M2 - ARA

TD Détecteur de fautes – élection de leader

On considère un ensemble de processus $\Pi = \{p_1, p_2, .., p_n\}$ communiquant par messages. Les liens de communication sont bidirectionnels et fiables. On ne considère que des fautes du type « crash ». Le réseau forme un graphe complet partiellement synchrone : après le temps GST (inconnu), il existe des bornes sur les délais de transmissions. Il y a au moins un processus correct.

Q1

Quelles propriétés doit assurer un détecteur de faute Ω ?

Considérez l'algorithme suivant :

Every process $p_i, i = 1, \ldots, n$ executes:
$trusted_i \leftarrow 1$
$\forall j \in \{1, \ldots, i-1\} : \Delta_{i,j} \leftarrow$ default timeout
cobegin
|| Task 1: **repeat periodically**
    if $trusted_i = i$ then send I-AM-THE-LEADER to $p_{i+1}, \ldots, p_n$
|| Task 2: **when** ($trusted_i < i$) and
        (did not receive I-AM-THE-LEADER from $p_{trusted_i}$ during the last $\Delta_{i,trusted_i}$ time units)
    $trusted_i \leftarrow trusted_i + 1$
|| Task 3: **when** (received I-AM-THE-LEADER from p_j) and ($j < trusted_i$)
    $trusted_i \leftarrow j$
    $\Delta_{i,j} \leftarrow \Delta_{i,j} + 1$
coend

Q2

Complétez l'algorithme avec la tâche T4 afin implémenter un détecteur Omega. A terme tous les processus doivent élire le même processus comme leader.

Task T 4 : upon the invocation of *leader ( )*

Q3

Supposons qu'aucun processus ne tombe en panne. Quel sera le processus leader ?
En présence de fautes, quel sera le processus élu ?

Q4

Est-ce que temporairement des processus différents peuvent être élus ? Illustrez votre réponse par un scénario

Q5

Quel mécanisme assure qu'à terme ces erreurs seront corrigées ?

On considère :
correct : l'ensemble qui contient les processus corrects
pleader : le processus correct élu comme leader
leader (t) : invocation de leader à l'instant t.

Q6

Montrez que :

$\exists t: \forall\, t' > t, \forall pi \in correct, leader(t') = pleader$

Q7

Si on ajoute à chaque processus une variable locale *suspected*$_i$ qui est mise à jour à $\Pi - \{trusted_i\}$ dans la Task3, est-ce que l'algorithme ci-dessus implémente un détecteur de défaillance ◊S ? Justifiez votre réponse.

En s'inspirant sur l'algorithme ci-dessus nous voulons maintenant implémenter un détecteur de défaillance ◊P. Les pseudo-codes de l'initialisation des variables et de la Task1 sont les suivants :

Every process $p_i, i = 1, \ldots, n$ executes:
$trusted_i \leftarrow 1$
$suspected_i \leftarrow \emptyset$ $\{suspected_i$ *provides the properties of* $\Diamond P\}$
$\forall j \in \{1, \ldots, n\} : \Delta_{i,j} \leftarrow$ default timeout $\{\Delta_{i,j}, j < i$ *are used to eventually agree on a common leader process*$\}$
$\{\Delta_{i,j}, j > i$ *are used by the leader to build the set of suspected processes*$\}$

cobegin
|| Task 1: repeat periodically
    if $trusted_i = i$ then
        send (I-AM-THE-LEADER, $suspected_i$) to $p_{i+1}, \ldots, p_n$
    else
        send I-AM-ALIVE to $p_{trusted_i}$

Notez que l'ensemble *suspect*$_i$ est ajouté dans le message I-*AM-THE-LEADER*

Q8

Complétez les autres tâches de l'algorithme. Vous pouvez ajouter le nombre de tâches que vous voulez. Cependant, la seule tâche qui peut envoyer des messages est la Task1.