TP 5: Consensus et Tolérance aux Pannes en Blockchain

November 10, 2024

Partie 1: Simulation de Consensus dans une Délibération Annuelle Electronique à l'Université

Objectif

L'objectif de ce TP est de simuler un processus de consensus dans une réunion entre enseignants d'une université. Nous utiliserons un modèle basé sur le vote pour représenter les différents niveaux de consensus (majorité, supermajorité, unanimité).

Exercice

Écrire un programme en Python pour simuler un vote dans une réunion universitaire et déterminer le consensus atteint. Le programme devra inclure les étapes suivantes :

- Génération aléatoire de votes pour 10 enseignants.
- Calcul des pourcentages de votes "Pour" et "Contre".
- Vérification du consensus atteint (majorité, supermajorité, unanimité).
- Affichage des résultats sous forme de graphique circulaire.

Code Python et Explications

1. Importation des bibliothèques

La première étape consiste à importer les bibliothèques nécessaires pour générer des votes et afficher les résultats sous forme de graphique.

```
1 import random
2 import matplotlib.pyplot as plt
```

Listing 1: Importation des bibliothèques

2. Définition des paramètres du vote

Nous définissons le nombre de votants et les seuils de consensus (majorité, supermajorité, unanimité).

```
num_voters = 10
proposal = "Acceptation de l'appel de note de l' tudiant "
majority_threshold = 0.51
supermajority_threshold = 0.66
unanimous_threshold = 1.0
```

Listing 2: Définition des paramètres du vote

3. Simulation des votes

Nous générons des votes aléatoires pour chaque enseignant, où 1 représente un vote "Pour" et 0 un vote "Contre".

```
votes = [random.choice([0, 1]) for _ in range(num_voters)]
yes_votes = sum(votes)
no_votes = num_voters - yes_votes
```

Listing 3: Simulation des votes

4. Calcul des pourcentages de votes

Nous calculons le pourcentage des votes "Pour" et "Contre".

```
yes_percentage = yes_votes / num_voters
no_percentage = no_votes / num_voters
```

Listing 4: Calcul des pourcentages de votes

5. Vérification du consensus

Nous vérifions le niveau de consensus atteint (majorité, supermajorité, unanimité).

```
def check_consensus(yes_percentage):
    if yes_percentage >= unanimous_threshold:
        return "Approbation unanime"
    elif yes_percentage >= supermajority_threshold:
        return "Approbation par supermajorit "
    elif yes_percentage >= majority_threshold:
        return "Approbation par majorit "
    else:
        return "Aucun consensus atteint"

result = check_consensus(yes_percentage)
```

Listing 5: Vérification du consensus

6. Affichage des résultats

Nous affichons les résultats du vote sous forme de graphique circulaire.

```
labels = ['Pour', 'Contre']
sizes = [yes_percentage * 100, no_percentage * 100]
colors = ['lightgreen', 'lightcoral']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%',
    startangle=140)
plt.title(f"R sultats du vote : {proposal}")
plt.axis('equal')
plt.show()
```

Listing 6: Affichage des résultats

Conclusion

En utilisant ce programme, nous avons pu simuler un processus de consensus basé sur le vote dans une réunion d'enseignants. Le graphique circulaire permet de visualiser facilement les résultats du vote, et le calcul du consensus nous aide à déterminer le niveau de soutien pour la proposition soumise au vote.

Partie 2: Introduction à la Théorie des Jeux, Consensus et Fault Tolerant Broadcast en Blockchain

Objectif du TP

Ce TP introduit les bases de la théorie des jeux et du consensus dans le contexte de la blockchain, ainsi que la notion de "Fault Tolerant Broadcast" (diffusion tolérante aux pannes). En particulier, les étudiants exploreront un modèle simple de consensus et comprendront comment la tolérance aux pannes permet de maintenir l'intégrité du réseau même en présence de nœuds défaillants ou malveillants.

Contexte

La blockchain est un réseau distribué où chaque participant doit maintenir un consensus sur l'état du système. Pour assurer la fiabilité du consensus même lorsque certains nœuds échouent ou agissent de manière malveillante, des protocoles de diffusion tolérante aux pannes sont utilisés. Dans ce TP, les étudiants appliqueront la théorie des jeux pour comprendre les motivations économiques des nœuds, ainsi que le rôle du "Fault Tolerant Broadcast" dans la sécurité des blockchains.

Fault Tolerant Broadcast (Diffusion Tolérante aux Pannes)

La diffusion tolérante aux pannes est un mécanisme par lequel un message peut être transmis à tous les nœuds d'un réseau, même si certains d'entre eux échouent ou se comportent de manière malveillante. Dans un contexte de blockchain, ce mécanisme garantit que les informations sont correctement partagées parmi les participants, ce qui aide à prévenir les attaques de désinformation (par exemple, la propagation de transactions non valides).

Principe du Jeu de Consensus et Fault Tolerant Broadcast

Chaque participant peut choisir entre deux actions :

- **Diffuser honnêtement** : Transmettre les transactions de manière correcte et fiable à travers le réseau.
- **Tromper** : Propager des informations erronées ou tenter de falsifier des transactions.

Chaque choix entraîne une **récompense** ou une **pénalité** basée sur le comportement des autres participants. Le "Fault Tolerant Broadcast" agit comme une couche de protection : il réduit l'impact des nœuds défaillants et protège le réseau en assurant une diffusion correcte malgré les fautes.

Matrice des Gains avec Diffusion Tolérante aux Pannes

	Honnête (B)	Trompe (B)
Honnête (A)	$(2, 2)$	$(-1, 3)$
Trompe (A)	$(3, -1)$	$(0.5, 0.5)$

Les valeurs de gain pour le cas où les deux participants trompent ont été ajustées pour refléter la résilience du réseau grâce au "Fault Tolerant Broadcast".

Code Python pour Simuler le Jeu de Consensus avec Fault Tolerant Broadcast

```
1
2  # D finir les gains pour chaque sc nario
3
4  avec Fault Tolerant Broadcast
5  rewards = {
6      ("Honn te", "Honn te"): (2, 2),
7      ("Honn te", "Trompe"): (-1, 3),
8      ("Trompe", "Honn te"): (3, -1),
9      ("Trompe", "Trompe"): (0.5, 0.5)
10
11     # gains r duits gr ce    Fault Tolerant Broadcast
12 }
13
14 def simulate_game(choice_A, choice_B):
15     return rewards[(choice_A, choice_B)]
16
17 # Tester diff rentes strat gies
18 strategies = [("Honn te", "Honn te"), ("Honn te", "Trompe"),
19               ("Trompe", "Honn te"), ("Trompe", "Trompe")]
20
21 for strategy in strategies:
22     gain_A, gain_B = simulate_game(strategy[0], strategy[1])
23     print(f"Strat gie A: {strategy[0]},
24     Strat gie B: {strategy[1]} -> Gains: A = {gain_A},
25
26     B = {gain_B}")
```

Listing 7: Simulation du jeu de consensus avec Fault Tolerant Broadcast

Questions

1. Quelle est l'importance du "Fault Tolerant Broadcast" dans un réseau blockchain ?
2. Comment le Fault Tolerant Broadcast influence-t-il le choix des stratégies par les participants ?
3. Que se passe-t-il si la majorité des participants choisissent de tromper malgré la présence de diffusion tolérante aux pannes ?
4. Comparez les résultats obtenus avec et sans diffusion tolérante aux pannes.

Objectifs pédagogiques

À la fin de ce TP, les étudiants devront être capables de :

- Comprendre le concept de consensus distribué avec tolérance aux pannes dans les blockchains.
- Appliquer les bases de la théorie des jeux pour analyser les comportements dans un réseau décentralisé.
- Expliquer comment le "Fault Tolerant Broadcast" augmente la résilience du réseau contre les comportements malveillants.

Conclusion

Ce TP fournit une première approche des concepts de consensus, de tolérance aux pannes, et de théorie des jeux appliqués à la blockchain. En analysant les impacts du "Fault Tolerant Broadcast" sur les gains des participants, les étudiants peuvent observer comment les blockchains résistent aux tentatives de falsification, renforçant ainsi la sécurité du réseau.