

TD – Consensus – M2 ARA

Coordinateur tournant (Chandra-Toueg 1996)

On considère un système à n processus $\{1, 2, \dots, n\}$. Les processus peuvent subir des fautes franches et les canaux de communication sont fiables. Le nombre de fautes est inférieur à $n/2$. Les primitives R-Broadcast et R-deliver implémentent une diffusion fiable.

Soit l'algorithme de Coordinateur Tournant dont le pseudo-code est le suivant pour le processus p :

```
1. procedure propose( $v_p$ )
2.    $estimate_p = v_p$ 
3.    $state_p = undecided$ 
4.    $r_p = 0$ 
5.    $ts_p = 0$ 
6.   while  $state_p = undecided$ 
7.      $r_p = r_p + 1$ 
8.      $c_p = (r_p \bmod n) + 1$ 
9.     Phase 1:
10.    send ( $p, r_p, estimate_p, ts_p$ ) to  $c_p$ 
11.    Phase 2:
12.    if  $p = c_p$  then
13.      wait for  $(n+1)/2$  estimations ( $q, r_p, estimate_q, ts_q$ ) from processes  $q$ 
14.       $msgs_p[r_p] = \{(q, r_p, estimate_q, ts_q) \text{ received}\}$ 
15.       $t = \text{largest } ts_q \text{ such that } (q, r_p, estimate_q, ts_q) \in msgs_p[r_p]$ 
16.       $estimate_p = \text{select one } estimate_q \text{ such that } (q, r_p, estimate_q, t) \in msgs_p[r_p]$ 
17.      send ( $p, r_p, estimate_p$ ) to all
18.    Phase 3:
19.    wait until received ( $c_p, r_p, estimate_{c_p}$ ) from  $c_p$  or  $c_p \in suspected_p$ 
20.    if received ( $c_p, r_p, estimate_{c_p}$ ) from  $c_p$  then
21.       $estimate_p = estimate_{c_p}$ 
22.       $ts_p = r_p$ 
23.      send ( $p, r_p, ack$ ) to  $c_p$ 
24.    else
25.      send ( $p, r_p, nack$ ) to  $c_p$  /*  $p$  suspects that  $c_p$  crashed */
26.    Phase 4:
27.    if  $p = c_p$  then
28.      wait for  $(n+1)/2$  ( $q, r_p, ack$ ) or ( $q, r_p, nack$ ) from processes  $q$ 
29.      if  $(n+1)/2$  ( $q, r_p, ack$ ) received then
30.        R-Broadcast ( $p, r_p, estimate_p, decide$ )
31.    R-deliver( $q, r_q, estimate_q, decide$ )
32.    if  $stat = undecided$  then
33.      decide( $estimate_q$ )
34.       $state_p = decided$ 
```

Question 1

Rappelez les propriétés de la diffusion fiable et du consensus

Question 2

Proposez un algorithme simple de diffusion fiable.

Question 3

Exécutez l'algorithme sans faute en considérant 3 processus.

Question 4

Exécutez l'algorithme en considérant 3 processus avec le processus 2 qui tombe en panne.

Question 5

Quel est l'impact d'une fausse suspicion. Est-il possible que deux coordinateurs différents diffusent une décision ? Illustrez votre réponse par un scénario.

Question 6

Qu'est-ce qu'un accord uniforme ? Démontrez quel type d'accord assure cet algorithme.

Question 7

Quelle est l'utilité du message nack ?

Question 8

On souhaite maintenant démontrer la terminaison de l'algorithme. On considère différents cas :

- a) Un processus correct décide : quelle propriété de la diffusion fiable assure la terminaison ?
- b) En vous appuyant sur la complétude forte du détecteur $\diamond S$, montrez que les processus ne peuvent rester indéfiniment bloqués dans une ronde.
- c) En vous appuyant sur la justesse faible de $\diamond S$, montrez qu'il existe un temps à partir duquel, tous les processus décident.

Question 9

On souhaite maintenant améliorer les performances de cet algorithme en réduisant sa latence (son nombre de phases). Proposez une version qui réduit la phase d'acquittement. Indication : on ne cherche pas à réduire le nombre de messages.

Consensus probabiliste (Ben-Or 1983)

On considère un système à n processus $\{1, 2, \dots, n\}$ avec au maximum f fautes franches, $f < 2n$. Le système est asynchrone (pas de borne sur les délais de transmission et de traitements des messages). Les processus ont accès à un générateur de nombre aléatoire (r.n.g) qui retourne uniformément soit 0 soit 1.

L'algorithme suivant est exécuté par chaque processus p . Cet algorithme réalise un consensus binaire (entre deux valeurs 0 ou 1)

```
1. procedure consensus( $v_p$ ) { $v_p$  is the initial value of process  $p$ }
2.    $x \leftarrow v_p$  { $x$  is  $p$ 's current estimate of the decision value}
3.    $k \leftarrow 0$ 
4.   while true do
5.      $k \leftarrow k + 1$  { $k$  is the current phase number}
6.     send ( $R, k, x$ ) to all processes
7.     wait for messages of the form ( $R, k, *$ ) from  $n - f$  processes
8.     if received more than  $n/2$  ( $R, k, v$ ) with the same  $v$ 
9.       then send ( $P, k, v$ ) to all processes
10.    else send ( $P, k, ?$ ) to all processes
11.    wait for messages of the form ( $P, k, *$ ) from  $n - f$  processes
12.    if received at least  $f + 1$  ( $P, k, v$ ) with the same  $v \neq ?$ 
13.      then decide( $v$ )
14.    if at least one ( $P, k, v$ ) with  $v \neq ?$ 
15.      then  $x \leftarrow v$ 
16.    else  $x \leftarrow 0$  or  $1$  randomly {query r.n.g.}
```

Question 10

Lors d'une ronde k , est-il possible qu'un processus propose 0 et un autre 1 ?

Question 11

Si un processus p décide v à la ronde k , montrez que tout processus q qui commence à la phase $k+1$ positionne sa variable x_q à v et décide v à la fin de la phase $k+1$

Question 12

Dans quel cas une valeur aléatoire est choisie ?

Montrez que cet algorithme termine avec une probabilité de 1.