Université de Batna 2
Département Informatique
CONCOURS D'ACCES AU DOCTORAT 3EME CYCLE EN INFORMATIQUE
Option : Systèmes Informatiques
Epreuve : Algorithmes Distribués (variante 2)
23 Octobre 2018 / Durée : 02h

Corrigé type

Exercice N°01 (05 points)

Var $etat_i$ (dehors, demandeur , dedans) init dehors
h_i : entier init 0
$last_i$: entier init 0
$prioritaire_i$: booleen init faux
$nbatt_i$: entier init 0
$compteur_i$: entier init 0
$differe_i$: ensemble de id_proc init {}
R_i : ensemble de id_proc init {1,2,..,n}-{i}

proc acquerir
debut
$etat_i$:= demandeur;
$h_i := h_i+1$; $last_i := h_i$;
$nbatt_i := card(Ri)$;
pourtout j **de** R_i
envoyer requete($last_i$, i) **a** C_j
fpourtout ;
attendre $nbatt_i = 0$;
$etat_i$:=dedans
fin

proc liberer
debut
pourtout j **de** $differe_i$
envoyer permission **a** C_j
fpourtout ;
$differe_i={}$
$compteur_i$=0 /// 1.5pts
$etat_i$:= dehors
fin

lors de reception requete(k,j) ///3.5 pts
debut
$h_i := \max(h, k)$;
$priorite_i := (etat_i \neq dehors) \wedge (last_i, i) < (k, j)$;

si $priorite_i$
alors
si $etat_i$**=dedans** $\wedge$ $compteur_i$<*n-4*
alors envoyer permission **a** C_j
$compteur_i$ = $compteur_i$ **+1**
sinon
$differe_i$:= $differe_i \cup \{j\}$
fsi
sinon envoyer permission **a** C_j
fsi
fin

lors de reception permission **de** *j*
debut
$nbatt_i$:= $nbatt_i$-1
fin

Exercice N°02 (05 points):

1. Pourquoi les détecteurs de défaillances proposés par Chandra et Toueg sont considérés comme un mécanisme de détection de défaillances non fiable ?

Puisque ils peuvent faire des erreurs au sujet de défaillances, c.à.d suspecter un processus correct et ne pas détecter le crash d'un processus défaillant. **1 points**

2. Quelle est la relation entre les classes de détecteurs de défaillances suivantes :
-*P* et *Q,*
-*S* et W,
-◊*P* et ◊*Q,*
-◊*S* et ◊W.

-*P* est équivalente *Q,* ***0.25 points***
-*S* est équivalente W, **0.25 points**
-◊*P* est équivalente ◊*Q,* ***0.25 points***
-◊*S* est équivalente ◊W. **0.25 points**

3. Quelles sont les propriétés d'un " Détecteur de Défaillances Inéluctablement Quasi-parfait ◊*Q* " ?

Deux propriétés :

Complétude faible : tous les processus défaillants finiront par être suspectés par au moins un processus correct. **0.75 points**

Précision inéluctablement forte : il existe un instant *t* à partir duquel aucun processus correct n'est suspecté. **0.75 points**

4. Quel est la classe de détecteur de défaillances la plus faible pour résoudre le Consensus ? Pourquoi ?

La classe <>W 0.75 points

Puisque elle est la plus générale, c'est-à-dire un algorithme qui se base sur <>W peut utiliser n'importe autre classe, non équivalente, de détecteurs de défaillances mais le contraires n'est pas correct. **0.75 points**

Exercice N°03 (05 points):

1. Définir les concepts suivants : Le consensus probabiliste, le consensus Uniforme et le k-consensus.

Formellement, **Le consensus probabiliste** est caractériser par la propriété de terminaison appelée R-terminaison (Random-termination), qui est définit comme suit :

- **R-terminaison :** tous les processus corrects décident avec une probabilité égale à 1. **0.5 points**

Le **consensus uniforme** a été défini et se diffère du consensus de base par la propriété d'accord :

- **Accord uniforme :** Si un processus décide une valeur *v*, alors tous les processus décident *v*. **0.5 points**

Dans le **consensus ensembliste** la propriété d'accord est remplacée par :

- **K-accord :** il existe au plus k valeurs différentes décidées par les processus. **0.5 points**

2. Quelles sont les propriétés qui caractérisent le problème de la diffusion Atomique ? est-t-il possible de résoudre le consensus en se basant sur la diffusion Atomique ? si oui comment ?

La diffusion atomique peut être définie par les cinq propriétés suivantes :

- **Terminaison :** Si un processus correct diffuse un message m, alors tous les processus corrects délivrent ce message. **0.25 points**
- **Validité :** Si un processus délivre un message *m*, alors m a été diffusé par au moins un processus. **0.25 points**
- **Intégrité :** un processus remis un message *m* au plus une fois ; **0.25 points**
- **Accord :** Si un processus correct délivre un message *m* alors tous les processus corrects délivrent *m* ; **0.25 points**
- **Ordre :** Si un processus correct délivre un message *m* avant un message *m',* alors tous les processus corrects délivrent *m* avant *m'*. **0.25 points**

Oui, il est possible de résoudre le consensus en se basant sur la diffusion Atomique **0.25 points**

Pour résoudre le consensus nous pouvons utiliser la diffusion atomique de la façon suivante : pour décider d'une valeur, un processus la diffuse de façon atomique. Pour décider d'une valeur un processus prend la valeur du premier message qu'il a délivré. Par la propriété d'ordre total de la diffusion atomique tous les processus corrects vont délivrer le même message. **0.5 points**

3. Quelle est la différence entre les deux protocoles 2PC (Two-Phase Commit) et le 3PC (Three-Phase Commit) ?

Le protocole 2PC (Two-Phase Commit) est pour résoudre la validation atomique AC et le protocole le 3PC (Three-Phase Commit) est pour résoudre la validation atomique non bloquante (NBAC pour Non-Blocking Atomic Commitment) dans les systèmes synchrones. **0.75 points**

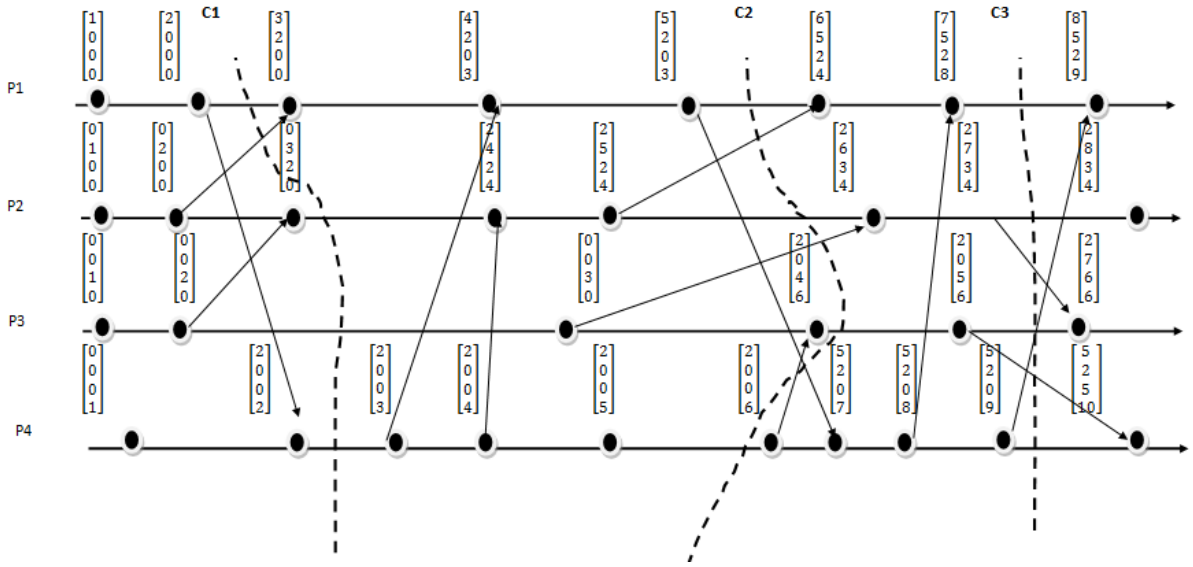
4. Le résultat d'impossibilité de FLP montre qu'un consensus n'est pas réalisable de façon déterministe dans un système asynchrone soumis à des crashes de processus, même si le système n'est soumis qu'à une seule défaillance et que les canaux sont fiables. De manière intuitive, ce résultat d'impossibilité est justifié par le fait qu'il est impossible de distinguer un processus lent d'un processus crashé. Pourquoi, il est impossible de distinguer entre un processus lent et un processus crashé ?

Car les **délais de transmission** des messages et **les vitesses relatives** des processus sont arbitraires. Donc, il est possible de décider d'attendre un message qui n'arrivera jamais et on reste bloquer indéfiniment ou décider de ne pas attendre un message et finalement on le reçoit. Par conséquence, on ne pas distinguer entre les deux situations et on ne pas savoir si on prêt la bonne décision. **0.75 points**

Exercice N°04 (05 points):

A)

1)



2) la coupure C1 est cohérente car $EV(c1)=\begin{bmatrix}2\\3\\2\\2\end{bmatrix} = EV_{sup}(c1)=\begin{bmatrix}2\\3\\2\\2\end{bmatrix}$

- La coupure C2 est incohérente car $EV(c2)=\begin{bmatrix}5\\5\\4\\5\end{bmatrix} \neq EV_{sup}(c2)=\begin{bmatrix}5\\5\\4\\6\end{bmatrix}$

- La coupure c3 est cohérente car $EV(c3)=\begin{bmatrix}7\\7\\5\\9\end{bmatrix} = EV_{sup}(c3)=\begin{bmatrix}7\\7\\5\\9\end{bmatrix}$

3) oui, les horloges de Mattern (vectorielles) suffisent pour vérifier la cohérence des coupures.

Justification : les horloges vectorielles sont équivalentes à la causalité.

B)

Solution 1

Si C4 et C5 sont des coupures cohérentes, alors que $C4 \cup C5$ et $C4 \cap C5$ le sont aussi. En effet :

si $e \in C4 \cap C5$ et e' -> e, alors : $e' \in C4$, car C4 cohérente et $e' \in C5$, car C5 cohérente donc $e' \in C4 \cap C5$ **(1pt)**

si $e \in C4 \cup C5$ et e' -> e, alors : **(1pt)**

ou bien $e \in C4$ et alors $e' \in C4$, car C4 cohérente

ou bien $e \in C5$ et alors $e' \in C5$, car C5 cohérente

donc $e' \in C4 \cup C5$

ou solution 2

intersection: Une coupure C est cohérente si $\forall e \in C : \forall e0 : e0 \prec e \Rightarrow e0 \in C$

on suppose que l'intersection C3 entre C1 et C2 ($C3= C1 \cap C2$) n'est pas une coupure cohérente, formellement $\exists e \in C3 : \exists e0 : e0 \prec e \Rightarrow e0 \notin C3$.

sachant que :

$\forall e1 \in C1 : \forall e01 : e01 \prec e1 \Rightarrow e01 \in C1$

$\forall e2 \in C2 : \forall e02 : e02 \prec e2 \Rightarrow e02 \in C2$

Si e ∈ C3 alors e ∈ C1 et e ∈ C2 donc ∀e0 : e0 ≺ e ⇒ e0 ∈ C2 et e0 ∈ C1 donc e0 ∈ C3 contradiction

union: on suppose que l'union C3=C1 ∪ C2 n'est pas une coupure cohérente,

formellement ∃ e ∈ C3 : ∃ e0 : e0 ≺ e ⇒ e0 ∉ C3.

sachant que :

∀e1 ∈ C1 : ∀e01 : e01 ≺ e1 ⇒ e01 ∈ C1

∀e2 ∈ C2 : ∀e02 : e02 ≺ e2 ⇒ e02 ∈ C2

Si e ∈ C3 alors e ∈ C1 ou e ∈ C2 donc ∀e0 : e0 ≺ e ⇒ e0 ∈ C2 ou e0 ∈ C1 donc e0 ∈ C3 contradiction