TIPE Les cryptomonnaies

Matthias Goffette

Lycée La Martinière Monplaisir

Lyon, 15 Mars 2017

Motivations et objectifs

- 9 milliards de dollars de bitcoins dans le monde
- Principe jeune et expérimental
- Champs variés d'application des principes informatiques utilisés
  - Différentes cryptomonnaies spécifiques
  - Cadastre
  - Lutte contre les spams
- Objectifs du TIPE
  - Programmer une cryptomonnaie simplifiée
  - Etudier l'évolution du système ainsi représenté en fonction de certains paramètres

Principes d'une cryptomonnaie

- Possibilité de réaliser des transactions (Alice envoie à Bob 10 BTC)
- Réseau de nœuds, possédant un historique des transactions (*blockchain*) et réalisant des opérations de vérification des nouvelles transactions (*Proof Of Work*) par des calculs (*minage*)
- Problème
  - Alice envoie 10 BTC à un marchand, retarde l'heure de son ordinateur, puis renvoie ces 10 BTC à elle-même

Sommaire

Proof of work

- *Hash* du bloc dépend du contenu et du *nonce* ; doit être inférieur à une difficulté
- Donc nécessité de tester tous les nonces un à un
- Exemples :
  - Alice envoie 10 BTC à un marchand, retarde l'heure de son ordinateur, puis renvoie ces 10 BTC à elle-même
  - Alice doit alors lutter seule contre la puissance générale du réseau lors de la *proof of work*
- Incitation en récompensant les mineurs par 25 BTC

Questionnements

- *Proof of Work* très gourmande en énergie
- *Blockchain* lourde (71 Go en juin 2016)
- Croissance exponentielle de la *blockchain* et du nombre de calculs à réaliser lors de la PoW

- *Proof of Stake* : fonction de vérification qui nécessite moins de calculs car se base sur des critères comme le solde de l'utilisateur ou son ancienneté selon la monnaie

Première modélisation

- Objectif : programmer une cryptomonnaie simplifiée
- Principe :
  - 2 utilisateurs qui réalisent des transactions entre eux
  - Calculent des PoW, les ajoutent à la blockchain, comparent les blockchains pour mettre à jour l'historique
- Réalisation
  - Code Python
    - Module *blockchain*, module *PoW* (fonction SHA256, librairie Python), module utilisateurs
  - Fonction SHA256 : fonction cryptographique ayant une grande résistance à la collision
- Résultats
  - Conception de l'architecture d'une cryptomonnaie
  - Vérification de la validité d'un bloc de la *blockchain*

Seconde modélisation : objectifs et principes

- Objectifs
  - Comparer les méthodes de *PoW* et *PoS*
  - Etudier la vitesse de remplissage de la *blockchain* et la rapidité de diffusion au sein du réseau d'utilisateurs en fonction des paramètres
    - nb d'utilisateurs, nb de tours de boucle, difficulté
- Principes
  - n utilisateurs regroupés dans une liste
  - Chaque tour, pour chaque utilisateur :
    - PoW ou PoS
    - Comparaison de la *blockchain* avec 10 utilisateurs choisis au hasard

Réalisation

- Code Python
- 1 module pour calculer l'état des n utilisateurs après t itérations
- PoW avec fonction SHA256 ; nombre zéros en première position
- Pour chaque utilisateur :

  `[blockchain_i, puissance, solde, date de création]`

Conclusion

- Premiers résultats
  - Diffusion plus importante si la difficulté est plus élevée
  - Dans notre modèle, si la puissance d'un seul utilisateur est plus élevée que celle des autres, sa *blockchain* a tendance à prédominer
- Suites du travail
  - Comparer la *Proof of Work* et la *Proof of Stake*
  - Quels sont les paramètres de PoS les plus économes tout en conservant la sécurité de la PoW