Ingénierie des Protocoles : Examen

Stéphane Devismes

Lisez attentivement ce qui suit :
- Aucun document n'est autorisé.
- Le barême est donné à titre indicatif uniquement. Il est susceptible de changer !

1 Détecteur de défaillances (9 points)

Nous rappelons que tout détecteur de défaillances *finalement parfait* $\diamond\mathcal{P}$ satisfait la **complétude forte** et l'**exactitude finalement forte** :

Complétude forte : Tout processus qui tombe en panne finit par être suspecté en permanence par *tous* les processus corrects.

Exactitude finalement forte : Il existe un temps à partir duquel plus aucun processus correct n'est suspecté par un processus correct.

Questions.

1. Un détecteur de défaillances *parfait* $\mathcal{P}$ satisfait la **complétude forte** et l'**exactitude forte**. Rappelez la définition d'exactitude forte. **(0,5 point)**
2. Que signifie $\diamond\mathcal{P} \prec \mathcal{P}$? Est-ce vrai ? (Justifiez) **(1 point)**

1.1 Algorithme

On suppose un réseau **complet** sujet à des pannes définitives **initiales** de processus, c'est-à-dire que tout processus qui tombe en panne n'a jamais participé à aucun calcul. De plus, les liens de communications sont sujets à **des pertes équitables**.

Questions.

1. Dans ce système, proposez un algorithme réalisant le détecteur de défaillance **finalement parfait**. **(2 points)**
2. Prouvez la correction de votre algorithme. **(2 points)**

1.2 Consensus

On suppose maintenant un réseau **complet** sujet à des pannes définitives où
- les liens de communications sont **fiables**,
- chaque processus dispose d'un détecteur $\diamond\mathcal{P}$, et
- $n > 2t$ (n est le nombre de processus et t le nombre maximum de pannes).

Nous rappelons que le détecteur Ω renvoie l'identité d'un unique processus supposé correct. Il garantit qu'en un temps fini, il existe un processus correct ℓ tel que $\Omega_p = \ell$ pour toujours pour tout processus correct p.

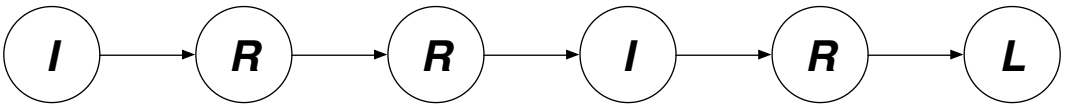

FIGURE 1 – Configuration initiale : p_0 est à gauche et p_{n-1} est à droite.

Questions.

1. Expliquez comment passer de $\diamond\mathcal{P}$ à Ω. **(0,5 point)**
2. Rappeler la spécification du consensus. **(1 point)**
3. Expliquez comment résoudre le consensus dans ce système. **(2 points)**

2 Autostabilisation (12 points)

2.1 Comprendre un algorithme auto-stabilisant

On considère un réseau en « ligne » : $p_0, p_1, \ldots p_{n-1}$. Le réseau est orienté : soit p_i un processus, on a

– si $i > 0$, alors p_i a un voisin de gauche, p_{i-1} ;
– si $i < n-1$, alors p_i a un voisin de droite, p_{i+1}.

Ci-dessous, nous proposons un algorithme, écrit dans le modèle à états, constitué de quatre règles. Dans cet algorithme, chaque processus p_i dispose d'une variable $p_i.S$ dont le domaine est $\{I, R, L\}$.

$$TR \quad :: \quad (p_i.S = I) \wedge (i = 0 \vee p_{i-1}.S = R) \wedge (i = n-1 \vee p_{i+1}.S = I) \quad \mapsto \quad p_i.S \leftarrow R$$

$$TL \quad :: \quad (p_i.S = R) \wedge (i = 0 \vee p_{i-1}.S = R) \wedge (i = n-1 \vee p_{i+1}.S = L) \quad \mapsto \quad p_i.S \leftarrow L$$

$$Clean \quad :: \quad (p_i.S = L) \wedge (i = 0 \vee p_{i-1}.S = L) \wedge (i = n-1 \vee p_{i+1}.S = I) \quad \mapsto \quad p_i.S \leftarrow I$$

$$Err \quad :: \quad (i \neq 0) \wedge [(p_i.S = R \wedge p_{i-1}.S \in \{I, L\}) \vee (p_i.S = L \wedge p_{i-1}.S = I)] \quad \mapsto \quad p_i.S \leftarrow I$$

Questions.

1. Donnez la trace d'exécution des 6 premiers pas de calculs de l'algorithme à partir de la configuration initiale de la figure 1 en supposant un démon synchrone. **(1,5 point)**
2. Donnez la trace d'exécution des 8 premiers pas de calculs de l'algorithme à partir de la configuration initiale de la figure 1 en supposant un démon central qui choisit toujours le processus activable le plus à droite. **(2 points)**
3. Dans le pire des cas, combien d'actions Err peuvent être exécutée en fonction de n ? Justifiez. (Vous pourrez illustrer votre propos avec un exemple.) **(1 points)**

2.2 Ecrire et prouver un algorithme auto-stabilisant

On considère un réseau en arbre bidirectionnel, enraciné en R et orienté. C'est-à-dire, chaque processus p peut lire l'état de n'importe quel voisin. De plus, chaque processus nonracine p connait le numéro de canal de son père dans l'arbre : ce numéro de canal est noté $père_p$. Nous désignons aussi par $\mathcal{N}_p$ l'ensemble des numéros de canaux des voisins de p.

En outre, on suppose que chaque processus p connaît son numéro de canal dans l'ensemble $\mathcal{N}_q$ de chacun de ces voisins q. Ainsi, p pourra tester si, par exemple, le pointeur $père_q$ de son voisin q le désigne comme son père dans l'arbre. Dans la suite, un tel test sera simplement écrit $père_q = p$.

Chaque processus p dispose enfin d'une entrée E_p booléenne de valeur constante (vrai ou faux).

Questions.

1. Ecrivez dans le modèle à états, un algorithme autostabilisant silencieux qui converge vers une configuration terminale où la variable Booléenne $Sortie_p$ de chaque processus p vaut vrai si et seulement s'il existe (au moins) un processus q tel que $E_q = vrai$. **(3 points)**
2. Prouvez que dans toute configuration terminale de votre algorithme, la variable Booléenne $Sortie_p$ de chaque processus p vaut vrai si et seulement si il existe (au moins) un processus q tel que $E_q = vrai$. **(2 points)**
3. Prouvez que votre algorithme converge vers une configuration terminale sous l'hypothèse d'un démon distribué faiblement équitable. **(2 points)**
4. Quel est le temps de stabilisation de votre algorithme en rondes ? **(0,5 point)**