**Tema 4: Algoritmi distribuiti**

1. **Algoritmi distribuiti de alegere leader**

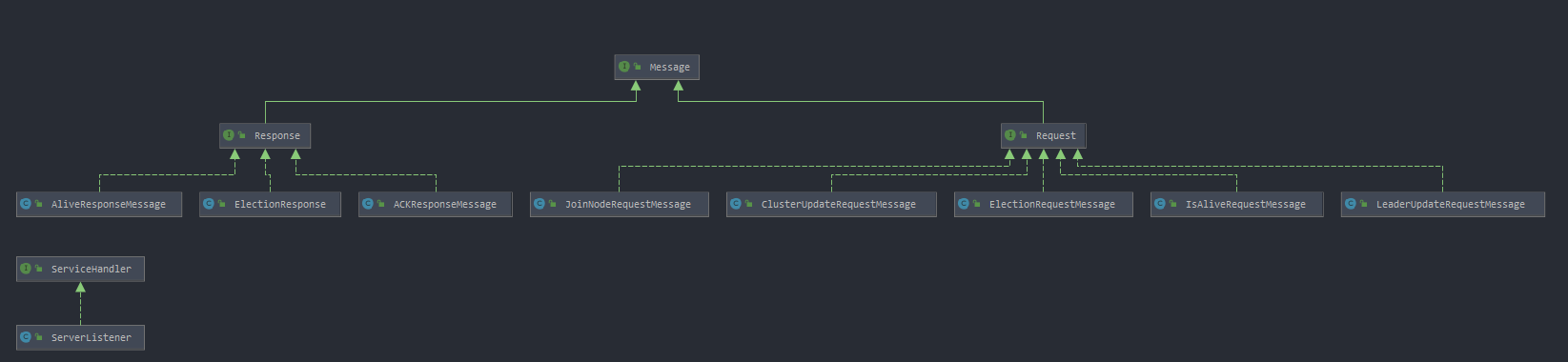
In context distribuit, este necesara existenta unui process cu rol de lider, proces ce va coordona intreaga procesare. Daca procesul ales ca lider esueaza, se va alege un nou lider pentru a fi posibila continuarea procesarii. Orice algoritm distribuit pentru alegerea liderului trebuie sa fie sigur, si longeviv. In continuare, se vor compara doi astfel de algoritmi: Bully si Ring.

Algoritmul Bully utilizeaza trei tipuri de mesaje: election, care anunta alegerea, answer, in care procesele raspund la alegere si leader, unde coordonatorii alesi sunt anuntati. Un proces incepe o alegere in momentul in care recentul coordonator a esuat sau daca primeste un mesaj de tipul election. Atunci cand incepe alegerea, un proces trimite un mesaj de tip election tuturor proceselor cu numar de identificare mai mare. Daca nu primeste niciun mesaj de tip answer, procesul ce a trimis mesajul de tip election va deveni coordonator si va anunta si restul proceselor. In cazul in care primeste un mesaj de tipul answer, procesul ce a initiat alegerea va astepta un mesaj de tipul coordonator. Daca un proces cunoaste faptul ca el are numarul de identificare cel mai mare, va anunta ca el este noul coordonator. In caz contrar, va incepe si el o alegere dintre procesele subordonate lui cu numerele de intentificare mai mari.

In continuare, va fi prezentat pseudocodul acestui algoritm:

1. Fiecare proces are un identificator unic si cunoaste identificatorii celorlalte procese, care pot fi active sau nu.
2. Un proces este lider, coordonand toate celelalte procese.
3. Daca liderul esueaza, va incepe procesul de alegere a unui nou lider.
4. Cand un proces observa faptul ca liderul a esuat, acesta va trimie un mesaj de tip election tuturor proceselor ce au un identificator mai mare decat al lui.
5. Atunci cand un proces primeste un mesaj de tip election, va raspunde cu un mesaj „OK” si apoi va trimite un mesaj de tip election proceselor cu numar de identificare mai mare decat al lui.
6. Daca un proces primeste un mesaj „OK”, mesaj de tip answear, va astepta sa fie anuntat noul lider.
7. Daca procesul nu primeste un raspuns de tip answear, se declara pe sine lider si anunta restul proceselor printr-un mesaj de tipul leader.
8. Procesul cu numarul de identificare maxim va fi ales intotdeauna ca lider.

Figura urmatoare prezinta o diagrama de clase a algoritmului prezentat mai sus:

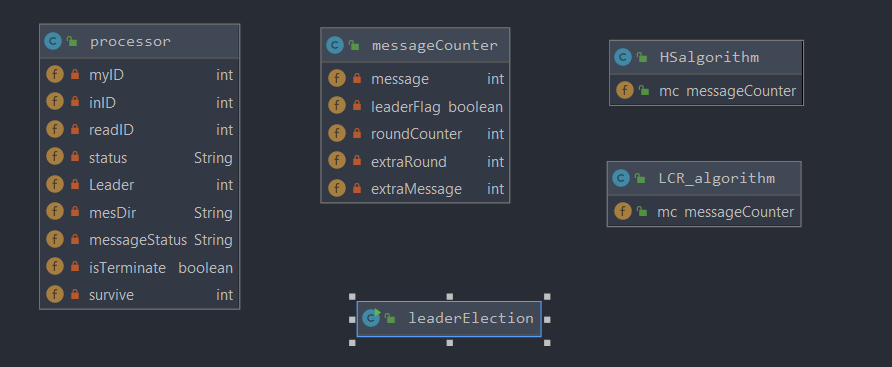


Algoritmul Ring reprezinta o alternativa a algoritmului Bully, procesele avand o organizare topologica de tip inel. Exista doar doua tipuri de mesaje: election si leader. La fel ca si in cazul algoritmului Bully, un proces incepe alegerea atunci cand observa faptul ca liderul a esuat. Mesajul de tip election va fi trimis procesului vecin de pe inel. Mesajul este mai apoi trimis prin inel, fiecare proces adaugandu-si propriul identificator la mesaj. Atunci cand mesajul ajunge inapoi la procesul ce a declansat alegerea, procesul cu identificatorl maxim va fi ales ca nou lider si restul proceselor vor fi anuntate prin mesaj de tip leader.

In continuare, se va prezenta pesudocodul algorimului:

1. Procesele sunt aranjate sub forma unui inel, in ordinea numarului.
2. Cand un proces observa faptul ca liderul a esuat, trimite un mesaj de tip election procesului vecin, lista ce contine initial numarul procesului ce a declansat alegerea.
3. Procesul asteapta sa primeasca o confirmare de la destinatar.
4. Daca nu a primit o confirmare, va trimite mesajul urmatorului proces din inel.
5. Cand un proces primeste un mesaj de tip election, numarul lui este adaugat in lista din mesaj si acesta este trimis mai departe.
6. Procesul ce a declansat alegerea primeste un mesaj de tip election ce contine numarul tuturor proceselor si alege noul lidel, trimitand un mesaj de tipul leader in jurul inelului.
7. Cand mesajul ajunge inapoi la procesul ce a declansat alegerea, aceasta se incheie.

Figura urmatoare descrie functional algoritmul prin diagrama de clase:



In cazul algoritmului Bully, daca procesul cu identificatorul cel mai mic detecteaza esecul coordonatorului, se vor realiza n-2 alegeri, toate procesele exceptand cel esuat si procesul cu identificatorul cel mai mare realizand sub-alegeri. Astfel, in cel mai rau caz, algoritmul Bully necesita O(n^2) mesaje. Daca procesul ce detecteaza esecul coordonatorului are cel mai mare identificator, se vor trimite n-2 mesaje de tipul coordonator, rezultand O(n-1) mesaje in cel mai bun caz.

In cazul algoritmului Ring, indiferent de identificatorul procesului ce a detectat esecul coordonatorului, se vor trimite n-1 mesaje de tip election si n-1 mesaje de tipul leader, rezultand astfel 2(n-1) mesaje trimise pentru o alegere.

1. **Algoritmi de excludere mutuala ce folosesc ceasuri logice**

Primul algoritm discutat este cel propus de Ricart si Agrawala, care presupune ca fiecare proces participant p1 retine un ceas Lamport si toate procesele trebuie sa poata comunica in perechi. Intr-un moment de timp, un proces de poate afla din una dintre cele trei stari: released, in afara zonei critice, wanted, asteptand sa intre in zona critica si held, in interoriul zonei critice.

Daca un proces doreste sa intre in zona critica, acesta realizeaza operatiunea de multicast a unui mesaj Lipi si asteapta pana la primirea unui raspuns de la oricare alt proces. In functie de stare, un proces va raspunde instantaneu la o cerere de intrare in zona critica daca e in starea released, va intarzia raspunsul pana la iesirea din zona cristica daca este held sau va raspunde instant doar daca timestampul cererii este mai mic decat cel al propriei cerei daca este in starea wanted. Algoritmul foloseste „ceasul Lamport”, necesita o metoda de asignare a timpilor catre evenimente si evenimentele locale sunt asignate in ordinea stricta a cresterii marcajelor de timp. Fiecare mesaj dintre procese e insotit de un marcaj de timp care indica timpul procesului care a trimis mesajul. Cand un mesaj este receptionat, marcajul sau de timp este comparat cu timpul local si in cazul in care acesta este mai mare, timpul local devine egal cu valoarea acestuia plus unu. Acest aspect impune o relatie „s-a intamplat inainte” intre evenimente.

In continuare, va fi prezentat pseudocodul algoritmului:

enterCS:

1. construct a request-to-enter message;
2. assign the current logical time to the request;
3. send the message to each other process;
4. wait for okay response from each other process;
5. receiveRequestMessage;

if this process is in the CS enqueue the request; else if this process is not waiting to enter the critical section send okay to requestor; else //this process is waiting to enter the critical section

if (this.request.timeStamp <incomingRequest.timeStamp)

enqueue the request;

else

send okay to requestor;

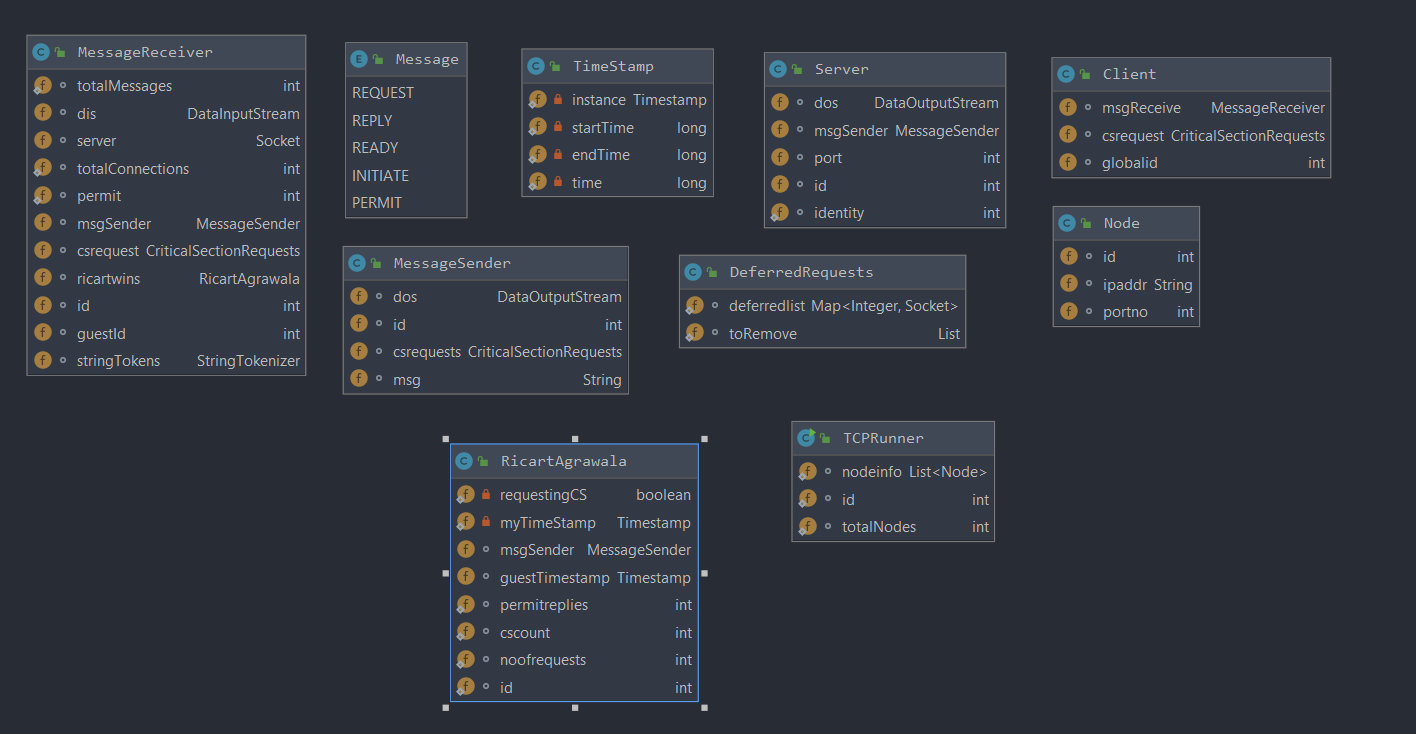
exitCS:

1. while(request queue not empty){

dequeue a request message;

send okay to requestor;

}Figura urmatoare prezinta diagrama de clase:



Algoritmul Maekawa presupune ca fiecare proces pi va avea o multime ce contine cereri, notata Ri, multime cu urmatoarele proprietati: intersectia lui Ri cu Rj nu trebuie sa fie vida, fiecare proces apartine multimii sale de cereri, fiecare multime are acelasi numar de elemente K si fiecare proces apartine la exact K multimi de cereri. Algoritmul foloseste trei tipuri de mesaje: request, ok si end. Daca un proces doreste sa intre in cona critica, el trebuie sa primeasca mesaje de tipul ok de la membrii multimii sala de solicitanti.

Pseudocodul algoritmului este prezentat in urmatoarea parte:

enterCS:

1. Send request message to each member of my request set.
2. Wait for okay messages from each member of my request set.

exitCS:

1. Send done message to ecah member of my request set.
2. receiveRequestMessage

grantedPi = false

if granted

enqueue the request;

else {

granted =true;

send okay to the requestor;

}

1. receiveDoneMessage:

if (queue is empty)

granted=false;

else{

dequeue the request with the earliest timestamp;

send okay top the requestor;

}

Algoritmul propus de Ricart si Agrawala foloseste 2(n-1) mesaje, avand nevoie doar de mesaje de n-1 mesaje de tipul request si n-1 mesaje de tipul de reply, in timp ce algoritmul Maekawa scade numarul de mesaje la 3\*radical(N), folosind trei tipuri de mesaje. Diferenta dintre cei doi algoritmi este data de tipul de coada folosita, algoritmul Ricart- Agrawala folosind o coada non-fifo, mesajele fiind plasate in ordine aleatoare, in timp ce algoritmul Maekawa foloseste o coada de tipul fifo.