Université Paris 7-Denis Diderot
M2 informatique:
Algorithmes distribués
2013-2014

Devoir 2:
à rendre pour le 18/3/2013
Chaque devoir peut avoir jusqu'à 3 auteurs

Ce deuxième devoir est une application du devoir précédent: à partir de l'algorithme de phases étudié dans le devoir on va construire une table de routage. Rappelons tout d'abord l'algorithme de phases.

Hypothèses Le réseau est représenté par un graphe orienté $G =< \Pi, V >$ fortement connexe. On notera $d(p,q)$ la longueur du plus court chemin de p à q (attention, en général $d(p,q) \neq d(q,p)$!). D le *diamètre* est défini par $D = \max\{d(p,q)|p,q \in \Pi\}$. Pour cet algorithme on suppose que D est connu des processus. Pour chaque processus p, $Out_p = \{q \in \Pi|(p,q) \in V\}$ est l'ensemble des processus auxquels p peut envoyer directement un message. $\preceq$ est la relation de causalité entre événements. Dans ce qui suit la communication est FIFO (les messages entre deux noeuds sont reçus suivant leur ordre d'émission).

Dans la suite toutes les variables des algorithme sont locales, l'indice indiquant l'identité du processus pour lequel la variable est définie (Var_p est locale au processus p).

Initialisations:
1 $\forall q \in In_p : Rec_p[q] := 0$
2 $Phase_p := 0$
CODE POUR LE PROCESSUS p:

3 $\{<>$ de $q\ \} \longrightarrow$ recevoir $<>$;
4 $\quad Rec_p[q] := Rec_p[q] + 1$

5 $\{\forall q \in In_p : Rec_p[q] \geq Phase_p \wedge Phase_p < D\} \longrightarrow$ send$<>$ à tous les $q \in Out_p$
6 $\quad Phase_p := Phase_p + 1$

7 $\{\forall q \in In_p : Rec_p[q] \geq D \wedge Phase_p \geq D\} \longrightarrow$ decider

---

Figure 1: Algorithme des phases

Il s'agit ici de calculer des tables de routage minimale à partir de l'algorithme de phases. Dans une table de routage R_p pour le processus p, $R_p[q]$ indique vers quel lien router les messages à destination de q: si $R_p[q] = r$, alors la première étape de la route de p vers q utilise le lien de p vers r (ce qui suppose que $r \in Out_p$). On suppose que $R_p[p] = p$ (les messages à destination de p depuis p sont délivrés directement!).

L'algorithme de phases a une propriété particulière qui est qu'on peut faire en sorte que le processus p entend parler pour le première fois d'un processus q au cours de la phase $d(p,q)$ pour p. Ainsi, si le premier message issu de q est (q,r) indiquant que ce premier message a emprunté le lien de q à r, et si au cours de l'algorithme de phases tous ces messages sont retransmis, la première fois que p reçoit un message de l'algorithme de phases contenant (q,x) pour un certain x, alors p sait que le plus court chemin de q à p a comme première étape le lien de q à x. En d'autres termes, $R_q[p] = x$ (la route des messages de q vers p commence par x). On peut remarquer que c'est p et non q qui a cette information. Aussi, dans un premier

temps, avec l'algorithme de phases chaque p va calculer une table T_p qui est telle que $T_p[q] = R_q[p]$ (p enregistre dans sa table T_p que la première étape de la route de q à p est $T_p[q]$). Dans un deuxième temps les informations pour le routage seront transmises aux processus concernés.

---

1 $i_p := \{(p,p)\}; T_p[p] := p$
2 $phase_p := 1$;
3 **forall** $j \in Out_p$ **do** $send(\{(p,j)\})$ **to** j
4 **while** *true* **do**
5 **if** tous les liens entrants contiennent un message **then**
6 $ancien_p := i_p$
7 Consommer un message sur chaque lien entrant
 /* chaque message est un ensemble (qui peut-être vide) de couples (i,j) */
8 $i_p := i_p \cup \{x | x$ lu dans l'instruction ligne 7 $\}$
9 $nouveau_p := \{(x,y) \in i_p | \nexists (x,z) \in ancien_p\}$
10 **forall** $(q,j) \in nouveau_p$ **do** $T_p[q] := j$
11 $phase_p := phase_p + 1$
12 **if** $phase_p \leq D$ **then**
13 **forall** $j \in Out_p$ **do** $send(nouveau_p)$ **to** j
14 **else** *termine*

Figure 2: Routage Minimal

Questions

On note i_p^k la valeur de i_p juste après la ligne 8 quand $phase_p = k$; i_p^0 est la valeur de i_p pour la phase 0 (Ligne 1).

1. Par induction sur k prouvez que: si $k = d(q,p)$ alors $q \in i_p^{d(q,p)}$.
2. Par induction sur k montrez que: si $d(q,p) > k$ alors $q \notin i_p^k$
3. En déduire que p n'écrit $T_p[q]$ qu'une seule fois et ce dans la phase égale à $d(q,p)$
4. Montrer par une induction sur $d(q,p)$ que si p écrit $T_p[q] = j$ alors j est la première étape d'un chemin de q à p.
5. Montrer que l'algorithme termine (on pourra utiliser les résultats prouvés dans le devoir 1). En déduire que chaque processus p termine avec une table T_p telle que $T_p[q]$ est la première étape d'une route de longueur minimale de q vers p.
6. Evaluez la complexité en nombre de messages de l'algorithme. En supposant que les identités des processus sont des entiers consécutifs $1, \ldots n$. Montrer que le nombre de bits échangés dans l'algorithme est en $O(en\log(n))$ (e étant le nombre de liens de communications).
7. Dans l'algorithme proposé les informations sur le routage ne sont pas connues des processus qui doivent assurer le routage: p connaît la première étape d'une route de longueur minimal de q vers p, mais cette première étape de la route doit être réalisée par q pas par p.

   Proposer un algorithme permettant à partir des tables $T_p[q]$ de construire des tables de routage c'est-à-dire des tables $R_p[q]$ (locales au processus p) telles que $R_p[q] = T_q[p]$.

   En considérant la "composition de ces deux algorithmes" combien de messages sont échangés? Quel est l'ordre de grandeur du nombre de bits échangés?