TP n°2 - Modélisation d'un Système de Confiance dans un Réseau Blockchain

Module: Blockchain

Objectif

L'objectif de ce TP est de modéliser un **système de confiance** dans un réseau Blockchain en utilisant un langage de programmation orienté objet, comme Java ou Python. Vous allez implémenter des nœuds (participants) et calculer la confiance entre eux en se basant sur l'historique des transactions.

Contexte

Chaque nœud dans la blockchain représente un participant du réseau. Ce participant dispose d'un historique d'interactions (transactions) avec d'autres participants. La **confiance** entre deux participants est calculée comme la proportion d'interactions réussies entre eux. Votre tâche est de modéliser cette relation de confiance et de coder un programme qui simule ces interactions et calcule la confiance.

Énoncé

1. **Classe `Participant` :**
   - Chaque participant doit avoir un identifiant unique.
   - Chaque participant a une liste d'interactions (transactions) avec d'autres participants.
   - Implémentez une méthode pour ajouter des transactions et une autre pour calculer la confiance basée sur l'historique.
2. **Classe `Transaction` :**
   - Chaque transaction doit avoir un `expéditeur` (`sender`) et un `destinataire` (`receiver`).
   - Chaque transaction doit être marquée comme `réussie` ou `échouée`.
3. **Classe `Blockchain` :**
   - La blockchain contient une liste de participants.
   - Elle contient également toutes les transactions du réseau.
   - Implémentez une méthode pour simuler une transaction et mettre à jour l'historique des participants.
4. **Calcul de la Confiance :**
   - La confiance entre deux participants i et j est définie par le nombre de transactions réussies entre eux divisé par le nombre total de transactions entre eux.

Consignes

- Implémentez les classes et méthodes nécessaires pour simuler des transactions entre les participants.
- Calculez la confiance entre deux participants et affichez-la à l'écran.
- Utilisez le langage de programmation de votre choix : `Java` ou `Python`.

Instructions supplémentaires

- Pensez à inclure des transactions réussies et échouées dans vos simulations.
- Testez différentes configurations d'interactions entre les participants pour analyser l'évolution de la confiance.
- Veillez à commenter votre code et à respecter les bonnes pratiques de programmation.

Exemple de Cas d'Utilisation

- Ajoutez deux participants, `Alice` et `Bob`.
- Simulez plusieurs transactions entre eux : certaines réussies et d'autres échouées.
- Calculez et affichez la confiance d'`Alice` envers `Bob` à la fin de la simulation.

Exemple de Calcul

Dans cet exemple, nous allons calculer la confiance entre deux participants, Alice et Bob, après une série de transactions. Supposons que nous ayons les informations suivantes concernant les transactions :

- Alice a effectué 5 transactions avec Bob.
- 3 de ces transactions ont réussi, et 2 ont échoué.

La confiance est calculée à l'aide de la formule suivante :

$$P(\text{Alice} \to \text{Bob}) = \frac{T_{\text{succ}}(\text{Alice}, \text{Bob})}{T_{\text{total}}(\text{Alice}, \text{Bob})}$$

où T_{succ} est le nombre de transactions réussies et T_{total} est le nombre total de transactions.

Résultat attendu :
Dans cet exemple, la confiance de Alice envers Bob sera :

$$P(\text{Alice} \to \text{Bob}) = \frac{3}{5} = 0.6$$

Ce qui signifie qu'Alice a un niveau de confiance de 60% envers Bob.

Code avec Structure Conditionnelle (SI...ALORS)

Maintenant, utilisons une structure conditionnelle simple pour décider si la confiance entre Alice et Bob est suffisante pour effectuer une autre transaction.

Listing 1: Exemple en pseudocode

```
# Nombre de transactions
transactions_totales = 5
transactions_reussies = 3

# Calcul de la confiance
confiance = transactions_reussies / transactions_totales

# D cision bas e sur la confiance
si confiance > 0.5 alors
    afficher("Confiance suffisante, nouvelle transaction possible.")
sinon
    afficher("Confiance insuffisante, transaction refus e.")
fin si
```

Explication du Code :

- Nous commençons par définir le nombre total de transactions (`transactions_totales`) et le nombre de transactions réussies (`transactions_reussies`).
- Nous calculons ensuite la confiance avec la formule `confiance = transactions_reussies / transactions_to`
- Ensuite, une condition est vérifiée : si la confiance est supérieure à 0.5 (c'est-à-dire 50%), alors nous autorisons une nouvelle transaction, sinon la transaction est refusée.

Annexe : Formalisation Mathématique de la Confiance dans les Systèmes Blockchain

La formalisation mathématique de la confiance dans les systèmes blockchain peut être abordée sous plusieurs perspectives, telles que la théorie des jeux, la cryptographie et la théorie des probabilités. La confiance dans une blockchain est souvent liée à la manière dont les participants interagissent, vérifient et contribuent au réseau de manière décentralisée. Voici une approche pour formaliser la confiance mathématiquement :

1. **Modélisation de la Confiance par Probabilités :**
   - Soit $P(i \to j)$ la probabilité que le participant i fasse confiance au participant j.
   - Cette probabilité peut être calculée comme le rapport entre le nombre de transactions réussies $T_{succ}(i,j)$ et le nombre total de transactions $T_{total}(i,j)$ entre i et j :

$$P(i \to j) = \frac{T_{succ}(i,j)}{T_{total}(i,j)}$$

2. **Modèle de Confiance Basé sur les Récompenses et Punitions :**
   - Dans un réseau blockchain, la confiance peut être affectée par des récompenses (transactions réussies) et des punitions (transactions échouées).
   - On peut définir un score de confiance $C(i \to j)$ pour chaque participant basé sur un modèle de récompenses $R(i,j)$ et de punitions $P(i,j)$:

$$C(i \to j) = \frac{R(i,j) - P(i,j)}{R(i,j) + P(i,j)}$$

   - Un score positif indique une bonne confiance, tandis qu'un score négatif suggère un manque de confiance.

3. **Théorie des Jeux et Stratégies de Confiance :**
   - La théorie des jeux permet de modéliser les interactions stratégiques entre participants. Chaque participant peut choisir une stratégie basée sur la confiance.
   - Par exemple, dans un jeu de la confiance, les participants i et j choisissent d'effectuer ou non une transaction en fonction du gain perçu $G(i,j)$

Approfondissement sur la Confiance

1. Importance de la Confiance dans les Systèmes Blockchain

La confiance est essentielle dans les systèmes blockchain, car ces systèmes reposent sur des participants décentralisés qui n'ont pas nécessairement de relations préalables. La confiance entre participants influe directement sur la validité et la sécurité des transactions. En mesurant la confiance, on peut mieux évaluer les interactions et minimiser les comportements malveillants.

2. Modélisation de la Confiance

2.1. Fonction de Confiance

La fonction de confiance peut être modélisée à l'aide de divers facteurs :

- **Historique des Transactions** : Les participants ayant un historique de transactions réussies avec un autre auront une confiance plus élevée.

- **Réputation** : Les participants peuvent avoir des scores de réputation qui influencent la confiance.
- **Récompenses et Punitions** : Les systèmes de récompenses pour les transactions réussies et les punitions pour les échecs peuvent encourager un comportement de confiance.

2.2. Exemple de Fonction de Confiance

Nous pourrions formaliser une fonction de confiance comme suit :

$$C(i \to j) = \alpha \cdot \frac{T_{succ}(i,j)}{T_{total}(i,j)} + \beta \cdot R(i) + \gamma \cdot P(j)$$

où α, β, γ sont des poids représentant l'importance relative de chaque facteur, et $R(i)$ et $P(j)$ représentent respectivement la réputation de i et les punitions reçues par j.

Exemples de Code pour Calculer la Confiance

Voici un exemple de code Python pour calculer la confiance entre deux participants, en tenant compte des facteurs de succès et d'échec des transactions :

Listing 2: Calcul de la Confiance

```
class Participant:
    def __init__(self, name):
        self.name = name
        self.transactions = []

    def add_transaction(self, recipient, successful):
        self.transactions.append((recipient, successful))

    def calculate_trust(self, recipient):
        successful_transactions = sum(1 for r, success in self.transactions i
        total_transactions = sum(1 for r, _ in self.transactions if r == reci
        if total_transactions == 0:
            return 0
        return successful_transactions / total_transactions

# Exemple d'utilisation
alice = Participant("Alice")
bob = Participant("Bob")

# Ajout de transactions
alice.add_transaction("Bob", True)
alice.add_transaction("Bob", True)
alice.add_transaction("Bob", False)
alice.add_transaction("Bob", True)

# Calcul de la confiance d'Alice envers Bob
trust_alice_bob = alice.calculate_trust("Bob")
print(f"Confiance d'Alice envers Bob: {trust_alice_bob:.2f}")
```

3. Scénarios d'Interaction

3.1. Scénario 1 : Transactions Réussies

Dans ce scénario, Alice effectue plusieurs transactions avec Bob, toutes réussies. La confiance d'Alice envers Bob va donc augmenter, et elle sera plus encline à effectuer des transactions futures avec lui.

3.2. Scénario 2 : Transactions Échouées

Si les transactions échouent, la confiance d'Alice envers Bob diminuerait. Cela pourrait entraîner une réticence à faire des affaires avec lui à l'avenir, ce qui affecterait potentiellement le réseau dans son ensemble.

4. Impact de la Confiance sur le Réseau Blockchain

Un réseau avec des participants de confiance élevés favorise la fluidité des transactions, réduit le besoin de vérifications supplémentaires et améliore l'efficacité générale du système. À l'inverse, une faible confiance peut entraîner des retards, des vérifications excessives et des pertes de ressources.

Conclusion Finale

Ce TP a permis d'explorer la modélisation de la confiance dans un réseau blockchain. À travers la création de classes, l'implémentation de la logique de confiance et l'analyse des résultats, vous avez pu constater comment la confiance joue un rôle crucial dans la dynamique des interactions des participants. Cette compréhension est essentielle pour le développement de systèmes blockchain robustes et fiables.