Ingénierie des Protocoles : Examen

Stéphane Devismes

Résumé

Lire attentivement ce qui suit :
- Aucun document n'est autorisé.
- Le barême est donné à titre indicatif uniquement. Il est susceptible de changer !
- Le barême est sur 60 points + 10 points bonus. Donc, votre note sera obtenue en divisant le total de vos points par 3.

1 Circulation de jeton : 30 points

Question 1 : 2 points. Rappeler la spécification de la *circulation de jeton*. En particulier, pour chaque propriété indiquer s'il s'agit d'une propriété de *sûreté* ou de *vivacité*.

Question 2 : 10 points. Proposer un algorithme de circulation de jeton fonctionnant dans le modèle à passage de message sous les hypothèses suivantes :
- Processus et canaux asynchrones.
- Canaux étiquetés de 1 à δ_p pour tout processus p.
- Pas de fautes.
- Topologie quelconque (connexe) d'au moins deux nœuds.
- Mono initiateur.

ATTENTION : En plus du code, il vous est demandé d'expliquer le fonctionnement de votre solution.

Question 3 : 8 points. Prouver votre algorithme.

Question 4 : 4 points. Justifier la complexité de votre algorithme.

Question 5 : 2 points. Rappeler la spécification de l'*élection de leader*.

Question 6 : 4 points. En supposant maintenant que les nœuds du réseau sont identifiés de manière unique, expliquer comment utiliser l'algorithme précédent pour réaliser un algorithme d'élection de leader dans un réseau identifié connexe quelconque.

2 Auto-stabilisation : 30 points

Dans un réseau, un *ensemble dominant* est un sous-ensemble de processus S tel que tout processus qui n'appartient pas à S a au moins un voisin qui appartient à S.

Un ensemble dominant est dit *minimal* si aucun de ses sous-ensembles propres n'est un ensemble dominant. Autrement dit, un ensemble dominant minimal S vérifie deux propriétés :

1. S est un ensemble dominant.
2. Si on enlève n'importe quel élément de S, on obtient un ensemble qui n'est pas dominant.

Par exemple, l'ensemble des nœuds blancs dans la figure 1 est un ensemble dominant minimal.

Question 7 : 4 points. Rappeler la définition de l'*auto-stabilisation*.

Question 8 : 2 points. Rappeler la définition d'un algorithme *silencieux*.

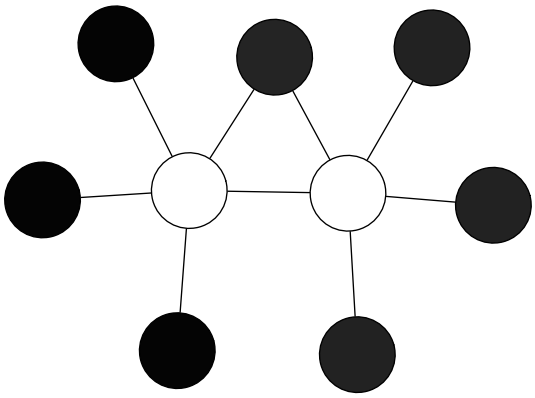

FIGURE 1 – Ensemble dominant minimal

Question 9 : 10 points. Proposer un algorithme auto-stabilisant silencieux construisant un ensemble dominant minimal fonctionnant sous les hypothèses suivantes :
- Modèle à états avec un démon *distribué faiblement équitable.*
- Topologie connexe quelconque.
- Processus identifiés.

ATTENTION : En plus du code, il vous est demandé d'expliquer le fonctionnement de votre solution.

Question 10 : 8 points. Prouver votre algorithme.

Question 11 : 2 points. Rappeler la différence entre un démon *faiblement équitable* et un démon *inéquitable.*

Question 12 : 2 points. Justifier pourquoi votre solution fonctionne ou ne fonctionne pas avec un démon *inéquitable.*

Question 13 : 2 points. Expliquer comment adapter votre solution pour quelle fonctionne dans un système à passage de messages.

3 Consensus : 10 points bonus

Une solution probabiliste du consensus a été proposée par Ben-Or. Cette solution assure les propriétés d'accord, validité, et intégrité, mais la propriété de terminaison de l'algorithme de Ben-Or est plus faible : la terminaison est assurée avec probabilité 1. Il est supposé qu'une majorité de processus sont corrects (c'est-à-dire ne tombent jamais en panne), et les hypothèses sur le système restent les mêmes que pour le résultat d'impossibilité : le réseau est complet, les liens sont fiables mais asynchrones.

Question 14 : 6 points. Rappeler les définitions des propriétés d'accord, validité, et intégrité du consensus.

Par souci de simplicité, l'algorithme 1 est sous forme itérative. De plus, pour limiter les cas, nous supposons qu'un processus peut s'envoyer un message à lui-même. Dans cet algorithme, t représente le nombre maximum de fautes, ainsi $n > 2t$. Chaque processus peut faire un tirage aléatoire uniforme en utilisant la fonction `Rand`. Par exemple, `Random(0,1)` renvoie une valeur aléatoire 0 ou 1, avec probabilité $\frac{1}{2}$.

Dans cet algorithme, les processus corrects exécutent une boucle infinie. Chaque itération de boucle est appelée *ronde* (la variable k donne le numéro de la ronde courante). Lors d'une ronde, les messages

Algorithme 1 Algorithme de Ben-Or pour tout processus p, entrée $v_p \in \{0,1\}$

1: $d_p \leftarrow \perp$
2: $k \leftarrow 0$
3: **Tant que** *true* **faire**
4: $\quad k++$
5: $\quad$ **envoie** (R, k, v_p) **à tous les processus (**p **y compris)**
6: $\quad$ **attendre de recevoir** $n-t$ messages de la forme $(R, k, \_)$ où $\ll \_ \gg$ peut être 0 ou 1
7: $\quad$ **Si** p a reçu plus de $\frac{n}{2}$ messages (R, k, x) avec la même valeur x **alors**
8: $\quad\quad$ **envoie** (P, k, x) **à tous les processus (**p **y compris)**
9: $\quad$ **Sinon**
10: $\quad\quad$ **envoie** $(P, k, ?)$ **à tous les processus (**p **y compris)**
11: $\quad$ **Fin Si**
12: $\quad$ **attendre de recevoir** $n-t$ messages de la forme $(P, k, \_)$ où $\ll \_ \gg$ peut être 0, 1, ou ?
13: $\quad$ **Si** p a reçu au moins $t+1$ messages (P, k, x) avec $x \neq ?$ **alors**
14: $\quad\quad$ **Si** $d_p = \perp$ **alors**
15: $\quad\quad\quad d_p \leftarrow x$
16: $\quad\quad$ **Fin Si**
17: $\quad$ **Fin Si**
18: $\quad$ **Si** p a reçu au moins 1 message (P, k, x) avec $x \neq ?$ **alors**
19: $\quad\quad v_p \leftarrow x$
20: $\quad$ **Sinon**
21: $\quad\quad v_p \leftarrow$ `Random(0,1)`
22: $\quad$ **Fin Si**
23: **Fin Tant que**

envoyés sont des triplets formés d'un type (R ou P), le numéro de ronde, et une valeur, 0 ou 1 pour pour les messages de type R et 0, 1 ou ? pour les messages de type P.

Les messages de type R sont appelés *rapport*. Les messages de type P sont appelés *proposition*. Ainsi, si un processus p envoie un message (R, k, x), nous dirons que p *rapporte* la valeur x dans la ronde k. Si un processus p envoie un message (P, k, x), nous dirons que p *propose* la valeur x dans la ronde k.

Chaque ronde se divise en deux *phases* : une *phase de rapport* où chaque processus envoie un rapport aux autres, et une *phase de proposition* où chaque processus envoie une proposition aux autres.

Question 15 : 4 points. Montrer que dans l'algorithme de Ben-Or, il n'existe pas deux processus différents qui proposent deux valeurs différentes 0 et 1 dans la même ronde.