Leader Election in Anonymous Rings: Franklin

**Abstract.** In cadrul acestei lucrari va fi descris un simulator al unui algoritm de leader election pentru topologii ring anonime, bidirectionale si asincrone. Algoritmul pentru care este construit simulatorul este algoritmul Franklin, care alege identificatorii unui nod in mod aleatoriu si numara hopurile pentru a detecta coliziuni. Se pot folosi diferite metode pentru a demonstra corectitudinea algoritmului si ca in final va fi ales un singur nod ca fiind leader-ul retelei. Se va prezenta si implementarea simulatorului care poate simula retele de orice dimensiune.

1. Introducere

Problema Leader election consta in alegerea unui unic lider in cadrul unor procese distribuite. Pentru aceasta, este necesar ca fiecare proces sa execute acelasi algoritm. Problema alegerii unui lider este o problema fundamentala in cadrul sistemelor distribuite si are foarte multe utilizari. De exemplu, este o unealta puternica pentru ruperea simetriei in cadrul unui sistem distribuit sau alegand un proces ca fiind lider se pot executa algoritmi centralizati intr-un mediu distribuit.

Sunt o multime de algoritmi pentru alegerea unui lider. Acesti algoritmi difera prin modul de comunicare (sincron vs. asincron), numele proceselor (identificatori unici vs. retele anonime), topologia retelei (ring, acyclic graph, complete graph etc.). In cadrul aceste lucrari va fi prezentat un algoritm care utilizeaza comunicare asincrona fara a tine cont de ordinea primirii mesajelor si care va functiona in cadrul unei topologii de tip ring bidirectionala.

Un algoritm clasic pentru topologii ring unidirectionale este algoritmul lui Chang si Roberts. Algoritmul necesita ca fiecare proces sa aiba un identificator unic si intre care sa existe o relatie de ordine; procesul cu cel mai mare identificator devine leaderul sistemului. Ideea de baza a acestui algoritm este aceea ca fiecare proces trimite identificatorul sau de-a lungul inelului si doar mesajul cu cel mai mare identificator se intoarce la nodul care a generat mesajul. Acest algoritm necesita in cel mai rau caz *O(n2)* mesaje, dar in medie necesita in jur de *O(n log n)* mesaje. Franklin a dezvoltat un algoritm pentru inele topologii ring bidirectionale si care necesita in cel mai rau caz *O(n log n)* mesaje. Algoritmul lui Franklin utilizeaza mai multe etape pentru a alege liderul inelului si fiecare nod al topologiei poate fi fie activ, fie pasiv. La inceputul fiecarei etape, fiecare proces activ trimite celor mai apropiati vecini activi identificatorul sau si primeste la randul sau mesaje asemanatoare de la vecini. Un nod activ va fi activ si in urmatoarea etapa doar daca identificatorul sau este mai mare decat identificatorii primiti de la vecinii sai. Chiar daca algoritmul este pentru inele bidirectionale, acesta a fost adaptat pentru inele unidirectionale.

1. Algoritmul lui Franklin pentru inele anonime

Consideram un inel de procese p0, ….., pn-1 for n≥2. Procese sunt anonime, adica nu au un identificator unic. Comunicarea intre procese prin intermediul mesajelor este asincrona si ordinea mesajelor nu este retinuta intre oricare pereche de procese. Canalale de comunicare sunt bidirectionale si orice process pi poate trimite mesaje vecinilor sai p(i+1)mod n si p(i-1) mod n; un mesaj este retinut in coada de mesaje a procesului destinatie. Se presupune ca primirea, procesarea si trimiterea unui mesaj nu introduce o latenta in retea (t=0). Canalele de comunicare sunt considerate a fi sigure si cozile de mesaje sunt gestionate de catre un scheduler ceea ce inseamna ca fiecare mesaj va fi pana la urma procesat de catre proceseul destinatie.

Fiecare process este fie activ, fie pasiv. In cadrul algoritmului Franklin fiecare proces pi poate avea trei stari:

* + idi {1, …., k}, pentru k ≥ 2, este identitatea unui nod nu neaparat unica
  + statei este {active, passive, leader}
  + biti este numarul selectiei curente modulo 2

Procesele pasive vor transmite mesajele mai departe crescand contorul de hopuri cu unu.

Toate mesajele sunt de forma (id, hop, bit) si pot fi transportate atat in sens trigonometric cat si in sens invers trigonometric unde:

* id este identificatorul nodului care transmite mesajul
* bit este un bit reprezentand etapa de alegere a leader-ului modulo 2
* hop este un contor care reprezinta numarul de hopuri ale unui mesaj si initial are valoare 1; este incrementat de fiecare data cand un proces trimite mesajul mai departe

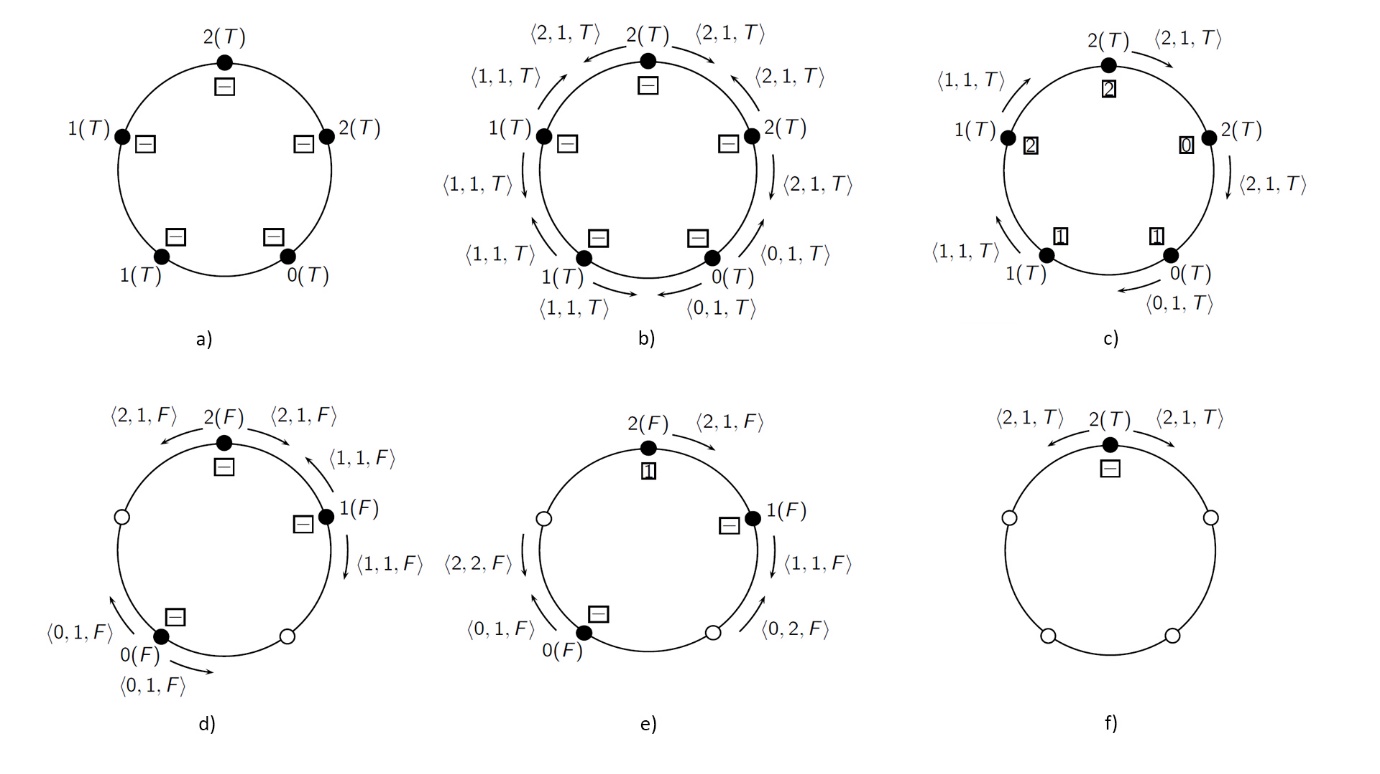
La inceputul unei etape fiecare process activ isi selecteaza in mod aleatoriu un identificator id si trimite un mesaj vecinilor sai; mesajul este de forma (idi, 1, biti). In continuare procesul pi va primi de la vecinii sai mesaje asemanatoare cu cel trimise de el. Dupa ce primeste cele doua mesaje de la vecinii sai, pi decide daca va mai fi activ in urmatoarea runda pe baza celor trei identificatori pe care ii are. Daca unul dintre mesajele primite are un identificator mai mare decat cel al nodului curent, atunci acesta devine inactiv. Daca un nod primeste un mesaj cu un hop egal cu numarul nodurilor din redea, acel proces devine liderul retelei (statei := leader).

In continuare vom da o descriere mai detaliata a algoritmului. Initial, toate procesele pi sunt active si biti este setat la T.

* La inceputul fiecarei runde cu numarul bit, un proces activ isi selecteaza un identificator id {1, …., k} si trimite mesajul (id, 1, bit) in ambele directii
* La primirea unui mesaj un proces pasiv trimite mesajul in aceeasi directie crescandu-i hop-ul cu 1. (id, hop+1, bit).
* Cand primeste un mesaj (id, hop, bit) cu biti = bit, un proces activ pi executa urmatorii pasi:
  + Daca hop = n, atunci pi devine leader (statei := leader)
  + Daca hop < n, atunci procesul stocheaza mesajul si asteapta un mesaj cu biti din directia opusa
* Un proces activ pi stocheaza mesajele care au un bit biti pentru a le procesa intr-o etapa cu un bit corespunzator
* Cand primeste un mesaj cu bitul biti din ambele directii, pi verifica daca unul dintre mesaje are un identificator mai mare dacat al sau. Daca se intampla ca unul din mesaje sa satisfaca conditia, procesul pi devine inactiv, altfel incepe o noua runda din cadrul algoritmului cu un bit inversat si cu un nou identificator.

1. Simulatorul

Pentru a intelege mai bine cum functioneaza algoritmul lui Franklin vom exemplifica prin intermediul unui exemplu fiecare pas al algoritmului; pasii pot fi urmariti in imaginea de mai jos.



1. Initial toate nodurile sunt active si au bitul de tura bit = T. La inceputul unei runde fiecare nod activ isi ia un identificator in mod aleatoriu
2. Trimite mesaje (id, 1, bit) in ambele directii
3. La primirea unui mesaj un nod activ stocheaza mesajul si asteapta mesajul din cealalta directie
4. La receptionarea mesajelor din ambele directii un nod devine inactiv daca identificatorul unuia din mesaje este mai mare decat identificatorul sau, altfel incepe o noua runda cu un nou indentificator si cu bitul de tura inversat
5. La receptionarea unui messaj un nod inactiv trimite mesajul mai departe in aceeasi directie incrementand hopul (id, hop+1, bit)
6. In ultima imagine mai ramane un singur nod activ care va trimite mesaje in ambele directii si fiind singurul nod activ acesta va primi mesajele cu numarul hopului egal cu numarul de noduri din retea. In acest caz el va deveni liderul retelei.

Pentru implementarea simulatorului s-a ales limbajul de programare Java datorita familiarizarii dezvoltatorilor cu acest limbaj. La pornirea simulatorului se vor verifica numarul de noduri date ca parametru si valoare maxima a identificatorilor nodurilor. Numarul de noduri trebuie sa fie minim 2 pentru a putea realiza un inel si valoare maxima trebuie sa fie tot minim 2. Daca cele doua conditii sunt indeplinite se va genera inelul cu n noduri si va incepe executia algoritmului.

Fiecare nod al algoritmului va fi reprezentat in cadrul simulatorului prin intermediul unui obiect de tip Nod care va contine starea procesului curent, bitul de iteratie, vecinii nodului, indentificatorul curent si o coada in care se vor tine mesajele primite. Mesajele dintre noduri vor fi reprezentate prin obicte de tip Message care contin identificatoul procesului care trimite mesajul, hopul, bitul de runda si directia mesajului.