Algorithmique Distribuée
Élection distribuée

Laurent PHILIPPE

Master 2 Informatique
UFR des Sciences et Techniques

2013/2014

Sommaire

Références

- Distributed Systems, Principle and Paradigms, A.Tanenbaum, Pearson Ed.
- Une introduction aux principes des systèmes répartis (3 tomes), M. Raynal, EyrollesEd. (1991)
- Cours de C.Kaiser du Cnam.
- Introduction to Distributed Algorithms, G. Tel, Cambridge University Press

Algorithmes d'élection

Objectif

- Elire un processus parmi d'autres

Pourquoi ?

- En AD : algorithmes utilisent un coordinateur ou initiateur
- Exemple : algorithme d'exclusion mutuelle centralisé
- Le plus souvent : coordinateur = processus de plus grand numéro
- Panne du coordinateur :
  nommer un nouveau coordinateur $\rightarrow$ élection

Algorithmes d'élection : systèmes sans mobilité, ni panne

Hypothèses

- Chaque processus a un numéro unique (ex : adresse réseau)
- Chaque processus connaît le numéro de tous les autres processus
- Mais un processus ne sait pas qui est actif et qui ne l'est pas
- Les réponses aux messages sont reçues au bout d'un temps borné par *TEMPO*
- Condition déclanche une élection
- La condition devient vraie sur 1 ou plus processus, pas forcément tous (par exemple le coordinateur ne répond plus)

Algorithmes d'élection

Exercice

Proposer un algorithme d'élection

Algorithmes d'élection

Différentes topologies

- Arbre ou arbre de recouvrement
- Anneau
- Quelconque

Sommaire

Algorithme du plus fort ou *Bully algorithm* (Garcia-Molina)

Déclanchement :

Quand un processus P s'aperçoit que le coordinateur ne répond plus à ses requêtes (time-out sur *TEMPO*), il lance l'algorithme d'élection

Lancement d'une élection par P :

Envoi d'un message `ELECTION` à tous les autres processus dont le numéro est plus grand que le sien

Réception d'un message `ELECTION` depuis P par un processus Q :

- Le processus Q envoie un message `ACK` à P lui signifiant qu'il est actif
- A son tour Q, lance une élection si ce n'est pas déjà fait

Algorithme du plus fort ou *Bully algorithm*

Sur le processus P :

- Si aucun processus ne lui répond avant *TEMPO*, P gagne l'élection et devient le coordinateur
- Si un processus de numéro plus élevé répond, c'est lui qui prend le pouvoir. Le rôle de P est terminé.

Annonce de l'élu

Le nouveau coordinateur envoie un message à tous les participants pour les informer de son rôle. L'application peut alors continuer à s'exécuter

Algorithme du plus fort ou *Bully algorithm*

Réveil d'un processus inactif

- Déclenche une élection
- S'il détient le plus grand numéro de processus en cours de fonctionnement, il gagne l'élection et devient le nouveau coordinateur

Algorithme du plus fort ou *Bully algorithm*

Déroulement de l'algorithme

- Application composée de 8 processus numérotés de 0 à 7
- Processus de numéro 7 tombe en panne, Processus de numéro 4 est le premier à détecter cette panne

Sommaire

Algorithme pour un arbre

Principe

- Un (ou plusieurs) processus détecte la panne du coordinateur
- Il informe l'ensemble de processus du début de l'algorithme avec un message $< wakeup >$ car la seconde partie de l'algorithme doit être initié par toutes les feuilles
- Lorsque le message $< wakeup >$ a parcouru tout l'arbre l'élection commence
- les feuilles émettent un message
- le processus ayant le plus grand numéro est élu

Algorithme pour un arbre

Variables

var ws_p : booléen **init** faux *(* ws_p est vrai si p est réveillé *)*
wr_p : integer **init** 0 *(* compte les messages de reveil reçus *)*
$req_p[q], \forall q \in Neigh_p$: booléen **init** faux
(vrai si p a reçu un message de q *)*
v_p : numéro de processus **init** p *(* plus grand processus *)*
$etat_p$: (*sleep*, *leader*, *lost*) **init** *sleep*
V_p : voisins du processus

Algorithme pour un arbre

Partie réveil :

début

- **si** p *est initiateur* **alors**
  - $ws_p := \text{true}$
  - **pour** $q \in Neigh_q$ **faire**
    - envoie $< wakeup >$ à q
- **tant que** $wr_p < \sharp V_p$ **faire**
  - reçoit $< wakeup >$
  - **si** $ws_p = faux$ **alors**
    - $ws_p := \text{true}$
    - **pour** $q \in Neigh_q$ **faire**
      - envoie $< wakeup >$ à q

fin

Algorithme pour un arbre

Partie election :

début

- **tant que** $\sharp(q : req_p[q] = faux) > 1$ **faire**
  - reçoit $< tok, r >$ de q
  - $rec_p[q] := vrai$
  - $v_p := \max(v_p, r)$
- envoie $< tok, v_p >$ à q_0 tel que $rec_q[q_0]$ est faux
- reçoit $< tok, v_p >$ de q_0
- $v_p := \max(v_p, r)$
- **si** $v_p = p$ **alors**
  - $state_p :=$ leader
- **sinon**
  - $state_p :=$ lost
- **pour** $q \in Neigh_p, q \neq q_0$ **faire** envoie $< tok, v_p >$ à q

fin

Sommaire

Un algorithme pour anneau

Structure d'anneau

- Processus sont organisés en anneau virtuel
- Chaque processus connaît la structure de l'anneau (donc son successeur $Suivant_p$)
- Comme pour l'arbre l'anneau peut être virtuel, et pour l'application, les processus communiquent entre eux directement, sans passer par l'anneau

Un algorithme pour anneau

Exercice

- Proposer un algorithme pour un anneau
- Quelle est sa complexité ?

Algorithme d'élection de LeLann

Hypothèses et Principe

- Les canaux de communication sont unidirectionnels et FIFO
- Un processus initiateur (qui a détecté une panne) envoie un jeton contenant son identité
- Un initiateur génère un jeton avant d'en avoir reçu un
- Chaque processus initiateur calcule une liste des initiateurs
- Lorsqu'un processus reçoit un jeton sans en avoir produit un il n'en initie pas un nouveau
- Lorsqu'un initiateur reçoit son propre jeton, il a déjà reçu tous les jetons des autres initiateurs, il calcule donc le vainqueur

Algorithme d'élection de LeLann

var $Liste_p$ sous-ensemble de P **init** $\{p\}$
$etat_p$ ($init$, $leader$, $perdu$, $sleep$)

début
- **si** p *est intiateur* **alors**
  - $etat_p := init$; envoie $< tok, p >$ à $Suivant_p$
  - reçoit $< tok, q >$
  - **tant que** $q \neq p$ **faire**
    - $Liste_p := Liste_p \cup \{q\}$
    - envoie $< tok, q >$ à $Suivant_p$
    - reçoit $< tok, q >$
  - **if** $p = max\ (Liste_p)$ **then** $etat_p := leader$ **else** $etat_p := perdu$
- **sinon**
  - **tant que** *vrai* **faire**
    - reçoit $< tok, q >$
    - envoie $< tok, q >$ à $Suivant_p$
    - **if** $etat_p = sleep$ **then** $etat_p := perdu$

Algorithme d'élection de LeLann

Exercice

Déroulement de l'algorithme

- Application composée de 8 processus numérotés de 0 à 7
- Processus 7 est initialement le coordinateur
- Coordinateur tombe en panne, Processus 2 détectent cette panne

Algorithme d'élection de LeLann

Questions

- Comment gère-t-on plusieurs élections simultanées ?
- Comment un processus qui reprend son activité connaît le coordinateur ?

Algorithme d'élection de LeLann

Propriétés

- Complexité de l'algorithme :
  - Nombre de messages : $O(N^2)$ (N initiateurs * N processus dans l'anneau)
  - Temps : $2N - 1$
- Finaliser par un dernier message pour annoncer la fin de l'élection aux perdants, les initiateurs eux connaissent le leader

Algorithme d'élection de Chang et Roberts

Hypothèses et Principe

- Evolution de l'algorithme de Le Lann
- Supprime les jetons des processus qui ne sont pas élus : ceux qui ont un numéro de processus plus petit
- Un initiateur perd quand il reçoit un jeton qui porte un numéro supérieur
- Un processus devient leader lorsqu'il reçoit son jeton

Algorithme d'élection de Chang et Roberts

var $etat_p$

début

  si p *est intiateur* **alors**

    $etat_p$:= cand ; envoie $< tok, p >$ à $Suivant_p$

    tant que $etat_p \neq leader$ **faire**

      reçoit $< tok, q >$

      if $q = p$ **then** $etat_p$:= leader

      else

        if $q > p$ **then**

          if $etat_p = cand$ **then** $etat_p$:= perdu

          envoie $< tok, q >$ à $Suivant_p$

  else tant que *vrai* **faire**

    reçoit $< tok, q >$

    envoie $< tok, q >$ à $Suivant_p$

    if $etat_p = sleep$ **then** $etat_p$:= perdu

Algorithme d'élection Chang et Roberts

Exercice

Déroulement de l'algorithme

- Application composée de 8 processus numérotés de 0 à 7
- Processus 7 est initialement le coordinateur
- Coordinateur tombe en panne, Processus 2 détectent cette panne

Algorithme d'élection de Chang et Roberts

Fin de l'algorithme

- Les processus avec un état *perdu* sont en attente infinie
- Le processus *leader* doit terminer l'élection en envoyant un message *elu* aux autres participants
- Qaund un processus participant reçoit un message *elu* il sort de la boucle $tantqueetat_p \neq leader$, il enregistre le leader et il fait suivre le message
- Qaund le *leader* reçoit le message *elu* il l'efface et l'élection est finie.

Algorithme d'élection de Dolev, Klawe et Rodeh

Hypothèses et principe

- Hypothèses de l'anneau
- Calcul du plus grand/petit numéro de processus
- Initialement tous les processus sont actifs
- A chaque round chaque processus compare son numéro avec ses voisins
- Si son numéro est plus petit que l'un de ses voisins (sens de la montre et opposé), passe à l'état passif
- A moins la moitié des identités actives ne survivent pas à un round $\rightarrow$ après au plus $logN$ round l'élection est finie
- MAIS : nécessite un anneau bi-directionnel ce qui n'est pas dans les hypothèses de départ.

Algorithme d'élection de Dolev, Klawe et Rodeh

Principe pour un anneau unidirectionnel

- Les processus passifs ne font que relayer les messages, quelque soit leur type
- Un processus actif envoie son numéro au prochain processus actif (relayé par les processus passifs) dans un message $< one >$
- Il obtient donc, dans le premier message reçu, le numéro du processus qui le précède
- Il envoie ensuite un second message $< two >$ avec le numéro reçu
- Son successeur reçoit ainsi le numéro du processus actif qui le précède à distance 2

Algorithme d'élection de Dolev, Klawe et Rodeh

Principe pour un anneau unidirectionnel

- Il compare les valeurs reçues et si la valeur reçue dans le message $< one >$ n'est supérieure à la sienne et à celle reçue dans $< two >$ alors il devient passif. Sinon il prend l'identité du voisin, car il ne peut pas dire aux précédents qui a gagné, et continue l'algorithme
- Lorsqu'un processus reçoit son identité dans un message $< one >$, il est le *leader* et difuse un message $< elu >$ pour en informer les autres.

Sommaire

Algorithme basé sur l'extinction

Principe

- Les processus initiateurs diffusent des messages avec leur numéro de processus,
- Les messages sont relayés par les récepteurs (voisins)
- Les messages parcourent le réseau et reviennent
- Seul le parcous du plus grand va finir, les autres sont arrêtés
- = Bully sur réseau quelconque

Algorithme basé sur l'extinction

Principe

- Chaque processus initiateur p diffuse à ses voisins un message marqué par son numéro p
- Chaque processus définit la plus grande valeur reçue, initialement la sienne
- Quand un message arrive, si la valeur contenue est inférieure à la valeur locale, le message est ignoré, si elle est supérieure le processus abandonne sa propre diffusion et enregistre la nouvelle valeur reçue comme la plus grande, si la valeur est égale à la valeur locale alors le processus est le leader.
- Complexité messages : N (nombre processus) * M (nombre messages par processus)