SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Zadanie 1

My blockchain

Anna Yuová

Predmet: Digitálne meny a blockchain

Akademický rok: 2020/2021

Semester: letný

OBSAH

Vysvetlenie jednotlivých fáz.....3
Opis riešenia.....4
Blokový návrh.....5
Používateľská príručka.....6
Voľba implementačného prostredia.....6
Záver.....6

VYSVETLENIE JEDNOTLIVÝCH FÁZ

1. FÁZA

Cieľom prvej fázy je vytvoriť verejný ledger a spracovať všetky transakcie. V každom bloku dostaneme nejaké transakcie, ktoré treba skontrolovať a následne aktualizovať zoznam platných transakcií. Dôležité sú funkcie Input a Output v triede Transaction.java. Vstupom je to, keď dostaneme nejaký coin a výstupom je jeho odoslanie - každá transakcia má vstup a výstup. Pri výstupe z transakcie je dôležité to, že odoslaný výstup musel byť už niekedy predtým prijatý. S tým súvisí trieda UTXO, pretože tie transakcie, ktoré robíme v súčasnosti závisia od tých, ktoré nám prišli v minulosti. Ak napríklad dostaneme 3 transakcie – po 1, 2 a 3 coinoch tak suma by bola 6 coinov, lenže tieto vstupy sú nezávislé.
Napríklad:
Alica pošle Bobovi 1 coin.
Andrea pošle Bobovi 2 coiny.
Adam pošle Bobovi 0,5 coinu.
Bobov UTXOPool po prijatých transakciách: [1; 2; 0,5]
Ďalej je trieda Crypto, v ktorej je funkcia verifySignature – na overenie podpisov. Každý má dva kľúče, s ktorými pracujeme – súkromný a verejný. Súkromný kľúč by mal mať iba vlastník coinu a používa sa na podpis pri vytvorení transakcie. Ostatní si vedia overiť pomocou verejného kľúča, či je tá transakcia naozaj v poriadku.

TESTOVANIE

Otestovať som sa to pokúsila najprv na klasickom prípade tak, aby boli splnené všetky podmienky, sedeli podpisy a aktualizovala som UTXOPool – vrátilo mi true. V druhom prípade som pozmenila verejný kľúč tak, aby nesedel Bobov podpis a tým pádom overenie podpisu vrátilo false.

2. FÁZA

Cieľom druhej fázy je vytvoriť distribuovaný koncept na dosiahnutie konsenzu a odolaniu sybil útokom. V sieti môžu byť podvodné alebo dôveryhodné uzly a treba navrhnúť to, ako sa majú správať všetky dôveryhodné uzly. Ak by sa stalo, že niekto zoberie už podpísanú transakciu a nakopíruje ju tam viackrát, tak ak je podpísaný originál, je podpísaná aj kópia. Treba sa preto zbaviť podvodných uzlov tak, že každý bude mať jedinečné id vďaka čomu bude mať každá transakcia iný podpis. Každý uzol má svoj zoznam sledovaných osôb, zoznam transakcií, ktoré môže poslať ďalej. Uzol bude teda sledovaný, ak si vrátim true z aktuálneho uzla. Konsenzus musí teda zabezpečiť to, aby sa všetky uzly zhodli bez nejakej jednej hlavnej (centrálnej) autority. Tým, že rozpošleme ten zoznam všetkých transakcií každému uzlu, tak každý má prístup k nemu a každý si ho vie pozrieť – znak distribúcie.

3. FÁZA

Cieľom tretej fázy je udržiavať aktualizovaný blockchain. Prijíma a spracováva transakcie tak, ako prídu a udržiava ich v blokoch. Uzly musia byť súčasťou distribuovaného protokolu, ktorý dosahuje konsenzus a vytvára štruktúru dát blokov. Čiže keď sem príde nový blok alebo nová transakcia, treba ju zaradiť do blokov alebo udržiavať v blokoch a takisto udržiavať aj aktualizovaný zoznam všetkých blokov.

OPIS RIEŠENIA

1. FÁZA

Vlastním vlastne UTXOPool, ktorý je naplnený coinami z predošlých transakcií. Keď chcem poslať niekomu nejaké coiny, musím vytvoriť novú transakciu, vyplniť jej vstup aj výstup, podpísať ju a odoslať. Upravila som triedu HandleTxs.java, ktorá zaisťuje platnosť transakcií (overenie, vybavenie transakcie, aktualizovanie utxoPoolu).
Táto trieda má 3 funkcie:
Pomocou konštruktora som iniciovala triedu, čiže som vytvorila verejný ledger, ktorého aktuálny UTXOPool je utxoPool z argumentu. TxIsValid – táto funkcia zaisťuje platnosť transakcií - na to aby bola transakcia platná musíme overiť či, sú podpisy platné - metóda overeniePodpisov, do ktorej pošlem dáta, verejný kľúč a podpis a vráti mi true, ak prebehlo všetko správne. Ďalej overujem či nedochádza k double-spendingu a dané utxo nebolo nárokované viackrát. Všetky vstupné hodnoty musia byť väčšie ako 0 a súčet všetkých vstupov nemôže byť väčší ako výstupy, pretože sa nedá ísť do mínusu. TxHandler – táto funkcia spracuje transakcie – prejdem všetky prijaté transakcie a ak sú správne pridám ich do poolu, najprv musím odoslaný coin z poolu odstrániť a potom aktualizovať pool s novým zostatkom, ktorý sa mi vrátil (moje coiny dokopy - fee = odoslaná hodnota).
MaxFeeHandleTxs.java – funguje rovnako ako HandleTxs.java (skopírovala som predošlé funkcie z HandleTxs, ktoré overujú transakcie) a pridala som navyše porovnanie všetkých transakcií na základe poplatkov a vrátim transakcie s maximálnymi poplatkami za vykonané transakcie.

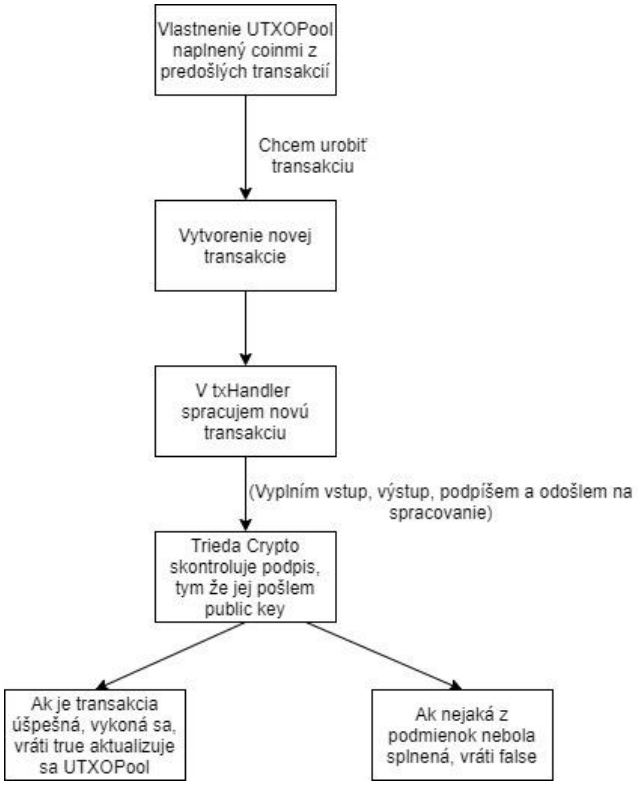
2. FÁZA

Najprv inicializujem triedu pomocou konštruktora (4 argumentový)– v Simulation potrebuje dostať 4 argumenty. Potom nastavím to, aby každý uzol dostal zoznam sledovaných osôb z booleovského poľa. Ak sa mi vráti true, tak uzol s folowees[i] je sledovaný, inak nie. Ďalej treba inicializovať skupinu transakcií, tým že ten zoznam s transakciami treba niekam zaradiť. V poslednej funkcii zistím čo prijali jednotlivé uzly (kandidáti). V simulation.java sa na konci vypíšu vygenerované id a podstatné je aby v každom uzle boli tie isté a neboli tam nakopírované niektoré pofidérne.

3. FÁZA

Pre správne spustenie poslednej fázy som si sem najskôr musela skopírovať niektoré triedy z prvej fázy. V UTXOPool mám už nejaké coiny z predošlých transakcií a UTXOPool sleduje moje UTXO. Ďalej som musela skopírovať aj triedu HandleTxs, ktorá spracováva transakcie a s ňou okrem utxo súvisí aj trieda Crypto, v ktorej sa kontrolujú podpisy. BlockHandler sa postará o to, že keď sem príde nová transakcia alebo blok, tak to vybaví a štruktúru blokom dodáva trieda Block. Keď príde nový blok, malo by sa to aktualizovať a výsledkom bude teda udržiavaný a aktualizovaný zoznam blokov. V triede Blockchain som musela najskôr vytvoriť prvý genesis blok. Vytvorila som si pomocnú triedu Blok, do ktorej pošlem základné parametre bloku, ktoré má mať a mám tam metódy set a get. Pridám transakciu do poolu musím získať utxo na to, aby som mohla môj blok vyťažiť a zaradiť do skupiny blokov. Blok pridám ak skontrolujem, že všetky transakcie v ňom sú platné.

BLOKOVÝ NÁVRH
1. FÁZA



2. FÁZA

3. FÁZA

POUŽÍVATEĽSKÁ PRÍRUČKA

Môj program som rozdelila do 3 balíkov (packge-ov) podľa jednodlivých fáz. V balíku faza1 sú všetky kódy potrebné k spusteniu prvej fázy a spúšťa sa pomocou triedy Main.java. V balíku faza2 sú všetky kódy potrebné k spusteniu druhej fázy a spúšťa sa pomocou triedy Simulation.java. Táto trieda vyžaduje aj nejaké argumenty, a preto treba zadať v Run->Run Configurations->Arguments vstupné argumenty (pravdepodobnosti p_graph, p_byzantine, p_txDistribution a počet kôl – napr. .1 .15 .05 10). V balíku faza3 sú všetky kódy potrebné k spusteniu poslednej fázy a spúšťa sa pomocou triedy Main.java, ktorú som skopírovala a upravila z prvej fázy.

VOĽBA IMPLEMENTAČNÉHO PROSTREDIA

Kód som implementovala v programe Eclipse (verzia 4.15.0) a tento program som si vybrala, pretože je to jednoduché prostredie na implementáciu v jazyku Java. S týmto programom mám skúseností z predošlých predmetov, kde sme sa učili Javu. Môj kód je spustiteľný.

ZÁVER

V zadaní 1 sme mali vytvoriť vlastný blockchain, ktorý bude vedieť robiť základné operácie s transakciami (spracuje, overí, aktualizuje UTXO), zaradí tieto transakcie do blokov a jednotlivé bloky do samotného blockchainu. Medzi každým uzlom by mal byť dosiahnutý konsenzus, aby sa dohodli aj bez centrálnej autority a tým pádom to bude distribuovaný systém, v ktorom vidí každý transakcie. Zadanie mi pomohlo hlbšie pochopiť veci z prednášok a vyskúšať si ich prakticky.