Projet ISOC631

Sujet 1- serveur de jeu – jeu d’Axelrod

Le but de ce projet est de mettre en place un contrat intelligent permettant de jouer au jeu du dilemme du prisonnier ou jeu d’Axelrod (voir le chapitre 1 du livre « THE EVOLUTION OF COOPERATION ». Le projet aura deux phases :

Phase 1-Mise en place du contrat intelligent

Ce contrat intelligent met en œuvre un jeu sécurisé d’Axelrod dans Solidity, prêt à être déployé sur la blockchain Ethereum. Le jeu suit les étapes suivantes :

1-Deux joueurs s'inscrivent et placent un pari.

2- Chaque participant choisit un coup et un mot de passe. Ils envoient le hachage de la chaîne concaténée au contrat qui la stocke.

3-Lorsque les deux joueurs ont joué leur coup, ils révèlent ce qu'ils ont joué. Pour ce faire, ils envoient leur coup et leur mot de passe en clair. Le contrat vérifie que le hachage de l'entrée reçue correspond à celui stocké.

4-Lorsque les deux joueurs ont révélé leur coup, le contrat détermine le gagnant et lui envoie le montant total de la mise.

5-Le jeu est réinitialisé et peut être rejoué.

*Mise en œuvre*

1-Phase d'enregistrement

Toute personne peut s'inscrire à condition qu'elle ne soit pas déjà inscrite et que sa mise soit supérieure à un minimum fixé, actuellement 1 finney.

Si un joueur a déjà été enregistré, un deuxième joueur souhaitant s'enregistrer doit placer une mise supérieure ou égale à la mise du joueur précédent. Cela permet d'éviter la stratégie consistant à toujours miser un montant inférieur à celui de l'adversaire afin de minimiser les risques tout en maximisant les gains. Il n'y a bien sûr aucun avantage à miser un montant strictement supérieur à la mise initiale, mais chacun doit être libre de gaspiller ses pièces comme il l'entend.

2-Phase d'engagement

Lorsqu'un joueur a été enregistré avec succès, il peut jouer. Comme décrit précédemment, le joueur fournit un hachage SHA256 de la concaténation d'un coup, représenté par un entier, et d'un mot de passe secret. Le contrat stocke ce hachage et personne d'autre que le joueur n'a accès au mouvement réel. Une fois qu'un tel hachage a été enregistré, il ne peut plus être modifié.

3-Phase de révélation

La phase de révélation commence lorsque les deux joueurs ont joué leurs coups. Un joueur révèle ce qu'il a joué en envoyant son mot de passe. Le contrat vérifie alors si c'est bien le coup qui a été joué en le hachant et en comparant le résultat au hachage stocké. S'ils sont égaux, le premier caractère de la chaîne (move, un entier) est sauvegardé et le contrat attend que le deuxième joueur révèle son coup.

La phase de révélation peut se terminer de deux façons : soit les deux joueurs ont révélé leur coup, soit le deuxième joueur ne soumet pas son coup avant la fin de la phase. Actuellement, la phase de révélation commence dès que l'un des joueurs révèle son coup et dure 10 minutes. Cela permet d'éviter une impasse dans laquelle le deuxième joueur se rend compte qu'il a perdu (puisqu'il peut maintenant voir le coup de son adversaire) et ne se donne pas la peine de terminer la partie.

4-Phase de résultat

À la fin de la phase de révélation, n'importe quel joueur peut déclencher la fonction getOutcome() pour que le contrat envoie les récompenses. Le gagnant, s'il y en a un, emporte tout. En cas d'égalité, les joueurs récupèrent leur mise. Un joueur qui n'a pas révélé son coup auparavant a automatiquement perdu.

Juste avant d'envoyer les pièces, le contrat réinitialise l'état du jeu. Cette opération est effectuée dans cet ordre afin d'éviter une éventuelle attaque par réentrance.

À tout moment, les joueurs ont accès aux variables d'état publiques et aux fonctions d'aide pour obtenir des informations sur l'état du jeu.

Les fonctions disponibles sont les suivantes

getContractBalance() : pour connaître la valeur actuelle de la cagnotte.

whoAmI() : pour voir l'ID du joueur : 1 ou 2 s'il est effectivement enregistré ou 0 s'il ne l'est pas.

bothPlayed() : retourne vrai si les deux joueurs ont joué et qu'ils peuvent maintenant passer à la phase de révélation.

bothRevealed() : retourne vrai si les deux joueurs ont révélé leur coup et que le contrat est prêt à envoyer les récompenses.

revealTimeLeft() : renvoie REVEAL\_TIMEOUT si le timer de la phase de révélation n'a pas encore démarré, renvoie le temps restant sinon

Les variables d'état publiques sont les suivantes

BET\_MIN : le montant minimal à envoyer au contrat pour s'enregistrer. Actuellement fixé à 1 finney.

REVEAL\_TIMEOUT : la durée de la phase de révélation en secondes. Actuellement fixée à 10 minutes (600 secondes).

initial\_bet : la mise du premier joueur enregistré s'il y en a un. Le deuxième joueur doit placer une mise supérieure ou égale à ce montant.

**L’interaction avec le jeu se fera par le bais d’une serveur web implantant une API REST. Les groupes faisant ce projet se mettrons d’accord sur les URLs implantant les fonctions disponibles et les variables d’état. Ceci permettra de faire la phase 2.**

Phase 2- Compétition d’Axelrod

Le contrat intelligent permet de jouer une fois le jeu d’Axelrod. Dans la suite on écrira un code interagissant avec l’API REST et jouant 200 fois le jeu d’Axelrod. Votre code implantera une stratégie de jeu (voir chapitre 2 du livre d’Axelrod).

Vous ferez des « battles » entre groupes afin de voir quelle stratégie gagne.

Projet 2- Étude de la structure de l’Internet mondial

Ce projet vise à développer un outil dont l’interface est donnée ci-dessous :

![Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement]()

Afin de faire cette interface, il faudra comprendre l’architecture d’Internet. Pour ceci lire le tutoriel

<https://www.kentik.com/kentipedia/bgp-routing/>

et

<https://au.int/sites/default/files/documents/31363-doc-session_4-1-_bgp_intro_-_fr.pdf>

et

https://fr.wikipedia.org/wiki/Internet\_Exchange\_Point

Vous avez sur ce site les tables de routage de nombreux IXP :

<https://www.pch.net/resources/Routing_Data/>

A faire :

1-Construire une fonction qui prend en entrée une longitude et latitude et retourne en sortie une liste des IXP ordonné par distance. Faire une seconde fonction qui utilise la première pour générer la liste des IXP ordonné par distance, mais à proximité d’une ville. Faite une troisième fonction qui prend un pays ou une liste de pays et sort la liste des IXP ordonné par distance qui sont situé dans ce pays. Finalement faire une dernière fonction qui prend une distance et ne sort que les IXPs qui sont à une distance inférieure à cette distance. Dans un premier temps vous trouverez la liste des IXPs et leur localisation sur le dump de peeringdb (voir <https://kdrive.infomaniak.com/app/share/808619/30edf4dd-736c-406d-af58-14d143bac581>)

2- Faire une carte qui représente les IXPs issue de la liste en coloriant les IXP en fonction de leur distance (vert les plus proches, rouges les plus éloignées)

3- les fichiers contenu dans l’URL <https://www.pch.net/resources/Routing_Data/> sont des tables de routage. Par exemple la ligne

\*> 1.0.138.0/24 194.42.48.80 0 0 6939 38040 23969 i

Signifie que le prefixe 1.0.138.0/24 est accessible de l’IXP de collecte (qui est determiné par le nom du fichier) en passant par le chemin d’AS 6939,38040,23969.

Nous allons travailler sur ces fichiers. Un prefixe 1.0.138.0/24 indique 2^(32-24)=256 adresses IP. En additionnant toutes les tailles d’adresse IP définies par toutes les lignes du fichier (en s’assurant qu’aucun préfixe n’est calculé deux fois) on obtient le nombre d’adresse IP accessible par cet IXP. Le nombre total d’adresse IP est 2^32 dont on peut calculer le pourcentage d’adresse IP accessible qui est nécessaire pour l’interface définie plus haut. Calculer cette valeur pour les IXPs choisi précédemment.

4-Le chemin d’AS 6939 38040 23969 signifie qu’il existe un chemin physique (un cable) entre les AS 6939 et 38040, ainsi qu’entre 38040 et 23969. On peut donc construire un graphe dont les nœuds sont les AS et les liens sont induit par les chemins d’AS. En utilisant la librairie python networkx et la table de routage construisez ce graphe et représenter de façon graphique dans la page web (on utilisera pour cela la librairie python [**forceatlas2**](https://github.com/bhargavchippada/forceatlas2).

3- Vous ajouterez des une interface REST afin de pouvoir interroger votre système automatiquement par le web.