Algoritmi paraleli și distribuiți

Alegerea liderului

Mitică Craus

Univeristatea Tehnică "Gheorghe Asachi" din Iași

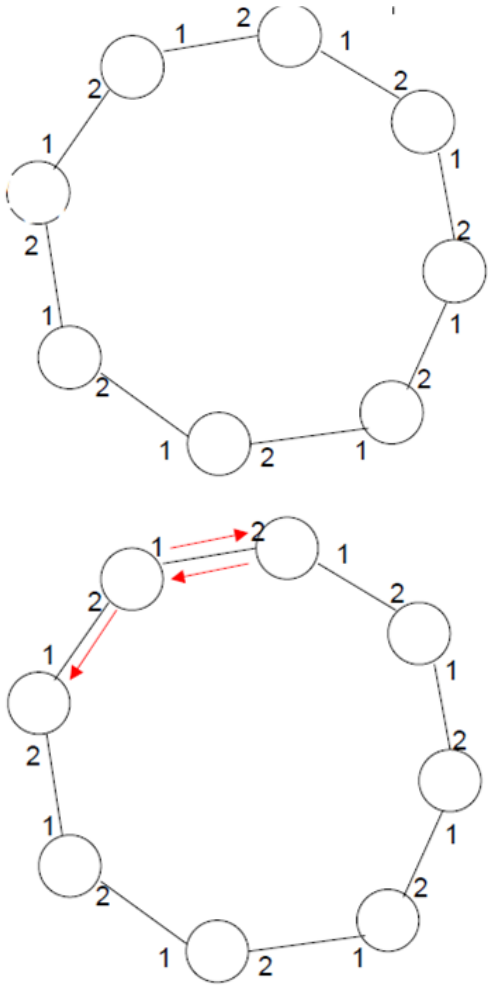
Alegerea liderului

Introducere

- *Problemă fundamentală: în sistemele distribuite este frecvent nevoie de un coordonator.*
- *Aplicații diverse:*
  - *defecțiuni ale componentelor unui sistem distribuit (de exemplu: alegerea unui nou server coordonator pentru continuarea funcționării unui serviciu);*
  - *excludere mutuală în sisteme bazate pe comunicarea prin mesaje;*
  - *rețele mobile – alegerea unui alt lider când liderul cunoscut părăsește rețeaua.*
- *Topologii de comunicare:*
  - *inelul;*
  - *graful.*

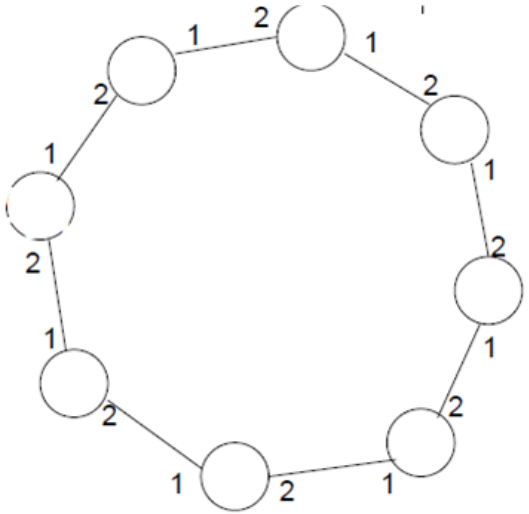
Alegerea liderului în rețele cu topologie de comuncare inel - aspecte generale

- Fiecare nod al inelului este asociat unei unități de procesare.
- Se stabilec sensuri în inel. De exemplul stânga și dreapta unui nod este fixată din perspectiva așezării nodului cu "fața" la centru.
- Legăturile între noduri pot fi bidirecționale.

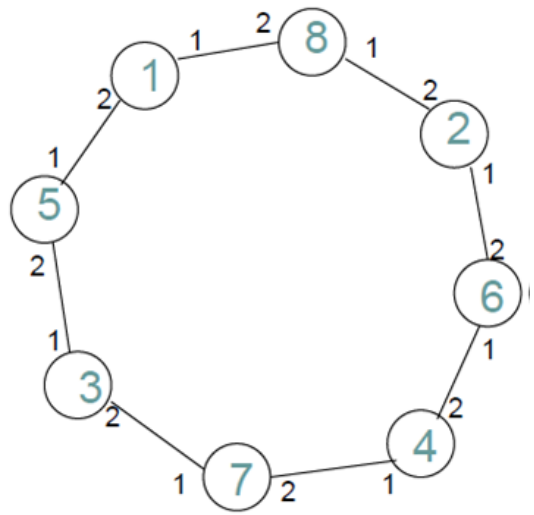


Aspecte generale - continuare

- Inel anonim:
  - Unitățile de procesare nu au indentificatori.
  - Au mecanisme identice pentru schimbarea stărilor, adică funcțiile de tranziție de stare sunt identice.
  - Sunt exprimate în termeni de stânga și dreapta.



- Inel etichetat:
  - Fiecare unitate de procesare $p_i,\ i \in \{0,1,\ldots,n-1\}$ are asociat un identificator unic id_i
  - Funcțiile de tranziție de stare sunt diferite.
  - Sunt identificate prin *id*-urile lor.



Aspecte generale - continuare

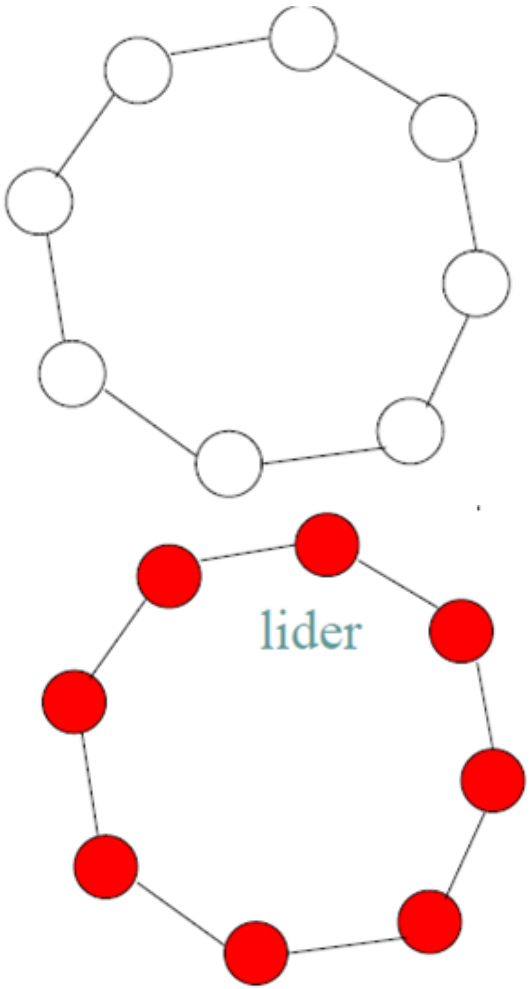
- Inel uniform:
  - Unitățile de procesare au identificatori unici.
  - Numarul nodurilor inelului (n) nu este cunoscut de către unitățile de procesare.
  - Mecanismele de schimbare a stărilor nu depind de n (Funcțiile de tranziție de stare nu au variabila n ca parametru).
- Inel neuniform:
  - Unitățile de procesare au identificatori unici.
  - Numarul nodurilor inelului (n) este cunoscut de către unitățile de procesare.
  - Mecanismele de schimbare a stărilor depind de n (Funcțiile de tranziție de stare au variabila n ca parametru).

Aspecte generale - continuare

- Algoritmii de alegere a liderului depind de:
  - Tipul inelului: anonim sau etichetat;
  - Numărul nodurilor inelului: cunoscut sau necunoscut;
  - Tipul de ceas: global (execuție sincronă) sau local (execuție asincronă).

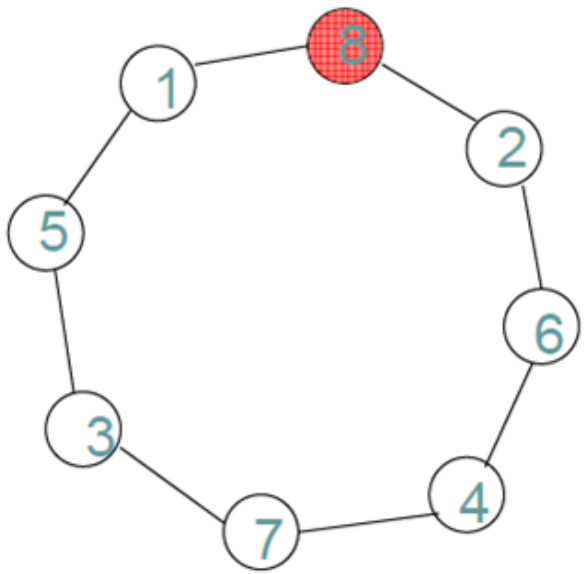
Alegerea liderului în inele anonime

- Inele anonime sincrone:
  - Ceasul este global.
  - Unitățile de procesare execută același program.
  - La un moment dat, unitățile de procesare efectuează aceeași operație (execută aceeași instrucțiune)
  - Unitățile de procesare au aceeași evoluție, deoarece sunt identice.
  - La final, toate unitățile de procesare sunt lideri.
- Inele anonime asincrone:
  - Ceasul este local.
  - Unitățile de procesare au aceeași evoluție, deoarece sunt identice.
  - La final, toate unitățile de procesare sunt lideri.



Alegerea liderului în inele etichetate

- Unitatea de procesare cu cel mai mare/mic identificator (*id*) este aleasă lider



Algoritm sincron de alegere a liderului în inele etichetate uniforme - descriere

- Startul nu este sincronizat. Aceasta înseamnă că unitățile de procesare se "trezesc" spontan sau după primirea unui mesaj, adică încep execuția programului de alegerea a liderului în momente diferite.
- La trezire, dacă nu a primit nici un mesaj, unitatea de procesare devine *participant*; dacă a primit un mesaj, devine *releu*.
- Atunci când o unitate de procesare cu rol de *participant* primește un mesaj de la p_i, dacă *id*-ul primit este mai mare decât identificatorul cel mai mic cunoscut (inclusiv propriul identificator), îl ignoră; altfel, îl reține $2^{id} - 1$ runde.
- Atunci când o unitate de procesare cu rol de *releu* primește un mesaj de la p_i, dacă *id*-ul primit este mai mare decât identificatorul cel mai mic cunoscut (fără propriul identificator), îl ignoră; altfel, îl retransmite imediat.

Algoritm sincron de alegere a liderului în inele etichetate uniforme

- *Notații*:
  - R_i memorează mesajele recepționate de unitatea de procesare p_i.
  - S_i memorează mesajele care urmează a fi trimise de către unitatea de procesare p_i.
  - W_i memorează mesajele întârziate (în așteptare).
  - $stare_i$ reprezintă starea curentă a unității de procesare p_i (*adormit*, *participant* sau *releu*).
  - $statut_i$ memorează starea finală a unității de procesare p_i (*lider* sau *non_lider*),
  - h reprezintă faza în care se află mesajul m.
- *Premise*:
  - Inițial $S_i = R_i = W_i = \emptyset$ și $stare_i = adormit$, $statut_i = nedefinit$.

Pseudocod pentru unitatea de procesare p_i

ASAL_IEU$(S_i, R_i, W_i, n, stare_i, statut_i, id_i, p_i)$

```
1  while true
2  do if stare_i = adormit
3        then if R_i = ∅
4                then stare_i ← participant
5                     min ← id_i
6                     memoreaza (id_i, 1) in S_i
7                else stare_i ← releu
8                     min ← ∞
9     for each (m, h) in R_i
10    do if h ≠ 3
11          then if m < min
12                  then min ← m
13                       if stare_i = releu and h = 1
14                         then memoreaza (m, 1) in S_i
15                         else memoreaza (m, 2) impreuna cu numarul rundei in W_i
16                  else if m = id
17                         then statut_i ← lider
18                              memoreaza (m, 3) in S_i
19          else if statut_i = lider
20                  then break /* termină execuția algoritmului */
21                  else statut_i ← non_lider
22                       memoreaza (m, 3) in S_i
23    for each (m, 2) in W_i
24    do if (m, 2) a fost primit in urma cu 2^m − 1 runde
25          then sterge (m, 2) din W_i si memoreaza in S_i
26    trimite S_i spre stanga
27    if statut_i = non_lider
28      then break /* termină execuția algoritmului */
```

Corectitudinea

Teorema (1)

Doar unitatea de procesare cu cel ai mic id primește inapoi propriul id.

Demonstrație.

Fie p_i unitatea de procesare participantă cu cel mai mic identificator (id_i). Cel puțin o unitate de procesare are statut de participant. Evident, nici o unitate de procesare nu poate "înghiți" (aruncă la coș, fără a-l transmite mai departe) mesajul id_i. Mai mult, mesajul id_i. este întârziat, în fiecare unitate de procesare, cel mult 2^{id_i} runde. Să presupunem că există o unitate de procesare p_j, $j \neq i$, care primește înapoi propriul identificator, id_j. Rezulă că id_j trece prin toate unitățile de procesare din inel, inclusiv prin p_i. Dar $id_i < id_j$ și p_i este o unitate de procesare participantă, care nu va retransmite id_j. Contradicție. □

Complexitatea

Mesajele care circulă prin inel pot fi clasificate în trei categorii:

1. mesaje în faza I ($h = 1$);
2. mesaje în faza II ($h = 2$) trimise înainte de intrarea mesajului liderului în faza II;
3. mesaje în faza II ($h = 2$) trimise după intrarea mesajului liderului în faza II.
4. mesaje în faza III ($h = 3$) trimise după ce a fost decis liderul.

Lema (1)

Numărul de mesaje din prima categorie este cel mult n.

Demonstrație.

Este suficient să demonstrăm că fiecare unitate de procesare retransmite cel mult un mesaj din prima categorie. Să presupunem că există o unitate de procesare p_i care retransmite două mesaje din prima categorie, id_j primit de la p_j și id_k de la p_k. Fără a restrânge generalitatea, presupunem că p_j este mai aproape de p_i decât p_k. Dacă id_k ajunge în p_j după ce p_j se "trezește" și trimite p_j, id_k trece în faza II și va fi întârziat în fiecare unitate de procesare participantă 2^{id_k} runde; altfel p_j nu participă și nu trimite identificatorul id_j. Prin urmare, fie id_k ajunge în p_i în faza II, fie id_j nu este trimis de p_j și prin urmare nu este primit de p_i. Contradicție. □

Complexitatea - continuare

Lema (2)

Fie r prima rundă în care se "trezește" măcar o unitate de procesare și începe execuția algoritmului. Fie p_i una dintre acestea. Dacă unitatea de procesare p_j este la distanță d față de p_i, atunci p_j primește un mesaj din prima categorie cel mult în runda $r+d$.

Demonstrație.

Inducție după d. Pentru $d=1$ este evident că vecinul lui p_i primește mesajul id_i (din categoria 1) în runda $r+1$. Să presupunem că unitatea de procesare p_k situată la distanța $d-1$ față de p_i primește un mesaj din prima categorie, cel mult în runda $r+d-1$. Dacă aceasta este " trezită", a trimis deja la vecinul p_j un mesaj din prima categorie; altfel, înseamnă că p_k are statut de *releu* și va retransmite în runda $r+d$ către p_j mesajul de categoria 1, primit anterior. □

Complexitatea - continuare

Lema (3)

Numărul de mesaje din categoria a doua este cel mult n.

Demonstrație.

Din Lema 1 rezultă că fiecare unitate de procesare retransmite cel mult un mesaj din prima categorie, adică pe o muchie este transmis cel mult un mesaj din prima categorie. Din Lema 2 rezultă ca în runda $r+n$ pe fiecare muchie a fost trimis un mesaj din categoria 1. Rezultă că după runda $r+n$ nu va mai fi transmis nici un mesaj din categoria 1. Din Lema 2 rezultă că mesajul liderului intră in faza a II-a după cel mult $r+n$ runde, adică nu mai târziun de a n-a rundă care urmează trezirii primei unități de procesare. Rezultă că mesajele din categoria 2 circulă cel mult n runde. Un mesaj m aflat în faza a II-a este întârziat 2^m-1 runde, după care este retransmis. Prin urmare, un mesaj m circulă cu statutul "categoria 2" cel mult de $\frac{n}{2^m}$ ori. Deoarece mesajul care conține cel mai mic identificator circulă de cele mai multe ori, rezultă că numarul maxim de mesaje se obține atunci când unitățile de procesare participante au identificatorii $0,1,\ldots,n-1$. Dacă ținem cont că mesajul liderului nu poate face parte din categoria 2, rezultă că numărul mesajelor din categoria 2 este $\sum_{i=1}^{n-1}\frac{n}{2^i} \leq n$ □

Complexitatea - continuare

Lema (4)

Nici un mesaj nu este retransmis după ce liderul p_i primește propriul identificator id_i.

Demonstrație.

Toate mesajele care urmează lui id_i sunt "înghițite". □

Complexitatea - continuare

Lema (5)

Numărul de mesaje din categoria a treia este cel mult $2n$.

Demonstrație.

Fie p_i liderul și p_j o unitate de procesare cu statut de *participant*. Din Teorema 1 rezultă că $id_i < id_j$. Din Lema 4 rezultă că nici un mesaj nu este retransmis după ce p_i primește înapoi propriul identificator id_i. Deoarece id_i este întârziat cel mult 2^{id_i} runde în fiecare unitate de procesare, sunt necesare cel mult $n2^{id_i}$ runde pentru ca p_i să primească înapoi propriul identificator id_i. Așadar, mesajele din categoria a treia sunt transmise de-a lungul a cel mult $n2^{id_i}$ runde. În timpul acestor runde, id_j este retransmis cel mult de $\frac{1}{2^{id_j}} n2^{id_i} = \frac{n}{2^{id_j - id_i}}$ ori. Astfel, numărul mesajelor din categoria 3 este cel mult $\sum_{j=0}^{n-1} \frac{n}{2^{id_j - id_i}}$. Pe baza faptului că numărul maxim de mesaje se obține atunci când unitățile de procesare participante au identificatorii $0, 1, \ldots, n-1$ (Lema 3) se deduce că $\sum_{j=0}^{n-1} \frac{n}{2^{id_j - id_i}} = \sum_{k=0}^{n-1} \frac{n}{2^k} < 2n$. □

Complexitatea - continuare

Lema (6)

Numărul de mesaje din categoria a patra este n.

Demonstrație.

Fiecare unitatea de procesare transmite mai departe mesajul liderului ($h = 3$), prin care acesta se declară câștigător. □

Complexitatea - continuare

Teorema (2)

Numărul de mesaje circulate prin algoritmul sincron de alegere a liderului în inele etichetate uniformeste cel mult $5n$.

Demonstrație.

Consecință imediată a lemelor 1,3,5 și 6. □

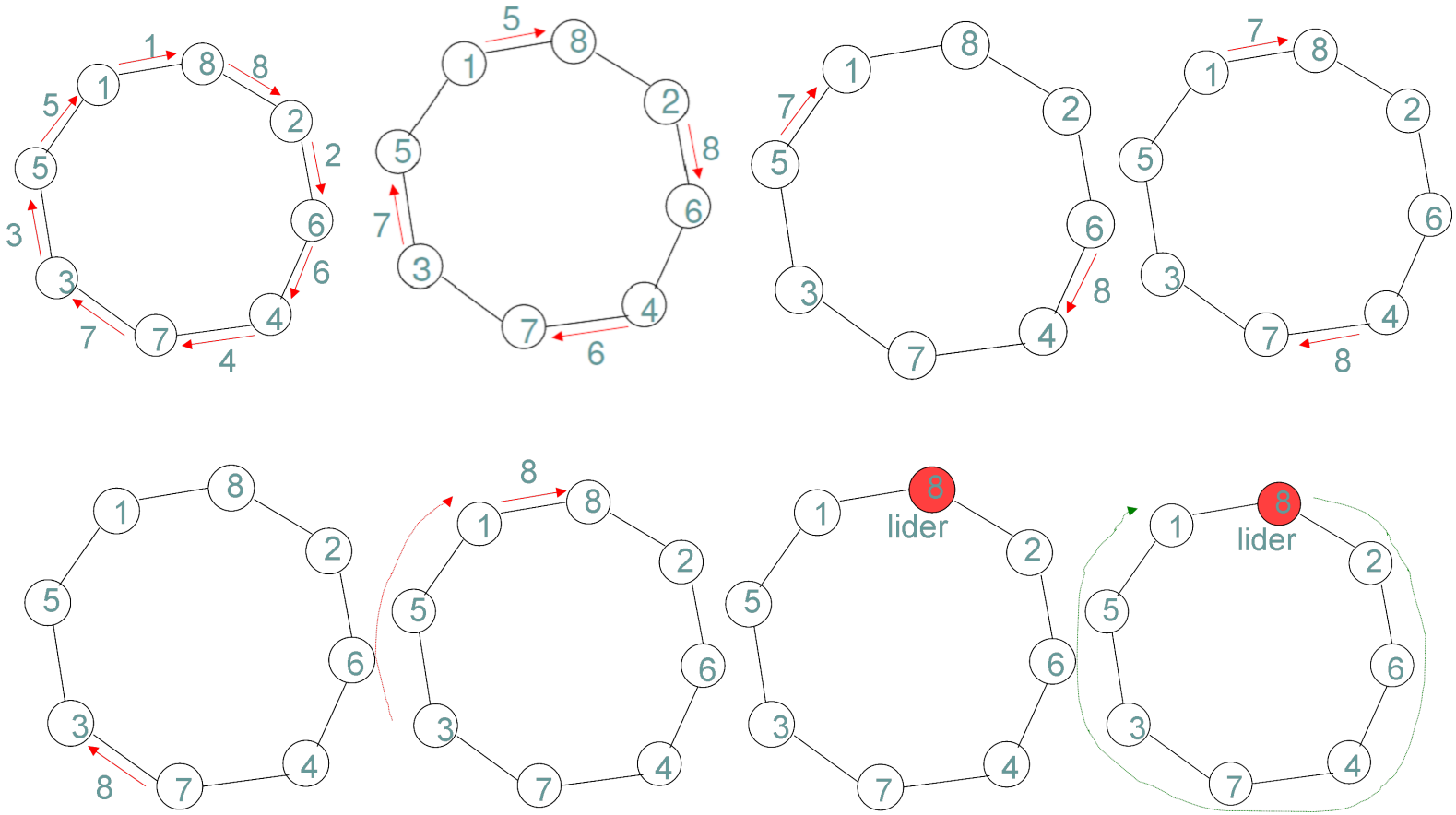
Algoritm sincron de alegere a liderului în inele etichetate neuniforme - descriere

- Algoritmul se desfășoară în faze. Fiecare fază este compusă din n runde.
- Startul este sincronizat. Aceasta înseamnă că toate unitățile de procesare se "trezesc" simultan, adică încep execuția programului de alegerea a liderului în același moment.
- Unitatea de procesare cu cel mai mic identificator este aleasă lider.
- În fiecare rundă, fiecare unitate de procesare efectuează următoarele operații:
  - analizează mesajele primite pe canalele din stânga și din dreapta;
  - își schimbă starea în funcție de starea curentă și de mesajele primite; dacă nu a primit niciun mesaj, își schimbă starea doar în funcție de starea curentă;
  - transmite mesaje la vecini, dacă are ceva de transmis.
- În faza i, dacă nimeni nu este ales, atunci unitatea de procesare cu id-ul i se autodeclară lider și transmite la vecini un mesaj prin care se declară lider; celelalte unități de procesare retransmit mesajul primit.

Algoritmul asincron LCR (LeLann, Chang si Roberts) - descriere

- Se consideră că unitățile de procesare "stau cu fața" spre centrul inelului.
- Fiecare unitate de procesare p_i transmite identificatorul său (id_i) vecinului din stânga.
- Apoi, așteaptă răspuns de la vecinul din dreapta
- Dacă identificatorul recepționat este mai mare decât propriul identificator, transmite identificatorul primit spre stânga.
- Dacă identificatorul primit e mai mic decat propriul identificator, ignoră ("inghite") mesajul.
- Dacă identificatorul recepționat este propriul identificator, unitatea respectivă se declară *lider* și transmite în inel un mesaj de terminare.
- Dacă o unitate de procesare primește mesaj de terminare, termina ca *non-lider*.

Algoritmul LCR - exemplu de execuție



Algoritmul LCR

Pseudocod pentru unitatea de procesare p_i

- *Notații*:
  - id_i reprezintă identificatorul unității de procesare p_i.
  - $stare_i$ memorează starea curentă a unității de procesare p_i (*adormit*, *trezit*).
  - $statut_i$ memorează starea finală a unității de procesare p_i (*lider* sau *non_lider*)
- *Premise*:
  - Inițial $stare_i = adormit$ și $statut_i = necunoscut$.

```
LCR_LE(p_i, stare_i, statut_i)
 1  stare_i ← trezit
 2  trimite <"id_nou", id_i> spre stanga
 3  while true
 4  do switch
 5       case primeste <"id_nou", id> dinspre dreapta :
 6            if id = id_i
 7              then statut_i ← lider
 8                   trimite <"lider", id> spre stanga
 9            if id > id_i
10              then trimite <"id_nou", id> spre stanga
11       case primeste <"lider", id> dinspre dreapta :
12            if id ≠ id_i
13              then statut_i ← non_lider
14                   trimite <"lider", id> spre stanga
15            break /* termină execuția algoritmului */
```

Corectitudinea

Teorema (3)

Doar unitatea de procesare cu cel mai mare id primește primește inapoi propriul id.

Demonstrație.

Doar mesajul unității de procesare cu cel mai mare *id* nu va fi "inghițit" (aruncat la coș). □

Complexitatea

Teorema (4)

Numărul de mesaje care circulă prin inel este cel mult n^2.

Demonstrație.

Cazul cel mai nefavorabil este acela în care unitățile de procesare au identificatori din mulțimea $\{0,1,\ldots,n-1\}$ și sunt plasate pe inel în ordinea $n-1,n-2,\ldots,0$. În această situație, mesajul $id_i = i$ al unității de procesare p_i este retransmis exact de $i+1$ ori. Astfel, numărul mesajelor care circulă prin inel (inclusiv mesajul de terminare) este $n+\sum_{i=0}^{n-1}(i+1) = O(n^2)$. □

Algoritmul FloodMax sincron - descriere

- Graful $G = (V, E)$ este conex. Unitățile de procesare cunosc diametrul$= d$.
- Fiecare unitate de procesare conține o înregistrare a identificatorului maxim cunoscut; inițial acesta este propriul identificator.
- La fiecare rundă, fiecare unitate de procesare trimite acest maxim vecinilor.
- După d runde, dacă maximul cunoscut este propriul identificator, atunci unitatea de procesare se declară *lider*; altfel se consideră *non-lider*.

Algoritmul FloodMax sincron

Pseudocod pentru unitatea de procesare p_i

- *Notații*:
  - id_i reprezintă identificatorul unității de procesare p_i.
  - R_i memorează mesajele recepționate de unitatea de procesare p_i într-o rundă.
  - max_id_i memorează cel mai mare identificator cunoscut de p_i..
  - $statut_i$ memorează starea finală a unității de procesare p_i (*lider* sau *non_lider*).
- *Premise*:
  - Inițial $max\_id_i = id_i$ și $statut_i = necunoscut$.

FloodMax_LE$(G, d, p_i, max_id_i, statut_i)$

```
1  for runda ← 1 to d
2  do for each (m) in R_i
3       do if max_id_i < m
4             then max_id_i ← m
5       trimite max_id_i vecinilor
6  if max_id_i = id_i
7    then statut_i ← lider
8    else  statut_i ← non − lider
```

Complexitatea

Teorema (5)

Numărul de mesaje circulă prin graf este $2d|E|$.

Demonstrație.

Într-o rundă, pe fiecare muchie circulă două mesaje. Numărul rundelor este d. □