®' Þ C(e) ® EC'')

  (4) C mereu în creștere

Ceasuri logice. Exemplu

Prezentați relațiile cauzale dintre toate perechile de evenimente din sistem.



Ceasuri logice. algoritmul lui Lampport

- Algoritmul lui Lamport: atribuirea numerică a ordonării cauzal
  - Ceas logic Lamport, C: contor software care crește monoton, ale cărui valori nu trebuie să aibă nicio relație specială cu niciun ceas fizic
  - Fiecare proces are propriul ceas logic
  - C(e) valoarea timpului asociat evenimentului

  e (a) Ci este crescută ÎNAINTE de fiecare eveniment, Ci

  =Ci +1 (b) Dacă pi trimite un mesaj, acesta include

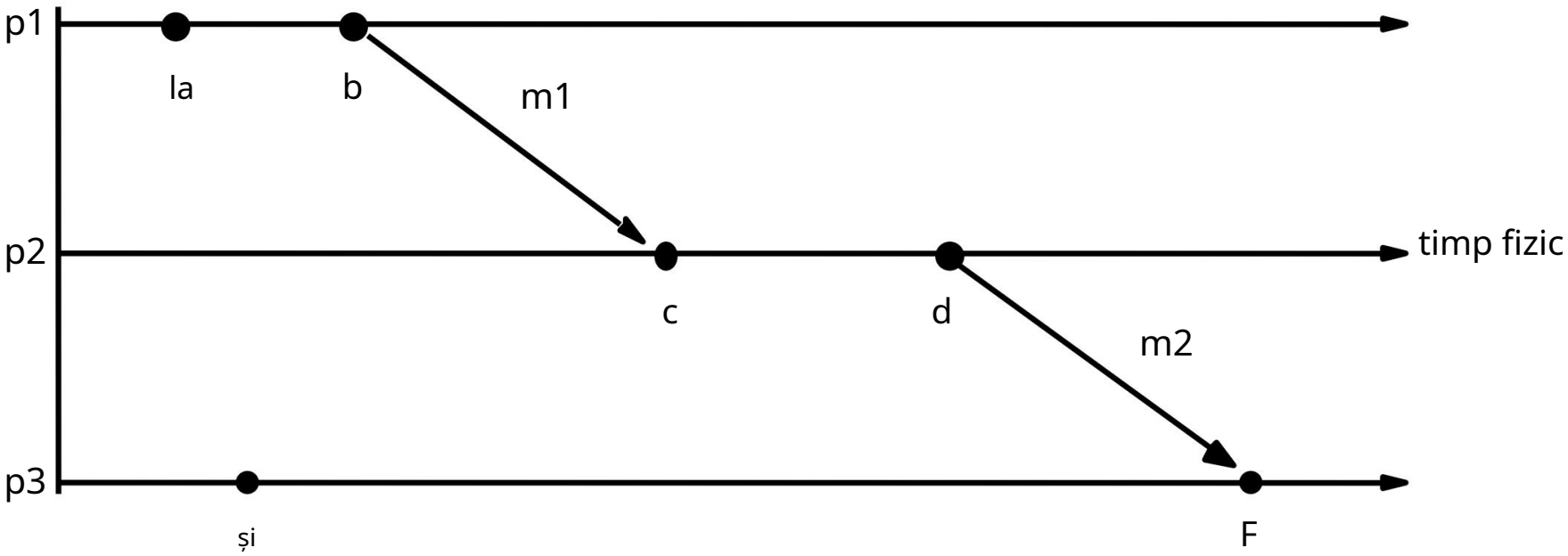
  valoarea t=Ci (c) Dacă pj primește un mesaj cu valoarea t, calculează noul Cj

=max(Cj ,t) și aplică (a) • Pentru a obține ordonarea totală a evenimentelor: o relație de prioritate între toate evenimentele tuturor proceselor –

Asociați numărul procesului în care se produce evenimentul la instantele de timp „repetate”. • " e, e' C(e)□ C(e')

Ceasuri logice. Exemplu

Aplicați algoritmul Lamport



Ceasuri logice. ceasuri vectoriale

- Problemă: cu Lampport, dacă C(e)<C(e') e ®e'? •

1989, Mattern and Fidge è ceasuri vectoriale

  – Un ceas vectorial într-un sistem de N procese este o matrice de N numere întregi (dimensiune=număr de procese)

  – Un ceas vectorial pentru fiecare pi

- Algoritm de ceas vectorial: (1) Inițial Vi [j]=0

  "i,j=1,2,..N (2) Înainte de fiecare

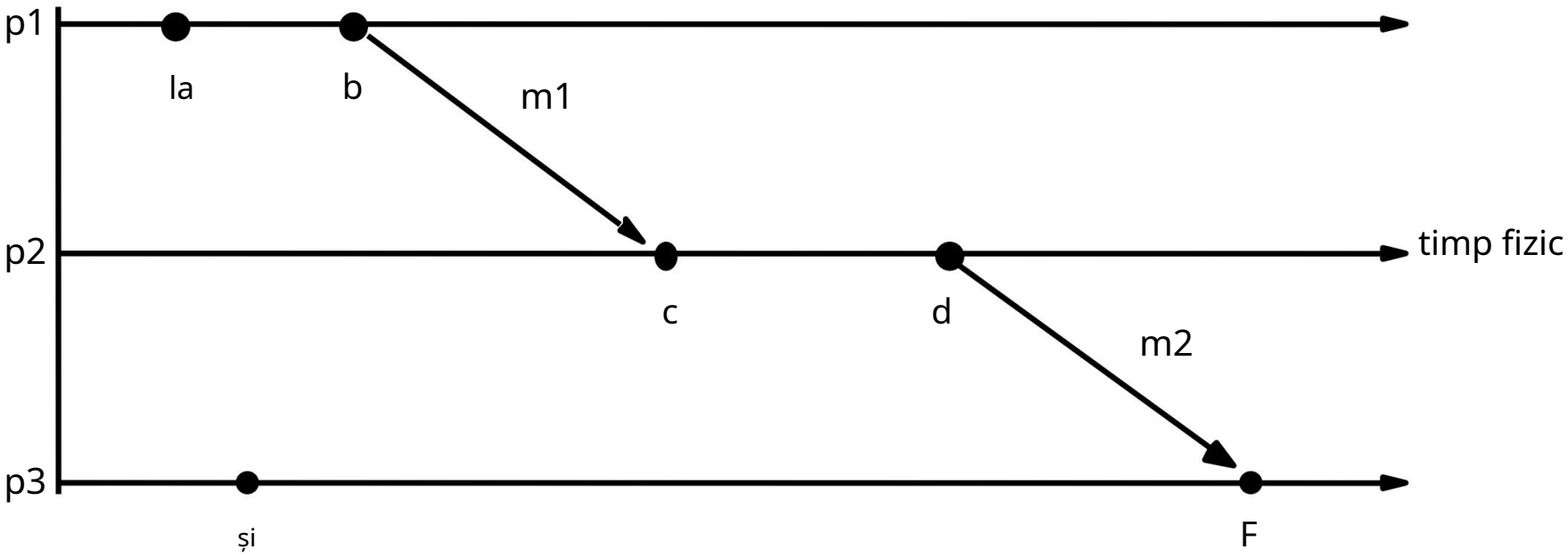
  eveniment din pi Vi [i]=Vi [i]+1 (3) pi include

  valoarea t=Vi în fiecare mesaj (4) Dacă pj

  primește un mesaj cu valoarea t, se calculează Vj [i]=max(Vj [i],Vi [i]) "i^{1}j

Ceasuri logice. Exemplu

Aplicați algoritmul Mattern și Fidge: ceasuri vectoriale



Ceasuri logice. Sincronizare.

- Comparații cu ceasuri vectoriale:
  - V=V' ð V[j]=V'[j] "j=1,2,..N – V£V'
    ð V[j] £ V'[j] "j=1,2,. .N
  - V<V' ð V£ V' ^ V□ V'
- Dacă V(e)/£ V(e') și V(e') ₤ V(e) ð e, e' concurente
- Dezavantajele ceasurilor vectoriale?
  –
  –

Cuprins

Starea globală a unui sistem distribuit

Cum să afli dacă o proprietate este adevărată într-un sistem distribuit?

- Soluție dificilă într-un sistem distribuit care este în funcțiune
- Astfel încât?
  - Obiecte nedorite: dacă nu mai există referințe la el în sistemul distribuit. Dar pot exista referințe într-un mesaj transmis (pe drum)
  - Detectare blocaj: Blocarea există atunci când unul sau mai multe procesele așteaptă un mesaj de la un proces și procesul așteaptă, de asemenea, un mesaj de la acele procese
  - Detectarea finalizării: problema este de a detecta că un algoritm distribuit s-a terminat. Exemplu: este posibil ca două procese să fi intrat în starea pasivă, dar este posibil să existe un mesaj de trezire pe drum

Starea globală a unui sistem distribuit

- Aceste probleme au o soluție dar necesită ca sistemul să fie observat într-un mod global.
- GÂND:

> Obțineți un stat global din statele locale care sunt înregistrate cu ore locale diferite

- Putem caracteriza execuţia unui proces prin intermediul acestuia istorie:

$$\text{istorie } ph(\ ) = =< \quad e^{0}_{i} \quad e^{1}_{i}, e^{2}_{i}, \quad , \ldots >$$

— O bucată de istorie

$$h_i^k =< e_i^0, e_i^1, e_i^2, \ldots, e_i^k >$$

Starea globală a unui sistem distribuit

- Fiecare proces poate înregistra evenimentele care au loc local și succesiunea stărilor prin care trece.

apare K înainte de • dacă starea procesului pi chiar

evenimentul K.

- Istoria globală H a lui P va fi: $H = h_0 \to h_1 \to h_2 \to \dots \to h_n$ • Starea globală S a lui P este S= (s1, s2, s3,..., sN)
  - Nu toate statele globale sunt consecvente. Trebuie să știm ce stări s-ar fi putut întâmpla în același timp
- O tăietură C a execuției unui sistem este a subset al istoriei sale globale, obținut din bucățile individuale ale istoriei procesului:

$$C = h_1^{c1} \to h_{Două}^{C2} \to \dots \to h_3^{C3} \to \dots \to c_n$$

Starea globală a unui sistem distribuit

- Starea dacă în starea globală S corespunzătoare celui cut C este acea stare a lui pi imediat după ce a procesat ultimul eveniment al tăierii
  - La ultimul eveniment: e_i^{Ci}
  - La setul de evenimente ei Ci ð chenar de tăiere
- O tăietură C este consecventă dacă pentru fiecare eveniment pe care îl conține conține și toate evenimentele care au avut loc înaintea sa:

$$„eÎC, f ® e Þ f ÎC$$

- O stare globală este consecventă dacă îi corespunde a tăietură consistentă

Cuprins

Algoritmul Chandy–Lamport

- Obiectiv: să înregistreze un set de stări de proces și canale astfel încât starea globală înregistrată să fie consecventă
  - Deși combinația de stări înregistrate este posibil să nu fi avut loc niciodată
- Se presupune că:
  - Nici canalele, nici procesele nu eșuează
  - Canalele sunt unidirecționale și FIFO
  - Există întotdeauna o cale între oricare două procese
  - Orice proces poate porni algoritmul
  - Procesele își pot continua execuția și pot trimite și primi mesaje în timp ce algoritmul rulează
- Nomenclatură
  - Canale de intrare, cele prin care alte procese trimit mesaje către pi
  - Canale de ieșire, prin care pi trimite mesaje către alte procese

Algoritmul Chandy–Lamport

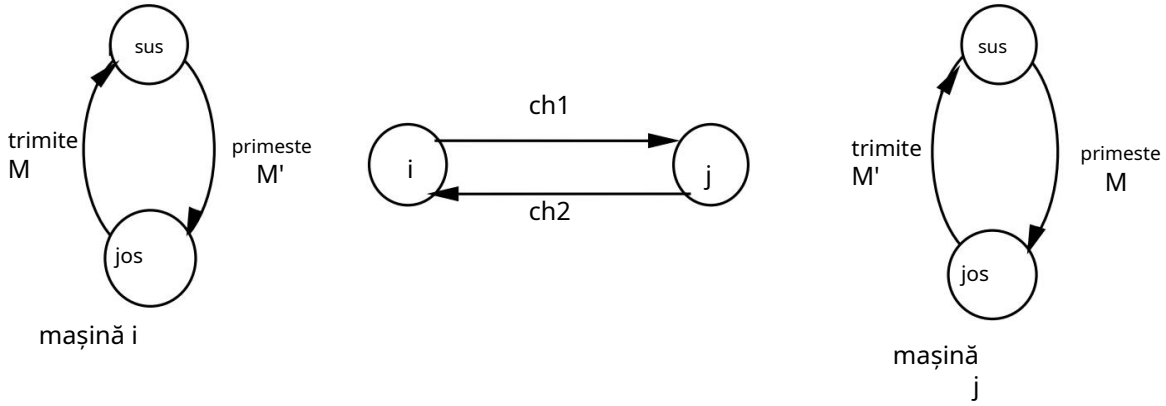
- Algoritm de instantaneu Chandy-Lamport
  - – Marker: mesaj gol util doar pentru algoritm
  - – Regula de trimitere a marcajelor: pi trimite un marcaj pe toate canalele sale de ieșire după ce își înregistrează starea și înainte de a trimite orice alte mesaje
  - – Regula de recepție a markerului: dacă pj primește un marker
    - a) dacă nu a fost înregistrat, se înregistrează și se salvează starea canalului prin care sosește markerul ca gol și se aplică regula de trimitere a markerului
    - b) dacă a fost deja înregistrat, salvează starea canalului prin care ajunge marcajul ca succesiune de mesaje primite de acel canal de la momentul în care a fost înregistrat până la primirea marcajului
  - - Adică, odată ce un proces a primit un marker pe un canal, acesta termină înregistrarea mesajelor primite pe canal
  - - Și odată ce ați primit un marker al tuturor canalelor, aveți a terminat de înregistrat statutul dvs

Algoritmul Chandy–Lamport

- Algoritmul poate fi pornit de orice proces
  - Se comportă ca și cum ar fi primit un marker și, prin urmare, urmează regula de recepție a markerului.
- Cele două reguli garantează că, dacă se primește un marker pe fiecare canal, fiecare proces își înregistrează starea și stările fiecărui canal de intrare.
- Algoritmul înregistrează o tăietură consistentă.
- Nu este necesar să opriți sistemul, acesta continuă să funcționeze normal în timp ce starea este înregistrată

Exemplu

- 2 aparate care trimit mesaje. • Desenați diagrama timpului
- Executați CL, pornind mașina i înainte de a trimite M prin ch1. Mașina j primește marcajul după ce a trimis M'



Secvența reală de evenimente și stări.

execuție originală

stare generală	i	ch1	j	ch2
S0	jos 0 jos			0
S1	sus M jos			0
S2	sus M sus			M'
S3	jos M sus			0

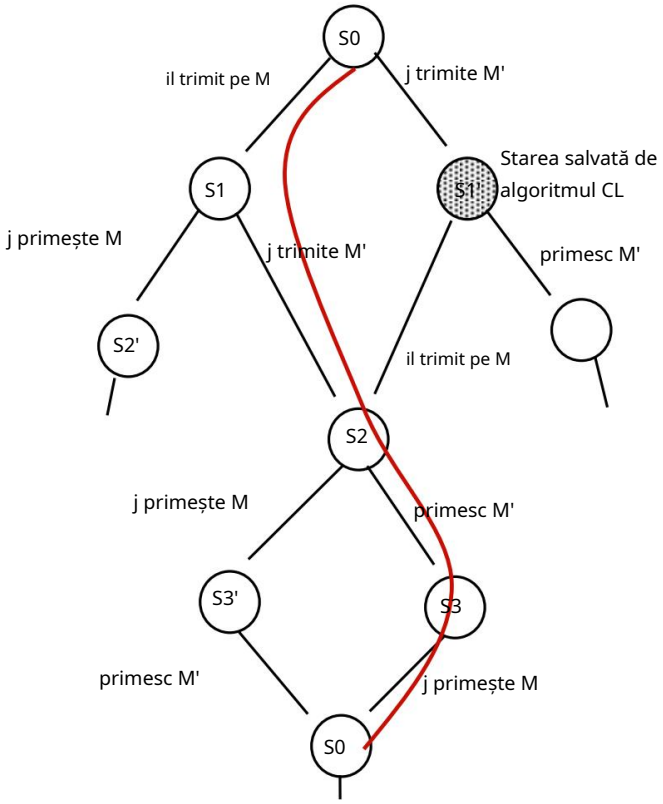
Exemplu

- Starea înregistrată de CL este: jos
  0 sus

M' • Sistemul nu a trecut niciodată
  prin această stare!

- Starea observată este fezabilă și
  poate fi atinsă din configurația
  inițială.
- Starea finală este întotdeauna
  atins din cele observate

- Pentru ce se utilizează CL?
  - Evaluează predicate stabile.
  - Restabiliți aplicația în caz de defecțiune
  - Sistem de audit



Cuprins

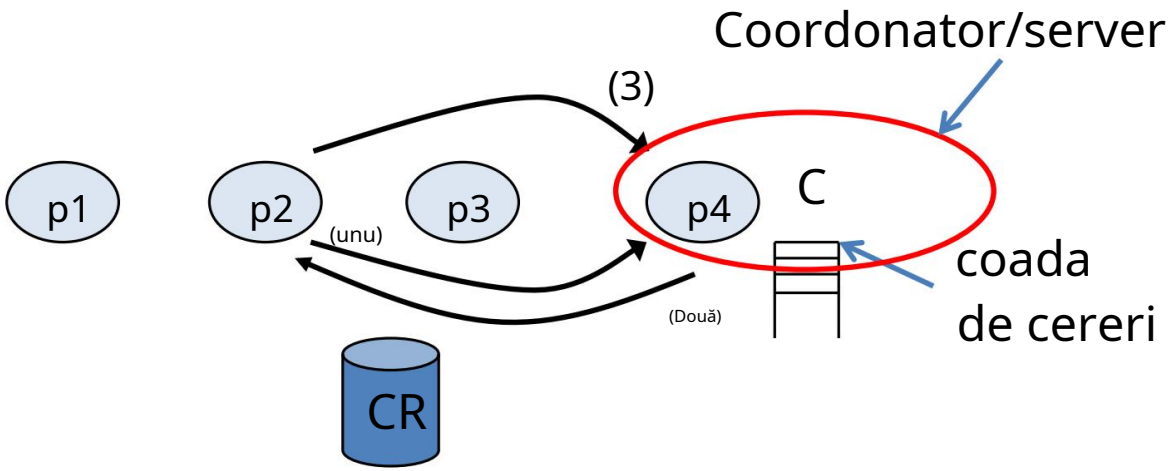
Tehnici de excludere reciprocă

- Dacă o colecție de procese partajează o resursă excluderea reciprocă este necesară pentru a preveni interferențele și pentru a asigura coerența datelor
  - Probleme regiuni critice pe sistemul de operare
    - Sistemul de operare controlează toate procesele și folosește semafoare, monitoare sau instrumente similare.
  - În SID avem nevoie de: soluție bazată doar pe pasul de postări
- Algoritmi distribuiți de excludere reciprocă:
  - Centralizat: bazat pe utilizarea unui server sau a unui coordonator
  - Distribuit: toate procesele au același rol, nu există coordonator

Cuprins

Tehnici de excludere reciprocă

- Algoritm centralizat
  - Coordonatorul decide cine are permisiunea de a folosi resursa
    - Este ales într-un fel (algoritmi de alegere)
  - Solicitare către coordonator de a intra în regiunea critică (1)
    - Răspuns pozitiv și introduceți (2).
    - Niciun răspuns sau răspuns negativ și este blocat. Coordonatorul lipeste petiția.
  - Procesul trimite un alt mesaj coordonatorului când iese (3)



Tehnici de excludere reciprocă

- Excluderea reciprocă este garantată deoarece coordonatorul permite doar un proces este în fiecare RC
- Avantaje
  - E corect: cererile sunt puse la coadă în ordinea sosirii
  - Niciun proces nu așteaptă la infinit
  - Ușor de implementat
  - Necesită câteva mesaje: 2 cereri + 1 eliberare
- Limitări:
  - Un punct de defecțiune în sistem
  - Gâtul de sticlă
  - Posibilități multiple de eșec: dacă procesul se blochează după o solicitare și nu primește răspuns, dacă nu părăsește regiunea critică etc.

Cuprins

Tehnici de excludere reciprocă

- Algoritm distribuit (Ricart-Agrawala): fără server/ coordonator
  - Ordinea totală necesară a evenimentelor (algoritmul Lampport, pt exemplu).
  - Dacă pi vrea să acceseze RC:
    1. Mesaj de compilare: numele RC, numărul procesului, ștampila de timp (Lamport)
    2. Trimiteți mesaj către toate procesele
    3. Dacă pj primește un mesaj:
       1. Dacă pj nu este în RC și nu dorește să acceseze, trimiteți un mesaj OK expeditorului
       2. Dacă pj este în RC , nu răspunde, cererea în coadă
       3. Dacă pj dorește să acceseze comparați marcajele de timp: primite vs trimise de pj anterior, cele mai mici câștiguri. Trimiteți OK dacă pierdeți sau puneți în coadă dacă câștigați
       4. Când pi are toate permisiunile (OK), accesați RC
       5. Când pi iese din RC, trimite un mesaj OK tuturor proceselor din coadă

Tehnici de excludere reciprocă

- Excluderea reciprocă este garantată deoarece:
  - Nu există foamete sau blocaj
- Limitări:
  - Câte puncte de eroare?
  - Câte mesaje?
  - Primitiva de comunicare de grup necesară: fie utilizați multicast/ difuzare (câte mesaje?) fie trimiteți un mesaj fiecărui proces
    - Necesită menținerea apartenenței la grup: cunoașterea tuturor proceselor care doresc să acceseze RC.
  - Există câteva îmbunătățiri: algoritmul Maekawa.

Cuprins

Algoritmi de alegere

- Obiectiv: alegeți un proces pentru a îndeplini o anumită funcție
  - Procesele multiple pot iniția o alegere concomitent
  - Nu toate procesele trebuie să participe la alegeri
  - În general, se alege procesul cu cel mai mare „identificator”:
    poate fi orice valoare atâta timp cât este unică și totală organizat
  - Procesul ales trebuie să fie unic.
  - Performanța unui algoritm de alegere este de obicei măsurată în utilizarea lățimii de bandă (numărul total de mesaje trimise) sau timpul de finalizare (timpul de transmitere a tuturor mesajelor între începutul și sfârșitul algoritmului).

Algoritmul Bully

- Algoritmul Bully

  – Funcționează chiar dacă procesele eșuează în timpul unei alegeri.

  – Se presupune că expedierea este fiabilă și sistemul este sincron, deoarece folosește temporizatoare (TO) pentru a detecta defecțiunile și că identificatorul (ID) al tuturor proceselor este cunoscut

  – algoritm:

    – pi detectează că coordonatorul nu răspunde și inițiază alegerile

    (1) pi trimite mesaje electorale tuturor proceselor cu ID mai mare

    (2) când un proces primește o alegere, returnează un mesaj OK și începe o nouă alegere (revine la pasul (1))

    (3) Dacă nimeni nu răspunde în T secunde, pi este noul coordonator și trimite mesajul coordonatorului tuturor proceselor

    (4) Dacă a primit un OK , așteaptă încă T' secunde pentru a primi un mesaj coordonator sau începe alte alegeri

Algoritm de alegere a inelului

- Algoritm de alegere de apel
  - Început spontan de pj
  - Toate procesele sunt marcate ca neparticipante
  - pj trimite mesajul electoral cu ID-ul său la următorul proces și se modifică la participant.
  - pk primește mesajul și compară identificatorii, dacă k>j schimbă identificatorul de mesaj cu al tău, îl trimite la următorul proces și se modifică pentru participant
    - Dacă un proces participant primește un mesaj cu un ID mai mic decât al său, nu îl transmite (elimină rapid alegerile concurente).
  - Dacă procesul primește un mesaj electoral cu propriul ID, acesta este noul coordonator și îl comunică celorlalți printr-un mesaj de coordonator
  - Când un proces primește mesajul coordonatorului , notează ID-ul, este marcat ca neparticipant și transmis.

Cuprins

algoritmi de consens

- Consens = acord. Procesele trebuie să fie
  acord asupra unei valori, după ce unul dintre ei a propus-o

  – În situații practice trebuie să ajungem chiar la un consens
    la eșecuri

  – Variante multiple: generali bizantini, consistență interactivă, multicast complet
    ordonat

- Fiecare proces pi începe într-o stare indecisă și propune o
  valoare vi dintr-o mulțime D.

- Procesele comunică prin schimbul de valori.

- Procesele aplică o funcție la valorile colectate
  (majoritate, minim, maxim etc.) și stabiliți o valoare pentru
  variabila de decizie, trecând la starea hotărâtă.

Caracteristici de consens

- Cerințe ale unui algoritm de consens
  - Terminare: toate procesele active la sfârșit stabilesc variabilă de decizie.
  - Acord: variabila de decizie a tuturor proceselor corecte e la fel.
  - Integritate: Dacă toate procesele corecte au propus aceeași valoare, atunci orice proces activ în stare hotărâtă a ales aceeași valoare.
- Consensul este banal în sistemele fără defecțiuni (mecanism anterior)
- Consensul este imposibil de garantat într-un sistem asincron defect .
  - Teorema lui Fischer.
  - Sistem asincron: nu se poate distinge un proces cu defecte unul lent
  - Nu înseamnă că nu se poate ajunge niciodată la un consens. Permite să fie lovit cu o probabilitate mai mare de 0.

Consens asupra sistemelor sincrone

- Algoritm de consens într-un sistem sincron cu defecțiuni dar cu canale fiabile:
  - Ei pot eșua f procese ale N sistemului
  - La fiecare pas fiecare proces trimite tuturor celorlalți valorile pe care le cunoaște și pe care nu le-a transmis înainte
  - Durata fiecărui pas este limitată de utilizarea cronometre (sistem sincron).
  - Consens atins în pași f+1 (demonstrabil)
  - După pașii f+1 fiecare proces alege o valoare aplicând a funcția de consens

algoritmi de consens

- Exemplu

f=1 Þ consens în 2 pași

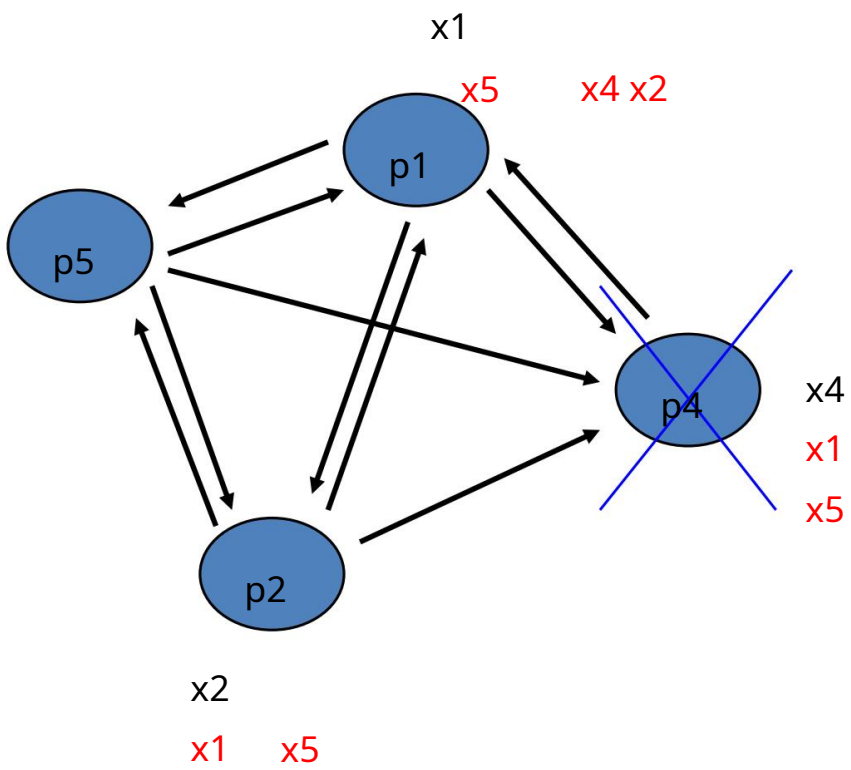
Pasul 1

x1=1

x2=1

x5=2

x4=0



algoritmi de consens

- Exemplu:

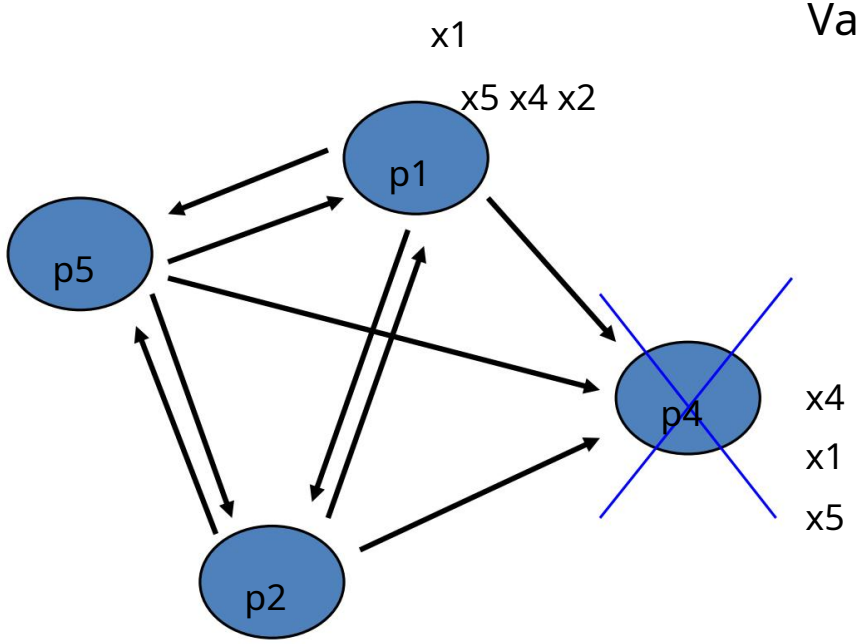
f=1 Þ consens în 2 pași

x1=1

x2=1

x5=2

x4=0



Pasul 2

Valoare=min()=0