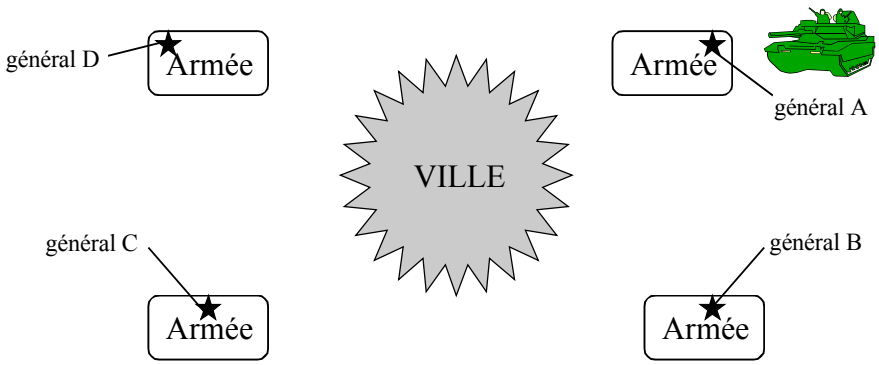
Problèmes des Généraux Byzantins

– Problème qui modélise une situation où la plupart des sites d'un réseau fonctionne correctement mais quelques sites fonctionnent de la manière la plus *maligne* possible.



Problématique

– N divisions de l'armée (Byzantine) encerclent une ville ennemie.
– Chaque division est commandée par un général. Les généraux communiquent entre eux par messagers fiables.
– Les généraux doivent décider d'un plan d'action commun (attaque ou retraite) ACCORD. Si les généraux sont unanimes par rapport à leur choix initial alors la décision = ce choix. (VALIDITE)
– Mais il peut exister des traîtres parmi les généraux qui essayent d'empêcher les généraux loyaux de se mettre d'accord.

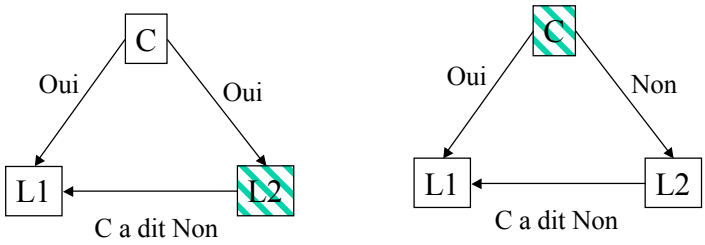
Variante

- Parmi les généraux on distingue le commandant en chef et les lieutenants. Les traîtres peuvent se trouver parmi les lieutenants et le commandant
- Le commandant envoie un ordre à ces N-1 lieutenants tel que
  - Tous les lieutenants loyaux obéissent au même ordre
  - Si le commandant est loyal, alors chaque lieutenant loyal obéit à l'ordre émis.

Hypothèses

- f sites possiblement fautifs
  - Un site fautif peut se comporter de manière complètement arbitraire. (ne pas envoyer de messages, en envoyer un autre, ne pas envoyer du tout …). La difficulté vient qu'un message reçu peut paraître plausible pour le receveur alors qu'il n'est pas correct…
- Les conditions de validité, accord et terminaison sont les mêmes énoncées dans le cas des pannes franches.

3 généraux dont un traître. Qui croire ?

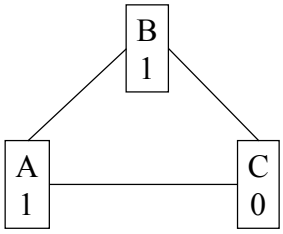


- L1 devrait décider Oui dans le 1er cas
- L1 devrait décider Non dans le 2ème cas

L1 ne peut pas différencier les 2 cas de figures.

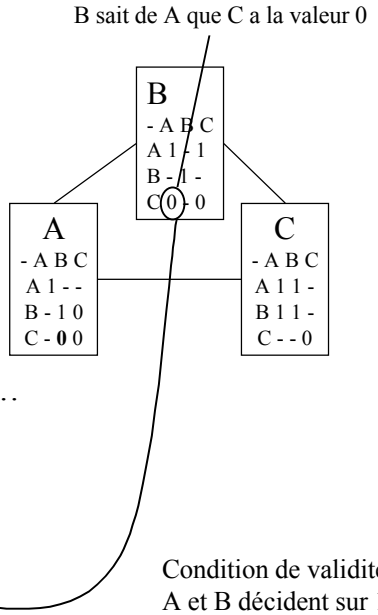
=> Généralisons le propos …

Résultat d'impossibilité: 1ère exécution



- 3 sites A, B et C résolvent le problème du consensus avec une faute byzantine
- Tour 1: chaque site envoie sa valeur initiale (pas de faute)
  - A → B,C la valeur 1
  - B → A, C la valeur 1
  - C → A, B la valeur 0

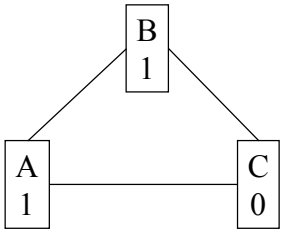
Résultat d'impossibilité



– Tour 2: A, B envoie la valeur qu'ils ont reçus, C se comporte bizarrement …
  - A → B la valeur 0 de C
  - A → C la valeur 1 de B
  - B → A la valeur 0 de C
  - B → C la valeur 1 de A
  - **C → A la valeur 0 de B**
  - C → B la valeur 1 de A

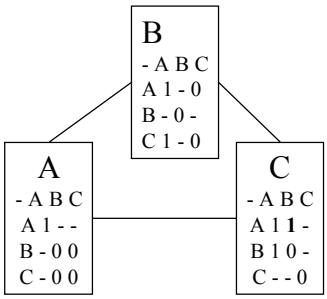
Condition de validité =>
A et B décident sur 1

Résultat d'impossibilité: 2ème exécution



– 3 sites A, B et C résolvent le problème du consensus avec une faute byzantine
– Tour 1: chaque site envoie sa valeur initiale (pas de faute)
  - A → B,C la valeur 1
  - B → A, C la valeur 0
  - C → A, B la valeur 0

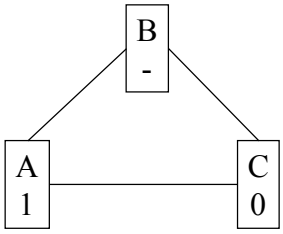
Résultat d'impossibilité



– Tour 2: B et C envoie la valeur qu'ils ont reçus,
A se comporte bizarrement …
  - B → A la valeur 0 de C
  - B → C la valeur 1 de A
  - C → A la valeur 0 de B
  - C → B la valeur 1 de A
  - A → B la valeur 0 de C
  - **A → C la valeur 1 de B**

Condition de validité =>
A et B décident sur 0

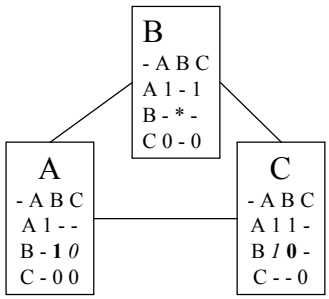
Résultat d'impossibilité: 3ème exécution



– 3 sites A, B et C résolvent le problème du consensus. B a un comportement byzantin dès le début.
– Tour 1: chaque site envoie sa valeur initiale
  - A → B,C la valeur 1
  - **B → A la valeur 1**
  - **B → C la valeur 0**
  - C → A, B la valeur 0

Résultat d'impossibilité

- Tour 2: Comportement normal
  - A → B la valeur 0 de C
  - A → C la valeur 1 de B
  - B → A la valeur 0 de C
  - B → C la valeur 1 de A
  - C → A la valeur 0 de B
  - C → B la valeur 1 de A



- Exécution 3
  - B envoie les mêmes messages à A que dans l'exécution 1
  - B envoie les mêmes messages à C que dans l'exécution 2 (dans les mêmes tours)
  - => Exec1 et Exec3 ne sont pas distinguables pour A
  - => Exec2 et Exec3 ne sont pas distinguables pour C

  - Dans exec1 A décide 1 donc de même dans Exec3
  - Dans exec2 C décide 0 donc de même dans Exec3

  - => ceci contredit l'hypothèse que A, B et C résolvent le problème du consensus.

Conditions pour résoudre le problème

- Il n'existe pas de solution pour le consensus avec fautes byzantines si $f \geq N/3$ où f = nombre de fautes et N = nombre de sites.
  - L'exemple précédent n'est que le premier pas de la preuve : le problème du consensus ne peut pas être résolu avec f = 1 et N = 3.
  - Il faut étendre la démonstration dans un cadre plus général

$$N > 3.f$$

Un algorithme

Le même que pour les pannes franches

- Hypothèses
  - n > 3.f
  - T (n,f)
  - Graphe complet, Système synchrone etc.
- Différences
  - Possibilité de recevoir des messages tronqués ou altérés : le site qui les reçoit peut détecter que ces messages sont anormaux
  - Procédure de décision va être différente car « on » ne peut pas faire confiance aux valeurs reçues. => utilisation d'un vote majoritaire.

Algorithme

Algorithme (local en i)

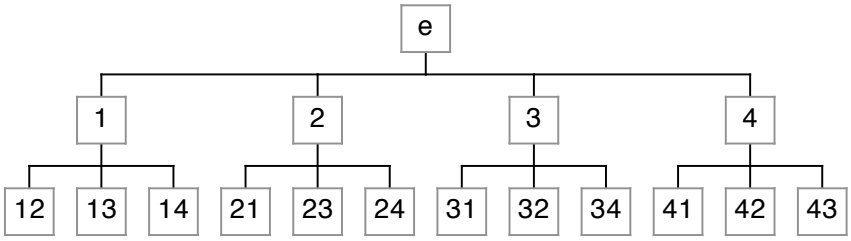
- i maintient T_i (n,f)
- val(x) désigne la valeur stockée dans le nœud d'étiquette x
- initialement val(e) = valeur d'entrée de i

- **Tour 1**
  - i diffuse val(e) à tous les sites (i lui-même inclus)
  - Réception d'un message contenant v de j
    - Si msg correct /*pas d'altération dans la forme*/ alors val(j) ← v
    - Sinon val(j) ← !
  - Si aucun message reçu de j alors val(j) ← ! /**vérifiable car synchrone**/

- **Tour k** , $2 \leq k \leq f+1$
  - i diffuse tous les couples (x,v) où x est une étiquette de niveau k-1 de T qui ne contient pas l'indice i
  - Réception de message de j avec val(x) = v ∈ V
    - Si msg correct alors val(x.j) ← v où x.j = étiquette de niveau k de T
    - Sinon val(x.j) ← !
  - Si aucun message reçu alors val(x.j) ← !
- A la **fin** du tour f+1, i applique une règle de décision

- Procédure de décision$_i$ (x)
  - Si x est une feuille alors decision$_i$ (x) ← val(x)
  - Sinon decision$_i$ (x) = Majorité(decision$_i$(x1),…, decision$_i$(xk)) où x1, …, xk étiquettes de fils de x dans T
  - Majorité : donne la valeur v qui apparaît en majorité (stricte) parmi les valeurs de tous les fils de x dans T. Si majorité n'existe pas alors valeur par défaut !
  - La décision finale est donné par decision$_i$ (e)

Exemple : n=4, f= 1



- 1 a 0 comme valeur initiale
- 2 a 0 comme valeur initiale
- 3 a 1 comme valeur initiale
- 4 a 0 comme valeur initiale
- Scénario : 3 est le site byzantin, il envoie au tour 2 les valeurs 1 à la place de 0 pour 1 2 et 4.

Complexité

- En nombre de messages

A chaque tour chaque site envoie n messages

Pour les n sites : n^2 messages échangés

Algorithme en f+1 tours : $(f+1).(n^2)$ messages

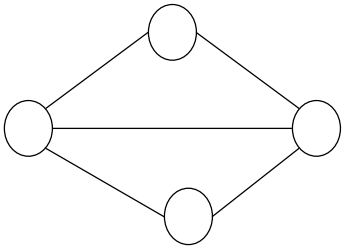
- En quantité d'informations échangées

A chaque tour, chaque site envoie un niveau entier de son arbre soit au pire $n(n-1)(n-2)\ldots(n-f-1) = \Theta\ (n^{f+2})$

- Ne pas oublier que l'on doit avoir $n > 3.f$

Consensus byzantin sur des graphes quelconques

- K-connexité:
  - Un graphe G est dit k-connexe ssi chaque couple de nœuds dans G est connecté par au moins k chemins disjoints
  - Arbre est 1-connexe
  - Clique est (N-1)-connexe



Nombre minimum de nœuds que l'on peut enlever pour rendre G connexe

- Théorème : Consensus byzantin peut être résolu dans un réseau de N sites avec f fautes si et seulement si

  $N > 3.f$

  G est au moins 2(f+1)-connexe (connexité(G) > 2f)

2f+1 connexe => il existe au une majorité de chemins disjoints fiables. On peut donc simuler un algorithme pour graphes complets qui résout le consensus

Reste à montrer que cela n'est pas possible pour G moins que 2f-connexe. Preuve par l'absurde en prenant f =1 puis en généralisant pour tout f.

Résultat d'impossibilité

- Fisher Lynch Paterson:Le consensus distribué n'a pas de solution dans un système asynchrone même s'il n'y a qu'un seul site fautif et même si le site le site n'est simplement fautif en arrêtant d'émettre des messages.

- Attention ce résultat ne signifie pas que le CD ne peut être atteint. Mais cela veut dire qu'il n'existe pas d'algorithme déterministe qui garantit que le consensus sera atteint.