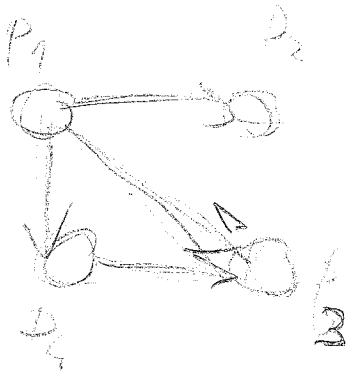
Université Denis Diderot Paris 7
UFR d'informatique
M2-algorithmique répartie



Examen d'algorithmique répartie
(14 mars 2010)
(documents autorisés-1 page)

Exercice 1.— On considère un système de n processus $\Pi : \{p_1, \cdots, p_n\}$ 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. Soit G le graphe de communication : (p, q) pour $p \in \Pi$ et $q \in \Pi$ est un arc de G si et seulement si il y a un lien de communication de p vers q. On suppose que G est un graphe fortement connexe et qu'un processus p connaît uniquement sa propre identité.

On veut définir un algorithme dans lequel tous les processus déterminent l'ensemble des identités du système : au départ chaque processus connaît uniquement sa propre identité et quand l'algorithme termine il connaîtra toutes les identités dans le système. Pour cela, chaque processus p maintient une variable locale ID_p initialisée à $\{p\}$, l'algorithme doit assurer (1) qu'il termine toujours et (2) quand il termine pour chaque processus p la variable ID_p est égale à Π (p connaît toutes les identités présentes dans le système).

1. Si G est un anneau orienté, proposer un algorihtme vérifiant les conditions précédentes.
2. Supposons que l'on dispose d'un algorithme de vagues pour un système ayant G comme graphe de communication, modifier (en modifiant le contenu des messages transmis par cet algorithme et en ajoutant du code local à chaque processus) cet algorithme pour obtenir un algorithme vérifiant les conditions précédentes (on notera e_p le premier événement de l'algorithme de vague et d_q une décision)
3. Montrer que de tout algorithme vérifiant les conditions (1) et (2) précédentes, on peut extraire un algorithme de vagues.

Exercice 2.— Soit $\Pi = \{p_1, \ldots, p_n\}$ 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 ici des crashs. On suppose que l'on dispose d'un algorithme de consensus, c'est-à-dire un algorithme de décision qui assure les propriétés de (1) *terminaison* : tous les processus corrects terminent, (2)*accord* : pour tout processus p, q si p décide v et q décide w alors $v = w$ et (3) *intégrité* : pour tout p, si p décide v alors v est la valeur initiale d'un processus.

On veut obtenir un algrorithme de *consensus généralisé* : il s'agit d'un algorithme 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 $V_p[p] = v_p$.

1. A partir d'un algorithme de consensus définir un algorithme de consensus généralisé.
2. A partir d'un algorithme de consensus généralisé construire un algorithme de consensus.