Examen d'algorithmique répartie
(19 mars 2012)
(documents manuscrits autorisés-2 pages)

Exercice 1.— On considère un système de n processus $\Pi : \{p_1, \dots, p_n\}$ sans défaillance communiquant par envoi-réception de messages. On suppose que la communication est point à point et qu'il n'y a pas de pertes de messages. On suppose de plus que les liens de communication sont bidirectionnels. Soit $G = \langle \Pi, A \rangle$ le graphe de communication : $\{p, q\}$ est une arête si et seulement il y a un lien de communication entre p et q . On suppose que G est un graphe connexe et que chaque processus ne connaît que son identité et celle de ses voisins (en particulier il ne connaît pas n le nombre de processus). On veut définir un algorithme permettant de calculer le nombre n de processus sur le réseau.

1. Supposons que G est un anneau, proposer un algorithme permettant à chaque processus de calculer n . (Dans cet algorithme les processus peuvent uniquement envoyer et recevoir des messages de leur voisins sur l'anneau). Combien de messages sont échangés avant que tous les processus aient calculé n ?
2. Supposons que l'on dispose d'un algorithme de vague fonctionnant sur G , comment utiliser cet algorithme de vague pour assurer qu'au moins un des processus calcule n .
3. On suppose maintenant que G est arbre (graphe connexe et sans cycle), proposer un algorithme efficace permettant à chaque processus de calculer n . Quel est le nombre de messages échangés? (on pourra utiliser l'algorithme de vague sur un arbre vu en cours).

Exercice 2.— Soit $\Pi = \{p_1, \dots, p_n\}$ un ensemble de processus communiquant par envoi-réception de messages. La communication est point à point, synchrone et sans perte de messages. Le graphe de communication est ici le graphe complet (tout processus peut communiquer avec tout processus). Les défaillances des processus sont quelconques (byzantines) on suppose qu'il y a au plus t processus byzantins. Dans la suite $\perp$ est une valeur particulière qui ne peut pas être une valeur initiale des processus.

1. On considère les spécifications suivantes de problèmes de décision (irrévocable) :
 - consensus : chaque processus p a une valeur initiale v_p , la décision doit assurer les propriétés suivantes : (1) terminaison : tous les processus corrects décident, (2) accord : pour tout processus correct p et q si p décide v et q décide w alors $v = w$ et (3) intégrité : si tous les processus corrects ont la même valeur initiale v alors un processus correct doit décider v .
 - consensus généralisé : il s'agit d'un problème de décision sur un vecteur. Plus précisément, chaque processus p a une valeur initiale v_p et un vecteur local à p , V_p , indicé par Π tel que chaque élément $V_p[q]$ ne peut être écrit qu'une seule fois et qui est initialisé à $\perp$ pour chaque coordonnée. L'algorithme doit assurer : (1) terminaison : tous les processus corrects terminent (2) accord sur le vecteur : pour tout p et q et r , si $V_p[r] \neq \perp$ et $V_q[r] \neq \perp$ alors $V_p[r] = V_q[r]$ et (3) validité : si p est correct alors $V_p[p] = v_p$

- *diffusion fiable pour s (DF_s)* : il s'agit d'un problème de décision où s a une valeur initiale m et le protocole doit assurer (1) *terminaison* : tous les processus corrects décident (2) *validité pour DF_s* : si s est correct la seule décision pour un processus correct est m et (3) *accord pour DF_s* si p et q sont corrects alors ils décident la même valeur (qui peut être $\perp$ si s n'est pas un processus correct).

- (a) Soit $\mathcal{A}$ un algorithme qui résout le consensus, construire à partir de $\mathcal{A}$ un algorithme qui résout le consensus généralisé.
- (b) Soit $\mathcal{A}$ un algorithme qui résout le consensus généralisé, construire à partir de $\mathcal{A}$ un algorithme qui résout la diffusion fiable pour p .
- (c) Soient $\mathcal{A}_p$ des algorithmes qui résolvent la diffusion fiable pour chaque p , construire à partir de ces $\mathcal{A}_p$ un algorithme qui résout le consensus généralisé.
- (d) Soit $\mathcal{A}$ un algorithme qui résout le consensus généralisé, en supposant que moins de la moitié des processus peuvent être incorrects ($n > 2t$) construire à partir de $\mathcal{A}$ un algorithme qui résout le consensus.

2. L'accord croisés est un problème de décision, pour lequel un processus particulier p_0 a une valeur initiale v_{p_0} , et qui doit assurer : (1) *terminaison* : tout processus correct décide, (2) *accord faible* pour tout processus correct p et q si p décide w et q correct décide w' alors soit $w = w'$ soit $w = \perp$ ou $w' = \perp$, (3) *validité* si p_0 est correct alors la seule décision possible pour un processus correct est v_{p_0} .

- (a) En supposant que $n > 3t$ proposer une solution en deux rondes pour résoudre l'accord des croisés. (On pourra s'inspirer des deux premières phases de la diffusion authentifiée).

$$V_{p_i} = f(V_{p_j}, id_{p_j}, id_{p_i})$$

et v_i ;

$$V(\tau)$$

$$v_i = f(v_j, id_{p_j}, id_{p_i})$$

soit $V_{p_i} = f(V_{p_j}, id_{p_j}, id_{p_i})$

(recevoir V_{p_j})

(recevoir v_j)

$$V_{p_i} = f(v_j, id_{p_j}, id_{p_i})$$

$$v_i = f(v_j, id_{p_j}, id_{p_i})$$

Terminaison

$v_i = v_j$