Corrigé

Master 1 Informatique, 2006–2007

MI5a : Algorithmes distribués

(A. Slissenko)

Contrôle continu

Le 01 Décembre 2006
9h40 – 11h40

1. [3 points] (a) Donnez la définition de l'ordre causal. Donnez la définition des algorithmes d'élection. S'agit-il des algorithmes synchrones ou asynchrones ?

Réponse. Voir le cours.

(b) Les processus d'un algorithme par vague sont les sommets d'un graphe connexe. Le nombre des sommets est n. Il y a un seul initiateur p. Quelle est la longueur minimale d'une exécution dans laquelle p décide ? Peut-elle être plus petite que $2n$?

Réponse. L'initiateur p exécute au moins 3 événements : l'envoi du premier message (comme initiateur), la réception d'un message (à cause de la dépendance) et la décision. La dépendance dit qu'avant que l'initiateur décide il y a un événement dans chaque processus qui précède cette décision. Pour que tel événement se produise dans un processus $q \neq p$, ce q doit recevoir un message (car il n'est pas initiateur) et il doit envoyer un message vers p via une chaîne d'envoi/réceptions (dépendance). Donc on a au moins 2 événement dans chaque non-initiateur. En total cela donne $(3 + 2(n-1)) = 2n + 1$, où n est le nombre de processus.

2. [4 points]

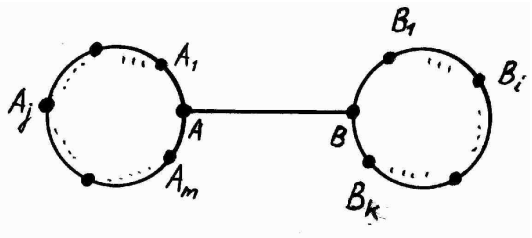

Fig. Φ.

Vous disposez d'un algorithme d'élection pour les anneaux uni-directionnels. C'est une boîte noire dont interface est disponible : on peut le lancer comme initiateur ou non-initiateur, on peut lui envoyer un message, on peut recevoir son message, on sait s'il décide, on sait s'il s'arrête. Décrivez un algorithme d'élection pour le graphe ayant la forme présentée sur la Fig. Φ et les canaux bi-directionnels, fondé sur l'algorithme d'élection pour les anneaux. Il vous faut redéfinir les voisins et le fonctionnement de l'algorithme dans les sommets A et B. Rappelons que votre algorithme doit marcher pour un ensemble d'initiateurs arbitraire.

Réponse. Soit α l'algorithme d'élection donné pour les canaux uni-directionnels. Le schéma de communication est sur la Fig. 1. Les carrés avec α représentent notre boîte noire d'élection. Les petits cercles représentent le traitement des communications par l'algorithme F qui gère l'utilisation de la boîte noire dans le sommets A et B. Il est le suivant. On fixe une orientation de notre nouveau réseaux, en utilisant le canal bi-directionnel AB dans deux sens et les autres canaux dans un seul sens. Considérons le fonctionnement du sommet A, pour B c'est pareil. Si A est un initiateur, alors F lance α comme initiateur, sinon α travaille comme non-initiateur. Si α envoie un message, alors F renvoie ce message selon la flèche, i. e. à $A1$. Si un message arrive de B, alors F le renvoie à α. Si α décide,

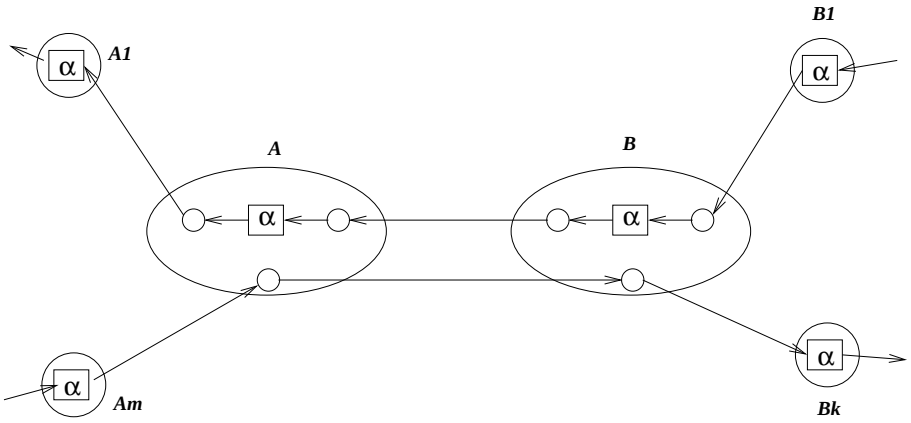

FIG. 1 – Organisation de communications

alors F décide. Si un message arrive de Am, alors F le renvoie à B. Remarquons que si α s'arrête, alors aucun message ne peut passer à travers α. Dans le cas où α s'arrête, F continue à renvoyer les messages qui arrivent de Am.

3. [5 points] Considérons 2 applications de l'algorithme de construction d'un arbre couvrant de Fig. 2 au graphe $G = (V, E)$, où $V = \{1, 2, \ldots, n\}$ et $E = \{\{1, 2\}, \{2, 3\}, \ldots, \{n-1, n\}$ (c'est une graphe-ligne). Prouvez que

(a) Si 1 est un seul initiateur, alors 1 décide.

(b) Si 1 et n sont initiateurs, alors l'un de deux initiateurs décide, et la fonction $father$ définit une forêt de deux arbres dont les racines sont les initiateurs. (Remarquons que le comportement de l'algorithme pour n arbitraire est pareil à son comportement pour $n = 3$, mais vous devez traiter le cas général).

Réponse.

(a) Le processus 1 envoie $\langle \mathbf{tok} \rangle$ à 2 selon les lignes 1, 2. Ensuite 2 reçoit $\langle \mathbf{tok} \rangle$ et le renvoie à 3 selon les lignes 9, 10 etc, jusqu'à n. Pendant ce passage i devient le père de $(i+1)$. Avoir atteint n l'algorithme envoie $\langle \mathbf{tok} \rangle$ vers 1 selon les lignes 11, 12. Avoir reçu $\langle \mathbf{tok} \rangle$ le processus 1 décide (lignes 4, 5).

(b) Les processus 1 et n envoient $\langle \mathbf{tok} \rangle$. Ces deux $\langle \mathbf{tok} \rangle$ avancent à la rencontre l'un de l'autre. Supposons que dans ce parcours le $\langle \mathbf{tok} \rangle$ de 1 atteint i et le $\langle \mathbf{tok} \rangle$ de n atteint $(i+1)$. Dans cette situation le processus i envoie $\langle \mathbf{tok} \rangle$ à $(i+1)$, et $(i+1)$ envoie $\langle \mathbf{tok} \rangle$ à i (lignes 9, 10).

Considérons le comportement de i. Avoir reçu $\langle \mathbf{tok} \rangle$ de $(i+1)$ il trouve que la garde le la ligne 6 est fausse. Donc il envoie $\langle \mathbf{tok} \rangle$ vers 1. Ce $\langle \mathbf{tok} \rangle$ arrive à 1 qui décide.

Pareil pour $(i+1)$, mais cette fois c'est n qui décide.

Donc on a même 2 processus qui décident, et on voit que la fonction $father$ définit 2 arbres : l'un avec la racine 1 et la feuille i, et l'autre avec la racine n et la feuille $(i+1)$. (Remarquons que les cas extrêmes, par exemple $i = 1$, ne sont pas exclus.)

4. [4 points] Donnez la définition de la signature numérique. Donnez la définition du consensus Byzantin.

Considérons la modification suivante de l'algorithme du cours de consensus byzantin avec signatures numériques. La modification concerne la notion de message valide :

On dit qu'un message $m = \langle \mathbf{val}, v \rangle : g : p_2 : \ldots : p_i$ reçu pendant l'impulsion i par le processus p est *valide* pour p si m a les signatures correctes de i processus dont le premier est le général, et p lui-même n'est pas parmi les processus qui ont signé le message (donc on ne demande pas que les signatures soient différentes).

À part de ces modifications l'algorithme reste le même. On a N processus dont $t < N$ sont malveillants. A l'impulsion 1 le général diffuse le message $\langle \mathbf{val}, x_g \rangle : g$ qui contient son entrée x_g et qui est signé par lui-même. Pendant les impulsions $2, 3, ..., (t+1)$ chaque processus correct p signe et renvoie 2 premiers messages valides reçus précédemment et met les valeurs contenues dans ces messages dans son ensemble W_p qui est initialement vide. La décision est faite à la fin de l'impulsion (t+1) et est égale à v si $W_p = \{v\}$, et est égale à 0 si $|W_p| > 1$.

Pour cette modification de l'algorithme prouvez que quelque soit N il y a une exécution sans consensus.

Quelle est la résistance et la complexité en messages de l'algorithme correct de consensus avec 2 renvois de messages ?

Réponse. Pour la signature numérique, le consensus Byzantin, la résistance et la complexité voir le cours.

Une exécution qui montre qu'avec la notion de la validité de message modifiée l'algorithme ne marche pas est la suivante. On suppose qu'il y a 2 processus correct, disons p et q (sinon on a toujours un consensus). Le général, qui est supposé d'être byzantin, ne fait rien jusqu'à l'impulsion $(t+1)$. Pendant l'impulsion $(t+1)$ il envoie à p un message $\langle \mathbf{val}, u \rangle : g : g : \ldots : g$ avec $(t+1)$ signatures, et envoie à q un message $\langle \mathbf{val}, w \rangle : g : g : \ldots : g$ avec $(t+1)$ signatures, où $u \neq w$. Donc à la fin de l'impulsion $(t+1)$ le processus p décide de u et q décide de w.

Remarquons que si le général est correct il envoie à tous la même valeur et personne ne peut falsifier sa signature et, en conséquence, personne ne peut changer la valeur envoyée dans les messages.

5. [4 points] Alice a N cartes. Bob veut choisir lui-même une carte, Alice aussi, et de plus chacun veut garder une copie du reste du jeu de cartes pour vérifier les choix plus tard quand les clefs de cryptages utilisées seront révélées. Bien sûr les choix doivent être faits sans voir les cartes, et chacun veut garder sa carte secrète jusqu'à la vérification finale.

Décrivez un protocole pour faire les choix en formules. Précisez le secret des clefs et les propriétés des cryptages utilisés.

Réponse. Les cryptages utilisés doivent être commutatifs. Les clefs utilisées doivent rester secrètes. Si, par exemple, Alice utilise un cryptage asymétrique et envoie à Bob les cartes cryptées avec sa clef publique, alors Bob peut choisir une carte en claire et crypter les cartes restantes avec la même clef publique d'Alice. Soit KA et KA^{-1} les clefs d'Alice et KB et KB^{-1} les clefs de Bob.

Soit $\{C_i\}_{i \in J}$ l'ensemble de cartes, où J est un ensemble d'indices, par exemple $J = \{1, 2, \ldots, 52\}$. On suppose que l'ordre donnés par la numérotation $C_1, C_2, \ldots, C_{|J|}$ est aléatoire pour qu'on ne puisse pas deviner de la valeur d'une carte à partir du numéro de message.

1. $A \to B : \{m_i\}_{i \in J}$, où $m_i = \{C_i\}_{KA}$.
2. B choisit $i_B \in J$.
3. $B \to A : \{m_{i_B}\}_{KB}$.
4. $A \to B : \{C_{i_B}\}_{KB} = \{\{m_{i_B}\}_{KB}\}_{KA^{-1}}$
5. B a sa carte C_{i_B}.
6. $B \to A : \{\{m_i\}_{KB}\}_{i \in (J \setminus i_B)}$. B garde une copie $\{m_i\}_{i \in J}$ et son choix.
7. A choisit $i_A \in (J \setminus i_B)$
8. $A \to B : \{m_{i_A}\}_{KB}$.
9. $B \to A : m_{i_A} = \{\{m_i\}_{KB}\}_{KB^{-1}}$
10. A a sa carte $C_{i_A} = \{m_{i_A}\}_{KA^{-1}}$.

6. [5 points] Considérons l'exécution de l'algorithme de synchronisation d'horloges (voir Fig. 3) pour $N = 2K + B$ processus dont $\{1, 2, \ldots, 2K\}$ sont corrects, et $\mathcal{B} = \{2K+1, \ldots, N\}$ sont byzantins. On suppose que le moment du début de travail est 0, $\delta_{min} = \delta_{max} = 0$ et la précision de la synchronisation

initiale δ est arbitraire (constante abstraite). Divisons les corrects en 2 ensembles : $\mathcal{C}_1 = \{1, \ldots, K\}$ et $\mathcal{C}_2 = \{K+1, \ldots, 2K\}$. La valeur initiale d'horloges des processus de $\mathcal{C}_1$ est v_1 et celle des processus de $\mathcal{C}_2$ est $v_2 = v_1 + \delta$. Les processus byzantins envoient aux processus de $\mathcal{C}_1$ la valeur $(v_1 - \delta)$ et aux processus de $\mathcal{C}_2$ la valeur $(v_1 + \delta)$. Trouvez la correction Δ_p pour les p corrects. Distinguez 2 cas : $B < \frac{N}{3}$ et $B \geq \frac{N}{3}$.

Réponse.

Les tableaux $D_p(q)$ et $A_p(q)$ initiaux pour p correct :
$D_p(q) = A_p(q) = 0$ pour $p, q \in \mathcal{C}_1$,
$D_p(q) = A_p(q) = \delta$ pour $p \in \mathcal{C}_1$ et $q \in \mathcal{C}_2$,
$D_p(q) = -\delta$ pour $p \in \mathcal{C}_1$ et $q \in \mathcal{B}$,
$D_p(q) = A_p(q) = 0$ pour $p, q \in \mathcal{C}_2$,
$D_p(q) = A_p(q) = -\delta$ pour $p \in \mathcal{C}_2$ et $q \in \mathcal{C}_1$,
$D_p(q) = 0$ pour $p \in \mathcal{C}_2$ et $q \in \mathcal{B}$.

Maintenant il nous faut calculer $A_p(q)$ pour $q \in \mathcal{B}$. Prenons max comme estimateur.

Cas $B < \frac{N}{3}$. Pour $p \in \mathcal{C}_1$ la valeur $-\delta$ a comme témoins les processus de $\mathcal{C}_1$ et de $\mathcal{B}$, mais pas de $\mathcal{C}_2$ (car $|-\delta - \delta| = 2\delta > \delta$). Remarquons que $K = \frac{N-B}{2}$. En total on a $(K+B) = \frac{N-B}{2} + B = \frac{N+B}{2}$ témoins. Pour retenir une valeur, il faut avoir au moins $(N-B)$ témoins. Dans notre cas il nous faut $\frac{N+B}{2} \geq (N-B)$ ce qui est équivalent à $B \geq \frac{N}{3}$.

Donc les processus de $\mathcal{C}_1$ remplacent les valeurs des byzantins par l'estimateur : $A_p(q) = \max\{0, \delta\} = \delta$ pour $p \in \mathcal{C}_1$ et $q \in \mathcal{B}$. La correction d'horloge pour $p \in \mathcal{C}_1$:

$\Delta_p = \frac{1}{N}\big(K \cdot 0 + K \cdot \delta + B \cdot \delta\big) = \frac{K+B}{N}\delta$.

Pour $p \in \mathcal{C}_2$ toutes les valeurs seront retenues. Ainsi, pour $p \in \mathcal{C}_2$ on a
$\Delta_p = \frac{1}{N}\big(K \cdot (-\delta) + K \cdot 0 + B \cdot 0\big) = -\frac{K}{N}\delta$.

Après la correction on a $C_p = v_1 + \frac{K+B}{N}\delta$ pour $p \in \mathcal{C}_1$, et $C_p = v_1 + \delta - \frac{K}{N}\delta$ pour $p \in \mathcal{C}_2$. La précision de synchronisation après la correction est

$|v_1 + \frac{K+B}{N}\delta - (v_1 + \delta - \frac{K}{N}\delta)| = |\frac{2K+B}{N} - 1| \cdot \delta = 0$.

Cas $B \geq \frac{N}{3}$. Dans ce cas, selon le calcul ci-dessus, il y a assez de témoins pour retenir toutes les valeurs reçues initialement. Donc $A_p(q) = D_p(q)$. Les corrections :

• pour $p \in \mathcal{C}_1$:
$\Delta_p = \frac{1}{N}\big(K \cdot 0 + K \cdot \delta + B \cdot (-\delta)\big) = \frac{K-B}{N}\delta$

• pour $p \in \mathcal{C}_2$: $\Delta_p = -\frac{K}{N}\delta$ (on reprends le cas précédent).

La précision de synchronisation après la correction dans le cas 2 est
$|v_1 + \frac{K-B}{N} - (v_1 + \delta - \frac{K}{N})|\delta = (1 - \frac{2K-B}{N})\delta = \frac{2B}{N}\delta$.
On voit que si B est assez grand les byzantins peuvent même désynchroniser davantage les horloges des corrects.

var $used_p[q]$: *boolean* **init** $false$ $\forall\, q \in Neigh_p$;
(* Indique si p a déjà envoyé $\langle$**tok**$\rangle$ à q *)
$father_p$: process **init** undef ;

Seulement pour l'initiateur (qui est unique) exécuter une fois :

begin
1 : $father_p := p$; **choose** $q \in Neigh_p$;
2 : $used_p[q] := true$; **send** $\langle$**tok**$\rangle$ à q
end

Pour tout processus après une réception de $\langle$**tok**$\rangle$ de q_0 :

begin
3 : **if** $father_p = undef$ **then** $father_p := q_0$;
4 : **if** $\forall\, q \in Neigh_p .\, used_p[q]$
5 : **then** *decide*
else
6 : **if** $\exists\, q \in Neigh_p . (q \neq father_p \wedge \neg\, used_p[q])$
then
begin
7 : **if** $father_p \neq q_0 \wedge \neg\, used_p[q_0]$
8 : **then** $q := q_0$
9 : **else choose** $q \in Neigh_p \setminus \{father_p\}$ tel que $\neg\, used_p[q]$;
10 : $used_p[q] := true$; **send** $\langle$**tok**$\rangle$ à q
end
else
begin
11 : $used_p[father_p] := true$;
12 : **send** $\langle$**tok**$\rangle$ à $father_p$
end
end

FIG. 2 – Recherche en profondeur classique.

$\underline{ClockSync_p}$:

Tout processus commence son travail après avoir reçu le message le lancement.

0 : **par forall** q **do** $A_p(q) = D_p(q) := \infty$ **endpar** ;
-- Initialisation des tableaux A_p et D_p ;

1 : $H_p := C_p$;

-- sauvegarde du temps du début de travail

2 : **par forall** q **do Send** $\langle \mathbf{val}, H_p \rangle$ **to** q **endpar** ;

3 : **while** $C_p \leq H_p + 2\delta_{max}$ -- Ici C_p joue le rôle du temps courant.
par forall q **do**
if $\langle \mathbf{val}, H \rangle$ est reçu de q
then $D_p(q) := (H - H_p)$
endpar ;
endwhile ;

4 : **par forall** q **do**
if $\#\{r : |D_p(q) - D_p(r)| \leq \delta + (\delta_{max} - \delta_{min})\} \geq (N - B)$
then $A_p(q) := D_p(q)$
endpar ;

5a : $i, q := 1$;

5 : **while** $q \leq N$ **do**
if $A_p(q) < \infty$ **then** $A'_p(i) := A_p(q)$; $i := i + 1$ **endif** ;
$q := q + 1$;
endwhile ;

6 : $esti_p := estimator(A'_p)$;

7 : **par forall** q **do**
if $A_p(q) = \infty$ **then** $A_p(q) := esti_p$
endpar ;

8 : $\Delta_p := \frac{1}{N} \sum A_p$;

9 : $C_p := C_p + \Delta_p$

FIG. 3 – Algorithme de synchronisation d'horloges