PPS à Trouville

Impossibilité Byzantine

(travail en cours et en collaboration **avec Maurice Herlihy)**

Christine Tasson
Christine.Tasson@pps.univ-paris-diderot.fr

le 11 septembre 2012

- **Systèmes distribués :** fiabilité par redondance du code (dans l'aéronautique, multiprocesseurs,...)
- **Le problème du (k-)consensus :** choisir le résultat correct, les processeurs encore valides,...
- **La topologie :** faire le lien entre solvabilité et existence de fonctions continues entre objets topologiques.

Résultat :

Une méthode mathématique pour montrer l'impossibilité de certaines tâches distribuées.

Feuille de route

1. Le modèle Byzantin
2. Le cas asynchrone
3. Le cas synchrone

Le modèle Byzantin

Quels systèmes distribués ?

- **Système :** $n+1$ processus s'échangent des messages (identifiant, valeur) via des canaux fiables.
  Un algorithme donné pour chaque processus.
- **Erreur :**
  - Par crash. Le processus arrête d'envoyer des messages.
  - Byzantine. Le processus n'exécute pas l'algorithme prévu et envoie des valeurs traitres.
- **Communication :**
  - Asynchrone. Aucune contrainte sur la durée d'une étape de l'algorithme
  - Synchrone. Les processus effectuent les étapes de leur algorithme en un temps déterminé.

Problème : Trouver un algorithme tel que

Chaque processus correct termine avec une valeur de décision compatible avec la tâche donnée quelles que soient les valeurs initiales, quel que soit l'entrelacement des traces d'exécutions et quelles que soient les erreurs commises par les autres processus.

Consensus : Tous les processus corrects ont la même valeur de décision.

k-consensus : Au plus k-valeurs de décisions différentes.

Dans les 2 cas : Si tous les processus commencent avec la même valeur, c'est la valeur de décision.

Le cas asynchrone

Comment attaquer le problème ?

Approche algorithmique :

- Proposer un algorithme ($\Rightarrow$ Problème résoluble)
- Supposer l'existence d'un algorithme et exhiber des traces d'exécution incompatibles ($\Rightarrow$ Problème non résoluble)

Approche topologique ($\Rightarrow$ Résultat d'impossibilité) :

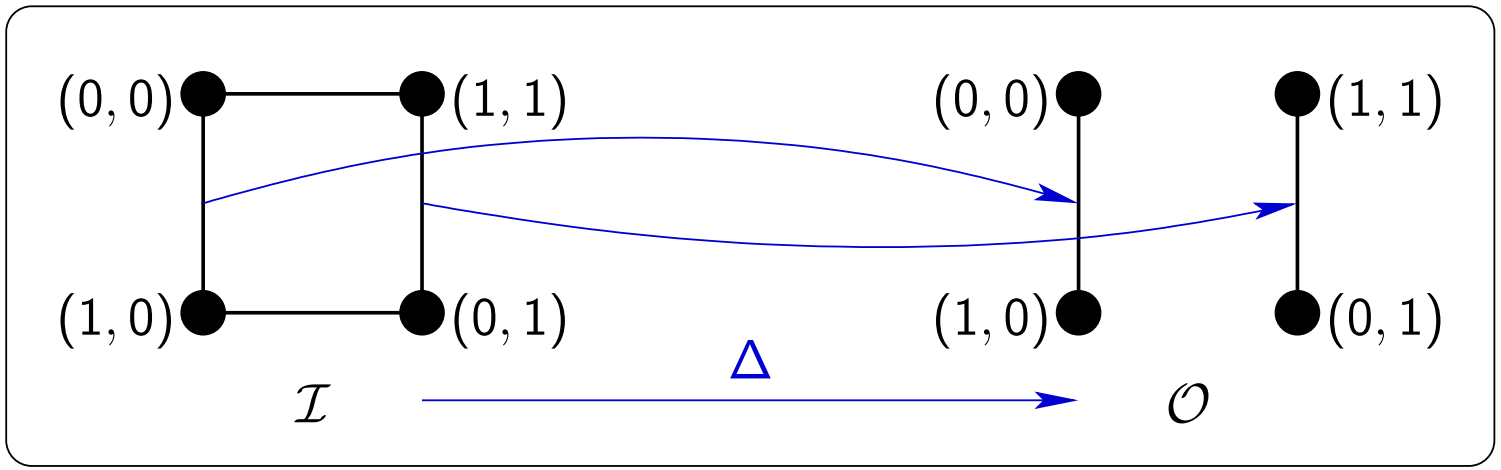
- Par l'absurde, supposer l'existence d'un algorithme
- Séparer cet algorithme en deux étapes :
  - le protocole historique

    ```
    Lire les messages
    Concaténer à l'historique
    Envoyer l'historique
    ```

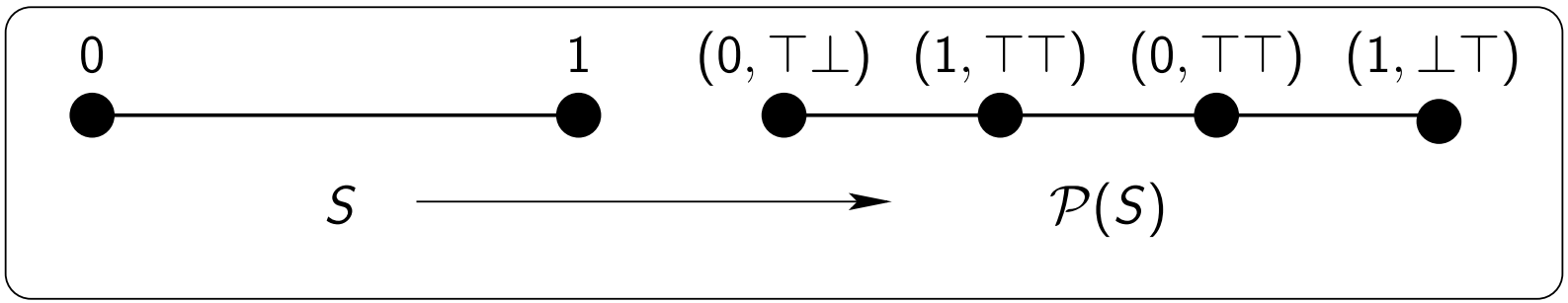
  - une procédure de décision

    δ : Historique $\mapsto$ Valeur de décision
- Modéliser le problème de façon topologique et trouver une obstruction topologique.

Représentation topologique du 1-consensus : $(\mathcal{I}, \mathcal{O}, \Delta)$



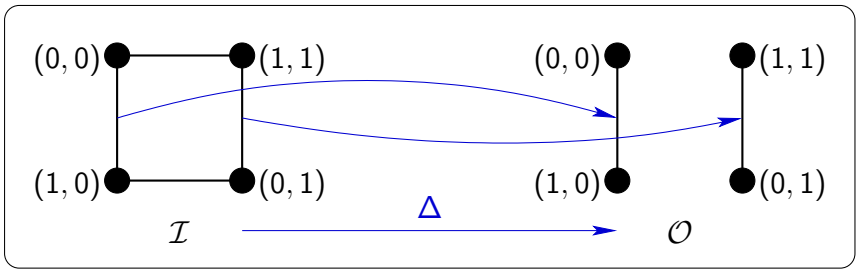
Le complexe du protocole historique : $\mathcal{P}$



Solution : $\delta : \mathcal{P}(\mathcal{I}) \to \mathcal{O}$ simpliciale et compatible avec Δ.

k-connexité : Une fonction simpliciale préserve la connexité.

- $\mathcal{C}$ subdivision de $\mathcal{I}$
- $\mathcal{C}$ est k-connexe
- $\delta(C) = \mathcal{O}$
- $\mathcal{O}$ n'est pas k-connexe



Application : Impossibilité du k-consensus avec $t \geq k$ crash :

- $\mathcal{P}_t$ complexe des exécutions avec exactement t crash
- $\mathcal{P}_t(\mathcal{I})$ est k-connexe si $t \geq k$
- $\mathcal{P}_t(\mathcal{I})$ contient les simplexes étiquetés par $i \in \{1, \ldots, k\}$)
- Un simplexe étiqueté par une seule valeur est préservé par δ
- Les simplexes étiquetés par i ne sont pas k-connexes dans $\mathcal{O}$.

CONTRADICTION

Le k-consensus byzantin n'est pas toujours résoluble

$$\mathbf{t \geq k}$$

Version algorithmique :

- Supposons l'existence d'un algorithme dans le cas Byzantin.
- Un processus Byzantin peut simuler une erreur par crash.
- Donc il existe un algorithme dans le cas des crash.

CONTRADICTION

Traduction topologique :

- $\mathcal{P}_t(\mathcal{I})$ est k-connexe si $t \geq k$
- $\delta : \mathcal{P}_t(\mathcal{I}) \to \mathcal{B}(\mathcal{I}) \to \mathcal{O}$ est simpliciale
- δ préserve le simplexe étiqueté par i (pour $i \in \{1, \ldots, k\}$)

CONTRADICTION

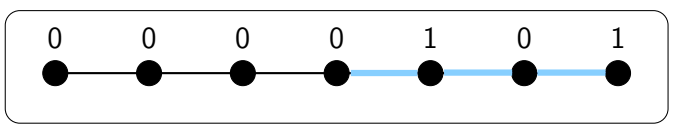
Le cas synchrone

Lemme de Sperner.

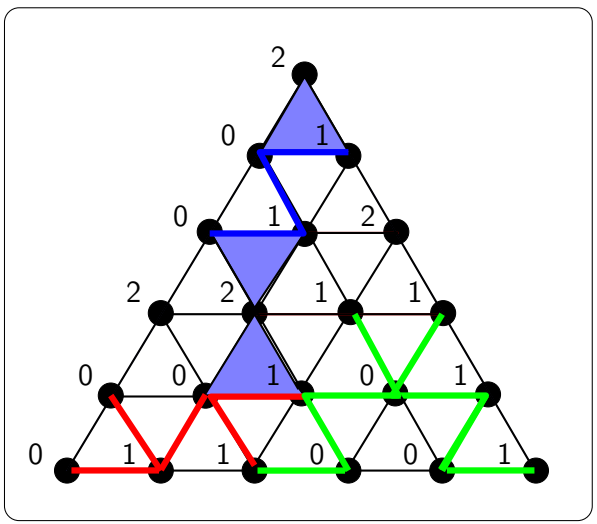
Une subdivision d'un k-simplexe étiqueté par un coloriage de Sperner, possède un nombre impair de simplexes étiquetés par $k+1$ valeurs différentes.

Démonstration :

$k = 1$



$k = 2$



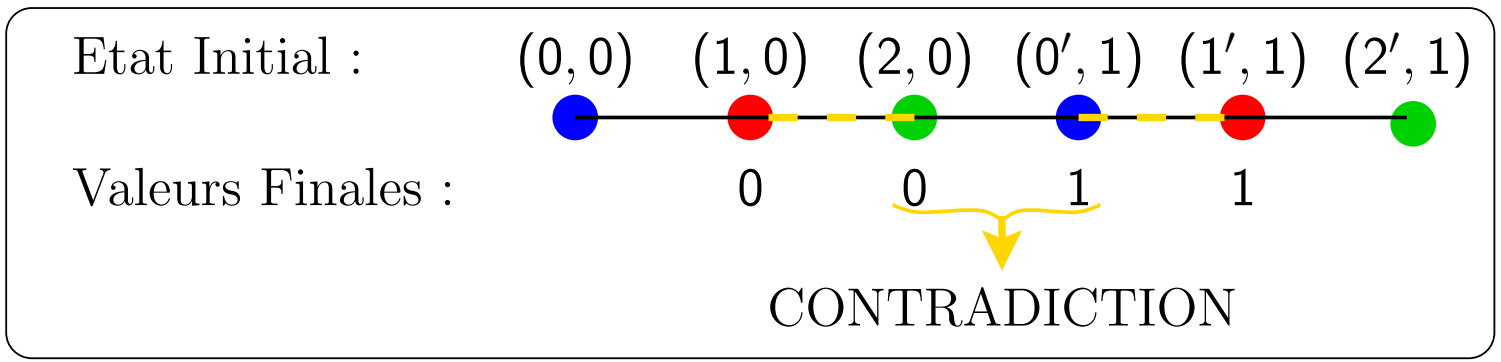
Ça marche pour $k = 1, 2, \ldots$

Le consensus synchrone, impossible si $n \leq 3t$

Théorème : Impossibilité du consensus synchrone ($k = 2$)

Il n'existe pas d'algorithme qui résolve le consensus dans un système distribué synchrone de $n = 3$ processus parmi lesquels $t = 1$ est byzantin.

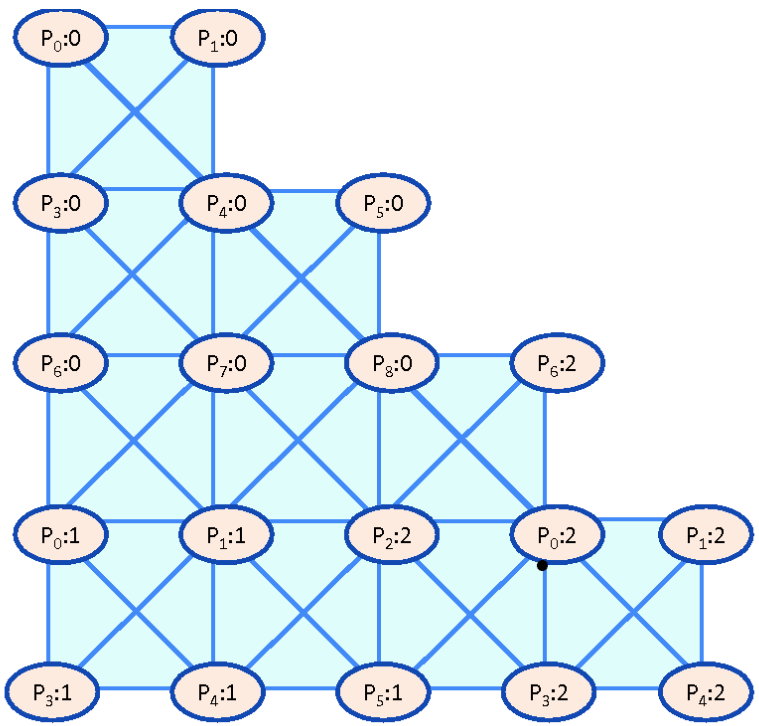
Démonstration. (Lynch - Distributed Algorithms)



Théorème : Impossibilité du 2-consensus synchrone

Il n'existe pas d'algorithme qui résolve le 2-consensus dans un système distribué synchrone de $n = 3^2$ processus parmi lesquels $t = 3^2 - 2^2$ est byzantin.

Démonstration.



Théorème (Impossibilité du consensus synchrone)

Il n'existe pas d'algorithme qui résolve le k-consensus dans un système distribué synchrone de $n = 3^k$ processus parmi lesquels $t = 3^k - 2^k$ sont byzantins.

Démonstration. Par l'absurde, on suppose l'existence d'un algorithme qui résout le k-consensus que l'on exécute sur un graphe de communication $\mathcal{I}'$ tel que :

- tout sommet est relié au plus une fois à un autre processus.
- tout sommet apparaît dans un simplexe de dimension $n - t$.
- $\mathcal{I}'$ est une subdivision d'un k-simplexe dont les extrémités sont colorées par $k + 1$ valeurs.

Conclusion

- Modèle asynchrone : classification complète des tâches.
- Modèle synchrone optimal ?
- Pertinence du modèle ? $=>$ vers les probabilités.