

# Les algorithmes de systèmes repartis

Université Virtuelle de Côte d'Ivoire



# Table des matières

<b>Objectifs</b>	3
<b>Introduction</b>	4
<b>I - Les algorithmes d'élection distribuée</b>	5
1. Présentation .....	5
2. Algorithme sur un arbre .....	6
3. Algorithme sur un anneau : .....	6
<b>II - Exercice</b>	8
<b>III - Les algorithmes d'exclusion mutuelle distribuée</b>	10
1. Présentation .....	10
2. Les mécanismes centralisés .....	10
3. Synchronisation distribuées avec des horloges logiques .....	12
4. Les algorithmes de synchronisation distribuée utilisant un jeton .....	12
<b>IV - Exercice</b>	14
<b>Solutions des exercices</b>	16

# Objectifs



À la fin de cette leçon, l'apprenant sera capable de :

- identifier les algorithmes d'élection distribuée;
- décrire les algorithmes d'exclusion mutuelle distribuée.

# Introduction



Il existe plusieurs algorithmes distribués en fonction du problème à résoudre et du contexte. Nous avons :

- Les algorithmes d'élection distribuée
- Les algorithmes d'exclusion mutuelle distribuée
- Les algorithmes distribués de terminaison
- Les algorithmes de calculs distribués

Dans le cadre de ce cours, nous verrons les algorithmes d'élection distribuée et d'exclusion mutuelle distribuée

# Les algorithmes d'élection distribuée



## 1. Présentation

### *Objectif*

Élire un processus parmi d'autres

### *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 une valeur TEMPO
- Condition déclenche une élection
- La condition devient vraie sur 1 ou plusieurs processus, pas forcément tous

### *Remarque*

- 
- Les algorithmes d'élection utilisent un coordinateur ou initiateur
  - Le plus souvent : coordinateur = processus de plus grand numéro
  - En cas de panne du coordinateur, un nouveau coordinateur est élu (élection)

### *Différentes topologies*

- Arbre ou arbre de recouvrement
- Anneau ou anneau virtuelle
- Quelconque

### *Les différents types d'algorithme d'élection*

Il existe plusieurs types d'algorithmes d'élection qui sont :

- algorithme sur un réseau complet : algorithme du plus fort ou Bully algorithm ou algorithme de Garcia-Molina
- algorithme sur un arbre
- algorithme sur un anneau :

- Algorithme d'élection de LeLann
- Algorithme d'élection de Chang et Roberts
- Algorithme d'élection de Dolev, Klawe et Rodeh
- algorithme sur un réseau quelconque :
  - algorithme basé sur l'extinction

## 2. Algorithme sur 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

### Variables

- $ws_p$  : booléen *init* faut (  $ws_p$  est vrai si p est réveillé )
- $wr_p$  : entier *init* 0 ( compte les messages de réveil 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

```

Partie réveil :
début
    si p est initiateur alors
         $ws_p := true$ 
        pour  $q \in Neigh_p$  faire
            envoyer < wakeup > à q
    tant que  $ws_p < \forall V_p$  faire
        reçoit < wakeup >
        si  $ws_p = faux$  alors
             $ws_p := true$ 
            pour  $q \in Neigh_p$  faire
                envoyer < wakeup > à q
fin

Partie election :
début
    tant que  $\exists (q : req_p[q] = faux) > 1$  faire
        reçoit < tok,  $r_p$  > de q
         $req_p[q] := vrai$ 
         $v_p := max(v_p, r)$ 
        envoie < tok,  $v_p$  > à  $q_0$  tel que  $req_p[q_0]$  est faux
        reçoit < tok,  $v_q$  > de  $q_0$ 
         $v_p := max(v_p, r)$ 
        si  $v_p = p$  alors
             $statut_p := leader$ 
        sinon
             $statut_p := lost$ 
        pour  $q \in Neigh_p, q \neq q_0$  faire envoie < tok,  $v_p$  > à q
fin
    
```

## 3. Algorithme sur un anneau :

### Structure d'anneau

- Les processus sont organisés en anneau virtuel

-



Chaque processus connaît la structure de l'anneau ( donc son successeur  $\text{Suivant}_p$  )

- Comme pour l'arbre, l'anneau peut être virtuel, et pour l'application, les processus communiquent entre eux directement, sans passer par l'anneau

## Algorithme d'élection de Le Lann

*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 autre
- Lorsqu'un initiateur reçoit son propre jeton, il a déjà reçu tous les jetons des autres initiateurs, il calcule donc le vainqueur

```

var Linit, sous-ensemble de  $\mathcal{P}$  init (p)
etatp (init, leader, perdu, sleep)

débuts
si p est initiateur alors
  etatp := init; envoie < tok, p > à Suivantp
  reçoit < tok, q >
  tant que q ≠ p faire
    Linitp := Linitp ∪ {q}
    envoie < tok, q > à Suivantp
    reçoit < tok, q' >
  si p = max(Linitp) then etatp := leader else etatp := perdu
sinon
  tant que vrai faire
    reçoit < tok, q >
    envoie < tok, q > à Suivantp
    si etatp = sleep then etatp := perdu

```

## Algorithme d'élection de Chang et Roberts

*Hypothèses et Principe :*

- Évolution 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

```

var etatp

débuts
si p est initiateur alors
  etatp := candi; envoie < tok, p > à Suivantp
  tant que etatp ≠ leader faire
    reçoit < tok, q >
    si q = p then etatp := leader
    else
      si q > p then
        si etatp = candi then etatp := perdu
        envoie < tok, q > à Suivantp
  else tant que vrai faire
    reçoit < tok, q >
    envoie < tok, q > à Suivantp
    si etatp = sleep then etatp := perdu

```

# Exercice



## Exercice

---

Quelques types d'algorithmes distribués sont (3 réponses) :

- ☐ élection distribuée
- ☐ processus distribués
- ☐ exclusion mutuelle distribuée
- ☐ mémoires distribuées
- ☐ calculs distribués
- ☐ processeur distribués

## Exercice

---

La topologie logique d'une architecture distribuées peut être (2 réponses)

- ☐ cursive
- ☐ connexe
- ☐ en anneau
- ☐ en trapèze
- ☐ en arbre

## Exercice

---

Les différents types d'algorithmes d'élection sont (4 réponses):

- ☐ algorithme sur un réseau complet
- ☐ algorithme sur un arbre
- ☐ algorithme sur une pyramide
- ☐ algorithme sur un trapèze



- ☐ algorithme sur un anneau :
- ☐ algorithmes cursifs
- ☐ algorithme ellipsoïdale
- ☐ algorithme sur un réseau quelconque :

### Exercice

---

Les différents types d'algorithme d'élection en anneau sont (3 réponses) :

- ☐ Algorithme d'élection de Dolev, Klawe et Rodeh
- ☐ Algorithme d'élection de Garcia-Molina
- ☐ Algorithme d'élection de Dizan Paulus
- ☐ Algorithme d'élection de Ricart et Agrawala
- ☐ Algorithme d'élection de Chang et Roberts
- ☐ Algorithme d'élection de LeLann



# Les algorithmes d'exclusion mutuelle distribuée

## 1. Présentation

## Rappels

- Une ressource partagée ou section critique n'est accédée que par un processus à la fois
- Un processus est dans les trois états possibles, par rapport à l'accès à la ressource :
  - *demandeur*
  - *dedans*
  - *dehors*
- Changement d'états par un processus :
  - de *dehors* à *demandeur*
  - de *dedans* à *dehors*
- Couche middleware gère le passage de l'état *demandeur* à l'état *dedans*

L'accès en exclusion mutuelle doit respecter 2 propriétés :

- *Sûreté* : un processus au maximum en section critique ( état dedans )
- *Vivacité* : toutes demande d'accès à la section critique est satisfaite en un temps fini

En système distribué, nous avons deux classes de mécanisme d'exclusion mutuelle :

- Les *mécanisme centralisés*
- Les *mécanisme distribués*
  - Synchronisation distribuée avec des horloges logiques
  - Synchronisation distribuée utilisant un jeton

## 2. Les mécanismes centralisés

### Définition

Les mécanismes centralisés de synchronisation en système distribué regroupe les algorithmes qui utilisent un coordinateur pour gérer l'exclusion mutuelle. Ce coordinateur :

- centralise les requêtes des différents processus de l'application
- réalise l'exclusion mutuelle en accordant la permission d'entrée en Section Critique
- gère une file des requêtes en attente de Section Critique

### *Algorithme : Processus $P_i$*

- Quand un processus  $P_i$  veut entrer en Section Critique, il envoie au coordinateur un message de "demande d'entrée en section critique". Il attend la permission du coordinateur avant d'entrer en section critique
- Lorsqu'un processus  $P_j$  reçoit la permission, il entre en section critique
- Quand  $P_i$  quitte la section critique, il envoie un message de sortie de section critique au coordinateur

### *Algorithme : Le Coordination*

- Si aucun processus en section critique, à la réception d'une demande d'accès de  $P_i$ , il (coordinateur) retourne "permission d'accès" à  $P_i$
- Si un processus  $P_j$  est déjà en section critique. Le coordinateur refuse l'accès. La méthode dépend de l'implantation :
  - Soit il n'envoie pas de réponse, le processus  $P_j$  est bloqué
  - soit il envoie une réponse : "Permission non accordée"

Dans les deux cas, le coordinateur dépose la demande de  $P_j$  dans une file d'attente

- A la réception d'un message de sortie de section critique, le coordinateur prend la première requête de la file d'attente de la section critique et envoie au processus un message de permission d'entrée en section critique

### *Avantages*

- Algorithme garantit l'exclusion mutuelle
- Algorithme juste (les demandes sont accordées dans l'ordre de la réception)
- Pas de famine (aucun processus ne reste bloqué)
- Pas de supposition sur l'ordre des messages
- Complexité : maximum 2 ou 3 messages

### *Inconvénients*

- Si le coordinateur a un problème, tout le système s'effondre
- Si processus bloqués lors de la demande d'accès à une section critique occupée : impossibilité de détecter la panne du coordinateur
- Dans de grands systèmes, le coordinateur = goulot d'étranglement

### 3. Synchronisation distribuées avec des horloges logiques

#### *Rappel*

- Les processus ne communiquent que par échange de messages
- Chaque processus connaît l'ensemble des processus du système
- Chaque processus a ses propre variable, pas de partage de variables
- On ne considère pas les cas de pertes de messages ni les pannes de machines
- Les échanges de messages sont FIFO : les messages ne se doublent pas.

#### *Principe*

- Diffuser la demande
- Il faut l'accord de tous les autres
- Une fois le consensus obtenu on accède à la SC (section critique)
- Difficulté : demande concurrentes

#### *Les algorithmes de synchronisation distribuées avec des horloges logiques*

- Algorithme de Lamport
- Algorithme de Ricart et Agrawala
- Algorithme de Carvalho et Roucairol

Le principe, le fonctionnement et l'écriture des algorithmes de Lamport, de Carvalho et Roucairol, et de Ricart et Agrawala ne seront pas abordés dans ce cours

### 4. Les algorithmes de synchronisation distribuée utilisant un jeton

#### *Algorithmes*

Différentes structures de communication :

- Anneau : Le Lann
- Diffusion : Suzuki-Kazami
- Arbre : Naimi-Trehel

#### *Algorithme de Le Lann*

- Algorithme utilisant un jeton pour gérer l'accès à une ressource critique
- Jeton (unique pour l'application) = permission d'entrer en section critique
- un seul processus détient le jeton à un moment donné

#### *Principe de l'algorithme :*

- Quand un processus reçoit le jeton de son prédécesseur dans l'anneau :

- s'il est demandeur de section critique, il garde le jeton et entre en section critique
- s'il n'est pas demandeur de section critique, il envoie le jeton à son successeur dans l'anneau
- A la sortie de la section critique, le processus envoie le jeton à son successeur dans l'anneau
- Pour des raisons d'équité, un processus ne peut pas entrer deux fois de suite en section critique

*Avantages :*

- Simple à mettre en œuvre
- intéressant si nombreux demandeurs de la ressource
- Équitable en terme de nombre d'accès et de temps d'attente

*Inconvénients :*

- Nécessite des échanges de messages même si aucun processus ne veut accéder à la section critique
- Le temps d'accès à la section critique peut être long
- Perte de jeton :
  - difficulté de détecter un tel cas
  - impossibilité de prendre en compte le temps écoulé entre deux passages de jeton

*Algorithme de Suzuki-Kasami*

- Chaque processus connaît l'ensemble des participants
- Quand un processus  $P_i$  veut accéder à la section critique, il envoie un message REQ(H) à tous les participants
- Le jeton contient l'estampille de la dernière visite qu'il a effectué
- Quand  $P_i$  sort de la section critique, il cherche le premier processus  $P_k$  tel que l'estampille de la dernière requête de  $P_k$  soit supérieure à celle mémorisée dans le jeton (correspondant à la dernière visite du jeton à  $P_k$ )

# Exercice



## Exercice

---

Il existe deux mécanismes d'exclusion mutuelle

- ☐ les mécanismes dynamiques
- ☐ les mécanismes rétrospectifs
- ☐ les mécanismes alvéoles
- ☐ Les mécanismes distribués
- ☐ les mécanismes embarqués
- ☐ Les mécanismes centralisés

## Exercice

---

Les mécanismes d'exclusion mutuelles distribuées sont ( 2 réponses ) :

- ☐ Synchronisation distribuée une clé de cryptage
- ☐ Synchronisation distribuée utilisant un jeton
- ☐ Synchronisation distribuée avec chiffrement
- ☐ Synchronisation distribuée centralisée
- ☐ Synchronisation distribuée avec des horloges logiques

## Exercice

---

Les algorithmes de synchronisation distribuées avec des horloges logiques sont ( 3 réponses ) :

- ☐ Algorithme de Carvalho et Roucairol
- ☐ Arbre : Naimi-Trehel
- ☐ Algorithme de Ricart et Agrawala
- ☐ Diffusion : Suzuki-Kazami

- ☐ Algorithme de Lamport
- ☐ Le Lann

# Solutions des exercices



### > Solution n°1

Exercice p. 8

## Exercice

Quelques types d'algorithmes distribués sont (3 réponses) :

- ☒ élection distribuée
- ☐ processus distribués
- ☒ exclusion mutuelle distribuée
- ☐ mémoires distribuées
- ☒ calculs distribués
- ☐ processeur distribués

## Exercice

La topologie logique d'un architecture distribuées peut être (2 réponses)

- ☐ cursive
- ☐ connexe
- ☒ en anneau
- ☐ en trapèze
- ☒ en arbre

## Exercise

Les différents types d'algorithmes d'élection sont (4 réponses):

- ☒ algorithme sur un réseau complet
- ☒ algorithme sur un arbre
- ☐ algorithme sur une pyramide



- ☐ algorithme sur un trapèze
- ☒ algorithme sur un anneau :
- ☐ algorithmes cursifs
- ☐ algorithme ellipsoïdale
- ☒ algorithme sur un réseau quelconque :

Exercice

Les différents types d'algorithme d'élection en anneau sont (3 réponses) :

- ☒ Algorithme d'élection de Dolev, Klawe et Rodeh
- ☐ Algorithme d'élection de Garcia-Molina
- ☐ Algorithme d'élection de Dizan Paulus
- ☐ Algorithme d'élection de Ricart et Agrawala
- ☒ Algorithme d'élection de Chang et Roberts
- ☒ Algorithme d'élection de LeLann

> **Solution** n°2

Exercice p. 14

Exercice

Il existe deux mécanismes d'exclusion mutuelle

- ☐ les mécanismes dynamiques
- ☐ les mécanismes rétrospectifs
- ☐ les mécanismes alvéoles
- ☒ Les mécanismes distribués
- ☐ les mécanismes embarqués
- ☒ Les mécanismes centralisés

Exercice

Les mécanismes d'exclusion mutuelles distribuées sont ( 2 réponses) :

- ☐ Synchronisation distribuée une clé de cryptage

- ☒ Synchronisation distribuée utilisant un jeton
- ☐ Synchronisation distribuée avec chiffrement
- ☐ Synchronisation distribuée centralisée
- ☒ Synchronisation distribuée avec des horloges logiques

#### Exercice

Les algorithmes de synchronisation distribuées avec des horloges logiques sont ( 3 réponses ) :

- ☒ Algorithme de Carvalho et Roucairol
- ☐ Arbre : Naimi-Trehel
- ☒ Algorithme de Ricart et Agrawala
- ☐ Diffusion : Suzuki-Kazami
- ☒ Algorithme de Lamport
- ☐ Le Lann