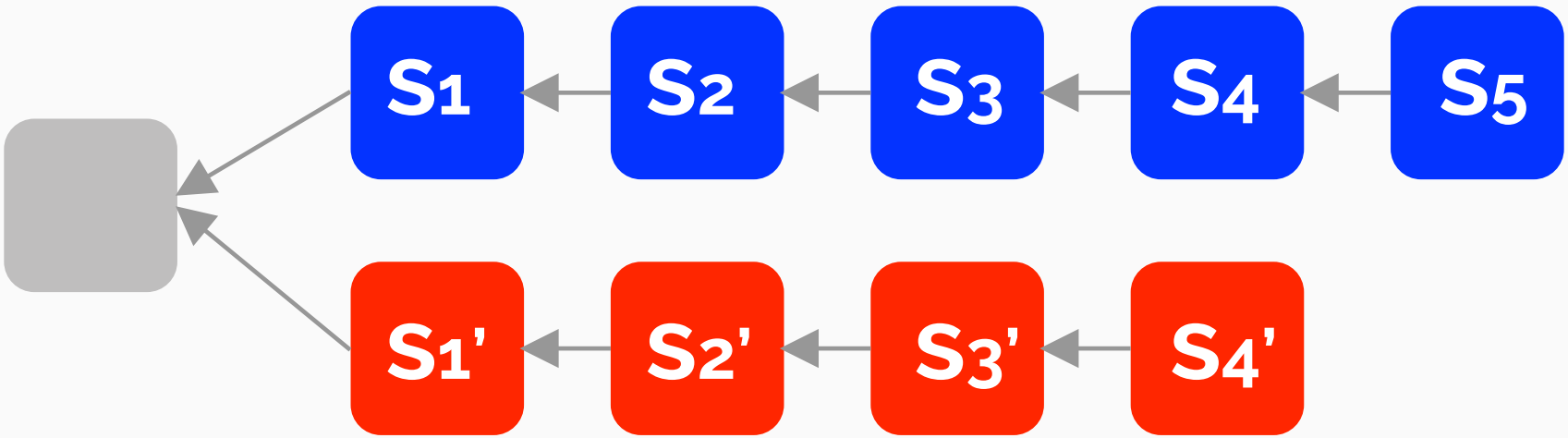
# Algorithmes de Consensus

3 octobre 2021

# Rappel : nécessité du consensus



# Contexte

**Problème fondamental de calcul distribué**

Tous les processus doivent se mettre d'accord (atteindre un consensus) afin d'être sûres qu'elles ont obtenu la même valeur.

**Objectif**

- Parvenir à la fiabilité d'un système.
- Doit pouvoir tolérer la présence de "pannes".

# Tolérance aux pannes

## Deux types de pannes

- Crash : un processus s'arrête brusquement.
- Panne Byzantine : un processus agit de manière imprévu mauvais messages envoyés, délais de réponse, etc.

## Tolérances aux pannes

Le système doit continuer à fonctionner même en présence de pannes.

# Problème des généraux Byzantins

Des généraux de l'armée Byzantine campent autour d'une cité ennemie. Ils ne peuvent communiquer qu'à l'aide de messagers et doivent établir un plan de bataille commun, faute de quoi la défaite sera inévitable. Cependant un certain nombre de ces généraux peuvent s'avérer être des traîtres, qui essayeront donc de semer la confusion parmi les autres. Le problème est donc de trouver un algorithme pour s'assurer que les généraux loyaux arrivent tout de même à se mettre d'accord sur un plan de bataille.

# Propriété attendue : sûreté

**Sûreté (Safety)**

- Finalité : pas de retour en arrière possible.
- Cohérence : les processus disposent de la même valeur.
- Entente : les processus sont d'accord sur la valeur.

$\Rightarrow$ Il n'y a pas de longues "branches" (*fork*) dans une chaîne.

# Propriété de finalité

**Définition**

- Un bloc (ou une transaction) est dite finale si elle **reste** sur la chaîne de chaque processus.
- Confirmations : nombre de blocs qui doivent arriver avant d'atteindre la finalité.

# Finalité probalistique

**Finalité probalistique**

- Après $n$ confirmations, un bloc est *probablement* final.
- La probabilité de *fork* décroit (exponentiellement) en fonction de $n$.

Exemple : Proof of Work (PoW)

# Finalité *à la* Nakamoto

**Nakamoto-style**

- Tolère au maximum, 1/2 d'acteurs *Byzantin*.
- La production de bloc dépend proportionnellement d'une ressource : *e.g.* puissance de calcul, *stake*
- Stratégie des attaquants : construire une plus longue chaîne en secret en utilisant ses propres ressources
- Les attaquant disposent d'une minorité des ressources : au bout d'un moment, ils vont perdre.

$\Rightarrow$ Attaquable si au moins 51% des ressources du réseau sont contrôlés par des attaquants.

# Finalité deterministe

**Finalité déterministe (ou absolue)**

- Après $n$ confirmations, un bloc est **final**.
- $n$ est une constante connue
  (finalité immédiate quand $n = 0$)

Exemple : Tendermint [1]

1. Algorithme de consensus de la blockchain *Cosmos*

# Algorithmes BFT classiques

## Classic BFT-style

- Tolère au maximum 1/3 d'acteurs *Byzantin*
- $n$ acteurs, $f$ acteurs Byzantins, $n = 3f + 1$
- Basé sur une notion de *quorum*
- Un quorum est atteint lorsque $2f + 1$ d'acteurs sont d'accords.

# Propriété attendue : Progrès

**Progrès (*Liveness*)**

- Le système doit être capable de progresser en un temps borné.
- Peut-être impossible sous certaines conditions :
  - Trop d'acteurs Byzantins.
  - Asynchronie des messages.

# Propriété attendue : Équité (Blockchains)

**Équité (*Fairness*)**

- Les acteurs honnêtes sont récompensés de manière équitable.
- Les acteurs malhonnêtes doivent être punis proportionnellement à leur nuisance (Proof of Stake).

# Modèles réseaux

**Synchrone**

- Tous les messages arrivent **et** en un temps borné.

**Asynchrone**

- Les messages peuvent dupliqués/retardés/perdus.

**Partiellement synchrone**

- Synchrone mais la borne de temps n'est pas connue.

# Théorème FLP

**Théorème FLP (1985 - Fischer, Lynch and Patterson)**

- Le consensus est impossible pour des processus asynchrones s'il peut se produire au moins une défaillance.

# Protocoles Proof of Work *à la* Bitcoin (*c.f.* Cours 1)

**Prover**

- "Mine" un hash de bloc qui sera préfixé par $n$ zéros.
- Peut prouver aux autres l'effort de calcul.
- Récompensé pour son travail.

**Verifier**

- Vérifie l'effort de calcul instantanément

# Protocoles Proof of Stake

**Séléction du producteur de bloc**

- Séléction basée sur la quantité de jetons (*stake*).
- Processus de décision déterministe via le protocole économique
- Aucune puissance de calcul requise : la validité dépend de l'identité du producteur.

Problème :
Que faire si le producteur publie deux blocs différents ?

# Problème du *Nothing at stake*

Créer deux blocs au même niveau :

- En Proof of Work, créer deux blocs au même niveau (fork) coûte cher en puissance de calcul.
- En Proof of Stake, si il n'y a pas de coût à créer un bloc, il n'y a rien à perdre à créer de fork (Nothing at stake).

Rappel : Fork $\Rightarrow$ Réduit la possibilité de finalité.

**Solution**

- Punir (économiquement) ce comportement.

# Algorithme de consensus : Emmy (Tezos)

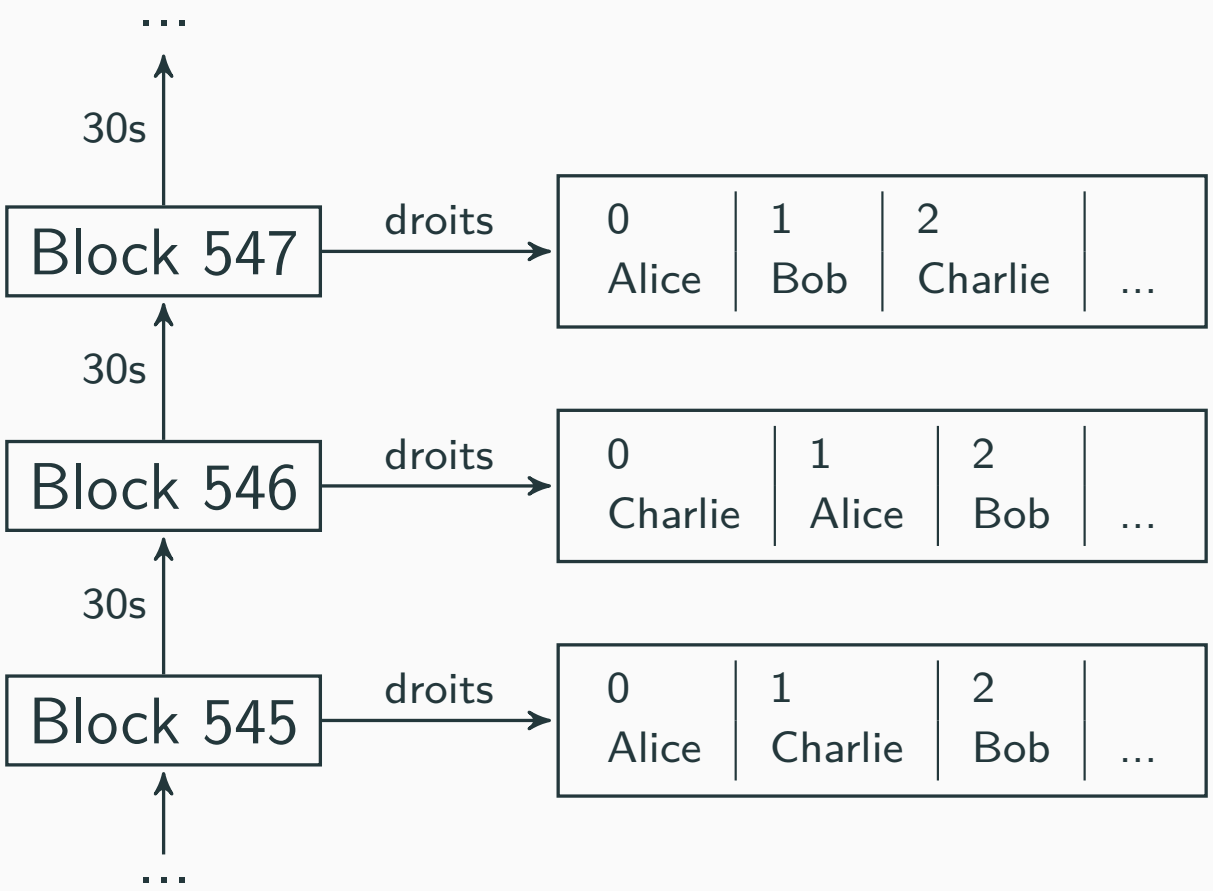
# Emmy : caractéristiques

- “Liquid” Proof of Stake
- Finalité probabiliste
- Droits déterminés proportionnellement en fonction du *stake*
- ~~Mining~~ $\Rightarrow$ **Baking**

# Droits de “baking”

- Pour chaque level, on tire, au hasard, une liste de comptes.
- Le premier compte de la liste peut commencer à “baker” au temps $t$ (30s après le prédecesseur), le deuxième compte à $(t + 30s)$, le troisième à $(t + 60s)$, etc.
- Liste infinie pour assurer la **liveness** : si un “baker” est inactif, on attend le prochain.

# Droits de “baking” – exemple



# Calcul des droits de baking

**Le montant de jetons varie dans le temps**

- "Snapshots" régulier de la distribution d'argents des participants

**Les droits de baking ne doivent pas être manipulable**

- Tous les 32 blocs, le baker désigné doit fournir le hash d'une graîne d'aléa (secret).
- Après un **cycle** (4096 blocs), tous les bakers ayant injecté des hashs de graînes doivent réveler leurs graînes.
- Toutes les graînes sont rassemblées pour former la prochaine graîne d'aléa qui permettra de tirer les nouveaux droits (à partir d'un snapshot aléatoire).

# Dépot de garantie

Pour résoudre le problème de “Nothing at stake” :

- Dès qu'un bloc est produit, un dépot de garantie est gelé ($640_{tz}$ par bloc).
- Si un “baker” produit deux blocs au même niveau, on enlève tous les dépots encore gelés.
- Les dépôts sont rendus au fur et à mesure, après un certain temps, si le “baker” n'a pas triché.
- Relation entre le stake et la somme des dépôts de garantie.

Incitation économique à rester honnête

# *Delegated Proof of Stake*

- Pour avoir des droits de production de bloc, les participants doivent avoir $8{,}000_{tz}$ : un **rouleau**
- On détermine les droits en fonction du nombre de rouleaux.

Si un participant n'ont pas assez de jetons pour participer (et donc obtenir des récompenses), il peut **déléguer** son *stake* ou encore l'associer à d'autres participants.

À l'heure actuelle, il existe des "services de délégations" qui acceptent les délégations et gardent une petite partie des récompenses (entre 5% et 15%) en échange du service.

# *Liquid Proof of Stake*

Delegated PoS mais :

- Les déléguants peuvent utiliser leurs jetons (pas de gel).
- Les “bakers” ne peuvent pas utiliser les jetons délégués.
- Les déléguants peuvent changer de délégués à tout moment.

# Activité

- Un baker devient inactif lorsqu'il n'a pas produit de bloc depuis un certain temps.
- Seuls les bakers actifs sont séléctionnés pour participer.

# Inflation

Chaque participant peut gagner jusqu'à 6% de ses jetons chaque année :

- Dès qu'un bloc est produit, la récompense est distribuée.
- L'inflation est maximisé à 6% (constante de protocole)

# Finalité probabiliste

Un bloc est final lorsqu'il a $+99\%$ de chances de faire partie de la chaîne finale.

Pour augmenter rapidement cette probabilité :

- On demande aux participants de désigner les blocs qu'ils souhaitent finaliser via des votes : **endorsements**.
- Plus un bloc est "endorsé", plus sa probabilité de finalité est grande :

$$\begin{aligned} \text{fitness de bloc} = \quad & \text{fitness du prédecesseur} \\ & + 1 \\ & + \text{nombre d'endorsements} \end{aligned}$$

# Finalité probabiliste (2)

Un bloc est final lorsqu'il a $+99\%$ de chances de faire partie de la chaîne finale.

Plus une séquence de blocs possède d'endorsements, plus sa probabilité d'être finale est grande.

**Hypothèse Nakamato-style**

- $+50\%$ des participants sont honnêtes.

# Endorsements

- Les droits d'endorsements sont déterminés de la même manière que les droits de baking.
- Un endorsement pour un bloc $n$ doit être inclus dans le bloc $n + 1$.
- On récompense égalements les endorsements.
- On prend également un dépôt de garantie.
- Deux endorsements à un même niveau pour deux blocs différents $\Rightarrow$ punition
- Seuls les délégués actifs sont séléctionnés.