ISTG - RICM2
Examen d'algorithmique distribuée
Durée 2h

Tous documents autorisés – Le barême est indicatif

23 avril 2002

1) Anneau (12 pts)

On se donne un réseau où les noeuds sont identifiés uniquement et sont reliés en anneau via des liaisons unidirectionnelles fiables. Il n'y a pas de relation entre l'identité d'un noeud et sa position dans l'anneau. Tout site peut émettre un message de données sans contrainte sur sa liaison de sortie (pas de jeton d'émission). Tout message porte le numéro du site destinataire. Chaque site est responsable de l'acheminement d'un message jusqu'à son destinataire selon le schéma suivant. Un message de donnée (étiquette `MSG`) reçu sur la liaison entrante `in` d'un site est traité localement si le site est le destinataire sinon est propagé sur la liaison de sortie `out`.

```
repeter alt
      besoin_émettre(dst,data)        ==> out!<MSG,dst,data>
      in?<MSG, dest,data> && dest!=ICI  ==> out!<MSG, dest, data>
      in?<MSG, dest,data> && dest==ICI  ==> traitement_local(data)
```

Figure 1: Routage en anneau

1.1) Vague sur anneau (2 pts) Rappeler l'algorithme permettant à tout noeud d'émettre une vague (étiquette `TOK`) sur l'anneau indépendamment des vagues issues des autres noeuds.
Il suffit de propager la vague si on n'est pas initiateur

```
repeter alt
      (decision_vague==true) ==> out!<TOK, ICI,...>decision_vague=false
      in?<TOK, I,> && I!=ICI  ==> out!<TOK, ICI,...> //propagation
      in?<TOK, I,> && I==ICI  ==>  fin_vague=true
```

1.2) Découverte des noeuds(2 pts) Proposer une spécialisation de vague qui permette aux noeuds de découvrir les identités des membres du réseau. N'importe quel noeud peut initier une vague de découverte (étiquette `DISC`) qui enregistre les identités des noeuds traversés. Le résultat de la vague est la liste des noeuds dans l'ordre de parcours.
Il suffit de rajouter son identité dans la vague et propager si on n'est pas initiateur

```
repeter alt
      (anneau_inconnu==true) ==>  out!<DISC, ICI,<ICI>>
                        anneau_inconnu=false;anneau_connu=false;
      in?<DISC, I,lst> && I!=ICI  ==> lst.append(ICI);out!<DISC,I, lst>
      in?<DISC, I,lst> && I==ICI  ==>
                Composition_anneau=lst; anneau_connu=true;
```

1.3) Elimination de vagues(2 pts) Dans le cas où des noeuds initient concurremment des vagues ayant le même rôle, on surcharge le réseau inutilement. Une solution est de ne conserver qu'une vague (élection) parmi les vagues en cours. Proposer une vague à extinction (étiquette `ELEC`)qui en cas de plusieurs vagues issues d'initiateurs différents[1] ne garde qu'une vague (par exemple celle de l'initiateur d'identité la plus élevée). **Un noeud laisse passser les vagues gagnantes et élimine les perdantes. L'algo le + simple est de considérer qu'à la reception d'une vague, le noeud récepteur devient candidat s'il ne l'est pas déja).**

```
repeter alt
      (decision_vague==true) ==>
            out!<ELEC, ICI,...> ; decision_vague=false ; election = true;
      in?<ELEC, I> && I<ICI   ==> // I gagne
            out!<ELEC, I,...> ; decision_vague=false; election = true;
      in?<ELEC, I> && I>ICI   ==> // ICI gagne
            if(!election) {out!<ELEC, ICI,...> ; election = true;} // on participe toujours
            decision_vague=false ;
      in?<ELEC, I,....> && I==ICI  ==> fin_vague=true ; election = false; // elu !!
```

1.4)Découverte des noeuds(2 pts) Modifier l'algorithme de découverte du réseau par de multiple vagues indépendantes pour utiliser une vague à extinction. A la fin de l'exécution de cet algorithme, tous les noeuds du réseau doivent connaitre la constitution de l'anneau.

***** donner brièvement l'idée générale de ce nouvel algorithme ? En combien de tours d'anneau, tous les noeuds sont ils informés de la constitution de l'anneau?

Un premier tour d'élection permet de ne garder qu'une vague et d'apprendre. Le noeud élu connait le réseau. Un second tour lui permet d'informer les autres noeuds. ***** donner l'algorithme

```
repeter alt
      (anneau_inconnu==true) ==>  out!<DISC1, ICI,<ICI>>
                        anneau_inconnu=false;anneau_connu=false;
      in?<DISC1, I,lst> && I<ICI  ==> // I gagne
            lst.append(ICI);out!<DISC,I, list>
            anneau_inconnu=true;anneau_connu=false;
      in?<DISC1, I,lst> && I>ICI  ==> // ICI gagne
            if(anneau_inconnu){lst.append(ICI);out!<DISC,I, lst>}
            anneau_inconnu=false;anneau_connu=false;
      in?<DISC1, I,lst> && I==ICI  ==>  // fin premier tour
                Composition_anneau=lst; anneau_connu=true;
                out!<DISC2,I, lst> // deuxième tour
      in?<DISC2, I,lst> && I!=ICI  ==>
                 Composition_anneau=lst; anneau_connu=true;
                out!<DISC2,I, lst> // progagation
     in?<DISC2, I,lst> && I==ICI  ==> // fin deuxième tour
```

1.5)Snapshot sur anneau FIFO (4 pts) On se pose une variante du problème du cliché (snapshot) sur un anneau à liaisons monodirectionnelles. On se limite au cas où les liaisons sont FIFO fiables. On considère qu'un seul site peut initier un cliché. Le cliché doit être pris en un tour d'anneau. Aucune autre demande de cliché ne peut être émise avant fin du cliché en cours. A son retour à l'initiateur, le message de cliché (préfixé par `SHOT`) contient le cliché complet.

***** que doit enregistrer l'initiateur après émission initiale du message de cliché `SHOT`?

tous les messages reçus avant retour de la vague

[1] Un initiateur n'émet qu'une vague à la fois.

et pourquoi?
Le jeton quitte l'initiateur avant leur réception et visitera tous les noeuds émetteur après leur émission (propriété FIFO)

***** que doit enregistrer un site recevant le message `SHOT` ? **son état local** au moment de la réception ? après ?**rien** et pourquoi?**tous les messages reçus ont été émis après passage du jeton (propriété FIFO)**

***** que contient le message de cliché ?**liste états locaux**

***** Compléter le comportement des sites pour les demandes de cliché issues de l'initiateur 0 .

```
repeter alt
      besoin_émettre(dst,data)        ==> out!<MSG,dst,data>
      in?<MSG, dest, data> && dest!=ICI ==>
   if (cliché_en_cours) MSG_SET= MSG_SET U <dest,data>;
         out!<MSG, dest, data>
      in?<MSG, dest,data> && dest==ICI ==>
         if (cliché_en_cours) MSG_SET= MSG_SET U <dest,data>;
         traitement_local(data)
      decision_de_cliché && !cliché_en_cours  ==>cliché_en_cours=true;
         out!<SHOT, ICI, {etat_local()}>; MSG_SET={};
      in?<SHOT,I,states>  && I!=ICI  ==> out!<SHOT, ICI, states U {etat_local()}>
      in?<SHOT,I,states>  && I==ICI  ==> cliché_en_cours=false; SNAPSHOT=states U MSG_SET;
```

2) Anneau cordal(10 pts)

L'inconvénient d'un anneau est que les délais de transmission peuvent être importants pour les noeuds les plus éloignés. Une façon d'accélérer les transmissions est de relier quelques noeuds distants par des liaisons supplémentaires : des cordes. Dans un schéma de ce type (cf. figure 2), il y des noeuds normaux, des noeuds routeurs et des noeuds concentrateurs. Les routeurs doivent choisir la route la plus courte pour atteindre le noeud destinataire : la voie normale ou une corde. Un noeud peut être routeur et concentrateur. On notera `Cout` l'ensemble des cordes sortantes d'un noeud et `Cin` l'ensemble des cordes entrantes. Ces ensembles peuvent être vides selon le type du noeud.

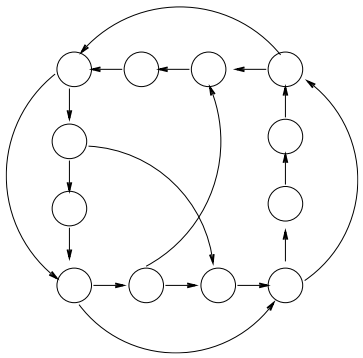

Figure 2: Un anneau cordal

2.1) Découverte des noeuds concentrateurs(3 pts) A la réception d'un message, ces noeuds devront donc décider de reémettre sur une corde ou sur la liaison normale. Pour cela, un routeur doit découvrir les identités des noeuds concentrateurs au bout de ses cordes. Connaissant la composition de l'anneau, il pourra donc décider de prendre une corde ou non pour atteindre un destinataire donné.

- Donner brièvement l'idée générale de la façon dont un noeud routeur concentrateur peut découvrir l'identité du concentrateur qui est au bout d'une corde.**Le routeur émet un message `WHO` avec son**

identité et l'identité du port de la corde sur la corde et le noeud concentrateur répond sur la liaison normale en rajoutant son identité par un message `IAM`

- Modifier le programme de découverte de l'anneau pour qu'une fois l'anneau connu, les noeuds routeurs enchaine par la recherche des concentrateurs de bout de cordes. .**On rajoute**

```
anneau_connu  && !vide(Cout) && cordes_inconnus    ==>
      cordes_inconnues=false; cordes_connues=false; waited=card(Cout);;
      pour tout l dans Cout l!<WHO, ICI, l>
exist l dans Cin l?<WHO,R,LR> ==> out ! <IAM,R,LR,ICI>
in?<IAM,R, RL , C> && R!=ICI  ==> out ! <IAM,R,LR,C>
in?<IAM,R, RL , C> && R==ICI  ==>
        apprend_route(R, RL , C);
        waited--;
       if( waited==0) cordes_connues=true;
```

2.2) Routage(3 pts)

***** Modifier l'algorithme de routage initial (cf algorithme 1)en supposant donnée la fonction de routage `getRoute` qui à partir d'une identité de noeud donne une liaison de sortie appartenant à l'ensemble `Cout U {out}` .

```
repeter alt
      besoin_émettre(dst,data)        ==> getRoute(dst)!<MSG,dst,data>
      exist l dans (Cin U {in})  l?<MSG, dest, data> && dest!=ICI ==>getRoute(dst)!<MSG, dest, data>
      exist l dans (Cin U {in}) l?<MSG, dest,data> && dest==ICI ==> traitement_local(data)
```

***** Donner brièvement l'idée générale de la façon dont la table de routage d'un noeud routeur peut être fabriquée.**On peut choisir un routage à la IP par les plus court chemin. Quand on a 1) la liste ordonné des noeuds de l'anneau et 2) les noeuds en bout de cordes, on peut calculer pour tous les noeuds de l'anneau par quel port (re)émettre un message. Tant que les cordes ne sont pas connues, on route sur l'anneau c'est à dire que getRoute renvoie** `out`

2.3)Snapshot sur anneau cordal (4 pts) On reste sous les mêmes hypothèses simplificatrices qu'en 1.5. Le noeud initiateur peut être routeur-concentrateur. Le cliché doit être pris en un tour d'anneau. Aucune autre demande de cliché ne peut être émise avant fin du cliché en cours.

***** que doit enregistrer l'initiateur après émission initiale du message de cliché `SHOT` sur `out` et `Cout` ? et pourquoi?

***** que doit enregistrer un site recevant le message `SHOT` sur `in`? sur une corde de `Cin`? au moment de la réception ? après ? et pourquoi?

***** que contient le message de cliché ?

***** Compléter le comportement des sites pour les demandes de cliché issues de l'initiateur 0 .

L'initiateur renregistre son état local et émet le jeton `SHOT` **sur toutes ses liaisons sortantes** `Coutet out`**. Il enregistre tous les messages reçus sur chacune des liaisons entrantes** `Cinet in`**. Il arrete l'enregistrement sur une liaison à la réception de** `SHOT` **sur la liaison. Tous les messages sont enregistrés lorsque il a reçu** `SHOT` **sur toutes les liaisons entrantes.**
Les autres noeuds déclenche la prise d'état dès la réception de `SHOT` **sur une liaison** `l0` **parmi les liaisons entrantes** `Cinet in`**. Il propage** `SHOT` **sur toutes ses liaisons sortantes** `Coutet out`**. Il enregistre tous les messages reçus sur toutes les liaisons entrantes dec** `Cinet in` **différentes de** `l0`**. Il arrete l'enregistrement sur une liaison à la réception de** `SHOT` **sur la liaison.**
Si on ne fait rien de plus, à ce moment, tous les noeuds ont enregisté leur état et les messages à enregistrer. Il faur maintenant collecter l'état. Soit tous les noeuds l'envoie à l'initiateur (plusieurs messages). Soit l'initiateur envoie une vague de collection.

```
repeter alt
      exist l dans (Cin U {in})  l?<MSG, dest, data> && dest!=ICI ==> .....
      exist l dans (Cin U {in})
           l?<MSG, dest,data> && dest==ICI ==>  .....  traitement_local(data)
      decision_de_cliché && !cliché_en_cours  ==> cliché_en_cours=true;.............
      exist l dans (Cin U {in})l?<SHOT,.....> && .....   ==>  ........
```