Problème des Généraux Byzantins

(Problème de la diffusion cohérente)

Z. Mammeri

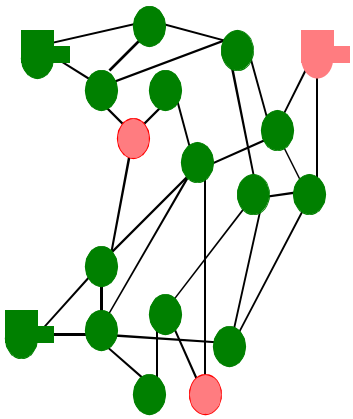
IRIT - UPS

Cours de DEA, 1998-2004

1. Spécificités du problème

- **Enoncé du problème**

Plusieurs armées, avec chacune à sa tête un général, assiègent une ville ennemie. Il s'agit pour ces généraux de décider, en fonction de leurs opinions, s'ils se retirent ou s'ils attaquent, et que cette décision soit bonne et unanime. Le problème vient du fait que les généraux ne communiquent pas directement deux à deux, et que des traîtres se sont glissés entre eux, dans le but d'empêcher les généraux loyaux de s'accorder ou de prendre une bonne décision.



- **Problème qui a suscité de nombreuses recherches d'algorithmes**

Problèmes à résoudre

- **Problème complet**
  - Chaque processus Général a une valeur initiale à transmettre
  - A la fin de la transmission de toutes les valeurs, on doit avoir :

    ACCORD : tous les processus corrects obtiennent le même vecteur constitué d'autant de valeurs qu'il y a de processus Généraux

    VALIDITE : dans le vecteur de valeurs d'un processus correct, toute valeur relative à un processus correct est la valeur initiale de ce processus

  ⇒ **Tous les processus corrects ont la même vue de l'état global**

- **Résoudre le problème précédent nécessite la résolution du problème de la diffusion cohérente**

Problème de la diffusion cohérente

- **Enoncé du problème**
  - Un processus (dit commandant) transmet une valeur à un ensemble de processus (dits lieutenants)
  - Après la transmission des valeurs par tous les sites, on doit avoir :

    ACCORD : tous les processus corrects s'accordent sur la même valeur

    VALIDITE : si le commandant est correct, tous les lieutenants s'accordent sur sa valeur

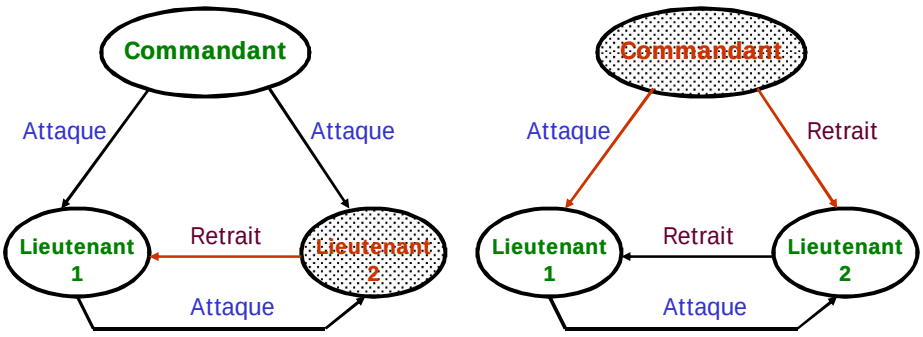
- **Résolution du problème par des Protocoles d'accord**

Problème des GB en informatique

- **Redondance dans les systèmes industriels et embarqués**
  - – Plusieurs équipements servent à fournir l'état d'un même élément (dans les avions, navettes spatiales, ...)
    Ex. Un capteur dit « train d'atterrissage sorti » et un autre dit le contraire. Que doit faire le pilote ?
- **Dans les systèmes répartis**
  - – Validation de transactions sur les BD réparties
  - – Commerce électronique
  - – Démarches administratives électronique (vote, ...)
  - – Contrôle d'armes ou d'armées
  - – …
- **Source du problème :** distribution aléatoire des erreurs ou malveillance

Existence de solution

- **Cas d'impossibilité de solution**



- **Condition d'existence de solution : $N \geq 3m + 1$**

  N : nombre de processus (personnes)
  m : nombre de processus défaillants (traîtres)

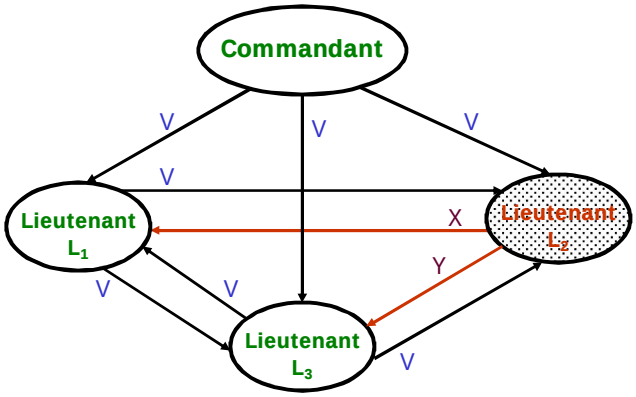
2. Algorithme « Oral Message » de Lamport

- **OM tolère m fautes si N ≥ 3m + 1**
- **Algorithme OM(0)** /* Pas de tolérance aux fautes */
  (1) : Le commandant envoie sa valeur **V** à tous ses lieutenants
  (2) : Si le processus i reçoit du commandant une valeur Val, alors v_i = Val
  Sinon v_i = Valeur_par_défaut
- **Algorithme OM(m)** /* m ≥ 1 */
  (1) : Le commandant envoie sa valeur **V** à tous ses lieutenants
  (1 ') : ∀i, si le processus i reçoit Val du commandant, alors v_i = Val
  (2) : ∀i, le processus i agit comme un commandant en exécutant OM(m-1) pour envoyer sa valeur v_i aux n-2 autres lieutenants
  (3) : ∀i, ∀k (i≠k), si pendant la phase (2) de OM(m-1), i reçoit Val_k de k, alors V_k = Val_k, sinon V_k = Valeur_par_défaut
  A la fin de toutes les réceptions, i utilise V_i = Majorité(V_1, ..., V_{n-1})
  (S'il n'y a pas de valeur majoritaire, prendre Valeur_par_défaut)

Récursivité

Exemple 1

(Situation avec 1 lieutenant traître)

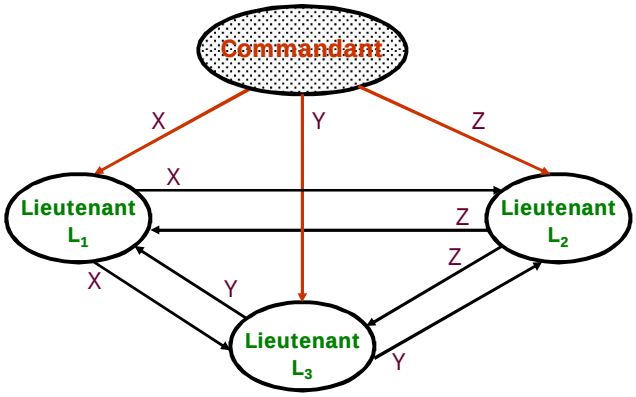


Pour L_1 : V_1 = Majorité(V, V, X) = V
Pour L_3 : V_3 = Majorité(V, V, Y) = V

} Accord sur la valeur du commandant

Exemple 2

(Situation avec 1 commandant traître) - Cas 1

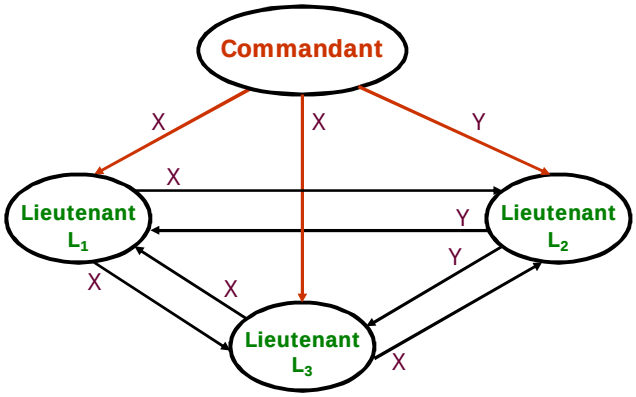


Pour L_1 : V_1 = Majorité(X, Y, Z) = ?
Pour L_2 : V_2 = Majorité(X, Y, Z) = ?
Pour L_3 : V_3 = Majorité(X, Y, Z) = ?

Pas de valeur majoritaire ⇒
Le commandant est un traître

Exemple 2

(Situation avec 1 commandant traître) - Cas 2



Pour L_1 : V_1 = Majorité(X, X, Y) = X
Pour L_2 : V_2 = Majorité(X, X, Y) = X
Pour L_3 : V_3 = Majorité(X, X, Y) = X

Les trois lieutenants se mettent d'accord sur la même valeur

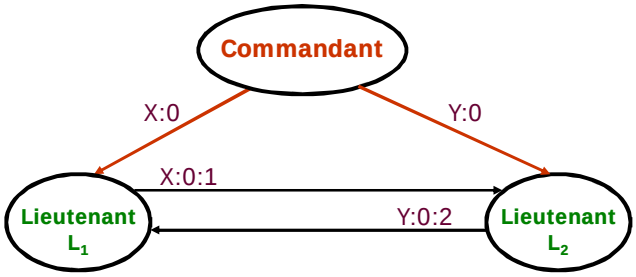
3. Algorithme « Signed Message » de Lamport

- **Idée de base :** Utilisation de signature pour interdire aux sites de modifier les messages pendant l'opération de relais
- **La signature est supposée infalsifiable**
- **Utilisation de clés** (pour tout processus i)
  - E_i : clé de cryptage du processus i connue seulement par ce processus
  - D_i : clé de décryptage du processus i connue par tous les autres processus
- **A l'émission du message Msg par le processus i :**
  - Générer MsgSigned = E_i(Msg)
  - Transmettre le message signé MsgSigned
- **Des messages arrivant à destination ne peuvent avoir des valeurs différentes que si le commandant l'a décidé (commandant traître)**
- **L'algorithme SM tolère m fautes ($m \leq N - 2$)**

- **L'algorithme SM ne tolère que les fautes des commandants**
  (puisqu'il suppose que les sites relayeurs ne peuvent pas modifier les messages signés)
- **Algorithme SM(m)** /* $m \geq 1$ */
  (0) $V_i = \emptyset$ /* V_i est un ensemble de valeurs */
  (1) Le commandant signe et envoie sa valeur V (soit V:0 le message émis)

  (2) : Pour chaque processus-lieutenant i
  (a) : Si le processus i reçoit un message de la forme V:0 du commandant, et il n'a reçu aucun ordre auparavant, alors :
  – $V_i = \{V\}$
  – il crée et signe un message V:0:i qu'il envoie aux n-2 autres lieutenants
  (b) : Si le processus i reçoit un message de la forme V:0:j_1:...j_k et V n'est pas encore dans l'ensemble V_i, alors :
  – $V_i = V_i \cup \{V\}$
  – Si k<m, alors il envoie un message signé V:0:j_1:...j_k:**i** à tous les lieutenants autres que j_1, ..., j_k
  (3) A la fin de toutes les réceptions, si V_i contient une seule valeur V, alors utiliser cette valeur, sinon utiliser Valeur_par_défaut (dans ce cas $V_i = \emptyset$ ou le commandant est défaillant).

Exemple



Pour L_1 : Majorité(X, Y) = ?
Pour L_2 : Majorité(Y, X) = ?

} Pas de majorité pour les deux lieutenants
La faute du commandant est détectée

4. Conclusion

- **Le problème des GB (et celui de la diffusion cohérente) se posent pour beaucoup d'applications réparties (bases de données, vote, ...)**
- **Solution à l'aide des protocoles d'accord OM et SM**
- **Surcoût (très important)**
  - **pour OM et SM : traitement et nombre de message en $O(n^m)$**
  - **pour SM : mécanisme de signature coûteux**
- **Beaucoup d'améliorations et extensions des algorithmes de Lamport** (cas d'applications particulières comme la synchro d'horloges, réseaux particuliers, ...) **pour réduire la complexité des algorithmes.**