Université Jopseph Fourier – UFR-IMA
MST ESI2 : Examen d'algorithmique distribuée

Tous documents autorisés – Le barême est indicatif

22 Avril 1998

1 Cliché périodique (6 pts)

1.1 Avec horloge globale(3 pts). Un système distribué dispose d'une horloge globale consultable par tous les noeuds (procédure `int Date()`). Sa finesse est suffisante pour dater uniquement les transitions d'état des processus. On désire prélever un cliché de l'état du système à la date **D0**. Pour ce faire, chaque noeud enregistre son état à la date **D0**. On dispose des procédures d'enregistrement suivantes :

- `RegisterLocalState(date)` sauve l'état interne à la date `date` date ,
- `RegisterLinkState( date, LinkState[])` enregiste l'état calculé des liaisons au moment du cliché. Cet état est dans le tableau `LinkState[]` des listes (initialement vides) de messages qui circulaient sur chacune des L liaisons au moment du cliché.

On suppose que toutes les liaisons de communication sont FIFO fiables.

1. Que doit contenir un message circulant sur une liaison ? (2 lignes)
2. Compléter le code suivant décrivant le comportement d'un noeud à la réception d'un message sur une liaison :

```
Quelque soit l(0<=l<L) de l'ensemble des L liaisons  de n
      l?<......> ==> .......
```

3. Quelle condition additive garantit que la prise d'un cliché termine.(2 lignes)

1.2 Sans horloge globale (3 pts). Pour pallier à l'absence d'une horloge globale, on peut utiliser des horloges logiques selon *Lamport*.

1. Modifier l'algorithme précédent pour substituer les horloges logiques à l'horloge globale,
2. Qu'est-ce qui garantit le bon fonctionnement de la prise de cliché ? (10 lignes maximum : mots clefs causalité, datation, horloge).
3. Comment un noeud peut-il forcer la prise de cliché ? (1 lignes)

2 Partage de données (6 pts)

Dans un système distribué, pour améliorer les temps d'accès en lecture à une information partagée par les noeuds, celle-ci est dupliquée sur tous les noeuds. Le problème est alors de garantir la cohérence de ces copies lorsque les noeuds la modifient. On définit la *cohérence séquentielle* comme le fait que les évolutions des copies sont exactement identiques c'est à dire que chaque copie subit exactement la même séquence de modifications que ses voisines. On peut essayer de ramener ce problème à la construction d'un ordre temporel de prise en compte des modifications.

2.1 Datation par horloge logique (2 pts). On suppose les liaisons FIFO fiables. Soit la solution suivante :

- chaque noeud possède un horloge logique locale incrémentée à chaque modification
- lorsqu'un noeud désire modifier la donnée, il envoie cette modification estampillée par son horloge à tous les autres noeuds,
- chaque noeud (y compris le noeud initiateur) enregistre cette modification dans un file triée par date croissante,
- chaque noeud fait les modifications dans l'ordre des estampilles.

Donner un exemple de comportement où cette solution est incorrecte ? Pourquoi ? Le recalage des horloges selon Lamport corrige-t-il le problème ? Donner un exemple.

2.2 Utilisation d'un anneau (4 pts) . On peut imaginer d'exploiter l'ordre induit par un anneau pour propager les modifications.

1. Décrire le comportement d'un noeud pour que la modification soit propagée en un tour d'anneau après passage du jeton et que le jeton ne soit jamais retenu par un noeud :
   - recevant une modification
     `FromRing?<MODIF,noeud modificateur, adresse, donnée> ==> .....`
   - recevant le jeton
     `FromRing?<JETON> ==> .....,`
   - quand un processus du noeud veut écrire dans la donnée
     `modifier(adresse, valeur)==>......`
2. pourquoi l'identité du site modificateur est elle nécessaire ? (2 lignes)
3. quelle contrainte pèse sur les liaisons constituant l'anneau ? pourquoi (5 lignes)?
4. l'anneau doit être un cycle eulérien ? hamiltonien ? Pourquoi ? (5 lignes)

3 Routage et Apprentissage (8 pts)

Soit un réseau maillé par des liaisons bidirectionnelles fiables. Chaque noeud possède un nombre L de liaisons numérotées de 0 à L-1. Les noeuds sont identifiés par un numéro de 0 à N-1. Tout noeud est indifféremment routeur, émetteur ou destinataire. Le protocole de routage mis en oeuvre est un routage par la source. Le format d'un paquet est le suivant :
`<PAQUET,étape,hop,chemin,donnée>` où `itinéraire` est la suite de longueur `hop` des numéros de liaison à emprunter. `étape`(initialement 0) est le rang (dans l'itinéraire) du routeur que le paquet traverse.

3.1 Routage par la source (2 pts). Compléter le comportement d'un noeud quelconque recevant un paquet.[1]

```
Quelque soit  n (0<=n<N) de l'ensemble des noeuds :
      Quelque soit l(0<=l<L) de l'ensemble des L liaisons  de n
           l?<PAQUET,étape,hop,chemin,donnée> ==> .....
```

3.2 Calcul de l'itinéraire de retour (2 pts). Dans le routage par la source, les routeurs sont très simples. Par contre les émetteurs doivent connaitre les différents chemins pour atteindre leurs interlocuteurs. Pour faciliter le dialogue entre noeuds, on souhaite que, lorsque le paquet arrive à destination, le contenu du champ chemin soit l'itinéraire que devra parcourir une éventuelle réponse au message.

Modifier le comportement d'un noeud pour ce faire(sans changer le format de paquet).

[1]On notera l'accès à un élément de chemin par `chemin[indice entier positif < hop]`.

3.3 Apprentissage (2 pts).

1. Quelles informations faut-il rajouter au message pour que les noeuds puissent *apprendre* les itinéraires atteignant émetteur et récepteur lors d'une opération de routage ?(2 lignes)
2. Modifier le comportement d'un noeud pour ce faire.
3. comment s'adapter aux reconfigurations du réseau.(2 lignes)
4. Comment éviter l'engorgement de la table de routage. (2 lignes)

On supposera disposer des procédures suivantes :
`Lookup( numéro noeud) -> chemin, hop` qui rend le chemin (et sa longueur) pour atteindre un noeud destinataire (0 chemin inconnu),
`Register( numéro noeud, chemin, hop)` qui enregistre le chemin pour atteindre un noeud donné.

3.4 Calcul d'un itinéraire (2 pts). Le mécanisme d'apprentissage repose sur le fait que des noeuds initient les dialogues et donc connaissent des chemins préalablement à tout apprentissage. Un moyen pour un tel noeud d'obtenir un chemin est d'explorer systématiquement le réseau par un calcul diffusant jusqu'à trouver le noeud destinataire. Le principe de ce calcul est simple, les paquets d'exploration progressent par duplication/reémission dans le réseau en enregistrant le chemin parcouru. Tous les paquets d'exploration parcourant un cycle sont éliminés. Seul le noeud destinataire envoie une réponse qui permet au noeud initiateur d'obtenir un itinéraire.

Les formats d'un paquet d'exploration et d'un paquet de réponse sont les suivants :
`<EXPLORE,étape,hop,chemin,initiateur,noeud cherché, noeuds visités>` où `noeuds visités` contient les identités des noeuds précédemment visités par le paquet d'exploration.
`<REPLY,étape,hop,chemin,noeud trouvé>`

1. Comment détecte-t-on qu'un paquet est dans une boucle ?(3 lignes)
2. Quand élimine-t-on un paquet d'exploration ? (5 lignes)
3. Un noeud recherché peut-il recevoir plusieurs fois le paquet d'exploration d'un initiateur? Pourquoi? (3 lignes)
4. Quel est l'intérêt (respct. l'inconvénient) que le noeud recherché réponde à tous les paquets d'exploration issus d'un même initiateur?
5. Décrire le comportement d'un noeud à la réception de ces 2 types de paquet.