Systèmes et algorithmes répartis

Consensus, détecteur de défaillances

Philippe Quéinnec

ENSEEIHT
Département Sciences du Numérique

20 janvier 2021

plan

Plan

Le consensus

Définition

Soit un ensemble de processus $p_1, \ldots, p_n$ reliés par des canaux de communication.
Initialement : chaque processus p_i propose une valeur v_i.
À la terminaison de l'algorithme : chaque p_i décide d'une valeur d_i.

- Accord : la valeur décidée est la même pour tous les processus corrects
- Intégrité : tout processus décide au plus une fois (sa décision est définitive)
- Validité : la valeur décidée est l'une des valeurs proposées
- Terminaison : tout processus correct décide au bout d'un temps fini

1. *The Byzantine Generals Problem*, Leslie Lamport, Robert Shostak and Marshall Pease. ACM Trans. on Programming Languages and Systems. 1982.

Remarques

Correct / défaillant

Un processus est dit correct s'il n'est et ne sera jamais défaillant.
Un processus incorrect peut fonctionner normalement avant de défaillir.

Valeur proposée / décidée

- Les valeurs proposées ne sont pas nécessairement toutes distinctes.
- Le consensus binaire (uniquement 0/1 comme valeurs possibles) est équivalent au consensus à valeur quelconque.
- La valeur décidée n'est pas nécessairement une valeur majoritaire, ni celle d'un processus correct.

Variantes – simultanéité

Démarrage

Quand un processus démarre-t-il l'algorithme de consensus ?

- Démarrage simultané (à une heure donnée)
- Démarrage initié par l'un des processus : diffusion d'un message d'initialisation de l'algorithme.
  (Attention aux propriétés de cette diffusion : fiable, ordonnée)
- Sur réception d'un 1er message de l'algorithme : ça complique !

Les trois formes sont équivalentes.

Consensus simultané

- Terminaison simultanée : tous les processus corrects décident *en même temps* (= au même tour, modèle synchrone).

Variantes – 2

Consensus uniforme

- Accord uniforme : la valeur décidée est la même pour tous les processus (corrects ou ultérieurement défaillants) qui décident

Correct = n'aura jamais de défaillance. Un processus incorrect peut décider puis devenir défaillant.

k-consensus

- k Accord : au plus k valeurs distinctes sont décidées pour l'ensemble des processus corrects

Consensus basique : $k = 1$

Consensus approximatif

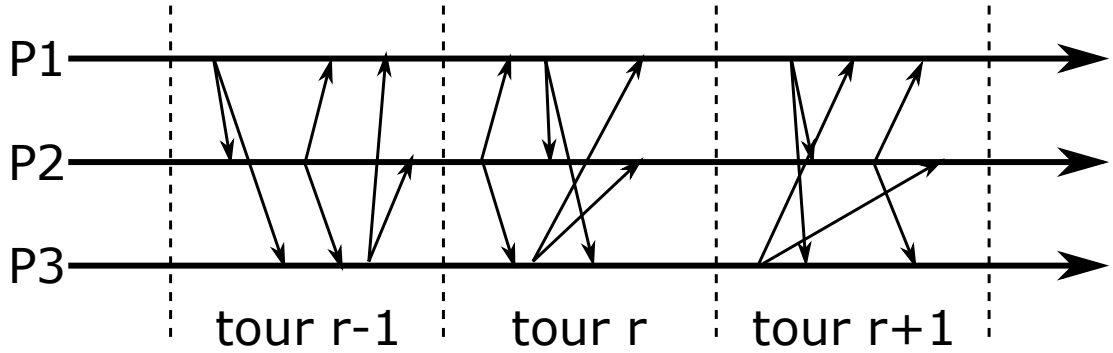
- ϵ-Accord : les valeurs décidées par les processus corrects doivent être à distance maximale ϵ l'une de l'autre.

Modèle temporel

Synchrone

borne supérieure connue sur le temps de transmission et sur l'avancement des processus.

Usuellement, algorithmes fonctionnant par tours, synchronisés sur tous les processus.



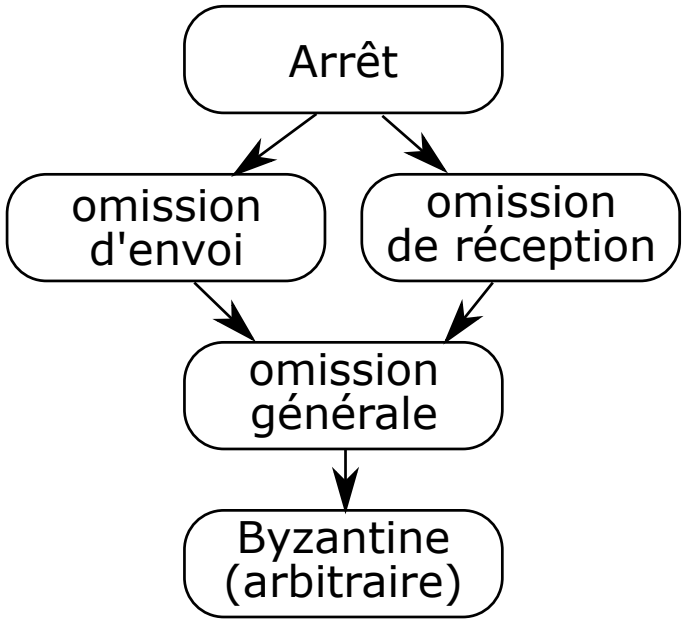
Asynchrone

Pas de borne connue : avancement arbitrairement lent des processus et du réseau.

Modèle moins contraint, plus réaliste (mais plus difficile)

Défaillances d'un processus

- Arrêt (*crash failure* ou panne franche) : le processus fonctionne correctement jusqu'à un point où il cesse définitivement d'agir.
- Omission
  - omission en émission : le processus omet certaines émissions qu'il aurait dû faire, ou cesse définitivement.
  - omission en réception : le processus ignore certains messages en réception, ou cesse définitivement.
- Arbitraire (*byzantine failure*) : le processus ment (par omission ou par contenu arbitraire des messages envoyés)



1. *Fault-Tolerant Broadcasts and Related Problems*, Vassos Hadzilacos and Sam Toueg. In *Distributed Systems*. 1993.

Communications

Défaillance

- réseau fiable : tout message finit par arriver
- perte : certains messages n'arrivent jamais
- ordre : respect de l'ordre d'émission ou d'un autre ordre
- arbitraire : duplication, modification du contenu...

Hypothèse de réseau fiable

Les défaillances réseau en asynchrone peuvent être modélisées par des défaillances de site $\Rightarrow$ on suppose le réseau fiable.

Utilité du consensus

Le consensus est un outil générique pour la tolérance aux fautes :

- Un système informatique est une machine à état :
  (nouvel état, sorties) $\leftarrow$ *fonction(état courant, entrée)*
- Assurer la disponibilité = réplication en n copies
- Transparence = équivalence avec une seule copie
- Consensus pour ordonner identiquement les entrées
  + si non déterministe, consensus pour décider identiquement des réponses sur les n copies

1. *The Implementation of Reliable Distributed Multiprocess Systems*, Leslie Lamport. Computer Networks. 1978.

Universalité

♪♪♪

Spécification séquentielle

Un objet possède une spécification séquentielle si ses comportements corrects sont exprimables par des séquences (= des traces) de ses opérations.

Universalité du consensus

Le consensus suffit pour implanter en réparti n'importe quel objet possédant une spécification séquentielle.

- En communication par message : consensus + diffusion générale (anonyme)

1. *Wait-Free Synchronization*, Maurice Herlihy. ACM Trans. on Programming Languages and Systems. 1991.

Problèmes réalisables avec le consensus ♪♪♪

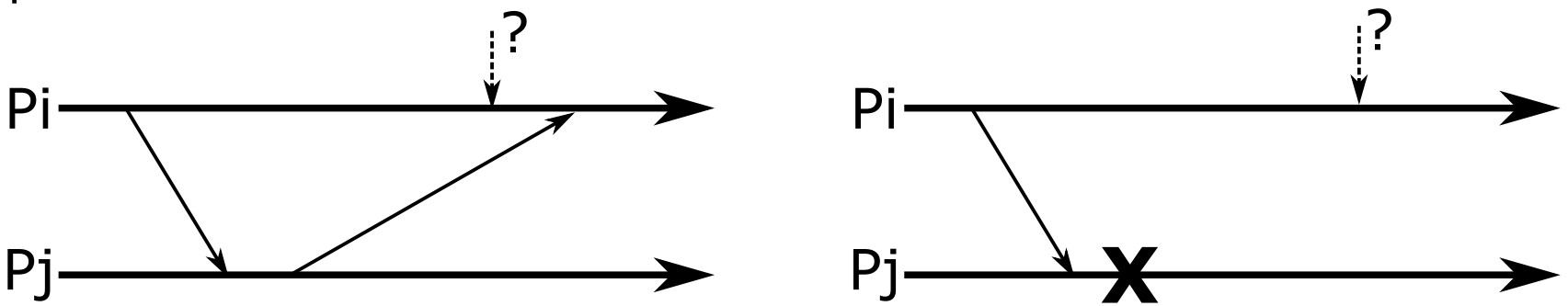
- Élection d'un leader = accord de tous sur un processus
- Diffusion fiable avec terminaison = tous les processus corrects délivrent un même message (éventuellement vide si l'émetteur s'est arrêté)
- Diffusion uniforme = tout les processus (corrects ou pas) délivrent ou aucun
- Construction de groupes
- Commit (validation) de transaction distribuée
- Calcul d'une fonction globale portant sur l'ensemble des sites

Impossibilité du consensus en asynchrone avec arrêt ♪♪♪

Résultat d'impossibilité (FLP85)

Le consensus est impossible à réaliser dans un système asynchrone où un seul processus peut subir une défaillance d'arrêt.

Intuitivement, il est impossible de distinguer un processus lent d'un processus arrêté.



1. *Impossibility of Distributed Consensus with One Faulty Process*, Michael Fischer, Nancy Lynch and Michael Paterson. Journal of the ACM. April 1985.

Impossibilité du consensus en synchrone avec perte de message ♪♪♪

Résultat d'impossibilité (Gray, 1978)

Le consensus est impossible à réaliser dans un système synchrone où les messages peuvent être arbitrairement perdus.

Intuition de la preuve : le dernier message avant décision peut être perdu sans changer la décision $\Rightarrow$ il était inutile. On le supprime, et on recommence le raisonnement.

(Piège : la perte de message peut être modélisée par une défaillance d'omission de *n'importe quel* processus. Le consensus est faisable en synchrone avec omission s'il y a $< n/2$ sites défaillants.)

1. *Notes on Data Base Operating Systems*, Jim Gray. Operating Systems, An Advanced Course, 1978.

Résultats d'impossibilité de réalisation du consensus

Processus communiquant par mémoire partagée

Défaillance / modèle	Arrêt de processus
synchrone	$\exists$ solution
asynchrone	impossible

Processus communiquant par messages

Défaillance / modèle	Arrêt de processus	Omission	Byzantine	Perte de message
synchrone	$\exists$ solution	$\exists$ solution	$\exists$ solution	impossible
asynchrone	impossible	impossible	impossible	impossible

Contourner FLP

♪♪♪

Affaiblir le problème

- Terminaison probabiliste
- k-consensus
- Consensus approximatif (ϵ-consensus)
- *Best effort*

Renforcer le système

- Système partiellement synchrone
- Système synchrone *suffisamment longtemps*
- Détecteur de défaillances

Réalisabilité du consensus

défaillance	synchrone	asynchrone
non	faisable	faisable
arrêt	faisable f défaillances $< n$ processus $\Omega(f+1)$ tours	impossible
omission	faisable f défaillances $< n/2$ processus	impossible
byzantine	faisable $f \leq \lfloor (n-1)/3 \rfloor$ processus $\Omega(f+1)$ tours	impossible

- k-consensus : faisable en asynchrone/arrêt avec $f < k < n$
- Consensus approximatif : faisable en asynchrone/arrêt avec $5f + 1 \leq n$