M2 - ARA

TD Détecteur de fautes – Omega = élection de leader ultime

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 ?

Réponses

A terme, (<>) retourner aux processus **corrects** l'id du **même processus** (le leader ultime) **correct**

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 ( )*

Réponses

Upon the invocation of leader()
    return trusted_i

Q3

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

Réponses

Sans faute et les messages arrivent à temps => p1 élu

Avec faute => le processus correct de plus petit identifiant est élu

Q4

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

Réponses

Plusieurs leaders possibles si les messages du leader courant arrivent hors délais

- ⇨ erreurs corrigées à la reception du message du leader

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

Réponses

A chaque erreur => **augmentation du timeout** => il existe un temps, après le GST, timeout > à la borne inconnue

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

Q6

Montrez que :

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

Réponses

Idée: A terme, tous les processus précédent pleader sont fautifs (n'envoient pas de messages), après GST, il existe un temps après lequel tous les timeout vont atteindre la borne inconnue sur les temps de transmission => les messages de pleader arrivent à temps => trusted = pleader pour tous les corrects.

Q6

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.

Réponses

<>S :
completude forte : toutes les fautes doivent être détectées : inclus dans Pi et comme trusted n'est pas fautif => propriété assurée
justesse finalement faible : à terme, il existe un correct non faussement suspectés, trusted à terme est correct et pas dans les suspects => propriété assurée

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*

Q7

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.

Réponses

<>P : completude forte + justesse finalement forte => à terme plus d'erreurs

Task 2 : when (trusted_i < i) and (did not receive (I-AM-LEADER, suspected_trustedi) for p_trustedi during Delta_i,trusted)
    trusted_i ++
    if (trusted_i =i)
        suspected_i = {p1, …, pi-1}

Task 3 : when (received (I-AM-LEADER,suspected_j) and (j <= trusted_i)
    suspected_i = suspected_j
    if (j < trusted_i)

trusted_i = j
Delta_i,j = Delta_i,j +1;

Task 4 : when (trusted_i =i) and (did not receive I-AM-ALIVE from pj during last Delta_i,j) and (j>i)
suspected_i = suspected_i U {pj}

Task 5 : when (trusted_i = i) and (received I-AM-ALIVE from pj) and (pj in suspected_i)
suspected_i = suspected_i – {pj}
Delta_i,j = Delta_i,j +1;