TD: le problème des pannes byzantines.

Consensus avec pannes byzantines.

Ce exercice est consacré à étudier le problème du consensus plus particulièrement avec pannes byzantines. Voici un exemple de problème de consensus : dans un système de gestion de transaction répartie, tous les processus ayant participé à une transaction doivent finalement décider de sa validation ou de son annulation. Ils doivent tous prendre la même décision. Un des problèmes majeur est d'être robuste vis à vis des pannes (de sites ou de communications).

Considérons le problème des généraux byzantins. L'armée byzantine assiège une ville : elle a n campements commandé par un lieutenant. Un général commande un parmi ces n campements et commande aussi les $n-1$ lieutenants. Par abus de notation, le général sera aussi considéré comme un lieutenant. Parmi ces n lieutenants, il y a f traites (ou byzantins). Cette armée doit attaquer cette ville. Pour réussir il faut que tous les campements commandés par des lieutenants loyaux doivent attaquer en même temps. Chaque jour, le général donne l'ordre d'attaquer ou d'attendre : il est noté ℓ_0. Les autres lieutenants sont notés $\ell_1,\ldots,\ell_{n-1}$.

Le problème est étudié dans le cadre suivant dans le mode synchrone et dans le monde se restreint au problème suivant :

1. **dans le mode de communication oral** : Les communications se font par l'intermédiaire de messagers (chaque destinataire d'un message connaît l'expéditeur).
2. **dans une nombre limité d'ordres** : deux possibles ⟨`ATTAQUER`⟩ ou ⟨`REPOSER`⟩.
3. **dans le mode de synchrone** : le temps de transmission ne peut pas être supérieure à δ. Les lieutenants se rendent compte quand il y a une perte de messages. Lors de non-réception de messages, la valeur reçue par défaut sera `DEF`.

Le résultat de l'algorithme doit satisfaire les deux conditions suivantes :

IC1 : tous les lieutenants loyaux prennent la même décision processus normaux doivent connaître/prendre la décision de P_0.

IC2 : Si le général ℓ_0 est byzantin, alors chaque lieutenant loyal obéissent à l'ordre du général.

Exemple (vu en cours) : Considérons l'exemple suivant : 4 processeurs $\ell_0,\ldots,\ell_3$ et ℓ_2 est le seul processeur malveillant (voir figure ci-dessous).

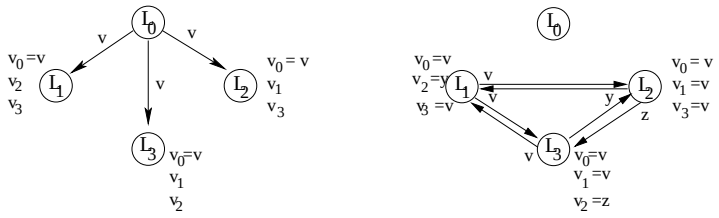

FIG. 1 – Échange des messages lors de l'algorithme

étape 1 : ℓ_0 envoie $\langle v \rangle$ à ℓ_1, ℓ_2 et ℓ_3
étape 2 : ℓ_2, ℓ_3 et ℓ_1 reçoivent $\langle v \rangle$.
étape 3 : ℓ_2, reçoit $\langle v \rangle$ de ℓ_3 et ℓ_1. il choisit $v = majorite(v, v, v)$
ℓ_1, reçoit $\langle v \rangle$ de ℓ_3 et $\langle y \rangle$ de ℓ_2. il choisit $v = majorite(v, y, v)$
ℓ_3, reçoit $\langle v \rangle$ de ℓ_1 et $\langle x \rangle$ de ℓ_2. il choisit $v = majorite(v, x, v)$
Finalement, ℓ_1 et ℓ_2 aboutissent à la même décision v celle initialement prise par ℓ_0.

Description de l'algorithme

Voici l'algorithme `P(0)` avec n campements et m participants byzantins ($n > 3m$) :

1. Cas `P(0)` (aucun participant est byzantin : $m = 0$) :
   (a) Le général ℓ_0 envoie la valeur $\langle v \rangle$ à chacun de ses lieutenants.
   (b) Si le lieutenant ℓ_j reçoit la valeur $\langle v \rangle$ alors $v_j = v$ sinon $v_j =$ `DEF` (détection d'une non réception d'un message).
2. Cas `P(m)` (m participants byzantins, $m > 0$, $n > 3m > 0$) :
   (a) Le général ℓ_0 envoie la valeur $\langle v \rangle$ à chacun de ses lieutenants.
   (b) Pour chaque lieutenant ℓ_j,
      i. si il reçoit la valeur $\langle v \rangle$ alors $v_j = v$ sinon $v_j =$ `DEF`
      ii. il lance la procédure `P(m-1)` en se comportant comme général et en envoyant v_j à ces $n - 2$ autres lieutenants.
   (c) Pour chaque lieutenant ℓ_i et tout $j \neq i$:
      i. Soit v_j = la valeur $\langle v \rangle$ que ℓ_i reçoit de ℓ_j lors de `P(m-1)` étape 2(b)ii ; $v_j =$ `DEF` si aucune valeur est reçue (avant δ).
   (d) $val = majorite(v_0, \ldots, v_{n-1})$ définie par

$$majorite(v_0, \ldots, v_{n-1}) = \begin{cases} v & \text{si la valeur } v \text{ est majoritaire} \\ \texttt{DEF} & \text{s'il n'existe aucune valeur majoritaire} \end{cases}$$

   (e) val est la valeur décidée dans ce tour

Questions

Le calcul du consensus se fait avec n lieutenants au total dont m sont byzantins ($3n > m$).

Question 0 Cet algorithme termine-t-il ?

Question 1 : Montrer que pour tout m et k, si $n > 2k + m$ avec k byzantins, l'algorithme `P(m)` satisfait la condition `IC2`.

Question 2 : Montrer que si il y a n lieutenants avec m byzantins ($n > 3m$), l'algorithme `P(m)` satisfait les deux conditions suivantes

IC1 : tous les lieutenants loyaux prennent la même décision processus normaux doivent connaître/prendre la décision de P_0.

IC2 : "si le général ℓ_0 n'est pas byzantin, alors les lieutenants loyaux aboutissent à la même décision que celle du général ℓ_0."

Question 3 : Calculer le nombre de messages échangés.

Diffusion asynchrone avec pannes byzantines.

Considérons la diffusion dans un système asynchrone en sachant que les communications sont fiables (pas de perte de messages). Dans cet exercice, nous allons décrire un algorithme de diffusion asynchrone avec pannes byzantines avec

- n sites/processus
- f pannes tolérées au maximun ($n \leq 3f + 1$)
- 2 valeurs possibles à diffuser `0` ou `1`.

Voici une description informelle.

Phase d'initialisation L'émetteur de la diffusion u commence à envoyer un message `<INITIAL>` à tous les autres processus afin de prévenir les autres du début de la diffusion. Ensuite, il envoit le message `<ECHO,valeur>` avec *valeur* correspondant à la valeur à diffuser.

Phase de réception de message – A chaque destinataire transmet la valeur reçue `v` à tous par l'intermédiaire du message `<ECHO,v>`.

- Si un processus a reçu $\begin{cases} \text{soit} & \text{plus de } \frac{n+f}{2} \text{ messages de } \texttt{<ECHO,v>} \\ \text{ou soit} & \text{plus de } f \text{ messages } \texttt{<READY,v>}. \end{cases}$,
  alors il transmet à tous y compris à lui-même un message `<READY,v>`
- Si un processus a reçu $2f + 1$ messages `<READY,v>` avec la même valeur `v`, alors il décide que la valeur `v` est la valeur diffusée.

Questions

Question 0 : Exécuter sur un petit exemple ($n = 4$, $f = 1$) l'algorithme

Question 1 : Considérons deux processus corrects p et q. Montrer par l'absurde qu'ils ne peuvent pas envoyer des messages `<READY>` avec des valeurs différentes.

Question 2 : Les processus corrects décident-ils que la même valeur est la valeur diffusée ?

Question 3 : Que concluez-vous si l'émetteur n'est pas byzantin.