Déploiement local de la plateforme d'échange décentralisée UniswapV2

Mathias Ramparison
Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, UMR 5104, 38000
Grenoble, France

***Abstract*—Nous présentons ici un travail pratique sur UniswapV2, un échange décentralisé (DEX) sur la blockchain Ethereum. La plate forme de travaux pratiques est une machine virtuelle tout-en-un, proposant le noeud blockchain Ethereum en local fonctionnant avec Hardhat. Le travail pratique consiste en la création et le déploiement de contrats intelligents (smart contracts) en Solidity, notamment pour des tokens ERC20 et des paires de tokens (liquidity pools). Le but est de comprendre les mécanismes de addLiquidity, removeLiquidity, et swap, tout en proposant des exercices pratiques utilisant Hardhat et MetaMask pour simuler des transactions et analyser le comportement d'un pool de liquidité. Enfin, il inclut une simulation de transactions réelles d'un pool USDC/WETH pour étudier l'évolution du taux de change. L'aspect sécurité et transparence des transactions est abordé à divers endroits dans le cours associé, et utilisé comme ligne de conduite tout au long des séances pratiques.**

I. Contexte

Dans le cadre d'un cours de la filière informatique pour la finance de Grenoble INP – Ensimag, nous avons réalisé une plate-forme fonctionnant sur une machine virtuelle permettant de faire fonctionner de façon locale sur sa machine l'échange décentralisé UniswapV2.

Le contenu de ce module[1], qui se déroule entre 6 et 9 heures selon le public, comprend :

- un cours d'introduction à la blockchain comprenant :
  - une introduction générale aux concepts liés à la blockchain
  - une présentation détaillée de l'échange décentralisé UniswapV2
- un travail pratique permettant de :
  - se familiariser avec le développement en Solidity sur la blockchain Ethereum
  - manipuler les tokens et comprendre le standard ERC20
  - comprendre les mécanisme de sûreté et de sécurité pour les mouvements de tokens
  - comprendre et manipuler des liquidity pools et comment y sont automatiquement calculés les taux de change

Le cours et le travail pratique mettent l'accent sur la sécurité et la sûreté à travers différents aspects :

- la blockchain en elle même
- la transparence des transactions
- la sûreté du code déployé, immuable par la suite
- la manipulation de transactions

Nous présentons en Section II d'abord le cours, puis le travail pratique en Section III, pour enfin conclure en Section IV et discuter des différents aspects pédagogiques et objectifs futurs en Section V.

II. Cours d'introduction à la Blockchain

Dans un premier temps, un cours présente une introduction à la blockchain inspiré de [2], ses paradigmes, concepts, usages, inconvénients, ainsi que les applications et la finance décentralisées. Il met en lumière les problèmes liés aux transactions centralisées, notamment en matière de gestion d'identité, de censure, de vulnérabilité et de coûts. Le cours explique ensuite les promesses de la décentralisation, avec un réseau pair-à-pair (P2P) à topologie aléatoire, tolérant aux pannes, et utilisant un registre partagé (shared ledger). En comparaison, on y explique que la blockchain s'appuie sur un consensus distribué pour la validation des transactions. Ce processus collectif assure la résistance à la fraude et au vol. La sécurité est assurée par la signature digitale, la validation distribuée, la réplication des données, et les blocs liés (historique). Nous décrivons également la notion de "coin" (jeton) comme étant immuable, créé par signature digitale et non modifiable. Les transactions consistent à consommer des coins et à en créer de nouveaux, avec des exigences de validité de la signature et de somme égale entre les coins consommés et créés. Nous passons rapidement sur la signature digitale, et indiquons que le protocole Bitcoin utilise l'algorithme de signature numérique à courbe elliptique (ECDSA). Par la suite, nous expliquons que les transactions sont regroupées en blocs liés par des pointeurs de hachage, validées par les mineurs via une preuve de travail dont la difficulté est ajustée périodiquement, assurant l'intégrité et l'immuabilité de la chaîne.

Nous abordons également dans le cours le concept de l'attaque des 51% où un acteur malveillant pourrait créer une chaîne privée et la diffuser pour invalider la chaîne publique. Pour prévenir ce type d'attaque, les validateurs de noeuds sont incités à être honnêtes grâce à des récompenses et des frais de transaction. Enfin les usages de la blockchain sont présentés, incluant le minage, le financement et le trading, l'identité décentralisée, et l'informatique décentralisée.

[1] disponible ici https://gricad-gitlab.univ-grenoble-alpes.fr/MRprojects/local-deploy-uniswapv2/

Nous abordons ensuite les applications décentralisées (dApps) visent à créer des applications autonomes et immuables avec des smart contracts. Le langage Solidity est utilisé pour développer ces smart contracts pour la machine virtuelle Ethereum (EVM). Finalement nous décrivons quelques inconvénients de la blockchain, tels que la scalabilité, la résistance à la centralisation (en raison de la consolidation du minage), la transparence (toutes les transactions sont publiques), la gouvernance (manque de structure claire pour la prise de décision), et la consommation d'énergie.

Dans une deuxième partie, nous abordons la finance décentralisée (DeFi), sujet principal des travaux pratiques qui suivent.

Grâce à un comparatif avec la finance traditionnelle, nous décrivons les notions de *order book*, market maker, et les AMM (Automated Market Makers) [1], des algorithmes qui agissent comme des market makers dans les *liquidity pools*. On y retrouve deux types d'acteurs : les fournisseurs de liquidité (liquidity providers) et les traders. Les fournisseurs de liquidité ajoutent des tokens et reçoivent des parts du liquidity pool, appelées *liquidity shares*. Les traders échangent un token contre un autre. Les AMM fonctionnent avec une fonction de conservation, par exemple où le produit des quantités de deux tokens A et B reste constant $(K = qA * qB)$. D'autres fonctions de conservation sont possibles [1]. Des considérations de sécurité et de confidentialité sont soulevées, avec des problèmes de manipulation des transactions comme le frontrunning et le sandwich attack. Des techniques de préservation de la vie privée comme les *zero-knowledge proofs* sont mentionnées (qui mériterait d'être développé dans la suite du cours). Nous concluons avec le fait que les fournisseurs de liquidités, les traders et le pool de liquidité forment un jeu à somme nulle. Enfin, en annexe, les notions de *divergence loss* (ou *impermanent loss*) et de *slippage* (ou *price impact*) sont introduites comme risques pour les acteurs des liquidity pools.

III. Travail pratique : manipulation d'UniswapV2 en local

Le travail pratique (TP) consiste en l'utilisation de l'échange décentralisé (DEX) UniswapV2, avec pour objectif de se familiariser avec ses mécanismes de base. Ce TP guide l'utilisateur à travers différentes étapes progressives, allant de la manipulation des éléments les plus simples (les tokens), à la création de paires de tokens, jusqu'aux échanges de tokens basés sur un taux de change calculé automatiquement.

Mise en place

Une machine virtuelle Linux Mint est fournie[2], avec UniswapV2 pré-déployé en utilisant l'environnement de développement Ethereum Hardhat. Les commandes pour lancer un noeud local et exécuter des scripts sont fournies. L'architecture du code d'UniswapV2 est expliquée, en particulier la structure des dossiers : contracts, artifacts, test et scripts. Les smart contracts sont écrits en Solidity et compilés pour être exécutés sur la machine virtuelle Ethereum (EVM).

[2]disponible dans le sujet de TP

Architecture du code d'UniswapV2

Architecture générale:

- **Le dossier `contracts`** contient les smart contracts en langage Solidity (extension .sol).
- **Le dossier `artifacts`** contient les smart contracts compilés (fichiers .json avec le byte-code pour l'EVM).
- Le dossier `contracts/test` contient les tests automatisés.
- Le dossier `scripts` contient les scripts de déploiement.

Dépendances entre les smart contracts:

- **`UniswapV2ERC20.sol`** : Contrat définissant un token ERC20. Fonctions : *transfer* (d'un compte vers un autre), *mint* (créer du token), *burn* (détruire du token).
- **`UniswapV2Pair.sol`** : Contrat définissant une paire de tokens (Liquidity Pool), dépendant de `UniswapV2ERC20.sol`. Fonctions : *mint* (de liquidity shares), *burn* (de liquidity shares), *swap* (échanger un token contre un autre).
- **`UniswapV2Factory.sol`** : Contrat permettant de créer des paires de tokens avec la fonction *createPair* et de définir les frais.
- **`UniswapV2Router.sol`** : Contrat de plus haut niveau, dépendant de tous les autres. Fonctions : *addLiquidity*, *removeLiquidity*, *swap* (et gestion de l'ETH).

Création et manipulation d'un token ERC20

Nous expliquons comment créer un smart contract de token ERC20 en suivant le standard ERC20, qui inclut des fonctions obligatoires comme totalSupply, balanceOf, transfer, transferFrom, approve, et allowance. La mise en place d'un contrat token est détaillée, avec un exemple de code Solidity (FunToken.sol). Le déploiement du contrat est guidé, en utilisant Hardhat et des scripts Javascript/Typescript, incluant l'utilisation de ethers.getSigners() pour récupérer les adresses des différents comptes (paires clé publique – clé privée) avec lesquels nous allons manipuler des tokens, et getContractFactory pour déployer les contrats. L'utilisation de Metamask est proposée pour interagir avec le token déployé sur la blockchain locale, ce qui donne un aspect interactif et visuel à la première manipulation de tokens. Des scripts pour tester le token sont introduits, incluant l'affichage des détails du token (nom, symbole, totalSupply, balance – la quantité de token) et les fonctions transfer, approve et allowance, en particulier pour transférer des tokens entre deux adresses (entre deux comptes).

L'accent est mis sur la nécessité d'utiliser `approve` et `allowance` afin d'autoriser un autre utilisateur à utiliser les tokens du créateur du token. En effet, par défaut seul le créateur peut manipuler un token. Les autres utilisateurs doivent l'autoriser à manipuler les tokens qu'ils possèdent sur leur compte.

Création d'une paire de tokens

La création d'une paire de tokens, qui correspond à la création d'un *liquidity pool*, est proposée à travers un smart

contract respectant également le standard ERC20, qui utilise les méthodes burn, mint et le paramètre totalSupply pour gérer les liquidity shares : en effet un liquidity pool doit émettre et recevoir des liquidity shares proportionnellement à la quantité de tokens qui y sont stockés. Le déploiement de la Factory Uniswap est expliqué, outil plus haut niveau nécessaire pour la création et la manipulation de paires de tokens de façon simplifiée. Un script est fourni pour créer une paire, en utilisant deux tokens ERC20 et en identifiant quel token correspond à quel token de la paire, avec affichage des balances (quantité de tokens que chaque compte possède).

Manipulation d'une paire de tokens

Les manipulations suivantes sont abordées:

- addLiquidity: décomposée en transfert des deux tokens au liquidity pool, et mint (création) de liquidity shares à envoyer à l'utilisateur en fonction des quantités déposées, avec une explication de la fonction mint dans UniswapV2Pair.sol et _mint dans UniswapV2ERC20.sol.
- removeLiquidity: décomposée en un transfert de liquidity shares au liquidity pool, burn (destruction) de ces liquidity shares, et transfert de la quantité de tokens correspondante au compte qui retire son placement.
- swap: un exemple de swap d'un montant fixe est donné, puis il est demandé de calculer le montant de token à recevoir en utilisant la fonction de conservation à produit constant ($K = qA * qB$). C'est justement la partie où on va pouvoir en pratique calculer le taux de change entre deux tokens : on a ici la possibilité de tester les formules de son choix.

Interactions haut niveau avec UniswapV2

Le concept du Router est introduit, un contrat de plus haut niveau qui simplifie l'ajout/retrait de liquidités et les échanges de tokens. Le déploiement du Router et des contrats nécessaires est expliqué, notamment WETH (Wrapped ETH, pour avoir un équivalent ERC20 de ETH). Un script est donné pour tester le Router, qui reproduit les opérations précédentes (déploiement de tokens, création de paires, ajout de liquidité, swap), mais en utilisant un autre compte pour les transferts et les swaps.

Une fois de plus, l'accent est mis sur l'importance des autorisations (approve et allowance) pour que le Router puisse agir au nom des utilisateurs est soulignée.

On illustre aussi la *profondeur de marché* en regardant l'évolution du prix en fonction des manipulations (ajout/retrait de liquidités et swaps).

Simulation de transactions dans un Liquidity Pool

Grâce à l'introduction du Routeur qui facilite grandement les interactions avec le DEX, le travail pratique se termine par de la simulation de transactions à partir d'un fichier CSV contenant des données réelles d'un pool USDC/WETH (une journée de transactions). Pour cela, nous fournissons un script permettant de parser le fichier, réaliser toutes les transactions automatiquement dans notre DEX UniswapV2 en local et d'exporter les données afin de dessiner un graphique montrant l'évolution du taux de change. Le script simule les transactions sur la base de 20 adresses en utilisant un fichier CSV. L'exécution du script est expliquée, avec la possibilité de limiter le nombre de transactions pour des tests. L'interprétation des résultats (graphique de l'évolution du taux de change et données exportées) est abordée, avec une comparaison possible avec les données réelles.

IV. Conclusion

En résumé, nous proposons dans un premier temps un cours d'introduction à la blockchain et aux échanges décentralisés (DEX), ainsi qu'un guide pratique pour comprendre et interagir avec UniswapV2. Il couvre la création de tokens, la création de paires, l'ajout et le retrait de liquidités, les swaps, et l'utilisation du router. Il offre une simulation de transactions pour mieux saisir l'évolution d'une pool. Le TP permet une approche pas à pas, de la base (smart contracts) vers les interactions de plus haut niveau (router) et se termine par une simulation avec des données réelles. Tout au long du cours et de ce travail pratique, l'accent est mis sur la sûreté et la sécurité aussi bien en parlant de la blockchain que des smart contracts déployés.

V. Aspects pédagogiques et objectifs futurs

Aspects pédagogiques: Les étudiants apprécient de poursuivre avec un projet plus axé sur une stratégie financière en blockchain (c'est un autre aspect du cours, non décrit ici). Ils apprécient particulièrement manipuler le taux de change à la main, car cela leur permet de mieux comprendre les dérives potentielles que cela peut engendrer dans des environnements financiers décentralisés. Il est important de noter que tout dérive des standards ERC20, même des concepts comme les liquidity pools, qui reposent sur cette base technologique. Au départ, ils aiment utiliser Metamask, car son interface visuelle rend les interactions plus intuitives, ce qui est idéal pour s'initier à l'écosystème blockchain. Cette approche leur permet d'explorer de manière concrète des mécanismes complexes, tout en combinant théorie et pratique pour approfondir leur compréhension des systèmes financiers décentralisés.

Objectifs: Nous avons pour objectif de partager cette plateforme avec un large éventail d'utilisateurs afin de recueillir leurs retours d'expérience et suggestions d'amélioration. Ces retours nous permettront de perfectionner les fonctionnalités existantes, en mettant un accent particulier sur les aspects liés à la sécurité et à la sûreté des transactions, qui sont au cœur de nos préoccupations. Par ailleurs, nous souhaitons explorer la possibilité d'enrichir la plateforme en y intégrant des outils avancés de surveillance des transactions, qui permettent de réaliser des transactions automatiques par la suite, par exemple des bots MEV (Maximal Extractable Value). Nous pourrions ainsi dans un cas pratique mettre à l'épreuve les méthodes actuelles de gestion et surtout les méthodes manipulation de transactions actuelles. Une autre piste d'amélioration est la portabilité de la plateforme de TP :

par exemple une VM moins lourde ou alors sa transformation en conteneur.

REFERENCES

[1] Jiahua Xu, Krzysztof Paruch, Simon Cousaert, Yebo Feng. (2023) "SoK: Decentralized Exchanges (DEX) with Automated Market Maker (AMM) Protocols". https://arxiv.org/pdf/2103.12732.pdf.

[2] Oleg Lodygensky. (2017) Blockchain tutorial. https://gitlab.in2p3.fr/lodygens/blockchain-tutorial.