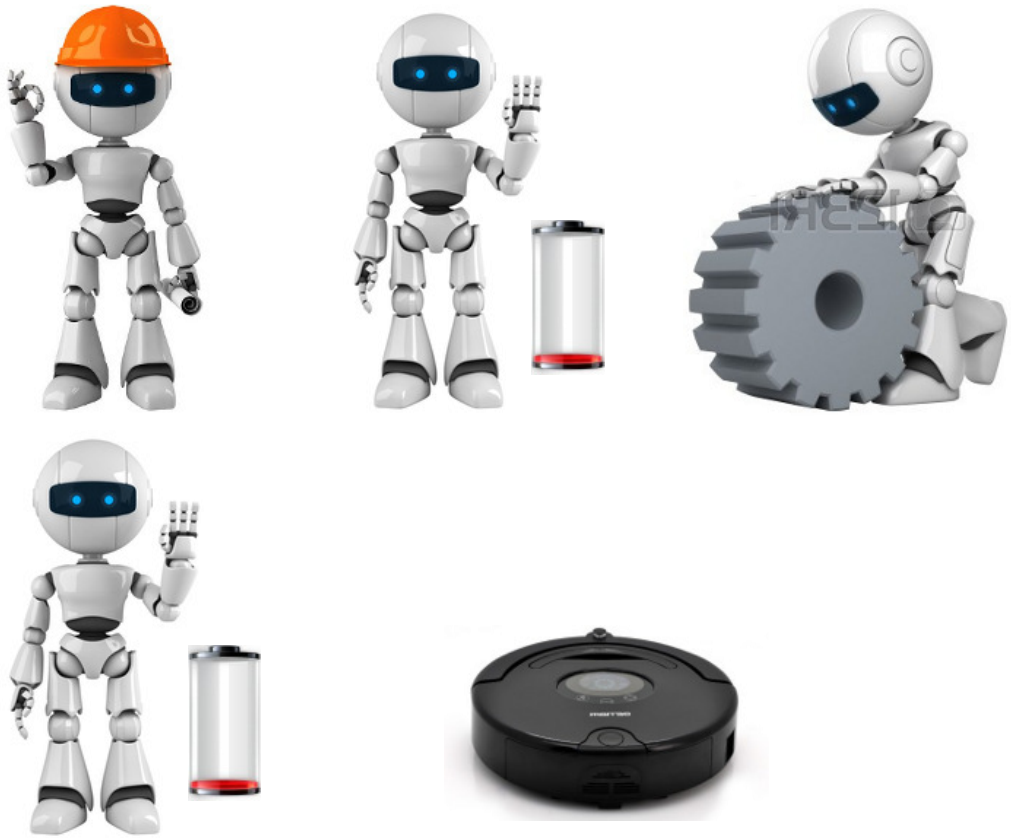
Algorithmique Distribuée Avancée
Élection distribuée

Laurent PHILIPPE

Master 2 Informatique
UFR des Sciences et Techniques

2020/2021

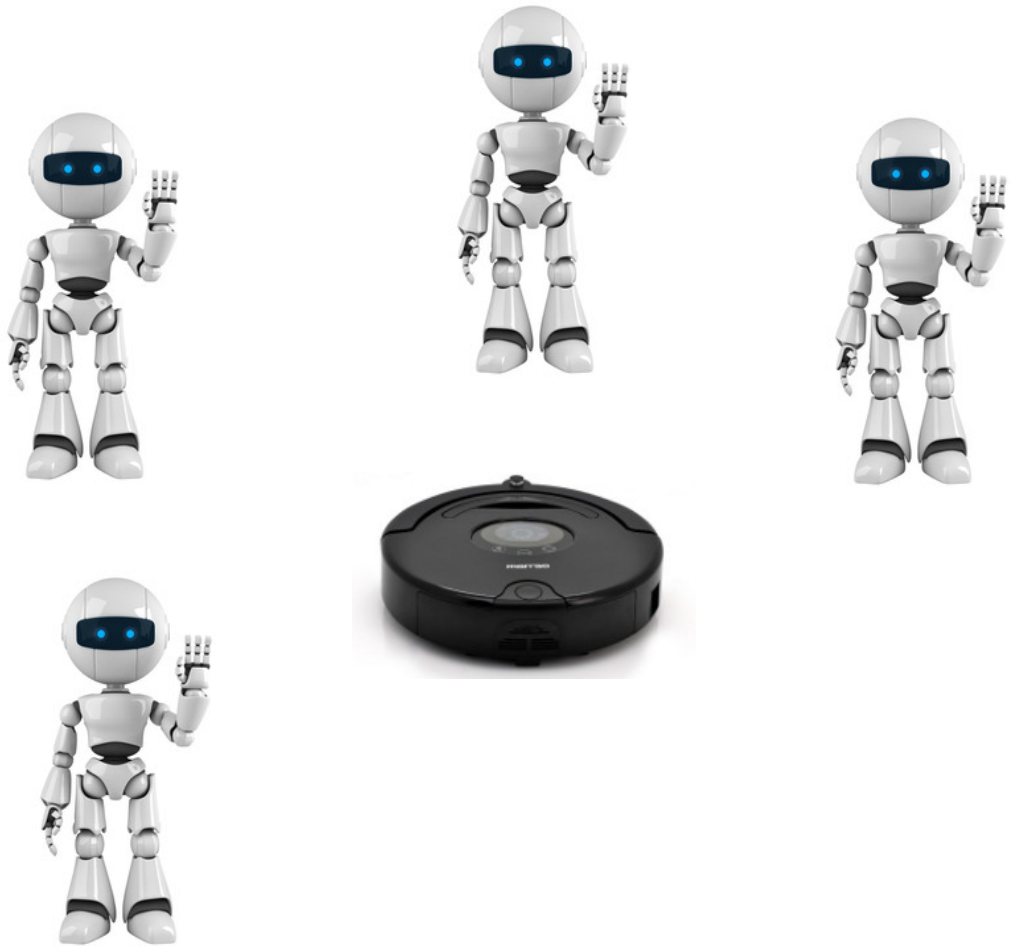
Algorithme centralisé des robots



Un robot maître décide qui accède à la base

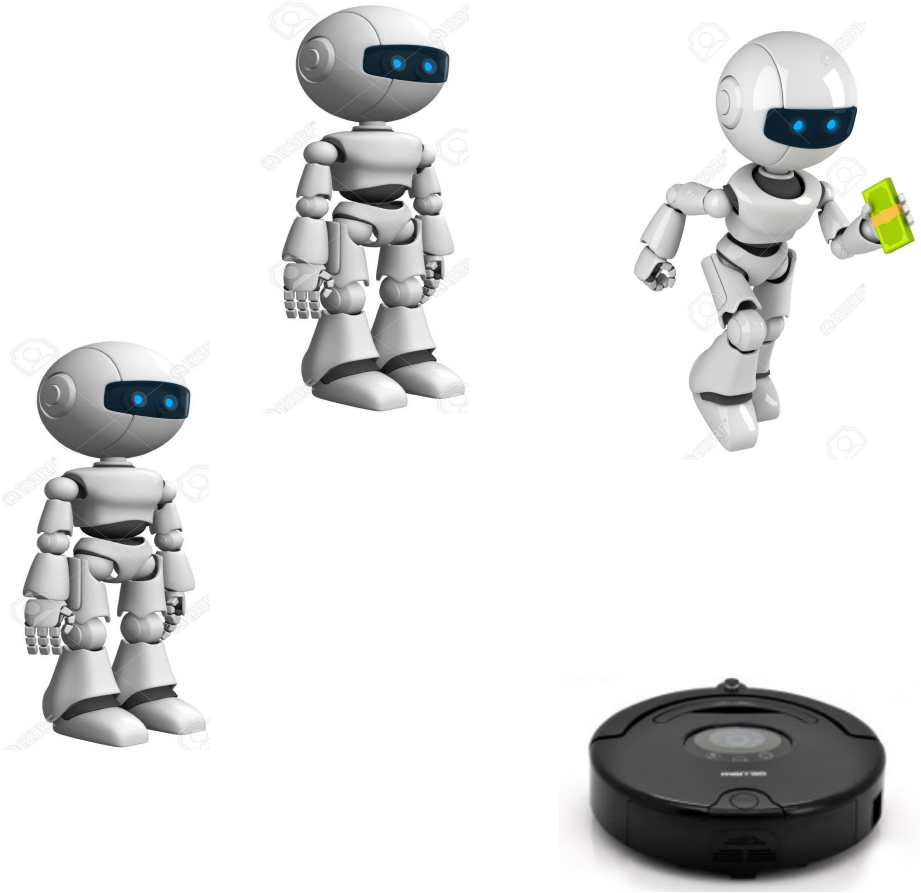
Algorithme centralisé des robots

Qui veut devenir le maître ?



Problème : comment choisir le robot maître ?

Algorithme à base de jeton



Algorithme à base d'anneau

Problème : à qui donner le jeton en premier ?





Sommaire

Algorithmes d'élection

Principe

- Suite à un changement de condition ou un évennement des processus lancent un algorithme d'élection
- Tous les processus ne lancent pas forcément l'élection, seuls ceux qui ont eu un changement de condition ou un évènement
- Objectif : élire un processus parmi d'autres
- A l'issue de l'élection un processus est dans l'état *LEADER* et les autres sont *LOST*

Algorithmes d'élection

Pourquoi ?

- En Algorithmique Distribuée : algorithmes utilisent un coordinateur ou initiateur
- Exemples :
  - Algorithmes centralisés : nommer le maître, trouver un nouveau maître en cas de panne
  - Algorithme à base de jeton : savoir qui possède le jeton au début de l'algorithme, à qui redonner le jeton en cas de panne.
  - Algorithmes de Consensus

Impossibilité de l'élection

Hypothèses

- Processus strictement identiques
- Liens bi-directionnels entre processus
- Communications fiables

Résultat

Il n'existe pas d'algorithme d'élection déterministe qui termine correctement en un temps fini.

Impossibilité de l'élection

Preuve

- Considérons deux processus strictement identiques ayant des délais de communication unitaires
- Si les deux processus commencent en même temps, comme ils sont dans le même état ils composent et envoient les mêmes messages
- Comme les temps de communication sont identiques alors ils reçoivent en même temps et leur état reste identique dans le temps
- Il ne peut donc pas y avoir un des processus qui décide qu'il est le maître et l'autre non

$\Rightarrow$ Il faut introduire un élément de différenciation entre les processus

Impossibilité de l'élection

Élements de différenciation

- Initiateur unique :
  - Processus avec numéro particulier (0, N)
- Identificateur par processus :
  - Fixé initialement : adresse MAC
  - Fixé dynamiquement :
    - Impossibilité de la numérotation, si processus identiques
    - UUID

Algorithmes d'élection

Propriétés

Comme pour les algorithmes de synchronisation un certain nombre de propriétés sont attendues

- Terminaison : l'algorithme élit un représentant
  - Ni deadlock
  - Ni livelock
- Sûreté : un seul représentant est élu
- Equité ? ? Pas vraiement de sens ici

Sommaire

Algorithmes d'élection

Hypothèses

- Les processus possèdent un ID mais ils ne connaissent que le leur
- Les processus ne tombent pas en panne pendant l'élection
- Le nombre de processus est fini et ne change pas au cours de l'élection
- Réseau complet : chaque processus peut communiquer avec tous les autres
- Les messages arrivent tous en un temps fini, pas de perte

Algorithmes d'élection

Exercice

Proposer un algorithme d'élection sur un réseau complet

Algorithmes d'élection

Solution simple

- Les processus sont numérotés
- Chacun envoie sa valeur à tous les autres
- Le plus fort gagne

Complexité

- En temps : $O(1)$
- En messages : $O(n^2)$

Algorithme du plus fort ou *Bully algorithm*

Hypothèses

- Chaque processus a un numéro unique
- 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)

Algorithme du plus fort ou *Bully algorithm*

Déclanchement :

Sur un processus p_i , il lance l'algorithme d'élection

Lancement d'une élection par p_i :

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

Réception d'un message `ELECTION` de p_j par un processus p_i :

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

Algorithme du plus fort ou *Bully algorithm*

Sur le processus p_i :

- Si aucun processus ne lui répond avant *tempo*, p_i gagne l'élection et devient le coordinateur
- Si un processus de numéro plus élevé répond, c'est lui qui prend la suite en charge. Le rôle de p_i est terminé.

Annonce de l'élu

Le nouveau leader 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 leader

Algorithme du plus fort ou *Bully algorithm*

Exercices

- Exercice 1 : écrire l'algorithme
- Exercice 2 : 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
- Exercice 3 : donner la complexité de l'algorithme
- Exercice 4 : proposer des solutions pour réduire la complexité

Algorithmes d'élection

Algorithme d'élection par niveaux

- Des processus se déclarent candidats (initiateurs)
- Les candidats essaient de capturer (convaincre de voter pour lui) les autres processus en envoyant des messages de capture
- Si un candidat échoue à capturer un processus il devient passif
- Celui qui a capturé tous les processus est le leader

Algorithmes d'élection par niveaux

Exercice

1. À quelle condition un processus peut-il en capturer un autre ?
2. Comment avoir une faible complexité en nombre de messages ?
3. Écrire l'algorithme d'élection par capture
4. Quelle est sa complexité en nombre de messages ?
5. Quelle est sa complexité en temps ?

Algorithme d'élection par niveaux

Plusieurs niveaux d'élection : stages

- Des processus se déclarent candidats (condition ou évènnement)
- Les candidats essaient de capturer les autres processus en envoyant des messages de capture, un par un
- Initialement les processus ont un niveau de 1 (eux-même) et à chaque processus capturé le candidat augmente son niveau
- Le candidat met son niveau dans les messages de capture
- Lorsqu'un processus essaie de capturer un autre processus ils comparent leurs niveaux.
- Si un candidat échoue à capturer un processus il devient passif
- Celui qui a capturé tous les processus est le leader

Algorithme d'élection par niveaux

Capture d'un processus j par un processus i

- si j est candidat
  - Si $stage_i > stage_j$
  - Si $stage_i = stage_j$ et $i < j$
  - i devient passif sinon
- si j est passif l'attaque réussit
- si j est déjà capturé alors i doit combattre son propriétaire k pour le capturer
  - si k est lui même candidat mais avec un niveau supérieur ou avec le même niveau et $k < i$ alors i perd, k notifie j qui notifie i
  - dans les autres cas l'attaque réussit et k notifie j qui notifie i ; si k était candidat il devient passif
- Si l'attaque réussit i incrémente son niveau de 1

Algorithmes d'élection par niveaux

Exercice

1. À quelle condition un processus peut-il en capturer un autre ?
2. Écrire l'algorithme d'élection par capture
3. Quelle est sa complexité en nombre de messages ?
4. Quelle est sa complexité en temps ?
5. Montrer qu'à un instant t la somme des niveaux peut dépasser le nombre de processus
6. Comment améliorer la complexité en nombre de messages ?

Algorithme d'élection par niveaux

Complexité

- A ce stade l'algorithme a encore une complexité élevée
- Le nombre de processus au niveau n peut rester grand
- Il faut limiter le nombre de processus par niveaux en garantissant qu'un processus ne peut être capturé que par un seul autre processus

Algorithme d'élection par niveaux

Optimisations

- Si un processus attaque un processus capturé, le processus répond directement si son propriétaire est plus fort
- On limite le nombre de messages remontés au propriétaire à un instant t à un, les autres sont mis en attente
- Si le propriétaire change les messages sont remontés au nouveau propriétaire

Algorithmes d'élection

Exercice

Dérouler l'algorithme d'élection par niveaux pour cinq processus $(p_1, p_2, p_3, p_4, p_5)$ avec trois initiateurs p_1, p_3, p_5 :

- Le processus p_3 commence en premier et envoie ses messages de capture dans l'ordre suivant : p_4, p_1, p_2, p_5
- Le processus p_1 commence en second et envoie ses messages de capture dans l'ordre suivant : p_4, p_3, p_2, p_5
- Le processus p_5 commence en dernier et envoie ses messages de capture dans l'ordre suivant : p_2, p_4, p_1, p_3

Algorithme d'élection par niveaux

Exercice

Montrer qu'à un instant t chacun des processus ne peut être capturé que par un seul processus

Algorithme d'élection

Algorithme sur un réseau complet

- Complexité en messages élevée
- Essayer d'autres topologies
- Mise en place d'une topologie au sein d'un algorithme peut-être justifiée pour réduire le nombre de messages et/ou simplifie l'algorithme (lié à la structure de l'algorithme)

Algorithmes d'élection

Différentes topologies

- Arbre ou arbre de recouvrement
- Anneau
- Quelconque

Sommaire

Algorithme pour un arbre

Principe

- Un (ou plusieurs) processus détecte la panne du coordinateur
- Il informe l'ensemble des 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*
$Neigh_p$: voisins du processus

Algorithme pour un arbre

Partie réveil :

début

- **si** p *est initiateur* **alors**
  - $ws_p \leftarrow true$
  - **pour** $q \in Neigh_p$ **faire**
    - envoie $< wakeup >$ à q
- **tant que** $wr_p < \sharp Neigh_p$ **faire**
  - reçoit $< wakeup >$
  - $wr_p \leftarrow wr_p + 1$
  - **si** $ws_p = faux$ **alors**
    - $ws_p \leftarrow true$
    - **pour** $q \in Neigh_q$ **faire**
      - envoie $< wakeup >$ à q

Algorithme pour un arbre

Partie election :

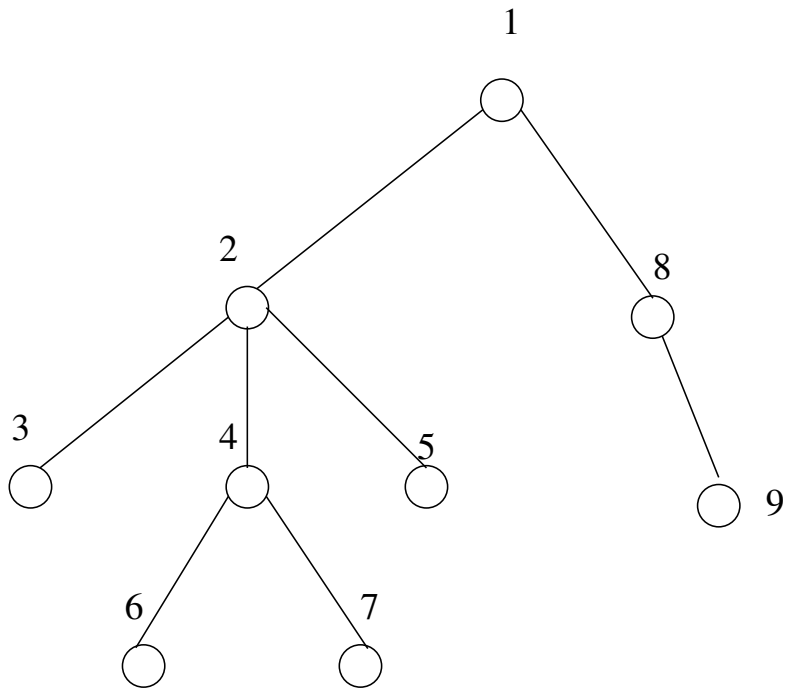
début

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

Algorithme pour un arbre

Exercice

Dérouler l'algorithme d'élection pour l'arbre suivant dans le cas où les processus p_5 et p_8 sont initialteurs.



Algorithme pour un arbre

Exercice

Écrire l'algorithme d'élection sur un arbre sous forme de règles

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 :
  - Poser les hypothèses
  - Donner les règles
  - Élements de preuve
  - Propriétés
- 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** $\{p\}$
    $etat_p$ ($init$, $leader$, $perdu$, $sleep$) **init** $sleep$
début
    si p *est intiateur* **alors**
        $etat_p \leftarrow init$;
        envoie $< tok, p >$ à $Suivant_p$
        reçoit $< tok, q >$
        tant que $q \neq p$ **faire**
            $Liste_p \leftarrow Liste_p \cup \{q\}$
            envoie $< tok, q >$ à $Suivant_p$;
            reçoit $< tok, q >$
        si $p = max\ (Liste_p)$ **alors** $etat_p \leftarrow leader$ **sinon** $etat_p \leftarrow perdu$
    sinon
        tant que *vrai* **faire**
            reçoit $< tok, q >$;
            envoie $< tok, q >$ à $Suivant_p$
            if $etat_p = sleep$ **then** $etat_p \leftarrow perdu$

Algorithme d'élection de LeLann

Exercices

1. Déroulement de l'algorithme
   - un anneau de 7 processus, numérotés de 0 à 6
   - les processus `1,4,5` lancent une élection
   - Dérouler l'algorithme de Lelann
2. Quelle est la complexité de l'algorithme ?

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**
  - $etat_p \leftarrow init$
  - envoie $< tok, p >$ à $Suivant_p$
  - **tant que** $etat_p \neq leader$ **faire**
    - reçoit $< tok, q >$
    - **si** $q = p$ **alors** $etat_p \leftarrow leader$
    - **sinon si** $q > p$ **alors**
      - **si** $etat_p = init$ **alors** $etat_p \leftarrow perdu$
      - envoie $< tok, q >$ à $Suivant_p$
- **sinon**
  - **tant que** *vrai* **faire**
    - reçoit $< tok, q >$
    - envoie $< tok, q >$ à $Suivant_p$
    - **si** $etat_p = sleep$ **alors** $etat_p \leftarrow perdu$

Algorithme d'élection Chang et Roberts

Exercice

Déroulement de l'algorithme

- Application composée de 8 processus numérotés de 0 à 7
- Les processus 1,2,4,6 veulent se choisir un coordinateur

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
- Quand un processus reçoit un message *elu* il sort du *tant que* $etat_p \neq leader$, il enregistre le leader et il fait suivre le message
- Quand le *leader* reçoit le message *elu*, l'élection est finie.

Propriétés

- Complexité de l'algorithme :
  - Nombre de messages : $O(N^2)$ (N initiateurs * N processus)
  - Temps : $O(N)$
- Finaliser par un dernier message pour annoncer la fin de

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 ni à la sienne et ni 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.

Algorithme d'élection de Dolev, Klawe et Rodeh

Exercices

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 `0, 2, 3 et 6` détectent cette panne

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_i diffuse à ses voisins un message marqué par son numéro p_i
- 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 en nombre de messages : N (nombre processus) * M (nombre messages par processus)

Algorithme basé sur l'extinction

Exercice

Écrire l'algorithme d'élection basé sur l'extinction