Proiect final la materia Sisteme Distribuite

Tema 1

- Asynchronous Consensus and Broadcast Protocols -

1. Generalizare

Lucrarea „Asynchronous Consensus and Broadcast Protocols” de Gabriel Bracha si Sam Toueg scrisa in 1985 ofera o imagine de ansamblu detaliata a problemelor consensului si difuzarii in sistemele distribuite si prezinta un cadru general pentru rezolvarea acestora.

Problema consensului este definita ca problema de a ajunge la un acord asupra unei valori intre un grup de procese dintr-un sistem distribuit. Autorii noteaza ca aceasta problema este deosebit de provocatoare in sistemele asincrone in care comunicarea este supusa intarzierilor si defectiunilor. Ele prezinta un cadru general pentru rezolvarea problemei consensului, care implica utilizarea unei combinatii intre un protocol de consens si un detector de esec pentru a se asigura ca toate procesele sunt de acord asupra unei valori.

Autorii descriu apoi mai multe protocoale care pot fi utilizate pentru a rezolva problema consensului. Ei discuta, de asemenea, problema difuzarii, care este procesul de trimitere a unui mesaj catre toate nodurile din sistem si prezinta un protocol pentru rezolvarea acestuia. Ei incheie lucrarea discutand implicatiile rezultatelor lor si subliniind unele probleme deschise in acest domeniu.

In ansamblu, lucrarea ofera o tratare cuprinzatoare a problemelor consensului si difuzarii in sistemele distribuite si ofera solutii generale pentru rezolvarea acestora. Prezinta o baza teoretica pentru intelegerea provocarilor acestor probleme si ofera o perspectiva asupra proiectarii sistemelor distribuite. Lucrarea este considerata o contributie importanta in domeniul calculului distribuit si a fost citata pe scara larga in literatura.

1. Concepte de introductie : descrierea generala a problemei si a rezolvarii

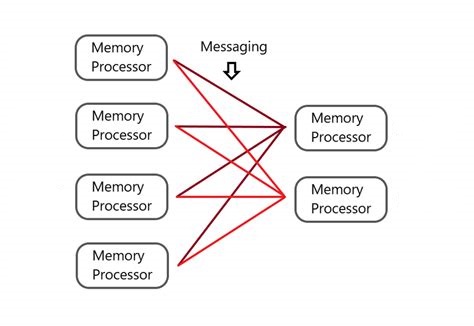
Un protocol de consesn permite un sistem de n procese asincrone, unele care sunt defecte, sa ajunga la un inteles. Sunt doua tipuri de defectiuni : procese fail-stop care doar mor si procese malicious care trimit mesaje false.

Pentru dezvoltarea protocoalelor descrise in aceasta lucrare au fost folosite metode probabilistice pentru sistemul de mesaje. In cazul fail-stop, protocolul poate rezista (n-1) / 2 defectiuni. In cazul proceselor malicious, protocolul poate rezista la (n-1) / 3 defectiuni. [*Aici n este numarul de procese totale*]

1. Descrierea detaliata a componentelor necesare pentru introducerea in cazul fail-stop

Modelul : pentru acest protocol consideram un sistem de n procese interconectate. Procesele comunica intre ele prin mesaje folosindu-se de un sistem de mesaje. Sistemul de mesaje mentine pentru fiecare proces un buffer de mesaje trimise dar nu primite. De asemenea suporta primitive pentru fiecare proces *q*. Acest lucru arata arata in pseudocod in modul urmator :

Send(p, m) - trimite un mesaj *m* in bufferul procesului *p*

Receive(m) - scoate un mesaj din bufferul procesului *q* si il returneaza in *m* daca poate daca nu returneaza valoare nula. 

Fiecare proces are un spatiu de depozitare intern numit *state*. Intr-un pas atomic (adica un pas care pare instant din perspectiva celorlalte procese) al sistemului, un proces poate primi un mesaj, performa un calcul si trimite un set finit de mesaje ca raspuns. Calculul si mesajele trimise sunt descrise de protocol care la randul lui este alcatuit din mesajele primite si state-ul local.

Un proces bun[1] urmareste acest protocol pana la finalul protocolului. Un caz fail-stop inseamna ca un proces moare (nu mai poate participa) in timpul executarii protocolului. Moartea unui proces nu are mesaj de eroare! Aceasta este cea mai mare problema, moartea nu poate fi detectata si nu poate fi deosebita dintr-un proces mort si unul incet.

Fiecare proces are 2 mari locatii de memorie : un input *ip* (input de proces p) si o decizie *dp* (dezicie de proces p). Un sistem incepe cu toate procese intr-un state oarecare, toate bufferurile goale, *dp* nedefinit si *ip* cu o valoare intre 0 si 1. O data ce *dp* ii este asignata o valoare v aceasta nu poate fi schimbata, astfel se spune ca *procesul p are decis v*. (mai exact procesul p are o valoare neschimbabila asignata de decizie)

Configuratia sistemului este colectia state-urilor si continutul bufferelor tuturor proceselor. Astfel vom denumi *C o configuratie* si *S un subset de procese.* O subconfiguratie *Cs* este o restrictie a lui C folosind doar membrii S. Notatia *Fi* denota orice configuratie unde toate procesele bune[1] au decis i. *FiS* denota orice subconfiguratie unde toate procesele bune in S au decis i.

O secventa de pasi atomici se numeste *schedule*. Daca o executie a unui schedule σ dintr-o configuratie C rezulta in configuratia D atunci notam acest lucru : C→D. Daca toate procese care urmeaza scheduleul σ apartin subsetului S atunci scriem CS→DS. Citim acesta operatie spunand D e *reachable* din C. In lucrarea “Asynchronous Consensus and Broadcast Protocols” este folosit clasa de schedule numita *fair schedule.*

Un protocol rezistent la k-procese rele satisface anumite conditii (cu conditia ca acesta sa nu depaseasca k procese rele) :

- *Bivalenta* : Daca toate procesele sunt bune, atunci condiguratiile F0 si F1 sunt reachable.

- *Consistenta* : Nu exista o configuratie reachable unde procese bune decid valori diferite.

- *Convergenta* : Pentru orice configuratie initiala, daca un proces nu a decis intr-un numar arbitrar de pasi t atunci rezultatul lui va fi 0. ( limt→∞proces = 0)

O limita inferioara a numarului de procese corecte : posibilitatea ca o moarte nedetectabila a unui proces in timpul executiei protocolului implica ca, la orice stadiu din protocol, procesele vor functiona pe baza unei informatii partial primite. Acest lucru duce la definitia urmatoarelor leme si teoreme :

Lema 1 : Cu un protocol rezistent la k-procese rele, pentru orice configuratie reacheable C si pentru orice substet S de procese care contin macar n - k procese bune (n - numar total de proces, k - numar procese rele) este adevarat fie CS→F0S sau CS→F1S.

Lema 2 : Pentru k>= 1, orice protocol de consens rezistent la k-procese rele are o configuratie initiala bivalenta.

Teorema 1 : Din Lema 1 si Lema 2 reiese ca nu exista un protocol rezistent la [n/2] procese rele pentru cazul fail-stop. Acest lucru este aplicat pentru orice tip de protocol, chiar si unul probabilistic sau nedeterministic. Astfel, cel mai bun protocol este unul rezistent la [n-1]/2 procese rele pentru cazul fail-stop.

Fair Schedulers : protocoalele pentru sisteme asincrone pot fi imaginate ca fiind create din mai multe runde. In fiecare runda *t* un proces poate trimite mesaje fiecarui alte proces si asteapta sa primeasca inapoi n - k mesaje inapoi (k aici inseamna procese rele aflate in modul fail-stop). Procesul dupa isi schimba stateul si incepe runda t+1.

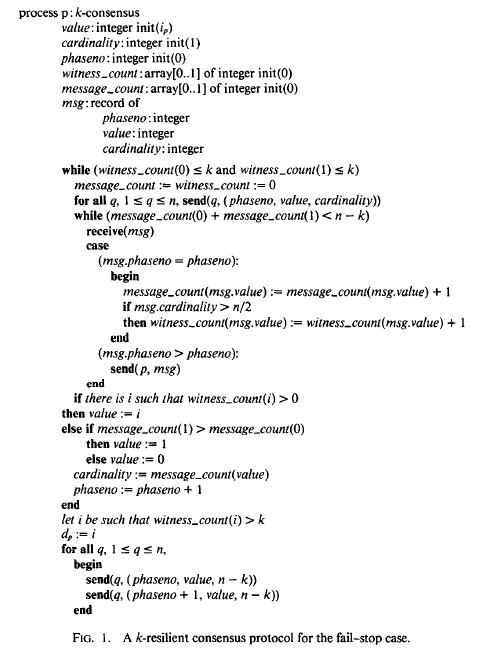
Avand un eveniment *R(q, p, t)* unde p primeste mesaje de la q in runda t, un scheduler fair are urmatoarele caracteristici:

1. Pentru orice proces p, q si runda t exista o constanta pozitiva ϵ astfel incat Pr[R(q, p, t)] > ϵ. (rezultatul unui proces trebuie sa fie mai mare decat constanta)
2. Pentru orice proces distinct r, p, q si runda t, evenimentele R(q, r, t) si R(q, p, t) sunt independente.
3. Descrierea detaliata a protocolului pentru cazul fail-stop

Protocolul va fi unul rezistent la (n-1) / 2 procese rele. Acesta contine runde precum cele mentionate mai sus. Atat stateul unui proces cat si mesajele schimbate intre procese contin un numar de faza, o valoare binara si o cardinalitate. In fiecare faza, un proces prima data trimite un mesaj cu stateul lui catre celelalte procese dupa care asteapta mesaje inapoi. Cand un proces primeste n - k, acesta numara mesajele cu valoare 0 respectiv 1. Un mesaj cu valoare i si cardinalitate mai mare decat n / 2 va fi numit un *martor* pentru i. Daca un proces primeste un martor pentru i acesta isi schimba valoarea in i. Altfel, isi schimba valoarea la i-ul cu cel mai mare set de mesaje. In ambele cazuri isi schimba cardinalitatea in marimea setului de i dupa care incepe o noua faza. Un proces decide i daca primeste mai multi de k martori pentru valoarea i. Acest lucru indica faptul ca sunt destui martori pentru acea valoare in sistemul de mesaje pentru a forta restul proceselor sa ajunga la aceiasi decizie.

Teorema 2 : Pentru orice k, 0<= k <= (n-1) / 2, protocolul descris mai jos este un protocol de consens k-rezistent pentru cazul fail-stop.

Pentru a demonstra ca procolul este rezistent la (n-1) / 2 procese rele se demonstreaza urmatoarele caracteristici : consistenta, lipsa de blocaje, convergenta si bivalenta.

Exemplu general pentru un astfel de protocol :

1. Descrierea in cuvinte proprii a problemei si a rezolvarii

Protocolul fail-safe descris in lucrarea "Asynchronous Consensus and Broadcast Protocols" de Gabriel Bracha si Sam Toueg este un sistem de protocol cu un grad ridicat de robustete si fiabilitate. Acesta asigura un consens asincron si un protocol de difuzare in conditii de esec si, prin urmare, poate fi utilizat in sisteme complexe pentru a preveni colapsul in cazul unui esec al unui singur proces. In acest mod, protocolul fail-safe contribuie la cresterea fiabilitatii sistemelor si la evitarea problemelor care pot surveni in cazul unui esec al acestora.

1. Descrierea detaliata a componentelor necesare pentru introducerea in cazul malicious

Modelul : un proces malicious poate trimite mesaje false sau contradictorii, poate sa nu trimita mesaje, isi poate schimba state-ul intern in oricare alt state. Un sistem de mesaje trebuie sa poata asigura o cale prin care se poate verifica identitatea fiecarui transmitator al fiecarui mesaj. Altfel, un singur caz malicious poate impersona tot sistemul, cauzand procese corecte sa ajunga la decizii contradictorii. Modelul se aseamana cu cel de la cazul fail-stop cu aditia ca un *schedule este legal* daca toti pasii lui sunt in conformitate cu protocolul si o configuratie C este reachable daca este reachable de un schedule legal.

O limita inferioara a numarului de procese corecte : posibilitatea ca informatiile primie gresite ale unui proces in timpul executiei protocolului implica ca, la orice stadiu din protocol, procesele vor functiona pe baza unei informatii partial primite sau complet gresite. Acest lucru duce la definitia urmatoarelor leme si teoreme :

Lema 3 : Pentru un protocol de consens rezistent la k-procese rele, pentru orice configuratie reachable C si pentru orice subset S de procese care contin cel putin n - k procese bune, fie CS→F0S sau CS→F1S.

Teorema 3 : Nu exista un protocol de consens rezistent la [n/3]-procese rele pentru cazul malicious.

1. Descrierea detaliata a protocolului pentru cazul malicious

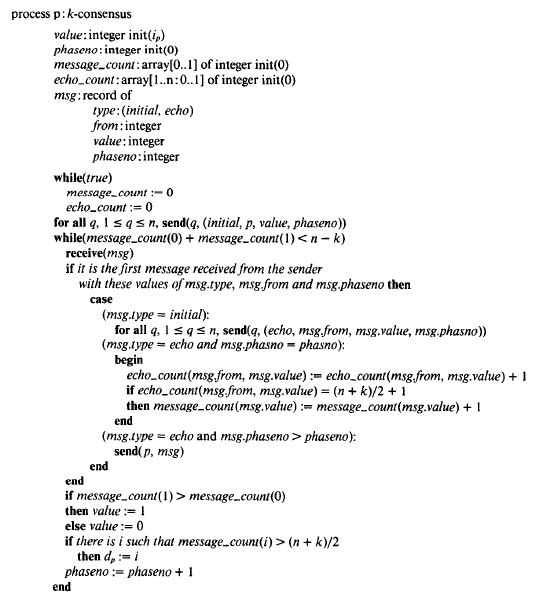
Protocolul va fi unul rezistent la (n-1) / 3 procese malicious. State-ul unui proces este alcatuit din un numar de faza si o valoare binara. Precum si la protocolul in cazul fail-safe, protocolul in cazul malicious este alcatuit din faze in care procesele isi trimit intre ele state-ul lor. Pentru a rezolva problema mesajelor false, informatia despre state este trimisa in felul urmator : exista 2 tipuri de mesaje : initial si echo. Un proces trimite tuturor celorlalte procese un mesaj initial cu numele si state-ul sau. La primirea mesajului initial, fiecare proces il transmite mai departe la celelalte procese. Procesul p, in faza t, accepta un mesaj cu valoarea i de la procesul q daca primeste mai mult de (n+k) / 2 mesaje in forma (echo, q, i, t).

Pe scurt, in fiecare faza, un proces isi trimite state-ul ( prin metoda descrisa mai sus) catre celelalte procese si asteapta pana primeste inapoi mesaje de la n - k procese. Dupa ce primeste mesajele, acesta isi schimba valoarea in cea majoritara reiesita din mesajele primite. Un proces decide i doar daca accepta mai mult de (n+k) / 2 mesaje cu valoarea i. Daca un proces decide i, atunci toate celelalte procese bune vor avea valoarea i.

Teorema 4 : Pentru orice k, 0<= k <= (n-1) / 3, protocolul descris mai jos este un protocol de consens k-rezistent pentru cazul malicious.

Pentru a demonstra ca procolul este rezistent la (n-1) / 2 procese rele se demonstreaza urmatoarele caracteristici : consistenta, lipsa de blocaje, convergenta si bivalenta.

Exemplu general pentru un astfel de protocol :



1. Descrierea in cuvinte proprii a problemei si a rezolvarii

Protocolul pentru cazul malicious abordeaza problema atacurilor intentionate la nivelul sistemelor de consens si difuzare. Acesta propune o metoda eficienta pentru detectarea si tratarea acestor atacuri, astfel incat sa se asigure un comportament corect al sistemului chiar si in prezenta unor comportamente malitioase. In acest mod, protocolul pentru cazul malitios contribuie la cresterea sigurantei sistemelor si la prevenirea problemelor cauzate de atacuri intentionate.

1. Acordul Bizantin Asincron

O problema majora a sistemelor distribuite este asigurarea de broadcasturi rezistente, fenomen cunoscut ca Acord Bizantin. Toate rezolvarile Acordului Bizantin pana la aceasta lucrare s-au ocupat cu un sistem sincron de n procese, unde k procese pot fi malicious. Un Acord Bizantin sincron se intampla atunci cand :

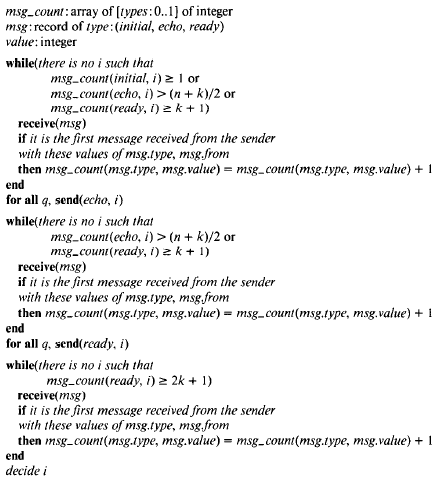
1. Toate procesele bune ajung la aceiasi valoare.
2. Daca transmitatorul este bun, toate procese bune ajung la valoarea lui.

Dar, cand consideram un sistem asincron, nu putem vizualiza astfel sistemul. Unele procese bune pot incepe folosind protocolul in timp ce unele nici nu stiu ca protocolul a inceput. Aceasta situatie poate fi rezolvata in 2 moduri : putem restrictiona comportamentul unui transmitator care este deja in cazul malicious (suficient sa il fortam sa transmita 2k + 1 mesaje cu aceiasi valoare) sau, cea prezentata mai jos, adoptarea unui fel anumit de a vizualiza sistemul ca fiind insuficient pentru a incepe protocolul. Procesele nu pot incepe decat daca sunt prezentate cu o vizualizare care garanteaza inceputul si intelegerea tuturor proceselor bune. Putem spune ca exista un Acord Bizantin asincron atunci cand :

1. Daca transmitatorul este bun atunci toate procesele bune decid valoarea acestuia.
2. Daca transmitatorul este malicious, atunci fie nici-un proces bun nu va decide sau toate procesele vor decide o valoare.

Teorema 5 : Este imposibil sa ajungi la un Acord Bizantin asincron cand k => n/3.

Teorema 6 : Imaginea de mai jos ajunge la un Acord Bizantin pentru (n-1) / 3 procese malicious.



1. Concluzie

In concluzie, lucrarea "Asynchronous Consensus and Broadcast Protocols" de Gabriel Bracha si Sam Toueg ofera o abordare sistematica a problemelor de consens si difuzare in sistemele asincrone. Autorii propun un set de protocoale eficiente si robuste care abordeaza problemele specifice acestor sisteme, cum ar fi esecul proceselor, comportamentele malitioase si asigurarea unui comportament corect in prezenta acestor probleme. Lucrarea ofera, de asemenea, o analiza a proprietatilor protocoalelor propuse si contribuie la intelegerea mai profunda a acestor probleme critice in sistemele asincrone. In ansamblu, aceasta lucrare reprezinta o contributie semnificativa la domeniul de cercetare a sistemelor de consens si difuzare.