Comment optimiser l'architecture d'un réseau pour résister aux attaques

Matthias Goffette

Lycée La Martinière Monplaisir

Lyon, 15 Mars 2017

Motivations et objectifs

- Réseaux dans tous les domaines : informatique, biologie, sociologie
- Sécuriser les réseau est un point primordial
  - Utilisés dans des systèmes critiques (finance, réseaux informatique d'entreprises...)
  - De plus en plus d'attaques pour récupérer les données des utilisateurs
  - Un réseau doit pouvoir être résistant
- Objectifs du TIPE
  - Modéliser des réseaux
  - Simuler des attaques, et en faisant varier certains paramètres, étudier la vulnérabilité

Sommaire

Les objets du réseau

- Agent : (id, *strategie*, informations)
- Tunnel : (emetteur récepteur)
- Information : (id, destinataire, texte, passeurs)
- Réseau : (liste d'agents, liste de tunnels)

Fonctionnement général

- Fonctionnement multi-agents, en effectuant de multiples *itérations* sur le réseau
- Itération :
  - Parcours des agents un à un
  - Si *normal* : passe son information à tous les voisins qui ne la possèdent pas
  - Si *attaquant* : envoie à tous ses voisins qui ne possèdent pas encore l'information une information de même `id`
- 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]`

Résultats : Diffusion des *blockchains* au sein des utilisateurs en fonction de la difficulté

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