Sisteme distribuite – Teorie
8. Excludere mutuala distribuita

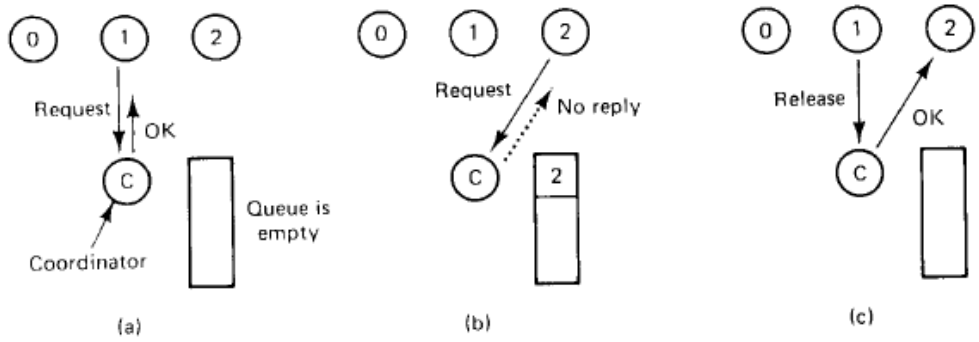
Problema

- Amintiti-va:
  - Cand un proces trebuie sa citeasca sau actualizeze o anume structura de date partajata, intra in sectiunea critica iar excluderea mutuala asigura ca nici un alt proces nu va utiliza structura de date partajata in acelasi timp
- Intr-un sistem cu un procesor:
  - Sectiunile critice sunt protejate utilizand semafoare, monitoare si contructori similari
- Cum poate fi implementate sectiunile critice si excluderea mutuala in sistemele distribuite?

Algoritmul centralizat

- Simuleaza ceea ce se intampla intr-un sistem mono-procesor.
- Un proces este ales ca fiind coordonator
  - Ex. cel care ruleaza pe masina cu adresa de retea cea mai mare
- Cand un proces doreste sa intre in sectiunea critica, expediaza un mesaj catre coordonator afirmand care este regiunea critica in care doreste sa intre si cerand permisiunea
- Daca nici un proces nu este in acel moment in sectiunea critica, coordonatorul expediaza un raspuns care acorda permisiunea
- Cand soseste raspunsul, procesul solicitant intra in sectiunea critica

Algoritmul centralizat - exemplu



- Procesul 1 solicita coordonatorului permisiunea de a intra in sectiunea critica si permisiunea este acordata
- Procesul 2 solicita sa intre in aceeasi sectiune critica
- Coordonatorul stie ca un proces diferit este deja in sectiunea crituca si nu acorda permisiunea
- Metoda exacta pentru a respinge permisiunea este dependenta de sistem
  - (b): coordonatorul doar refuza sa raspunda, astfel blocand procesul 2 care este in asteptarea raspunsului
  - Alternativ, poate trimite un raspuns cu "permisiune respinsa."
  - In ambele cazuri, pune in coada cererea de la 2
- Cand procesul 1 iese din sectiunea critica, trimite un mesaj coordonatorului eliberand accesul sau exclusiv - (c)
- Coordonatorul ia primul articol din coada de cereri refuzate si expediaza procesului corespunzator un mesaj de permisie
  - Daca procesul este blocat (adica acesta este primul mesaj catre acesta), se deblocheaza si inra in sectiunea critica.
  - Daca un mesaj explicit de refuz a fost deha expediat, procesul va rebui sa extraga mesajul nou din trafic sau sa se blocheze mai tarziu in asteptarea acestui mesaj
  - In ambele cazuri cand vede mesajul de acceptare, intra in sectiunea critica

Algoritmul centralizat – pro si contra

- Garanteaza:
  - Excluderea mutuala: coordonatorul lasa doar un proces la un momentar dat sa intre in sectiunea critica
  - Este corect pentru ca cererile sunt acordate in ordinea in care au fost receptionate
  - Nici un proces nu va astepta infinit (fara infometare).
- Schema este usor de implementat, si necesita numai trei mesaje per o utilizare a sectiunii critice (cerere, permisiune, eliberare).
- Poate fi utilizata si in cazul mai general al *alocarii de resurse*
- Probleme:
  - Coordonatorul este un punct singular de esec: daca este distrus, intregul sistem este cazut
  - Daca procesele in mod normal se blocheaza dupa o cerere, nu pot distinge intre un coordonator cazut si permisiune resinsa deoarece in ambele cazuri mesajul de raspuns nu este primit
  - Intr-un sistem mare, un singur cordonator poate deveni o gatuire a performantei

Alg. distribuit Ricart & Agrawala (1/2)

- Necesita o ordine totala a tuturor evenimentelor din sistem:
  - Pentru oricare perechi de evenimente, precum tratarea mesajelor, trebuie sa fie care se intampla inainte
  - Algoritmul lui Lamport este o modalitate de a atinge aceasta ordonare si poate fi utilizat pentru a oferi marci de timp
- Cand un proces doreste sa intre intr-o sectiune critica,

1. Construieste un mesaj continand numele sectiunii crituce in care vrea sa intre, numarul sau de proces si timpul curent
2. Expediaza apoi mesajul la toate procesele, inclusiv el insusi
   - Expedierea mesajelor este considerata de incredere, adica un mesaj este confirmat
   - Daca este disponibila comunicarea de incredere in grup este de preferat mesajelor individuale

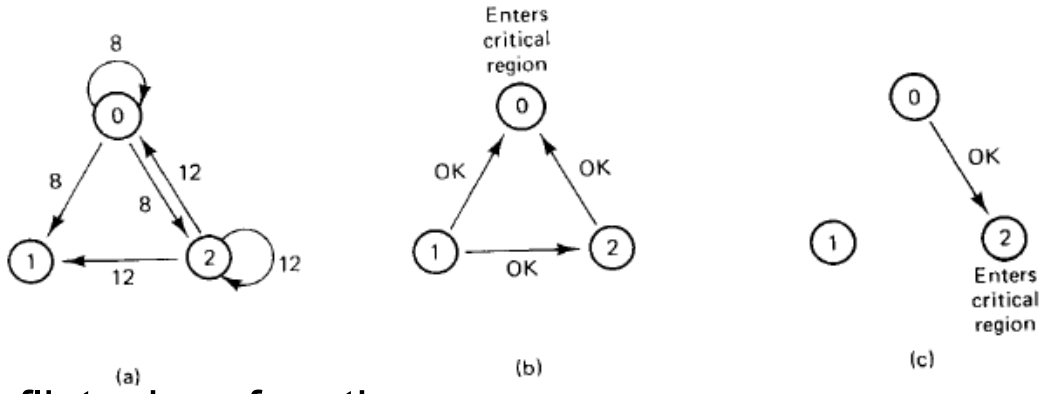
Alg. distribuit Ricart & Agrawala (2/2)

- Cand un proces receptioneaza un mesaj de cerere de la alat proces, actiunea sa depinde de starea sa relativ la sectiunea critica numita in mesaj:

1. Daca receptorul nu este in sectiunea critica si nu doreste sa intre in aceasta => expediaza inapoi un mesaj OK catre expeditorul initial.
2. Daca receptorul este deja in regiunea critica => nu raspunde + pune in coada cererea
3. Daca receptorul doreste sa intre in sectiunea critica dar inca nu a a facut acest lucru =>
   - Compara marca de timp din mesajul sosit cu cea continuta in mesajul e care l-a trimis la toate procesele
   - Cea mai joasa marca invinge:
     - Daca marca mesajului sosit este mai mica, receptorul trimite un mesaj de OK
     - Daca propriul mesaj are o marca mai mica, receptorul trece in coada mesajul sosit si nu raspunde

- Dupa expedierea cererilor de permisiune de a intra in sectiunea critica
  procesul asteapta ca toate celelalte sa-l acorde permisiunea
- De indata ce toate permisiunile sosesc,
  procesul poate intra in sectiunea critica
- Cand iese din sectiunea critica
  procesul expediaza mesaje de OK la toate procesele din coada sa & sterge coada.

Exemplu



- Daca nu exista nici un conflict, sigur functioneaza.
- Sa presupunem ca doua procese incearca sa intre in aceeasi sectiune critica simutan :
  - procesul 0 expediaza la toate procesele o cerere cu marca de timp 8,
  - in aproape acelasi timp, procesul 2 expediaza o cerere cu marca de timp 12.
- Procesul 1 nu este interesat sa intre in sectiunea critica asa ca trimite OK la ambii expeditori.
- Procesele 0 si 2 ambele vad conflictul si compara marcile de timp
  - Procesul 2 vede ca a pierdurt si acorda permisiunea expediind OK.
  - Procesul 0
    - Pune in coada cererea de la 2 pentru procesare ulterioara
    - Intra in sectiunea critica – (b)
    - Cand termina elimina cererea de la 2 din coada sa
    - Expedieaza OK la procesul 2, permitand acestuia sa intre in sectiunea critica - (c).

Pro si contra

- Excluderea mutuala este garantata fara impas sau infometare
- Nr. mesaje necesare per intrare este acum 2(*n* -1) daca nr. de procese din sistem - *n*.
- Nu exista un punct singular de esec – a fost inlocuit cu *n* puncte de esec!
  - Daca unul dintre procese este cazut, nu mai raspunde cererilor.
  - Aceasta tacere este interpretata (incorect) ca respingere a permisiunii, astfel blocandu-se toate incercarile ulterioare ale altor procese de a intra in sectiunea critica
  - Probabilitatea ca una dintre cele *n* procese sa esueze este de *n* ori mai mare decat cea a unui singur coordonator => s-a inlocuit un algoritm slab cu unul si mai slab care necesita un trafic si mai mare
- O alta problema:
  - Fie se utilizeaza o primitiva de comunicare in grup sau
  - Fiecare proces trebuie sa mentina o lista a proceselor, incluzand procesele care intra in grup, parasesc grupul sau care esueaza

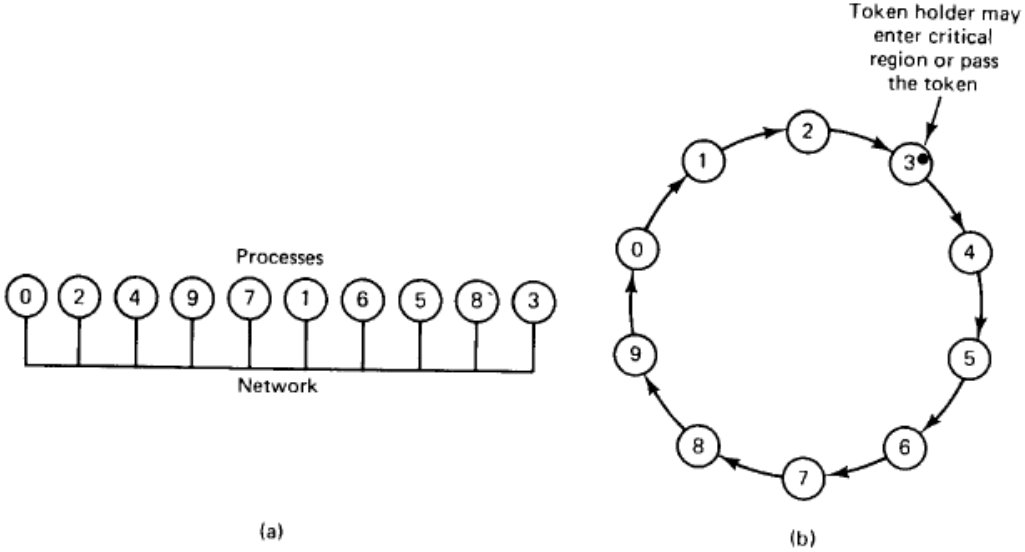
  => Metoda este functionala in cazul unor grupuri mici de procese care nu-si schimba apartenenta la grup
- Toate procesele sunt implicate in luarea deciziilor referitoare la intrarea in sec. critice
  - Daca un proces este incapabil sa trateze incarcarea, este improbabil ca fortarea ca toate celelalte procese sa faca acelasi lucru in exact acelasi timp in paralel nu ajuta

Imbunatatiri minore

- Remediu:
  - Cand este primita o cerere, receptorul trimite intotdeauna un raspuns, fie acordand sau respingand permisiunea
  - Daca o cerere sau un raspuns este pierdut, expeditorul foloseste expirarea timpului pentru a concluziona ca destinatorul nu mai este functional
  - Dupa respingerea unei permisiuni, expeditorul ar trebi sa fie blocat in asteptarea unui mesaj Ok ulterior
- Obtinerea permisiunii de la fiecare pentru a intra in sectiunea critica este o problema mare => ? Alta metoda pentru a preveni ca doua procese sa intre in sectiunea critica in acelasi timp.
  - Permisiune catre un proces sa intre in sectiunea critica daca a colectat majoritatea simpla de la celelalte procese in loc de a o obtine de la toate
- Concluzie:
  - Algoritmul propus este mai lent, mai complicat, mai costisitor si mai putin robust fata de cel centralizat

Algoritmul semnului in inel (1/2)

- Presupune:
  - Retea bazate pe bus, fara o ordonare inerenta a proceselor
  - Este construit un inel logic in care fiecarui proces ii este asignata o pozitie in cerc
  - Pozitiile in inel pot fi alocate de exemplu in ordine numerica a adreselor
  - Nu conteaza cum este realizata ordonarea
  - Tot ceea ce conteaza este ca fiecare proces sa-si cunoasca procesele vecine.



Algoritmul semnului in inel (1/2)

- Cand cercul este initializat, procesul 0 ii este acordat un semn.
- Semnul circula in inel
  - Este pasat de la procesul k la procesul k+1 (modulo dimensiunea inelului) in point-to-point messages.
- Cand un proces primeste semnul de la vecinul sau,
  - Verifica daca este pe punctul de a intra intr-o sectiune critica
  - Daca da,
    1. Procesul intra in sectiune,
    2. Isi face treaba in sectiune,
    3. Paraseste sectiunea.
    4. Paseaza semnul in inel.
  - Daca nu,
    1. Paseaza semnul in inel [=> daca nici un proces nu doreste sa intre in nici o sectiune critica, semnul circula doar la viteza mare in inel]
- Nu este permisa intrarea intr-o a doua sectiune critica prin utilizarea aceluiasi semn

Pro si contra

- Corectitudine: numai un proces are semnul la un moment dat, astfel numai un proce poate intra in sectiunea crituca
- Deoarece semnul circula intre procese intr-o ordine bine definita, infometarea nu poate interveni
- Deoarece un proces decide daca doreste sa intre intr-o sectiune critica, in cel mia rau caz va astepta ca toate celelalte procesele sa intre in sectiunea critica si s-o paraseasca
- Probleme:
  - Daca semnul este pierdut, trebuie regenerat
  - Detectarea daca a fost pierdut este dificila, deoarece cantitatea de timp dintre aparitii succesive ale semnului in retea este nemarginita
    - Faptul ca semnul n-a fost receptionat pentru o ora nu inseamna ca a fost pierdut; unul dintre procese e posibil sa-l utilizeze
- Algoritmul are probleme daca un proces cedeaza, dar recuperarea este mai usoara decat in celelalte cazuri
  - Daca se solicita ca un proces care receptioneaza un semn sa confirme receptionearea, un proces cazut va fi detectate de vecinul care i-a trimis semnul
  - In acel moment procesul cazut poate fi eliminat din grup si detinatorul semnulu poate trece la urmatorul membru din inel, sau mai departe daca este necesar
  - Fiecare mentine configuratia curenta a inelului

O comparare a celor trei algoritmi (1/2)

Algoritm	*Mesaje*	*Intarziere inainte de intrare*	*Probleme*
Centralizat	3	2	Cadere coordonator
Distribuit R&A	2(*n*-1)	2(*n*-1)	Caderea oricarui proces
Semnal in inel	1 to infinity	0 to *n* -1	Semn pierdut, proces cazut

Mesaje:

- Algoritmul centralizat este cel mai simplu si cel mai eficient:
  - Necesita numai trei mesaje pentru a intra si parasi o sectiune critica: o cerere si o permisiune pentru a intra, si o eliberare la iesire
- Algoritmul R&A
  - necesita *n* -1 mesaje de cerere, unul la fiecare alt proces,
  - in plus *n* -1 mesaje de permisiune
- In algoritmul semnului, numarul este variabil.
  - Daca fiecare proces doreste in mod constant sa intre intr-o sectiune critica, atunci fiecare trecere a semnului va rezulta intr-o intrare si o iesire pentru, in medie, un mesaj per sectiunea critica
  - La cealalta extrena, semnul poate circulate ore intregi fara ca vreun proces sa fie interesat de acesta => nr. mesaje per intrare in sectiunea critica este nemarginit

O comparare a celor trei algoritmi (2/2)

- Intarzierea cu care un proces intra in sectiunea critica variaza de asemenea la cei 3 algoritmi:
  - Cand sectiunile critice sunt scurte si rar utilizare, factorul dominant in intarziere este mecanismul curent de intrare in sectiunea critica
  - Cand sunt lungi si des utilizate, factorul dominant este asteptarea ca tot ceilalti sa ajunga la rand
  - Sunt necesare, presuunand ca reteaua poate trata un singur mesa la un moment dat
    - doar 2 timpi pentru mesaje pentru a intra in sectiunea critica in cazul centralizat,
    - 2(n -1) timpi in cazul distribuit
    - pentru semnul in inel, timpul variaza de la 0 (semn de abia sosit) la n-1 (semnul tocmai a plecat).
- Evenimente de esec
  - Algoritmii distribuiti sunt mai sesnsibili la esecuri decat cel centralizat
  - Intr-un sistem tolerant la esecuri, nici unul dintre cei 3 algoritmi nu este adecvat, dar daca esecurile sunt nefrecvente, sunt acceptabili