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).

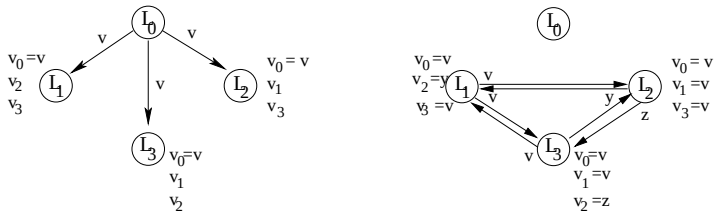


FIG. 1 – Échange des messages lors de l'algorithme

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 ?
Correction :
- A chaque appel récursif, la valeur `m` est postifive et décrémentée de 1.
- le cas où $m = 0$ est traité.

Question 1 : Montrer que pour tout m et k, si $n > 2k + m$ avec k byzantins, l'algorithme `P(m)` satisfait la condition `IC2`.
Correction : Nous allons prouver la condition `IC2` par récurrence sur m (le nombre de byzantins). Soit n le nombre de lieutenants

1. Cas $m = 0$: `P(0)` fonctionne uniquement si
   (a) le général n'est pas byzantin
   (b) et qu'il n'y a pas de messages non-délivrés.
2. Cas $m > 0$: Supposons par hypothèse de récurrence que `P(m-1)` satisfait la condition `IC2`. Prouvons que que `P(m)` satisfait la condition `IC2`. Regardons le comportement de l'algorithme :

(a) A l'étape 1 : le général ℓ_0 envoie la valeur $\langle v \rangle$ à chacun de ses lieutenants.

(b) A l'étape 2 : chaque lieutenant loyal ℓ_j exécute de nouveau `P(m-1)` à l'étape 2(b)ii (et devient général). De plus, on peut noter que $n-1 > 2k+(m-1)$ car $n > 2k+m$ et $m > 0$. Donc on peut appliquer l' hypothèse de récurrence.

Chaque lieutenant loyal ℓ_i obtient $v_j = v$ de chaque lieutenant loyal ℓ_j.

Sa liste $(v_0, \ldots, v_{n-1})$ contient $1 + n - 1 - k = (n-k)$ éléments ayant la valeur v.

(c) Donc pour que le lieutenant loyal ℓ_i choisit v, il faut que la valeur v soit majoritaire dans $(v_0, \ldots, v_{n-1})$.

il faut $n - k > \frac{n}{2}$

(d) Comme $n > 2k+m$, on a $n-1 > 2k+(m-1)$ et $m > 0$ ensuite , $\frac{n-1}{2} > k$

(e) Comme $\frac{n-1}{2} > k$, on a $n - k = n - 1 + 1 + k > \frac{n}{2}$

(f) Donc v est majoritaire dans $(v_0, \ldots, v_{n-1})$. et chaque lieutenant loyal ℓ_i choisit la valeur v.

(g) Ce qui vérifie 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."

Correction : Nous allons prouver la condition `IC2` par récurrence sur m (le nombre de byzantins). Soit n le nombre de lieutenants

1. Cas $m = 0$: `P(0)` fonctionne uniquement si
   (a) le général n'est pas byzantin
   (b) et qu'il n'y a pas de messages non-délivrés.

   `P(0)` satisfait `IC2` et `IC1`.
2. Cas $m > 0$: Supposons par hypothèse de récurrence que `P(m-1)` satisfait `IC2` et `IC1`. Prouvons que que `P(m)` satisfait `IC2` et `IC1`

   Considérons tout d'abord le cas où le général n'est pas byzantin. D'après la question précédente, `P(m)` satisfait la condition `IC2`. Comme `IC2` implique `IC1`, alors `P(m)` satisfait `IC2` et `IC1`.

   Maintenant, considérons tout d'abord le cas où le général est byzantin. Il y a $m-1$ lieutenants byzantins. Regardons le comportement de l'algorithme :

   (a) Au début, le général ℓ_0 exécute l'étape 1 à sa facon (il va envoyer à n'importe quelle valeur à n'importe qui selon sa guise). Ensuite les lieutenant vont rentrer dans l'étape 2 : chaque lieutenant exécute de nouveau `P(m-1)` en considérant qu'il est le général et qu'il a $n-2$ lieutenants. Remarquons, que

   $n > 3m$ implique $n - 2 > 3m - 2 > 3(m-1)$

   (b) Donc on peut appliquer l' hypothèse de récurrence sur les exécutions de l'algorithme `P(m-1)` : l'exécution de `P(m-1)` satisfait `IC2` et `IC1`. Considérons la valeur de v_j après pour l'exécution de `P(m-1)`

      i. Si j est un byzantin, alors pour toutes les valeurs v_j des lieutenant loyaux sont identiques (par `IC1`).

      ii. Si j n'est pas un byzantin, alors pour toutes les valeurs v_j des lieutenant loyaux sont identiques (par `IC2`).

      Donc

      tous les lieutenants loyaux possèdent la même liste $(v_0, \ldots, v_{n-1})$ (pour les éléments correspondants aux lieutenants loyaux)

   (c) Comme ils appliquent la même fonction majorité, ils obtiennent le même résultat. Ce qui vérifie la condition `IC2`

Question 3 : Calculer le nombre de messages échangés.
Correction : Soit n le nombre de lieutenants

algorithme `P(m)`	messages échangés
P(0)	$n-1$
P(1)	$(n-1)(n-2)$
P(2)	$(n-1)(n-2)(n-3)$
...	...
P(m)	$(n-1)(n-2)(n-3)\dots(n-m-1)$
P(m)	$O(n^m)$

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>`.

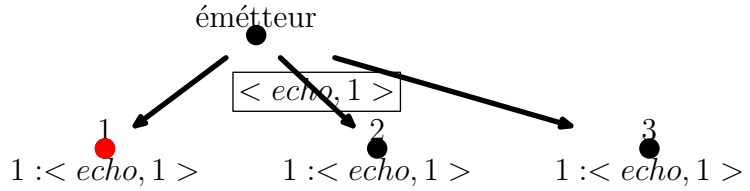
- Si un processus a reçu $\begin{cases} \text{soit} & \text{plus de } \frac{n+f}{2} \text{ messages de } \texttt{<ECHO,v>} \\ \text{ou soit} & \text{plus de } f \text{ messages } \texttt{<READY,v>}. \end{cases}$,
  alors il transmet à tous y compris à lui-même un message `<READY,v>`
- Si un processus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

Questions

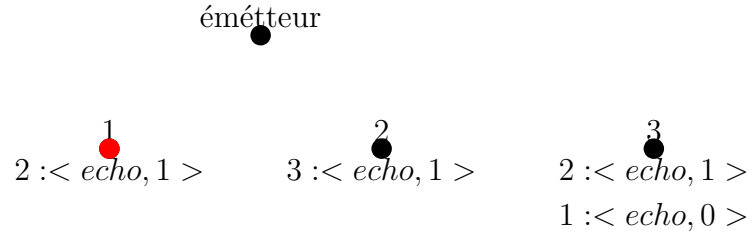
Question 0 : Exécuter sur un petit exemple ($n=4$, $f=1$) l'algorithme
Correction :
Supposons que le noeud 0 est l'émetteur et que le noeud 1 est byzantin

processeur	0	1	2	3
phase d'initialisation	émission de <INIT > émission de <ECHO,1 >		Reception de <INIT > R de <ECHO,v >	
phase réception	 3R <ECHO,1 > E <READY,1 > 3R <READY,1 > décide <1 >		E <ECHO,1 > 3R <ECHO,1 > E <READY,1 > 3R <READY,1 > décide <1 >	E <ECHO,1 > 3R <ECHO,1 > E <READY,1 > 3R <READY,1 > décide <1 >

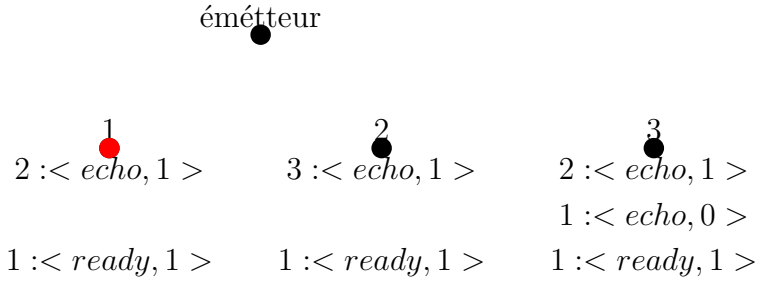
– phase d'initialisation :



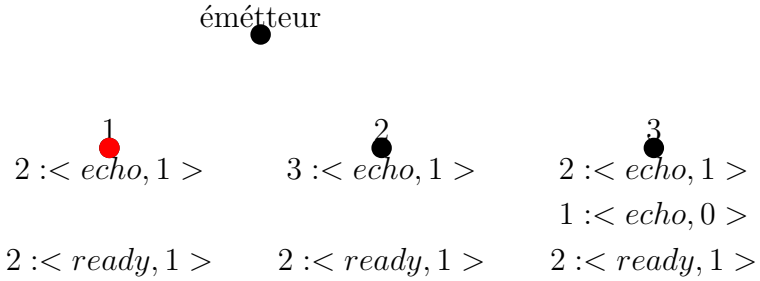
– le site 1 envoie `<ECHO,0>` à 3 et `<ECHO,1>` à 2 :



– le site 2 envoie `<READY,1>` à tout le monde :



– le site 2 envoie `<READY,1>` à tout le monde :



– les sites 2 et 3 décident donc 1

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.
Correction : Supposons que p et q envoient des messages `<READY>` avec des valeurs différentes (l'un envoie `<READY,1>`, l'autre envoie `<READY,0>`).

1. Cela signifie que p a reçu $\begin{cases} \text{soit} & \text{plus de } \frac{n+f}{2} \text{ messages de } \texttt{<ECHO,1>} \\ \text{ou soit} & \text{au moins } f+1 \text{ messages } \texttt{<READY,1>}. \end{cases}$

   Au moins un processus correct a envoyé un de ces messages incorrects (car il y a au plus f processus byzantins). Le même raisonnement peut s'appliquer pour q. q a reçu
   soit plus de $\frac{n+f}{2}$ messages de `<ECHO,0>`
   ou soit au moins $f+1$ messages `<READY,0>`.
   On peut aussi en conclure qu'au moins un processus correct a envoyé un de ces messages incorrects
2. Considérons le cas où p a reçu au moins $f+1$ messages `<READY,1>`. Au moins un processus p_1 correct a envoyé un message `<READY,1>`. Ainsi, on peut construire une suite des de processus $\{p = p_0, p_1, \ldots, p_i ...\}$ tel que

– le processus p_i a reçu au moins $f+1$ messages <READY,1>.
– le processus p_i a reçu un message <READY,1> d'un processus correct de p_{i+1}.
Comme le nombre de processus est borné, cette suite est finie. Cela signifie qu'il existe un processus correcte (autre que q) tel qu'il a reçu plus de $\frac{n+f}{2}$ messages de <ECHO,1>.

3. Donc deux processus corrects différents ont reçu plus de $\frac{n+f}{2}$ messages de <ECHO,0>,et plus de $\frac{n+f}{2}$ messages de <ECHO,1>.
Or l'intersection I de ces deux ensembles contient plus de f processus car leur somme est supérieure strictement à $n+f$ et n est le nombre de processus.
Donc cette intersection contient au moins un processus correct. Par conséquence, ceci est en contradiction avec le fait qu'un processus correct ne peut pas envoyer à la fois 0 et 1 (par définition d'un processus correcte).

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

Comme il y a au moins $n-f$ processus non-byzantins, d'après la question précédente, ils envoient les messages <READY,v> avec la même valeur. Un processus correct reçoit $n-f$ messages <READY,v>. Comme $n \geq 3f+1$, on a $n-f \geq 2f+1$. Alors tous les processus non-byzantins décident que v est la valeur diffusée car ils recoivent au moins $2f+1$ messages <READY,v>.

Donc deux processus corrects ne peuvent pas décider des valeurs différentes.

Si le processus correct p décide v, alors il a reçu $2f+1$ messages <READY,v> dont au moins $f+1$ messages proviennent de sites non-byzantins. Donc tout processus correct va lui aussi recevoir au moins $f+1$ messages et va envoyer un message <READY,v>. Ainsi au moins $n-f$ processus vons envoyer <READY,v>.

Ainsi au moins $n-f$ processus vont envoyer <READY,v> et tout processus non-byzantins recevra au moins $2f+1$ messages <READY,v> et décidra v.

Donc si un processus non-byzantin décide v alors tous les processus décident v

Question 3 : Que concluez-vous si l'émetteur n'est pas byzantin.
Correction : Si l'émetteur n'est pas byzantin, alors il émet la même valeur v à tous les autres processus. Donc après la réception du message de l'émetteur (communication fiable), tous autres les processus non-byzantins (au moins $n-f-1$) envoient le message <ECHO,v>. Ils recoivent au moins $1+n-f-1$ messages <ECHO,v>.

Il faut noter que $n-f \geq 2f+1$ et que $\frac{n-f}{2} \geq f+1/2$

$$\frac{n-f}{2} > f$$