UJF - UFR-IMA - MST ESI2
ISTG - RICM2
Examen d'algorithmique distribuée
Durée 2h

Tous documents autorisés – Le barême est indicatif

18 avril 2001

I) Vague ECHO Il s'agit de proposer une variante de la vague pour réseau quelconque à liaisons bidirectionnelles (sans hypothèse FIFO). On rappelle à la suite la variante de l'algorithme ECHO proposée en cours/TD qui supporte à un instant donné :

- des vagues issues d'initiateurs quelconques
- une seule vague par initiateur. Seule la dernière vague d'un initiateur est conservée. Les autres s'éliminent par extinction.

Il y a N noeuds possibles. Tout noeud exécute le même algorithme :

```
// Wait[N] tableau d'ensembles de liaisons
// Up[N] tableau des liaisons de retour vers les initiateurs
// last[N] tableau des dates des dernières vagues
// NodeId indificateur du noeud
// HL horloge locale

état_initial ==>HL=0;  for(n=0;n<N;n++)last[n]=0, Wait[n]=L
loopALT
  decision_vague ==>  HL++ ;
      Wait[NodeId]=L;last[NodeI]=HL;
      forall l in  L l!<TOK,NodeId,last[NodeId] >

  forall l in  L l?<TOK, i, h> && L=={l}  ==> // noeud terminal
      l!< RPL, i, h>

  forall l in L l?<TOK, i, h> && L!={l}  ==>
      if(h>last[i])  // nlle vague
         last[i]=h; Up[i]=l; Wait[i]=L - {l};
         forall l in Wait[i] l!<TOK,i,h >
      elseif(h==last[i]) // boucle
         l!< RPL, i, h>

  forall l in  L l?<RPL, i, h>   ==>
      if(h==last[i])  // retour vague
         Wait[i]-={l}
         if (Wait[i]=={} && i== NodeId)   FIN_VAGUE(h)
         if (Wait[i]=={} && i!= NodeId)    up[n]!<RPL, i , h>
end loopALT
```

question 1 (4 points) Modifier l'algorithme précédent pour supporter plusieurs vagues à la fois issues d'un même initiateur. Ces vagues ne doivent donc plus être éliminées. Un des points clés de la solution est d'éviter de conserver sur les noeuds la description des vagues terminées[1].

question 2 (1 points) Votre algorithme supporte t il les liaisons non FIFO ? Pourquoi? (5 lignes maximum)

II) Vague TORE 2D Il s'agit de généraliser le comportement des noeuds d'un anneau FIFO à une grille d'anneaux FIFO (cf figure 1).

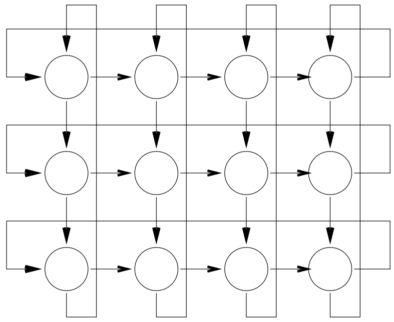

Figure 1: Exemple de tore 2D 4*3

On rappelle le comportement des noeuds d'un anneau FIFO :

```
// li liaison entrante et lo liaison sortante
loopALT
 //Partie routage
    li?<MSG,d,data> && d!=NodeId ==> lo!<MSG,d,data> // reémession
    li?<MSG, d,data> && d==NodeId ==>  traitement local (data)

  // Partie vague
    décision_vague ==> HL++; LAST =HL; lout!<TOK, NodeId, LAST>;
    li?<TOK, i, h> && i!==NodeId ==>  lo!<TOK,i,h,>
    li?<TOK, i, h> && i==NodeId ==> FIN_VAGUE(h)
end loopALT
```

Adaptez l'algorithme précédent de façon à ce qu'il fonctionne sur un tore 2D. On notera Li et Lo les deux tableaux de liaisons. L'indice 0 repère la ligne et l'indice 1 la colonne. Le problème comprend 2 parties indépendantes :

question 3 (2 points) modifier la partie routage pour router dans un tore 2D. On supposera disposer d'une fonction *Coord(int)->int,int* qui fournit à partir de l'identité d'un noeud ses coordonnées $< x, y >$ dans la grille.

question 4 (4 points) modifier la partie vague pour réaliser une vague sur tore 2D. On pourra introduire différents types de `TOK` si nécessaire.

III) Vague PHASE Le principe de la vague PHASE pour graphe quelconque à liaisons monodirectionnelles est rappelée à la suite[2]. Le diamètre du réseau est supposé connu.

[1] On se donnera les gestionnaires de données appropriés(table de hachage, etc..) en précisant simplement la sémantique des opérateurs utilisés (insertion, retrait, etc..).

[2] Il ne supporte qu'une vague dans le réseau.

```
Li (Lo)  ensemble des liaisons entrantes (sortantes) d'un noeud
CTi[Li] tableau de nombres de message reçus par liaison entrante
CTo nombre des phases d'émission

élprtat_initial ==> CTo=0;  forall l in Lin CTi[l]=0
loopALT
  decision_vague ==>
      CTo++
      forall l in Lo l!<TOK > // première phase

  forall l in Li l?<TOK> && Cto<DiamètreRéseau ==>
      CTi[l]++
      if(minimumDes(CTi)>=Cto)  // nouvelle phase d'émission
         CTo++
         forall l in Lo l!<TOK >

  forall l in Li l?<TOK> && Cto>=DiamètreRéseau ==> FIN_VAGUE
end loopALT
```

L'algorithme est basé sur un mécanisme de répétition de phase. Une phase (sauf la phase initiale) est la diffusion sur les liens de sortie de ce qui a été reçu sur tous les liens entrants à la phase précédente.

question 5 (3 points) Modifier l'algorithme pour le rendre capable de supporter plusieurs initiateurs ainsi qu'une vague par initiateur selon la méthode d'extinction des vagues plus anciennes.

question 6 (2 points) Pourquoi l'algorithme consistant à reémettre sur les liaisons de sortie chaque message arrivant sur une liaison entrante ne marche-t-il pas ?

question 7 (2 points) Pourquoi l'algorithme de phase ne marche-t-il pas avec une seule répétition[3]?

question 8 (2 points) Pourquoi le nombre de phases nécessaire est il la valeur du diamètre du réseau ?

Justifiez vos trois dernières réponses à l'aide du schéma(cf. figure 2)

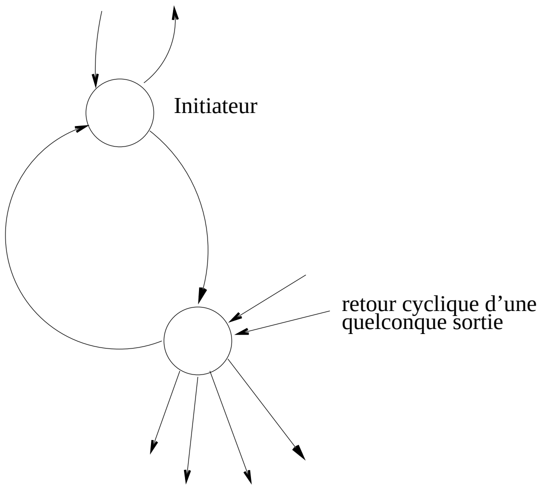

Figure 2: Exemple de sous-réseau monodirectionnel

[3]équivalent à un diamètre de 1.