Plan

Réalisation en absence de défaillance

Principe

Tous les processus diffusent leur valeur, chacun garde la plus petite reçue.

Synchrone $\Rightarrow$ borne supérieure de communication $\Delta \Rightarrow$ décision en un tour et n diffusions.

```
Processus P_i(v_i),  0 ≤ i < n
local V_i
on round 0 :
    V_i ← {v_i}
    envoyer(v_i) à tous les autres
on réception(v) :
    V_i ← V_i ∪ {v}
on round 1 :
    décider min(V_i)   (toute fonction déterministe)
```

Réalisation en défaillance d'arrêt

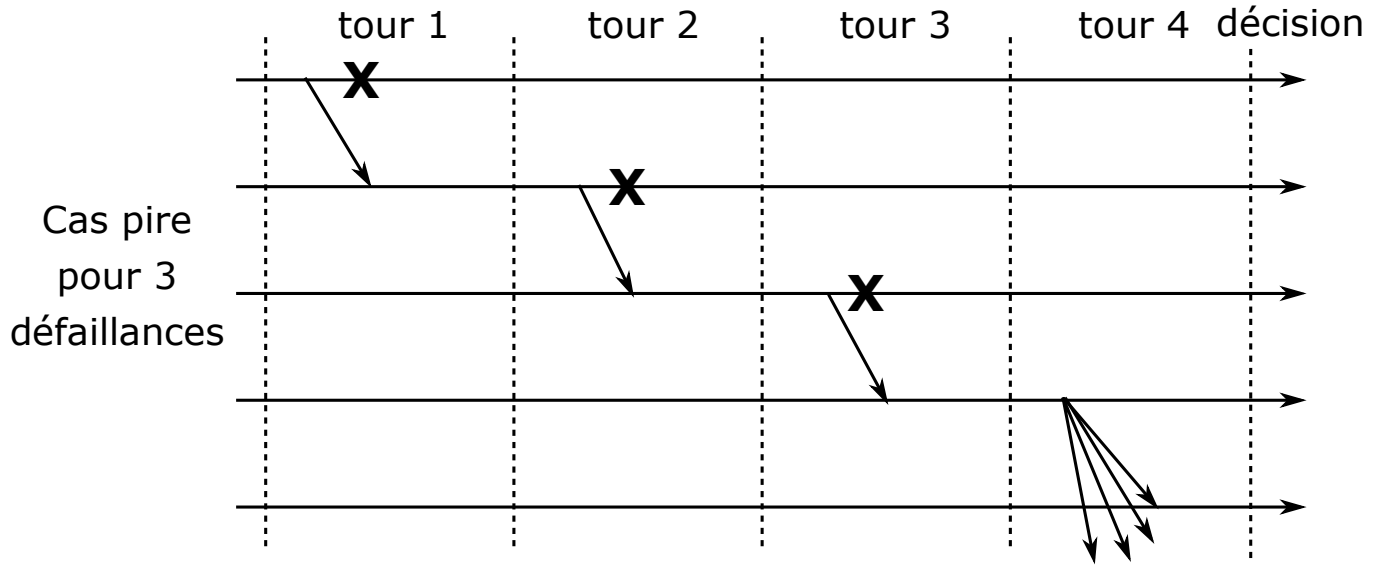
Tolérance à f défaillances ($f < n$).

Principe

Au i-ième tour, le processus i diffuse sa valeur. Après $f+1$ tours, on est sûr qu'au moins un processus correct a diffusé une valeur reçue par au moins $n-f$ processus corrects.

```
Processus P_i(v_i),  0 ≤ i < n
local x
on start :
    x ← v_i
on réception(v) :
    x ← v
on round i, 0 ≤ i ≤ f :    // au i-ième tour pour P_i
    envoyer(x) à tous les autres
on round (f + 1):
    décider x
```

Réalisation en défaillance d'arrêt



- $f + 1$ tours, $f + 1$ diffusions
- décision simultanée
- non équitable : les valeurs des processus $f + 1, \ldots, n$ ne sont pas considérées $\Rightarrow$ ajouter un tour préliminaire :

  `diffuser(`v_i`)`
  $x \leftarrow$ `l'une des valeurs reçues dans ce tour`

Algorithme équitable à $f+1$ tours

Principe

À chaque tour, chaque processus diffuse la plus petite valeur qu'il connaît (uniquement si elle a changé).

```
Processus P_i(v_i), 0 ≤ i < n
local x, prevx, received
on start :
    x ← v_i, prevx ← ⊥
on réception(v) :
    received ← received ∪ {v}
on round k, 0 ≤ k ≤ f :          // à chaque tour
    prevx ← x
    x ← min(received ∪ {x})
    received ← ∅
    si x ≠ prevx alors diffuser(x)
on round (f + 1):
    décider x
```

Nombre optimal de tours

Borne inférieure

Il n'existe pas d'algorithme synchrone basé sur des tours qui résolve le consensus avec f défaillances d'arrêt en moins de $f + 1$ tours.

Le pire cas est qu'un seul processus défaille à chaque tour.

Existence

La borne "$f + 1$ tours" est atteignable.

1. *A Lower Bound for the Time to Assure Interactive Consistency*, Michael Fischer and Nancy Lynch. Information Processing Letters. 1982.

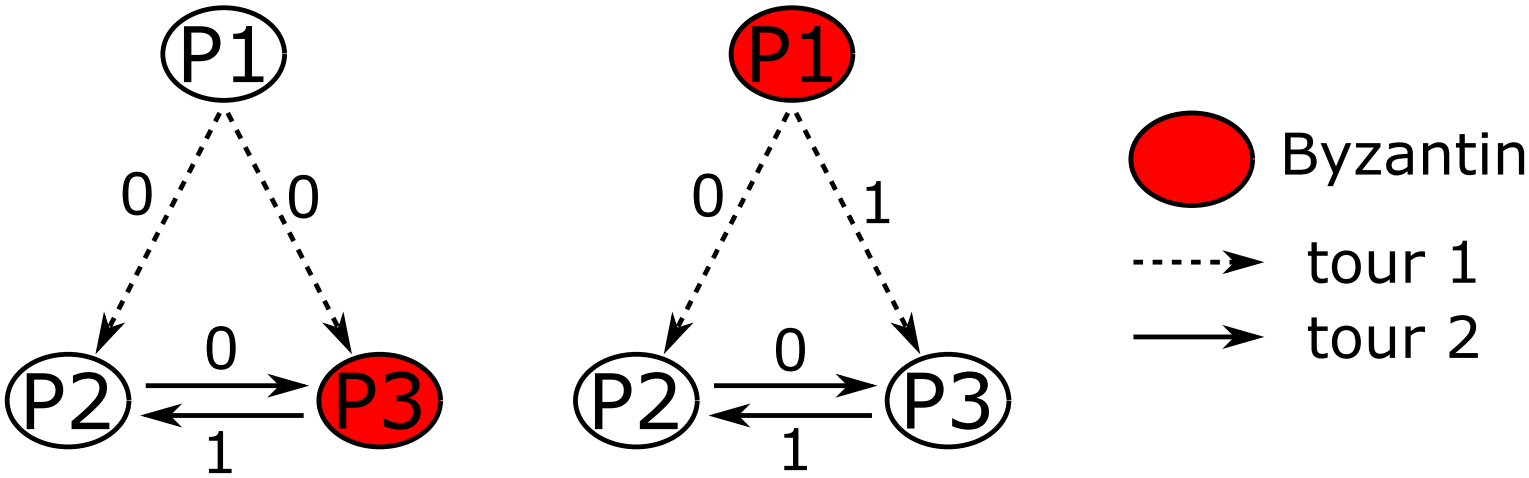
Le problème des généraux byzantins

Les généraux byzantins

Des divisions de l'armée Byzantine assiègent une cité et doivent décider d'attaquer ou pas. Chaque division est commandée par un général qui communique avec les autres par des messagers fiables.
Chaque général doit éventuellement décider d'un plan d'action (terminaison) ; le plan doit être le même pour tous (accord) ; un général ne doit pas changer de décision une fois prise (intégrité) ; la décision retenue doit avoir été proposée et en particulier s'ils sont unanimes, ceci doit être la décision finale (validité).
Certains généraux sont des traîtres qui veulent empêcher les généraux loyaux de conclure. Pour cela, ils peuvent envoyer des messages contradictoires aux autres généraux ou mentir sur ce qu'ils ont reçu des autres généraux. Les traîtres peuvent même se coaliser pour conspirer de manière coordonnée.

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

Impossibilité du consensus à 3 avec une défaillance byzantine



- Défaillance byzantine = le processus peut mentir
- P_2 ne peut pas distinguer entre les deux scénarios
- Des échanges supplémentaires ne changent rien
- P_2 ne peut pas nécessairement décider identiquement à l'autre processus correct

Algorithme avec défaillances byzantines

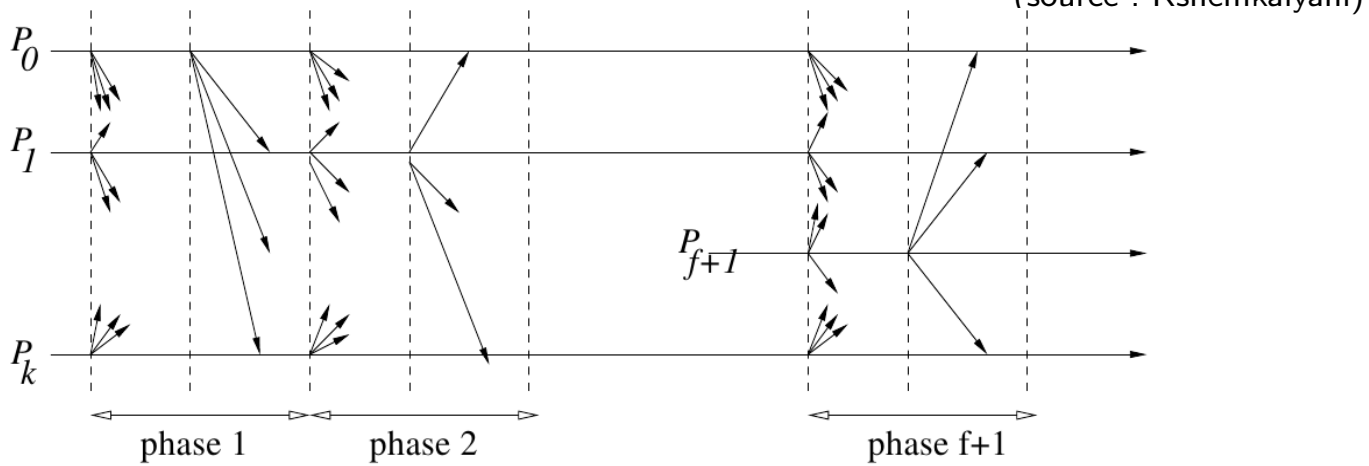
Le consensus en synchrone avec défaillance byzantine est impossible pour $n \leq 3f$.

Le consensus en synchrone avec défaillance byzantine est possible si $f \leq \lfloor \frac{n-1}{3} \rfloor$.

Algorithme avec "roi de phase" (*Phase King*) ♪♪♪

- $f + 1$ phases, chaque phase a un unique roi, fixé a priori
- Chaque phase contient deux tours :

(source : Kshemkalyani)



  - Tour 1 : chaque site diffuse son estimation à tous.
    Chaque site reçoit les valeurs proposées puis détermine si une valeur est proposée par une majorité qualifiée ($> n/2 + f$), ou une majorité simple ($> n/2$)
  - Tour 2 : le roi fixe son estimation à sa valeur majoritaire (sa propre valeur si pas de majorité) et la diffuse.
    Chaque site fixe sa nouvelle estimation à la valeur reçue en tour 1 avec majorité qualifiée, ou sinon à la valeur reçue du roi.

Algorithme avec "roi de phase" – justification

$f+1$ phases, $(f+1)(n+1)(n-1)$ messages,
$f < \lceil n/4 \rceil$ défaillances

- Parmi les $f+1$ phases, au moins une où le roi est correct
- Dans cette phase, les processus corrects obtiennent la même estimation que le roi : soit p_i et p_j corrects :
  - ou p_i et p_j ont chacun une majorité qualifiée ($> n/2+f$) et le roi a alors aussi cette même estimation
  - ou p_i a une majorité qualifiée ($> n/2+f$) et p_j utilise la valeur du roi ($> n/2$)
  - ou p_i et p_j utilisent l'estimation du roi
- Si tous les processus corrects ont la même estimation au début d'une phase, ils garderont cette même valeur à la fin (même si le roi est byzantin).