Université de Montpellier — 10 janvier 2022
MasterM1 d'Informatique — Durée: 3h

Algorithmes Distribués
Examen 12 janvier 2022

Les documents sont autorisés. La note prendra en compte la clarté des explications.

Exercice 1

On considère une exécution répartie sur trois sites, pour laquelle on n'a pu récupérer que l'horloge vectorielle du dernier événement sur chacun des sites :

$\begin{bmatrix} 3 & 3 & 2 \end{bmatrix}$ sur le site 1 ; $\begin{bmatrix} 1 & 3 & 2 \end{bmatrix}$ sur le site 2 ; $\begin{bmatrix} 1 & 0 & 2 \end{bmatrix}$ sur le site 3.

1. Combien y a-t-'il eu d'événements sur chacun des sites ?
2. Peut-on connaître le nombre de messages émis par le site 1 ?
3. Peut-on savoir si le site 2 a envoyé un message au site 1 ?
4. Compléter le schéma donné par la figure 1 pour reconstruire l'historique des événements jusqu'aux dates indiquées. Vous donnerez la date (matricielle) de chaque événement.

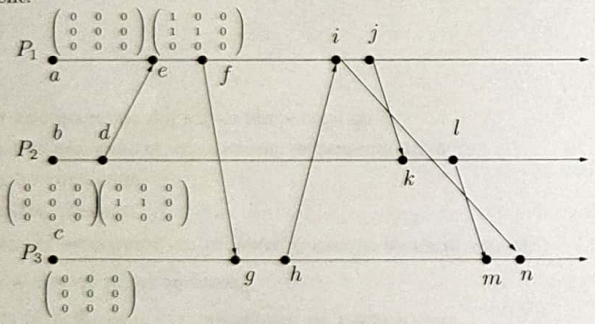

FIGURE 1 – *Schéma à compléter.*

Chacune de vos réponses devra être clairement justifier.

□

Exercice 2 Exclusion mutuelle dans les arbres : algorithme de Raymond

Nous utiliserons les messages $REQUEST$ pour demander à utiliser la ressource et le message $JETON$ qui représente la ressource (ou l'autorisation de l'utiliser). L'idée de ce protocole est que chaque site p a toujours un «pointeur» ($Racine_p$) qui est dirigé vers le jeton dans l'arbre.

```
Procédure acquisition
    Si (non Avoir_jeton_p) alors
        Si (File_vide(Request_p)) alors Envoyer(< REQUEST >) à Racine_p;
        Ajouter(Request_p,p);
        Attendre(Avoir_jeton_p);
    En_SC_p := vrai;

Procédure libération
    En_SC_p := faux;
    Si (non File_vide(Request_p)) alors
        Racine_p := Défiler(Request_p);
        Envoyer(< JETON >) à Racine_p;
        Avoir_jeton_p := faux;
        Si (non File_vide(Request_p)) alors
            Envoyer(< REQUEST >) à Racine_p;

Lors de la réception de < REQUEST > de q
    Si (Avoir_jeton_p) alors
        Si (En_SC_p) alors Ajouter(Request_p,q);
        Sinon
            Racine_p := q;
            Envoyer(< JETON >) à Racine_p;
            Avoir_jeton_p := faux;
    Sinon
        Si (File_vide(Request_p)) alors Envoyer(< REQUEST >) à Racine_p;
        Ajouter(Request_p,q);

Lors de la réception de < JETON > de q
    Racine_p := Défiler(Request_p);
    Si (Racine_p = p) alors Avoir_jeton_p := vrai;
    Sinon
        Envoyer(< JETON >) à Racine_p;
        Si (non File_vide(Request_p)) alors Envoyer(< REQUEST >) à Racine_p;
```

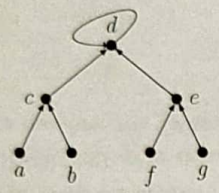

FIGURE 2 – Anti-arborescence.

1. Donner le principe et le fonctionnement de l'algorithme.
2. Quel est le rôle de la file *Request* ?
3. Appliquer cet algorithme sur la figure 2, avec le scénario suivant :
   - c fait une demande.
   - g fait une demande
   - b fait une demande et arrive avant la demande de g.
   - a fait une demande qui arrive après celle de g.

□

Exercice 3

Dans cet exercice, nous nous intéressons à un algorithme résolvant le problème de l'élection dans un anneau unidirectionnel. C'est l'algorithme de LeLann.

Chaque site p dispose d'une variable $List_p$ qui calcule la liste des identités des initiateurs, et un état $state_p$ qui définit l'état du site p avec $state_p \in \{cand, sleep, lost\}$.

Voici l'algorithme qui est proposé :

```
Pour un initiateur p, à exécuter qu'une seule fois
    state_p := cand;
    Envoyer(jeton,p) à Next_p

Pour un initiateur p, lors de la réception de (jeton,q)
Si q ≠ p
    Alors list_p := list_p ∪ {q};
    Envoyer(jeton,q) à Next_p;
Sinon Si p = min(list_p);
```



```
    Alors state_p := leader;
        Sinon state_p := lost;

Pour un non initiateur p, Lors de la réception de (jeton) de q
Envoyer (jeton,q) à Next_p;
Si state_p = sleep;
    Alors state_p := lost;
```

1. Donner un exemple d'exécution sur l'anneau donné par la figure 3 tel que tous les sites sont initiateurs.

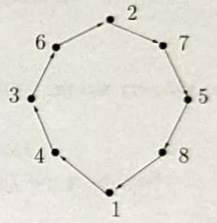

FIGURE 3 – *Anneau orienté*

2. Donner le principe de l'algorithme.
3. Donner sa complexité en nombre de messages échangés, et en temps au pire cas.
4. Pourquoi l'hypothèse des communications FIFO est primordiale ?
5. Expliquer la finitude de l'algorithme.

□

Exercice 4

Dans cet exercice on s'intéresse à calculer le diamètre d'un arbre. Le diamètre d'un graphe est la plus grande distance (en nombre d'arêtes) d'un plus court chemin entre chaque paire de sommets.

1. Calculer le diamètre de l'arbre donné par la figure 4.
2. Proposer un algorithme distribué qui détermine le diamètre d'un arbre.
3. Procéder à une exécution de votre algorithme sur la figure 4.

4. Notion de site saturé : un site est dit saturé lorsqu'il a reçu tous les messages de ces voisins.
   (a) Dans un arbre, combien y-a-t'il au plus de sommets saturés ?
   (b) Si il en existe plusieurs, quels sont les relations entre les sommets saturés ?

□

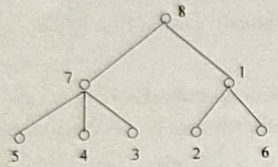

FIGURE 4 – Un réseau en arbre.