Aix*Marseille université
Socialement engagée

Année 2023-24

Domaine Sciences et Technologies
Master informatique
Fiabilité Logicielle : TD 4
Code UE : SINBU22
TD 4 : consensus

Algorithmes de consensus

La tâche distribuée du consensus consiste pour plusieurs processus (n processus nommé $P_1, P_2, \dots P_n$) à se mettre d'accord sur une même valeur.

Plus formellement, chaque processus P_i commence avec une valeur entière initiale v_i et doit décider d'une valeur entière d_i. Une fois qu'un processus a décidé une valeur, il ne peut plus changer d'avis et donc n'a pas le droit de décider une autre valeur. On appelle processus non défaillants, les processus qui ne sont pas affectés par une panne.

Un algorithme de consensus pour être correct doit respecter les trois propriétés suivantes :

- **Accord :** Les valeurs décidées par les processus non défaillants doivent être identiques.
- **Intégrité :** Les valeurs décidées par les processus non défaillants doivent faire partie des valeurs initialement proposées.
- **Terminaison :** Tous les processus non défaillants décident en un temps fini.

Afin de communiquer et décider de valeur, les processus ont accès aux primitives suivantes :

- `sendTo(j, m)` permet au processus exécutant cette primitive d'envoyer le message `m` au processus P_j.
- `receiveFrom(j)` retourne le dernier message envoyé par le processus P_j au processus exécutant cette primitive. Cet appel n'est pas bloquant : s'il n'y a aucun message reçu de P_j alors l'appel renvoie l'ensemble vide $\emptyset$.
- `sleep(m)` met le processus en pause pendant `m` millisecondes.
- `decide(x)` fait décider la valeur `x` au processus exécutant cette primitive.

Les processus et le réseau de communication peuvent subir des pannes :

- **pertes de messages** : des messages envoyés peuvent être perdu ;
- **crash de processus** : un processus peut arrêter de fonctionner (n'exécute plus de code ni n'envoie de messages) à tout moment.

Mauvaises approches pour le consensus

Avant d'étudier de véritables algorithmes de consensus, on va d'abord voir ce qui pose problèmes avec les solutions naïves.

Algorithme décider sa propre valeur

On considère un algorithme dans lequel chaque processus décide sa valeur initiale, c'est-à-dire que chaque processus P_i exécute `decide`(v_i).

Question 1 : Parmi les trois propriétés nécessaires pour le consensus (accord, intégrité et terminaison), quelles sont celles qui sont respectées par cet algorithme et quelles sont celles qui ne sont pas respectées ?

Algorithme valeur fixe

On considère un algorithme dans lequel chaque processus décide la valeur 1, c'est-à-dire que chaque processus P_i exécute `decide(1)`.

Question 2 : Parmi les trois propriétés nécessaires pour le consensus (accord, intégrité et terminaison), quelles sont celles qui sont respectées par cet algorithme et quelles sont celles qui ne sont pas respectées ?

Algorithme avec coordinateur

Une première approche raisonnable pour résoudre la tâche de consensus est de définir un processus fixé (le processus P_1) qui aura le rôle de coordinateur. Le coordinateur récupère toutes les valeurs des autres processus afin de calculer la valeur à décider. Ensuite, il diffuse la valeur calculée à tous les autres processus. Les processus non coordinateurs envoie leur valeur au coordinateur puis reçoive la valeur du coordinateur qui sera la valeur qu'ils vont décider. Le pseudo-code ci-dessous décrit de manière précise le fonctionnement de cet algorithme.

input : the processus number i, an initial value v_i and the number of processus n
output: a decided value d_i

```
1  if i == 1 then                                          // processus coordinateur
2  |  sleep(500);
3  |  m = v_i;
4  |  for j from 2 to n do  m = max(m, receiveFrom(j)) ;
5  |  decide(m);
6  |  for j from 2 to n do  sendTo(j, m) ;
7  else                                                    // processus non coordinateur
8  |  sendTo(1, v_i);
9  |  sleep(1000);
10 |  m=receiveFrom(1);
11 |  if m ≠ ∅ then
12 |  |  decide(m);
13 |  else
14 |  |  decide(v_i);
```

Algorithm 1: Algorithme avec coordinateur

Question 3 : Quel est le nombre de messages envoyés (exprimé en fonction de n) par l'ensemble des processus lors de l'exécution de l'algorithme s'il n'y a pas de pannes ?

Question 4 : Quelles sont les propriétés du consensus (accord, intégrité et terminaison) qui sont toujours garanties par cet algorithme dans toutes les exécutions possibles ?

Question 5 : Que peut-il se passer si un message met plus d'une seconde à être transmis ?

Question 6 : Que peut-il se passer si un message est perdu, c'est-à-dire qu'il n'est jamais reçu par le destinataire ?

Question 7 : En supposant que tous les messages soient reçus à temps (les appels à `receiveFrom` renvoient toujours une valeur différente de $\emptyset$), que se passe-t-il si un processus crashe avant le début de l'exécution de l'algorithme ?

Question 8 : Sous quelles conditions cet algorithme respecte toutes les propriétés du consensus ?

Algorithme messages symétriques

Une deuxième approche raisonnable pour résoudre la tâche de consensus est de faire que chaque processus transmet à chaque autre processus sa valeur initiale. Ensuite, chaque processus tente de récupérer le message de chaque autre processus, puis calcule la valeur qu'il décide à partir des valeurs récupérées. Le pseudo-code ci-dessous décrit de manière précise le fonctionnement de cet algorithme.

input : the processus number i, an initial value v_i and the number of processus n
output: a decided value d_i

```
1  m = v_i;
2  for j from 1 to n do
3  |  if j ≠ i then
4  |  |  sendTo(j, m)
5  end
6  sleep(500);
7  for j from 1 to n do
8  |  if j ≠ i then
9  |  |  r = receiveFrom(j);
10 |  |  if r ≠ ∅ then
11 |  |  |  m = max(m, r)
12 end
13 decide(m);
```

Algorithm 2: Algorithme messages symmétriques

Question 9 : Quel est le nombre de messages envoyés (exprimé en fonction de n) par l'ensemble des processus lors de l'exécution de l'algorithme ?

Question 10 : Quelles sont les propriétés du consensus (accord, intégrité et terminaison) qui sont toujours garanti par cet algorithme ?

Question 11 : Que peut-il se passer si un message met plus d'une seconde à être transmis ?

Question 12 : En supposant que tous les messages soient reçus à temps (les appels à `receiveFrom` renvoient toujours une valeur différente de $\emptyset$), que se passe-t-il si un processus crashe avant le début de l'exécution de l'algorithme ?

Question 13 : Sous quelles conditions cet algorithme respecte toutes les propriétés du consensus ?

Algorithme messages symétriques avec boucle

On considère une variante de l'algorithme précédent dans lequel chaque processus attend d'avoir reçu un message de chaque autre processus. Le pseudo-code ci-dessous décrit de manière précise le fonctionnement de cet algorithme.

input : the processus number i, an initial value v_i and the number of processus n
output: a decided value d_i

```
1  m = v_i;
2  for j from 1 to n do
3  |  if j ≠ i then
4  |  |  sendTo(j, m)
5  end
6  for j from 1 to n do
7  |  if j ≠ i then
8  |  |  repeat
9  |  |  |  sleep(500);
10 |  |  |  r = receiveFrom(j);
11 |  |  until r ≠ ∅;
12 |  |  m = max(m, r);
13 end
14 decide(m);
```

Algorithm 3: Algorithme variante messages symmétriques

Question 14 : Quelles sont les propriétés du consensus (accord, intégrité et terminaison) qui sont toujours garanties par cet algorithme dans toutes les exécutions possibles même en cas de panne ?

Question 15 : Sous quelles conditions cet algorithme respecte toutes les propriétés du consensus ?

Algorithme de consensus en rondes

On considère l'algorithme suivant (écrit en pseudo-code) et pour lequel la variable r (définissant un nombre de rondes) est un paramètre de l'algorithme qui est le même pour chaque serveur.

```
   input : the server number i, an initial value v_i, a number of rounds r and a number of servers n
   output: a decided value d_i
 1 V = {v_i};
 2 W = V;
 3 for k from 1 to r do                                          // faire r rondes
 4 |   for j from 1 to n do
 5 |   |   if j ≠ i then
 6 |   |   |   sendTo(j, W);
 7 |   |   end
 8 |   end
 9 |   sleep(500);
10 |   R = ∅;
11 |   for j from 1 to n do
12 |   |   if j ≠ i then
13 |   |   |   R = R ∪ receiveFrom(j)
14 |   |   end
15 |   end
16 |   W = R \ V;                                  // On retire les valeurs déjà connues
17 |   V = V ∪ R;
18 |   sleep(500);
19 end
20 decide(min(V));
```

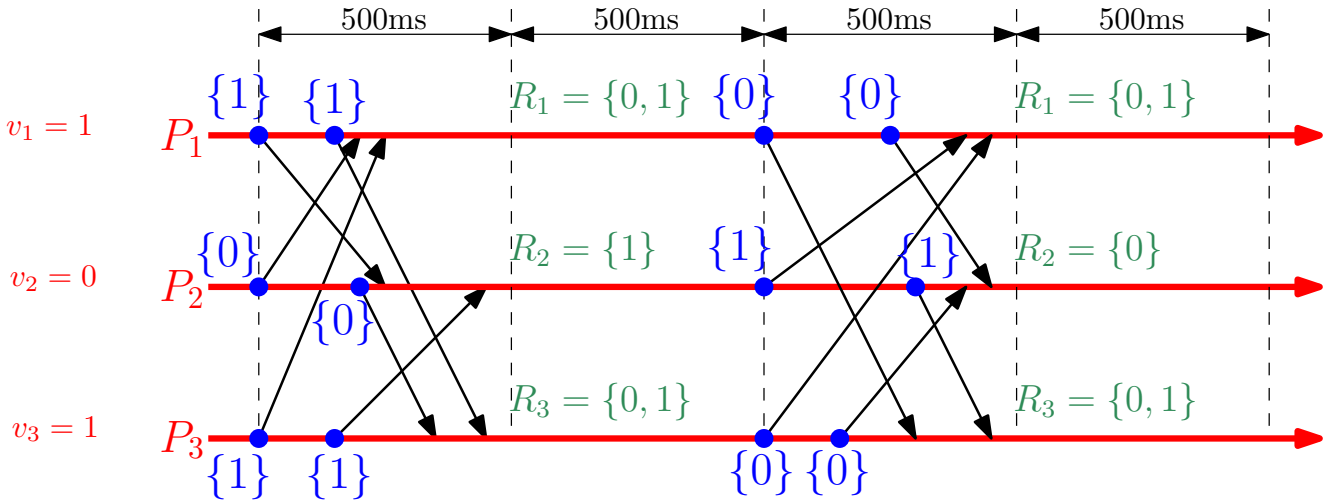
Algorithm 4: Algorithme avec rondes

> **Question 16 :** Quel est le nombre de messages envoyés (exprimé en fonction de n et r) par l'ensemble des processus lors de l'exécution de l'algorithme ?

> **Question 17 :** Pour toute valeur de r, est-ce que la propriété de terminaison du consensus est toujours respectée par l'algorithme ?

On note R_i, W_i et V_i les valeurs de R, W et V pour le processus P_i.

On considère ci-dessous le diagramme d'une exécution avec trois serveurs et deux rondes ($n = 3$, $r = 2$).



Question 18 : Pour chaque processus P_i, quelle est la valeur décidée d_i pour cette exécution ? Est-ce que cela respecte les propriétés d'accord et de validité du problème du consensus ?

On suppose maintenant que certains des serveurs peuvent tomber en panne définitive au début ou pendant l'exécution de l'algorithme entre deux instructions de l'algorithme.

Question 19 : On considère trois serveurs (dont un peut crasher) et une ronde ($r = 1$). Décrivez une exécution en précisant le scénario des pannes, c'est-à-dire en indiquant quand et quelles pannes se produisent, pour laquelle les valeurs décidées par l'algorithme ne respectent pas une des trois propriétés du consensus.

Question 20 : On considère cinq serveurs (dont deux peuvent crasher) et deux rondes ($r = 2$). Décrivez une exécution en précisant le scénario des pannes pour laquelle les valeurs décidées par l'algorithme ne respectent pas une des trois propriétés du consensus.

Question 21 : On considère cinq serveurs (dont trois peuvent crasher) et quatre rondes ($r = 4$). Est-ce que pour toutes les exécutions avec tous les scénarios de pannes possibles (en considérant que les messages sont reçus en moins de 500 millisecondes), l'algorithme décide des valeurs respectant les trois propriétés du consensus ?