Election

Master 2 Informatique - UFR S.A.T

Pr. Ousmane THIARE

ousmane.thiare@ugb.edu.sn
http://www.ousmanethiare.com/

10 mai 2024

1 Introduction

2 Election sur un anneau unidirectionnel

3 Election sur un arbre couvrant

4 Election dans un graphe quelconque

Chapitre 9 : Election

Introduction

Il se pose souvent le problème du choix d'un processus parmi N afin de résoudre un problème :

- remplacement d'un serveur : reprise sur panne par exemple ;

Or, suivant le problème à résoudre, les capacités d'un processus ou site à résoudre le problème peuvent changer : par exemple, un serveur NFS peut être très capable de résoudre le problème de la panne d'un serveur de fichier. Par contre, sa capacité à déterminer si une application est terminée, peut être nulle.
Donc, lorsque le choix doit être fait, il faut tenir compte du type de problème à résoudre. On suppose donc, que pour chaque type de problème, chaque processus Pi dispose d'une capacité ci à répondre à un problème de ce type.
On impose qu'il existe une relation d'ordre sur les capacités soit par $c_i < c_j$ si $i < j$ soit par $(c_i, i) < (c_j, j)$ si

Introduction

Il se pose souvent le problème du choix d'un processus parmi N afin de résoudre un problème :

- remplacement d'un serveur : reprise sur panne par exemple ;
- choix d'un initiateur d'un algorithme quelconque ;

Or, suivant le problème à résoudre, les capacités d'un processus ou site à résoudre le problème peuvent changer : par exemple, un serveur NFS peut être très capable de résoudre le problème de la panne d'un serveur de fichier. Par contre, sa capacité à déterminer si une application est terminée, peut être nulle.

Donc, lorsque le choix doit être fait, il faut tenir compte du type de problème à résoudre. On suppose donc, que pour chaque type de problème, chaque processus Pi dispose d'une capacité ci à répondre à un problème de ce type.

On impose qu'il existe une relation d'ordre sur les capacité soit par $c_i < c_j$ si $i < j$ soit par $(c_i, i) < (c_j, j)$ si

Introduction

Lors du moment de faire ce choix deux possibilités : soit l'utilisateur intervient directement ou a fixé a priori le processus à choisir, soit il a autorisé l'application à choisir elle-même ce processus en fonction des capacités de ceux-ci à résoudre le problème qui se pose. Pour cela, on utilise des algorithmes d'*élection*.

Le site initiateur P_{i_0} transmet le message $El(c_{i_0}, i_0, i_0)$
Un site $P_{j \neq i_0}$ qui reçoit le message $El(c_i, i, i_0)$ transmet $El(c_j, j, i_0)$ si $c_i < c_j$ au processus suivant sur l'anneau, sinon il transmet $El(c_i, i, i_0)$.
Lorsque P_{i_0} reçoit $El(c_i, i, i_0)$, il élit le processus $P_i \Rightarrow$ phase de proclamation $\Rightarrow$ message Pl(i).

Phase 1 - Diffusion de la demande des capacités

- init : P_{i_0} effectue :

Phase 1 - Diffusion de la demande des capacités

- init : P_{i_0} effectue :
  - (cmax,imax) = (c_{i_0}, i_0) ; N_{fils} = nombre de fils

Phase 1 - Diffusion de la demande des capacités

- init : P_{i_0} effectue :
  - (cmax,imax) = (c_{i_0}, i_0) ; N_{fils} = nombre de fils
  - SI $N_{fils} > 0$, P_{i_0} émet le message *ElDemande*() à l'ensemble de ses fils
    SINON Proclamation(cmax, imax)

Phase 1 - Diffusion de la demande des capacités

- init : P_{i_0} effectue :
  - (cmax,imax) = (c_{i_0}, i_0) ; N_{fils} = nombre de fils
  - SI $N_{fils} > 0$, P_{i_0} émet le message *ElDemande*() à l'ensemble de ses fils
    SINON Proclamation(cmax, imax)
- Lorsqu'un site P_i reçoit le message *ElDemande*()

Phase 1 - Diffusion de la demande des capacités

- init : P_{i_0} effectue :
  - (cmax,imax) = (c_{i_0}, i_0) ; N_{fils} = nombre de fils
  - SI $N_{fils} > 0$, P_{i_0} émet le message *ElDemande*() à l'ensemble de ses fils
    SINON Proclamation(cmax, imax)
- Lorsqu'un site P_i reçoit le message *ElDemande*()
  - (cmax,imax) = (c_i, i) ; N_{fils} = nombre de fils

Phase 1 - Diffusion de la demande des capacités

- init : P_{i_0} effectue :
  - (cmax,imax) = (c_{i_0}, i_0) ; N_{fils} = nombre de fils
  - SI $N_{fils} > 0$, P_{i_0} émet le message *ElDemande*() à l'ensemble de ses fils
    SINON Proclamation(cmax, imax)
- Lorsqu'un site P_i reçoit le message *ElDemande*()
  - (cmax,imax) = (c_i, i) ; N_{fils} = nombre de fils
  - SI $N_{fils} > 0$, P_i émet le message *ElDemande*() à l'ensemble de ses fils
    SINON il transmet *REP(cmax, imax)* à son père

Phase 2 - Remontée des informations

- Lorsqu'un site P_i, reçoit le message $REP(c_j, j)$ d'un de ses fils, il effectue :

Phase 2 - Remontée des informations

- Lorsqu'un site P_i, reçoit le message $REP(c_j, j)$ d'un de ses fils, il effectue :
  - (cmax, imax) = $max((c_j, j), (cmax, imax))$; $N_{fils} = N_{fils} - 1$
    SI $N_{fils} == 0$
    SI P_i n'a pas de père alors Proclamation(cmax, imax)
    SINON Pi transmet REP (cmax, imax) à son père
    SINON NOP

Phase 3 - Proclamation :

- Proclamation(cmax, imax) :

Phase 3 - Proclamation :

- Proclamation(cmax, imax) :
  - P_{i_0} émet le message ELU(cmax,imax) vers tous ses fils SI N_{fils} = nombre de fils > 0 SINON FIN

Phase 3 - Proclamation :

- Proclamation(cmax, imax) :
  - P_{i_0} émet le message ELU(cmax,imax) vers tous ses fils SI N_{fils} = nombre de fils > 0 SINON FIN
- Lorsqu'un site P_i, reçoit le message ELU(cmax,imax)

Introduction
Election sur un anneau unidirectionnel
Election sur un arbre couvrant
Election dans un graphe quelconque

Phase 3 - Proclamation :

- Proclamation(cmax, imax) :
  - P_{i_0} émet le message ELU(cmax,imax) vers tous ses fils SI N_{fils} = nombre de fils > 0 SINON FIN
- Lorsqu'un site P_i, reçoit le message ELU(cmax,imax)
  - il mémorise l'élu ; N_{fils} = nombre de fils

Phase 3 - Proclamation :

- Proclamation(cmax, imax) :
  - P_{i_0} émet le message ELU(cmax,imax) vers tous ses fils SI N_{fils} = nombre de fils > 0 SINON FIN
- Lorsqu'un site P_i, reçoit le message ELU(cmax,imax)
  - il mémorise l'élu ; N_{fils} = nombre de fils
  - SI $N_{fils} > 0$, P_i transmet ce message ELU(cmax,imax) à l'ensemble de ses fils
    SINON il transmet OK à son père

Phase 3 - Proclamation :

- Lorsqu'un site P_i, reçoit le message OK d'un de ses fils, il effectue :

Phase 3 - Proclamation :

- Lorsqu'un site P_i, reçoit le message OK d'un de ses fils, il effectue :
  - $N_{fils} = N_{fils} - 1$
    SI $N_{fils} == 0$
        SI P_i n'a pas de père alors FIN
        SINON Pi transmet OK à son père
    SINON NOP

Utilisation de l'algorithme précédent

P_{i_0} demande à la racine de faire l'élection [1] ce qui revient en gros à l'algorithme précédent, avec une phase d'initialisation de l'algorithme.

Phase 0 - Initialisation de la demande d'élection : P_{i_0} émet le message InitEL() vers son père. Lorsqu'un site autre que la racine reçoit le message InitEL() il transmet ce message à son père. Lorsque la racine reçoit ce message, elle effectue l'algorithme précédent.

1. ce qui revient à dire implicitement, que la capacité à résoudre le problème de l'élection est maximale pour la racine !

Principe d'un algorithme plus général

Phase 1 - Initialisation

P_{i_0} émet le message ElDemande() vers tous ses fils et vers son père :

Lorsqu'un site non feuille reçoit le message ElDemande()

- de son père, il passe à l'état descendant et il transmet ce message à l'ensemble de ses fils
- d'un de ses fils, il passe à l'état ascendant et il transmet ce message a ? son père et à l'ensemble de ses fils excepté au processus fils P_f qui lui a transmis ce message.

Lorsqu'un noeud feuille P_i reçoit le message ElDemande(), il transmet à son père le message $El(c_i, i)$.

Principe d'un algorithme plus général

Phase 2 - Remontée des informations

Lorsqu'un site descendant $P_{j \neq i_0}$ a reçu tous les messages $El(c_i, i)$ de ses fils, il transmet à son père le message $El(c_k, k)$ tel que $c_k = max(\{c_i\}, c_j)$

Lorsqu'un site ascendant $P_{j \neq i_0}$ a reçu le message $El(c_p, p)$ de son père et $El(c_i, i)$ de tous ses fils excepté P_f il transmet à son fils P_f le message $El(c_k, k)$ tel que $c_k = max(\{c_i\}, c_p, c_j)$

Lorsque la racine P_{racine} a reçu le message $El(c_i, i)$ de tous ses fils excepté P_f il transmet à son fils P_f le message $El(c_k, k)$ tel que $c_k = max(\{c_i\}, c_{racine})$

Principe d'un algorithme plus général

Phase 3 - Élection

Lorsque le site P_{i_0} a reçu le message $El(c_p, p)$ de son père et $El(c_i, i)$ de tous ses fils, il élit P_k tel que $c_k = max(\{c_i\}, c_p, c_{i_0})$

Phase 4 - Proclamation

P_{i_0} proclame le résultat par l'émission du message ELU(k) vers son père et vers l'ensemble de ses fils

Lorsqu'un site reçoit le message ELU(k) , il mémorise l'information puis s'il est non feuille :

s'il a eu le message de son père, il transmet ce message ELU(k) à l'ensemble de ses fils

sinon il transmet ce message à son père et à l'ensemble de ses fils excepté au processus fils P_f qui lui a transmis le message ELU(k).

On se place dans le cas d'un graphe de communication FIFO à liaisons bidirectionnelles.
On suppose disposer d'un algorithme de diffusion stable.
Principe de "Bully algorithm" de Garcia-Molina
Le site initiateur Pi0 diffuse le message $El(c_{i_0}, i_0)$ à tous les processus . Puis il attend.
A la réception d'un $El(c_k, k)$ un processus P_i

- répond Ack(i) à P_k si sa capacité ci est inférieure ou égale à c_k.

On se place dans le cas d'un graphe de communication FIFO à liaisons bidirectionnelles.

On suppose disposer d'un algorithme de diffusion stable.

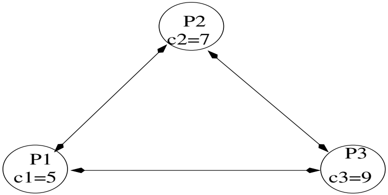
Principe de "Bully algorithm" de Garcia-Molina

Le site initiateur Pi0 diffuse le message $El(c_{i_0}, i_0)$ à tous les processus . Puis il attend.

A la réception d'un $El(c_k, k)$ un processus P_i

- répond Ack(i) à P_k si sa capacité ci est inférieure ou égale à c_k.
- émet $El(c_i, i)$ en diffusion si sa capacité ci est strictement supérieure à c_k.

Si tous les processus répondent par un Ack à P_{i_0}, alors de façon évidente P_{i_0} à la meilleure capacité et donc il passe dans l'état Elu, et peut proclamer le résultat. Par contre, si un site P_i répond par $El(c_i, i)$ c'est que sa capacité est supérieure (P_i "prend le pouvoir") et donc, P_{i_0} arrête d'attendre et doit répondre un acquittement.
Dans l'exemple suivant :



un déroulement possible est :

Site	réception	action
P1	–	diffuse El(5,1)
P2	El(5,1)	diffuse El(7,2)
P3	El(5,1)	diffuse El(9,3)
P1	El(7,2)	Ack(1)––––––>P2
P1	El(9,3)	Ack(1)–––––––>P3
P2	Ack(1)	–
P3	Ack(1)	–
P2	El(9,3)	Ack(2)–––––––> P3
P3	El(7,2)	diffuse El(9,3)
P1	El(9,3)	Ack(1)–––––––>P3
P2	El(9,3)	Ack(2)–––––––> P3
P3	Ack(2)	Proclame
P3	Ack(1)	
P3	Ack(2)	re–proclame

Il y a 13 messages échangés (hors proclamation). En fait, on voit que le pire des cas est quand les diffusions se font dans l'ordre croissant des capacités (si c'était P_3 qui initiait l'élection le nombre de message échangés serait de 4).
Malheureusement, on ne peut rien faire pour le choix de l'initiateur. Par contre, on voit que P_3 diffuse deux fois sa demande. On peut limiter le nombre de messages en transit par "si lorsqu'un site à déjà émis un El , il ne le fait plus".

L'algorithme

D'où, le nouvel algorithme par :

Initialement, tous les P_i sont dans l'état *Peut-être*

Le site initiateur P_{i_0} diffuse le message $El(c_{i_0}, i_0)$ à tous les processus et passe à l'état Enquête. A la réception d'un $El(c_k, k)$, un processus P_i :

- s'il est dans l'état *Peut-être* :
  - il répond Ack(i) à P_k si sa capacité ci est inférieure ou égale à c_k.
  - il émet $El(c_i, i)$ en diffusion si sa capacité ci est strictement supérieure à c_k et passe dans l'état *Enquête*
- s'il est dans l'état Enquête :
  - il ignore ce message si sa capacité ci est supérieure à c_k.
  - il répond Ack(i) à P_k si sa capacité ci est inférieure ou égale à c_k et passe dans l'état Perdu [2]

2. Il ne pourra plus être élu : il peut jeter tous les Ack qu'il a déjà

L'algorithme

s'il est dans l'état Perdu :

il répond Ack(i) à P_k si sa capacité ci est inférieure ou égale à c_k.

Ce qui nous donne sur l'exemple :

Site	état	réception	nouvel état	action
P1	Peut etre	–	Enquete	émet El(5,1)
P2	Peut etre	El(5,1)	Enquete	émet El(7,2)
P3	Peut etre	El(5,1)	Enquete	émet El(9,3)
P1	Enquete	El(7,2)	Perdu	Ack(1)––––> P2
P1	Perdu	El(9,3)	Perdu	Ack(1)––––> P3
P2	Enquete	Ack(1)	Enquete	–
P3	Enquete	Ack(1)	Enquete	–
P2	Enquete	El(9,3)	Perdu	Ack(2)––––––>
P3	Enquete	El(7,2)	Enquete	
P3	Enquete	Ack(2)	Elu	Proclame

Coût de l'algorithme

Le pire des cas est le cas où les demandes d'élection est faite par le processus de plus petite capacité car cela peut entraîner une demande d'élection de tous les sites si ceux-ci la reçoivent avant tout autre message. Ce qui nous donne :

- N - 1 messages (diffusion initiale)

Coût de l'algorithme

Le pire des cas est le cas où les demandes d'élection est faite par le processus de plus petite capacité car cela peut entraîner une demande d'élection de tous les sites si ceux-ci la reçoivent avant tout autre message. Ce qui nous donne :

- N - 1 messages (diffusion initiale)
- Les N - 1 autres sites émettent alors en diffusion, c'est-à-dire, chacun N - 1 messages, soit au total : (N - 1)(N - 1) messages

Coût de l'algorithme

Le pire des cas est le cas où les demandes d'élection est faite par le processus de plus petite capacité car cela peut entraîner une demande d'élection de tous les sites si ceux-ci la reçoivent avant tout autre message. Ce qui nous donne :

- N - 1 messages (diffusion initiale)
- Les N - 1 autres sites émettent alors en diffusion, c'est-à-dire, chacun N - 1 messages, soit au total : (N - 1)(N - 1) messages
- Enfin, il y a $0 + 1 + 2 + \cdots + (N - 1)$ acquittements, soit N(N-1) messages.

Coût de l'algorithme

D'où au total $\dfrac{3N(N-1)}{2}$ messages.

A noter qu'il serait plus simple que P_{i_0} demande à chaque site sa capité et calcule le maximum à partir de ces réponses : cout en (N-1)

On suppose que l'on ne dispose pas d'un algorithme de diffusion atomique fiable : un processus ne connaît que les processus avec qui il est directement connecté dans le graphe de communication. Il connaît par contre le nombre N de processus.

Principe Le site initiateur va chercher à calculer la capacité maximale des sites qu'il connait. Il va donc leur demander directe- ment celles-ci.

Phase 1 - Diffusion de la demande des capacités
Ces sites vont essayer de faire la même chose (forme de récursivité). Problème : les demandes risquent de se diffuser sans limite. En effet, un site peut recevoir une demande par différents canaux. Pour résoudre cela, un site ne refera jamais une demande. Lorsqu'il recevra une demande supplémentaire, il répondra "NOK". En effet, se capacité sera prise en compte dans la réponse qu'il fera à son premier demandeur. Elle n'a plus besoin d'être prise en compte par ce demandeur supplémentraire.

Phase 2 - Remontée des informations
Lorsqu'un site aura reçu une réponse (soit une capacité soit un NOK) de tous ses voisins, il répondra la capacité maximale à celui qui lui a fait la demande en premier.
Phase 3 - Proclamation :
Le site initiateur enverra le proclamation à tous les sites qui connait. Par contre, un site n'enverra cette proclamation qu'au sites qui ne lui ont pas répondu "NOK".

- Coût : Lors du calcul de la capacité maximale, chaque site émet une demande par site qu'il connait et reçoit une réponse. Le coût est donc égal au nombre M de liens entre les sites. Pour la proclamation, il n'y a plus que (N-1) liens "actifs" car on obtient une structure d'arbre. D'où un coût total de (M+N-1). A noter que $M \leq N^3$

3. il ne pourra plus être élu : il peut jeter tous les Ack qu'il a déjà reçu