

# Algorithmes Répartis - Jean-Michel Couvreur

Alexandre Masson

14 Janvier 2013

**Sujets traités**

- Modele synchrone
- 

**Évaluation**

- Controle continu
  - controle sur table
  - mini-projet
- Controle terminal
  - un controle sur table
  - la nature du sujet sera développée en cours

## 1 Modèle Synchrone

Processus sont attachés à des noeuds d'un graphe orienté et communique par des messages.

Graphe  $G = (V, E, n = |V|$

- outnbr, innbr
- distance(i,j) taille du plus court chemin de i à j
- diametre

M : alphabet des messages plus false pour rien.

pour tout i dans V, un processus est donnée par

- States : ensemble d'états
- start : états initiaux
- msgs : states x out-nbrs  $\rightarrow M \cup \text{false}$
- trans : ensemble des transitions

Exécution d'un tour ;

- appliquer msgs pour déterminer les messages à envoyer
- envoyer et recevoir les messages,
- Appliquer trans , pour déterminer l'état suivant.

**Remarques**

- pas de restriction sur la durée des calculs locaux,
- Déterministes,
- on peut définir des "états d'arrêt",
- Plus tard nous examinerons quelques problèmes :
  - temps de démarrage variable
  - défaillance
  - choix aléatoires

**Exécution**

- une exécution est un objet mathématique servant à décrire comment un algorithme fonctionne.

- Définition :
  - un état global
  - Messages
  - exécution :  $C_0, M_1, N_1, C_1, M_2, N_2, \dots$ 
    - $C^*$  : sont des états globaux
    - $M^*$  : sont les messages envoyés
    - $N^*$  : sont les messages reçus
    - Séquence infinie (mais on peut considérer des préfixes finis)

## 2 Problème de l'élection d'un leader

- Réseau de processus
- Vous voulez distinguer un processus , le leader.
- Finalement, exactement un processus sera désigné leader
- Motivation : le leader peut prendre en charge :
  - les communications
  - coordination des traitements des données (par exemple ; dans les protocoles de validation)
  - Allocation de ressources
  - etc...

### Cas simple l'anneau

- Variantes :
  - bidirectionnel ou unidirectionnel
  - taille de l'anneau n connu ou inconnu

### Nous avons besoin de qqc de plus

- besoin de distinguer les processus
- supposons un UID , s'il connaît
- chaque processus démarre en stockant son pid
- il sont comparable et pouvoir faire de l'arithmétique simple dessus est un plus
- dans le réseau tous les identifiants sont différents

### Un algorithme

- auteurs : LeLann , Chang, Roberts
- hypothèse
  - comm unidirectionnelle
  - les proc ne connaissent pas n
  - comparaison d'UID seulement
- Idée :
  - chaque processus envoie son uid dans un message, à relayer étape par étape, autour de l'anneau.
  - le proc compare l'id reçu avec le sien , et envoie à son voisin le plus grand des deux entre le sien et celui reçu.

### Preuve , complexité, terminaison

- M, l'alphabet de messages : UID
- état : valeur des variables :
  - u : a pour valeur son pid
  - send, son pid ou false *status : ? ou leader, initialement ?*
- start : défini par l'init des variables
- msgs : transmet la valeur send à son voisin
- trans : défini par le pseudo-code :

- if incoming = v, a UID, then
- case :
- $v > u$  : send := v ;
- $v = u$  : status := leader
- $v < u$  : no-op
- endcase

### Réduire le nombre de communication

- autor : Hirschberg, Sinclair
- hypothèse :
  - bidirectionnel
  - les processus ne connaissent pas n
  - comparaisons d'id seulement
- Idée :
  - Stratégie du doublement
  - Chaque processus envoie son UID dans les deux sens, à des distances de plus en plus grandes (successivement deux fois plus grande à chaque fois)
  - phase aller : un jeton est ignoré si il atteint un noeud dont l'uid est plus grand
  - phase retour : tout le monde passe le jeton *un processus débute la phase suivante que si ses deux jetons reviennent. le processus qui reçoit son propre jeton dans la phase aller est l'élu.*

### Élection d'un leader

- Hypothèse
  - UID avec comparaisons.
  - pas d'hypothèse sur la répartition des UID
  - les processus connaissent un majorant du diamètre

## 3

14 Février 2013

**défaillance de processus** défaillance byzantine : chaque processus envoie sa valeur, le cas de l'arrêt fonctionne, nb rang = nb panne + 1. byz : un menteur au moment du choix, va envoyer au suivant une mauvaise valeur .

**collecte d'information exponentielle** valeur transmise le long d'un réseau en forme d'arbre. la première couche correspond au résultats reçu à l'étape numéro 1., le seconde correspond à la 2eme étape. le dernier chiffre du numéro marqué au dessus de la "feuille" c'est le dernier processus à avoir envoyé le message reçu. on peut ainsi remonter le chemin parcouru par le message.

Chaque processus utilise la même structure d'arbre.

les noeuds sont étiquetés par des valeurs

**initialement** les têtes sont étiquetées par la valeur du noeuds

**round  $r > 1$**

- envoyer le niveau  $r-1$  à tous noeuds (soi même aussi)
- utiliser les messages reçu pour étiqueter le niveau  $r$
- si pas de message utiliser  $\perp$

**règle de choix pour le cas de la panne d'arrêt :**

- triviale
- on prend toutes les étiquettes de l'arbre
- si un seul élément , on le prend
- sinon on prend par défaut  $v_0$

**preuve et complexité**

- preuve : il y aura toujours un moment où il n'y a pas de panne
- complexité : nombre de panne  $+1$  rounds, il envoie à tous ces voisins
- complexité en nombre de message :  $n-1$  \* nombre de pannes
- défaut : nombre de feuilles peut croître exponentiellement

### 3.1 Algorithmes : pannes byzantines

- conditions :
  - accord : pas deux processus sans panne décide différemment
  - si tous les processus sans panne une valeur identique alors elle est choisie par tous les processus sans panne
  - Terminaison : tous les processus sans panne font un choix
- EIG algorithme utilise :
  - un arbre
  - $f+1$  rounds
  - $n > 3f$

**mauvais exemple** 3 processus, 1 panne supposons  $n > 3f$  :

nouvelle règle de décision :

- notons  $val(x)$  , la valeur du nœud  $x$  de l'arbre.
- redécorer l'arbre
  - parcours de bas en haut pour trouver la valeur la plus fréquente.

18 Février 2013

## 4 Modèles asynchrones