VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

TESTOVÁNÍ BEZPEČNOSTI A VÝKONU PROOF-OF-STAKE PROTOKOLŮ POMOCÍ SIMULACE

SECURITY AND PERFORMANCE TESTBED FOR SIMULATION OF PROOF-OF-STAKE PROTOCOLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JURAJ HOLUB**
AUTHOR

VEDOUCÍ PRÁCE **Ing. IVAN HOMOLIAK, Ph.D.**
SUPERVISOR

BRNO 2021

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Kľúčové slová

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Citácia

HOLUB, Juraj. *Testování bezpečnosti a výkonu Proof-of-Stake Protokolů pomocí simulace.* Brno, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ivan Homoliak, Ph.D.

Testování bezpečnosti a výkonu Proof-of-Stake Protokolů pomocí simulace

Prehlásenie

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Juraj Holub
29. decembra 2021

Poďakovanie

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.).

Obsah

Kapitola 1

Úvod

TODO

Kapitola 2

Blockchain

Táto kapitola vysvetľuje základné koncepty a pojmy spojené z technológiou blockchain, ako aj samotnú dátovú štruktúru blockchain. Sekcia 2.1 vysvetľuje pojmi distribuovaná účtovná kniha a blockchain. Ďalej rozoberá vlastnosti a využitie blockchainu. Sekcia 2.4 vysvetľuje kryptografiu používanú v blockchaine (hashovanie, a asymetrická kryptografia). Sekcia 2.3 popisuje peer-to-peer siete a ich využitie v blockchaine. Na záver sú v sekcii 2.5 spojené všetky vysvetlené koncepty dokopy a je popísaná samotná dátová štruktúra blockchain.

2.1 Distribuovaná účtovná kniha

Účtovná kniha (anglicky *ledger*) sa v histórii ľudstva dlhodobo používa na záznam rôznych položiek, najčastejšie peňazí a majetku. Príchod digitalizácie a globalizácie presunul tento známy koncept z papierovej podoby do elektronickej. Toto prináša nové výzvy z hľadiska bezpečnosti. *Distribuovaná účtovná kniha* (anglicky *distributed ledger*) je všeobecne technológia, ktorá poskytuje dôveryhodnú a bezpečnú databázu zdieľanú naprieč viacerými inštitúciami, krajinami a to typicky verejne. Veľmi bežným odvetvím využitia distribuovanej účtovnej knihy je bankovníctvo. Banka poskytuje centralizovanú autoritu, ktorá zabezpečuje bezpečnú manipuláciu s peniazmi klientov. Tento koncept označujeme ako centralizovaná účtovná kniha. [34]

V roku 2008 bola publikovaná práca [29], ktorá navrhla *decentralizovanú* distribuovanú účtovnú knihu. Práca navrhla koncept elektronického platobného systému, ktorého bezpečnosť je založená na kryptografickom dôkaze namiesto dôvere v centralizovanú autoritu. Takáto distribuovaná účtovná kniha sa nazýva **blockchain** (ďalej **BC**).

2.1.1 Vlastnosti BC

BC je dátová štruktúra, ktorá má nasledujúce vlastnosti [43]:

- **Decentralizácia** (anglicky *decentralization*): BC funguje nad peer-to-peer sieťou, ktorá nepotrebuje centralizovanú dôveryhodnú autoritu, ale na zabezpečenie konzistencie používa konsenzus protokol.
- **Auditovateľnosť** (anglicky *auditability*): BC v sebe nesie celú histórie zmien dátového obsahu a teda každú zmena stavu dát uložených v BC je možné sledovať.
- **Nemennosť** (anglicky *persistency*): Modifikácia alebo zmazanie časti dátového obsahu BC nie je možná.

- **Anonymita** (anglicky *anonymity*): Užívatelia pracujúci s BC používajú na identifikáciu asymetrickú kryptografiu s digitálnym podpisom. Takýto kryptografický identifikátor neodhaľuje skutočnú identitu užívateľa a pritom umožňuje nepopierateľne určiť vlastníka elektronického zdroja.

Tieto vlastnosti BC sú zabezpečené pomocou peer-to-peer siete na ktorej je BC postavený (viď sekcia 2.3) a taktiež pomocou samotnej dátovej štruktúry, ktorá využíva modernú kryptografiu (viď sekcia 2.5). [1]

2.1.2 Aplikačné využitie

BC bol navrhnutý a po prvýkrát implementovaný za účelom poskytnúť elektronickú peňažnú menu nezávislú od centralizovaného bankovníctva. Tento prvý a najznámejší BC je Bitcoin [29]. Avšak vlastnosti BC technológie nachádzajú uplatnenie vo veľkom množstve odvetví. Nasledujúci zoznam vymenúva niekoľko aplikácií, ktoré BC môže riešiť [22]:

- **Elektronická peňaženka**: Elektronické peňaženky pre obchod s nejakou formou digitálnych peňazí (typicky v podobe tokenov). Takéto tokeny sú vlastnené pomocou privátneho kľúča, ktorý má uschovaný majiteľ. Ten môže vlastníctvo tokenov presúvať na iné subjekty v danej sieti.

- **Zmenárne**: V dnešnej dobe existuje veľké množstvo elektronických peňažných mien postavených nad BC. Takéto meny všeobecne označujeme ako kryptomeny. Z dôvodu veľkého množstva kryptomien sa prirodzene zvyšuje dopyt po zmenárni medzi jednotlivými kryptomenami. Klasická zmenáreň je riešená tradične centralizovanou autoritou. Avšak BC je vhodnou technológie aj pre decentralizovanú zmenáreň.

- **Súborové systémy**: V dnešnej dobe už existujú decentralizované súborové systémy založené na peer-to-peer sieťach. Implementácia takéhoto decentralizovaného súborového systému ako BC by nám umožnila nepopierateľne a trasovateľne verzovať zmeny v obsahu.

- **Správa identít**: Táto služba je typicky zabezpečovaná centrálnou autoritou, ktorá prideľuje pre konkrétne entity určité zdroje na ktoré majú právo. Ide o schému podobnú banke. BC by v tomto prípade opäť umožnil náhradu tejto centralizovanej autority za decentralizovanú sieť.

- **Voľby**: Elektronické hlasovanie je ďalším vhodným príkladom, kde sa dá efektívne využiť BC. Hlasujúce entity predstavujú decentralizovanú sieť a vlastnosti BC zase poskytujú transparentnosť a verejnú overiteľnosť.

- **Reputačné systémy**: Tieto služby merajú úroveň dôvery v určité entity. Typickým príkladom je reputácia rôznych predajcov na základe hlasovania zákazníkov. Transparentnosť a nemennosť BC histórie by znížila možnosť manipulácie s reputáciu v prospech nejakej entity.

- **Aukcie**: Elektronická aukcia je služba veľmi podobná elektronickej peňaženke alebo zmenárni s podobnými bezpečnostnými požiadavkami. Tieto vlastnosti by opäť dokázala pokryť technológia BC.

2.2 Viacvrstvová abstrakcia BC

Existuje veľké množstvo BC protokolov, ktoré majú rôzne využitie a technologické riešenie. Avšak všetky tieto implementácie sú založené na spoločnom koncepte distribuovanej účtovnej knihy. Pre ich jednotnú klasifikáciu použijeme nasledujúci model abstrakcie so štyrmi vrstvami [22]:

1. **Sieťová vrstva** predstavuje najnižšiu vrstvu abstrakcie a zaoberá sa peer-to-peer sieťou. Sieť rieši pripájanie nových peerov a komunikácia medzi uzlami v sieti (šírenie transakcií a blokov). Táto vrstva má kritický dosah na výkonnosť BC. Napríklad, verejný BC ako je Bitcoin tvorí veľmi rozsiahlu sieť s tisíckami aktívnych uzlov. V takejto sieti sa už vlastnosti ako stratovosť dát alebo priepustnosť nezanedbateľne prejaví na rýchlosti a stabilite celého BC. Sieťová vrstva je popísaná v sekcii 2.3. [16]
2. **Konsenzus vrstva** definuje protokol pomocou, ktorého sa ustanovuje dohoda na stave BC. Konsenzus v BC je popísaný v kapitole 3.
3. **Dátová vrstva** (alebo tiež úložisko) definuje model transakcií (binárny hashovací strom, hashovacie a kryptografické algoritmy). Sekcia 2.4 popisuje kryptografické primitíva používané v technológii BC a sekcia 2.5 popisuje BC ako dátovú štruktúru.
4. **Aplikačná vrstva** definuje využitie v konkrétnej službe (napríklad kryptomena).

2.3 Peer-to-peer sieť

Technológia BC je postavená na peer-to-peer sieťach. Peer-to-peer sieť sa podieľa na decentralizovanosti, nemennosti a auditovateľnosti BC.

Peer-to-peer sieť je dynamický súbor nezávislých uzlov (anglicky *peers*), ktoré sú prepojené do grafu. Každý uzol obsahuje zdroje, ktoré zdieľa všetkým ostatným uzlom v sieti. [13, 35] Dôvod existencie peer-to-peer sietí je teda decentralizovaný spôsob zdieľania zdrojov ako sú súbory, fyzické zariadenia, výpočtový výkon alebo aj elektronické finančné zdroje. Dnes existuje množstvo peer-to-peer sietí. Veľmi známe sú napríklad Gnutella, Kazaa alebo BitTorrent. [2]

2.3.1 Referenčný model

Najbežnejšie technické riešenie peer-to-peer siete je navrstvenie siete (anglicky *overlay network*) na už existujúcu sieť, ktorou je typicky Internet. Takúto sieť potom môžeme definovať ako päticu (P, R, I, F_P, F_R), kde:

- P je množina uzlov
- R je množinu zdrojov
- I je priestor identifikátorov
- $F_P : P \to I$ je funkcia, ktorá mapuje uzoly na identifikátory
- $F_R : R \to I$ je funkcia, ktorá mapuje zdroje na identifikátory

Obrázok 2.1 ukazuje princíp fungovania takto definovanej siete. Tvorba siete s týmto modelom je potom závislá od šiestich návrhových aspektov:

1. Voľba priestoru identifikátorov.
2. Mapovanie zdrojov a uzlov na identifikátory.
3. Správa priestoru identifikátorov v réžii uzlov siete.
4. Tvorba grafu (štruktúra siete).
5. Stratégia smerovania (anglicky *routing*).
6. Stratégia údržby.

Konkrétne riešenie pre popisovaných šesť aspektov je zavislé od požiadaviek na efektivitu, škálovateľnosť, samoorganizovateľnosť, odolnosť voči chybám a kooperáciu. [2]

2.3.2 Využitie v BC

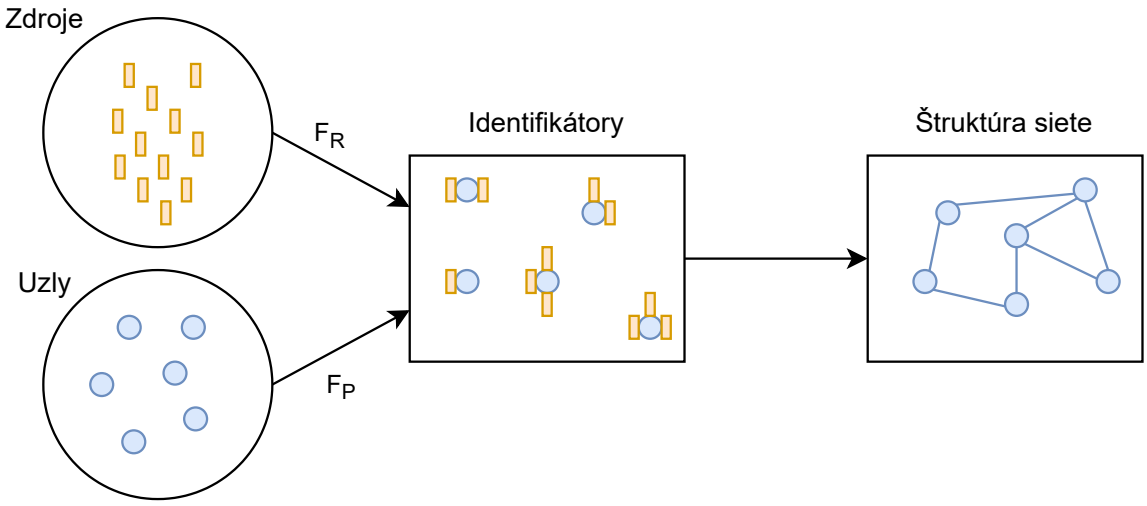
Peer-to-peer sieť umožňuje BC uchovávať jeho obsah decentralizovane a pritom bezpečne. Tento koncept si vysvetlíme na prípade BC, ktorý sa využíva ako kryptomena.

Elektronické financie sú typicky reprezentované pomocou elektronických mincí. Takáto minca je definovaná nejakou unikátnou sekvencie bitov. Avšak narozdiel od fyzických mincí, elektronické mince umožňujú jednoduchú falzifikáciu. Útočník skopíruje bitový reťazec danej mince a zaplatí ním viacnásobne rôzne produkty. Tento útok sa volá zdvojnásobenia výdavkov (anglicky *double-spending attack*). Proti tomuto útoku existuje tradičné zabezpečenie pomocou centrálnej autority. Banka je centrálna autorita, ktorá schvaľuje všetky manipulácie s elektronickými mincami a teda neumožní použiť mincu takýmto podvodným spôsobom. Avšak toto riešenie nie je možné použiť v decentralizovanej sieti, kde centrálna autorita neexistuje. V prípade decentralizovanej siete je možné zabrániť tomuto útoku pomocou použitia peer-to-peer siete v kombinácii s BC. [21]

Kryptomena Bitcoin ako prvá navrhla použitie peer-to-peer siete v spojení s BC technológiou pre zabránenie double-spending útoku. V takejto sieti je jediný zdroj na zdieľanie a to je dátová štruktúra BC v ktorej sú uložené všetky informácie o elektronických financiách. Zjednodušene môžeme povedať, že majorita uzlov siete zdieľa rovnaký zdroj (rovnakú kópiu BC). Ak chce niektorý uzol vykonať finančnú transakciu tak zašle správu s navrhovanou zmenou BC do siete. Uzly v tejto sieti nie je potrebné identifikovať pretože správy posielané v tejto sieti nie sú smerované na žiadne konkrétne miesto. Keď uzol prijme správu s nejakou modifikáciu tak si overí či ide o validnú požiadavku na finančnú transakciu. Štruktúra BC používa modernú kryptografiu na overenie validnosti transakcie (pozri sekciu 2.4). BC vlastnený väčšinou siete je ten, ktorý sa považuje za pravdu. Útočník by musel teda vlastniť aspoň 51 % uzlov v sieti aby mohol vykonať double-spending útok. Ak je daná sieť dostatočne veľká tak by toho útočník nemal byť schopný dosiahnuť. [29]

2.4 Kryptografia v BC

Pre pochopenie technológie BC je potrebná základná znalosť modernej kryptografie. V tejto sekcii je popísaný kryptografická hashovacia funkcia (pozri 2.4.1) a jej využitie na tvorbu dátových štruktúr zabezpečených proti modifikácii obsahu (viď sekcia 2.4.2). Ďalej je vysvetlený koncept asymetrickej kryptografie a digitálneho podpisu (viď sekcia 2.4.3). Tieto kryptografické primitíva sú základom na ktorom stojí nemennosť, auditovateľnosť a anonymita BC.



Obr. 2.1: Referenčný model peer-to-peer siete. [2]

2.4.1 Hashovacia funkcia

Hashovacia funkcia je taká funkcia h, ktorá má ako parameter x reťazec bitov ľubovolnej dĺžky a vracia reťazec y s konštantnou dĺžkou(viď rovnica 2.1). Reťazec y voláme hash. Hashovacia funkcia vracia pre konkrétny vstup vždy rovnaký hash.

$$h(x) = y \tag{2.1}$$

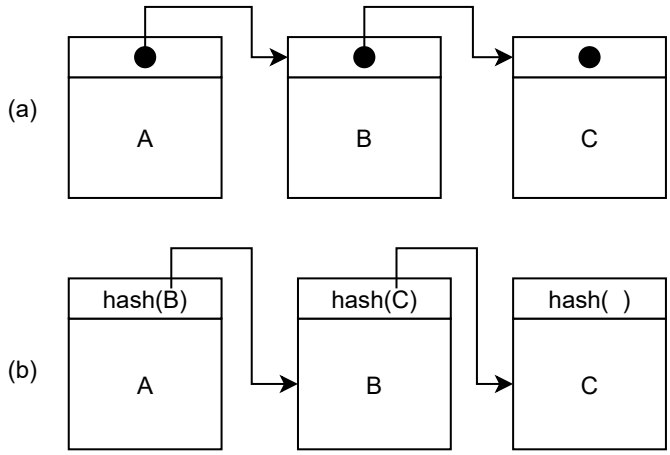
Kryptografická hashovacia funkcia, alebo tiež jednocestná funkcia (anglicky *one way function*), je taká hashovacia funkcia pre ktorú platia nasledujíce tri vlastnosti:

1. Pre daný hash x je výpočetne nezvládnuteľné nájsť správu takú, že $h(x) = y$. Anglicky voláme túto vlastnosť *first preimage resistant*.
2. Pre danú správu je výpočetne nezvládnuteľné nájsť inú správu s rovnakým hashom. Anglicky voláme túto vlastnosť *second preimage resistant*.
3. Pre ľubovoľnú správu je výpočetne nezvládnuteľné nájsť inú správu s rovnakým hashom. Anglicky voláme túto vlastnosť *collision resistant*.

Hashovacie funkcie majú v oblasti počítačovej bezpečnosti dôležité využitie:

- Bezpečné ukladanie hesiel: Digitálna služba neukladá v databáze heslo, ale len jeho hash. Pri ukradnutí databázy nedochádza k odhalenie hesiel užívateľov.
- Integrita dát: Hashovacia funkcia môže byť použitá na ochranu integrity ľubovoľných dát. Ak spočítate hash veľkého súboru a bezpečne ho uložíte tak ste schopný detekovať, že niekto tento súbor zmenil.
- Digitálny podpis: Hashovacia funkcia je kryptografické primitívum potrebné pre vytvorenie digitálneho podpisu.

Existuje množstvo hashovacích funkcií. Medzi veľmi známe a používané patrí napríklad MD5 (128 bitový výstup), SHA256 (256 bitový výstup), SHA512 (512 bitový výstup). [27, 36]



Obr. 2.2: (a) Zoznam pomocou ukazovateľov (b) Zoznam pomocou hash ukazovateľov

2.4.2 Hash ukazovateľ

Hash ukazovateľ (anglicky *hash pointer*) je primitívom pre tvorbu dátových štruktúr s kryptografickým zabezpečením proti manipulácia s obsahom (anglicky *tamper-evident*). Hash ukazovateľ funguje ako klasický ukazovateľ v zozname či strome. Navyše však neumožňuje meniť už pridané prvky. Jediná povolená operácia je pridanie ďalšieho prvku do dátovej štruktúry.

Obrázok 2.2 demonštruje rozdiel medzi zoznamom vytvoreným pomocou klasických ukazovateľov a pomocou hash ukazovateľov. Bežný zoznam umožňuje pozmeniť ľubovoľný už existujúci prvok nezávisle na zvyšku zoznamu. Naopak, hash pointer referencuje pomocou samotného dátového obsahu. Ak by sme zmenili dátový obsah prvku B, tak by sa narušila referencia v predchádzajúcom prvku. [1, 30]

2.4.3 Digitálny podpis

Digitálny podpis (anglicky *digital signature*) je kryptografický koncept používaný na autentifikáciu, autorizáciu a nepopierateľnosť. Digitálny podpis jednoznačne prepojí určitú entitu s informáciu. V technológii blockchain slúži digitálny podpis na určenie vlastníctva zdrojov, ktoré blockchain uchováva. [27, 29]

Moderná kryptografia používa pre zaistenie dôvernosti šifrovanie pomocou tajného kľúča. Pre zašifrovanie a dešifrovanie tajnej správy je potrebná znalosť tajného kľúča. Tento mechanizmus zaisťuje dôvernosť avšak nezaisťuje nepopierateľnosť pretože obe komunikujúce strany poznajú tajný kľúč a teda nie je možné právne dokázať kto správu napísal. Na zaistenie nepopierateľnosti sa používa asymetrické šifrovanie, ktoré používa dvojicu kľúčov:

- **Privátny kľúč** je tajný a pozná ho len odosielateľ správy. Odosielateľ používa tento kľúč na zašifrovanie správy.
- **Verejný kľúč** je dostupný komukoľvek. Ktokoľvek s týmto kľúčom dokáže dešifrovať správu.

Tieto dva kľúče tvoria dvojicu prepojenú matematickým spôsobom. Zo znalosti verejného kľúča je výpočetne nezvládnuteľné zistiť privátny kľúč. Zašifrovaná správa nie je dôverná

pretože ktokoľvek môže použiť verejný kľúč na jej dešifrovanie. Avšak zašifrovaná správa je nepopierateľne napísaná vlastníkom privátneho kľúča.

Tento koncept je základom digitálneho podpisu. Ak chceme nepopierateľne dokázať, že nejaký dátový obsah (napríklad pdf dokument) sme vytvorili mi, tak vypočítame jeho hash (viď sekcia 2.4.1) a zašifrujeme ho naším privátnym kľúčom. Zašifrovaný hash priložíme k dokumentu. Príjemca dokumentu si následne pomocou verejného kľúča dešifruje hash priložený k správe a porovná si ho s tým ktorý vypočítal sám z danej správy. Ak sú hashe rovnaké tak nikto správu nezmenil a dokument je jednoznačne vytvorený vlastníkom tajného kľúča. Najznámejšie algoritmy na digitálny podpis sú RSA, DSA, ECDSA. [27]

2.4.4 Prahový digitálny podpis

Prahový digitálny podpis (anglicky *threshold signatures*) je špeciálna schéma digitálneho podpisu. V tejto schéme je n účastníkov a každý vlastní časť privátneho kľúča. Každý účastník môže použiť svoju časť tajného kľúča na čiastočné podpísanie správy M. Kompletný podpis môže byť zostrojený ak aspoň t účastníkov poskytlo svoju časť podpisu. Potom hovoríme o t-z-n prahovom podpise. Takýto koncept sa dá využiť pri kryptografickom hlasovaní. [10]

2.5 Dátová štruktúra blockchain

Blockchain je dátová štruktúra podobná zoznamu (anglicky *linked list*). Blockchain organizuje dáta do podmnožín, ktoré sa volajú bloky. Blok je podobný uzlu v zozname. Každý blok obsahuje referenciu na ďalší blok. Rozdiel medzi zoznamom a blockchainom je v tom, že referencia blockchainu je zabezpečená proti manipulácia (anglicky *tamper-evident*) pomocou modernej kryptografie. Bežný zoznam používa referenciu pomocou ukazovateľov (anglicky *pointers*), ktoré môže ktokoľvek a kedykoľvek pozmeniť bez toho aby pozmenil dátový obsah. Naopak, blockchain vôbec neumožňuje meniť už pridané bloky. Jediná povolená operácia je pridanie ďalšieho bloku na koniec blockchainu. [1]

Každý blok obsahuje dáta, ktoré sú typicky vo forme transakcií. Kryptograficky bezpečný blockchain by mohol fungovať aj tak, že v každom bloku bude uložená práve jedna transakcia. Z dôvodu optimalizácie je ale v jednom bloku uložené množstvo transakcií. Vďaka tejto optimalizácii nemusí celá sieť vytvárať konsenzus po každej transakcii. Samotné transakcie v rámci jedného bloku sú ukladané v ďalšej dátovej štruktúre, ktorá taktiež používa kryptografické hashovanie (viď sekcia 2.5.3). [30]

2.5.1 Blok

Blok sa skladá z hlavičky a tela. Telo bloku obsahuje dáta a hlavička obsahuje metadáta. Dáta v tele bloku sú uložené vo forme transakcií. Transakcie sú popísané v sekcii 2.5.2. Počet transakcií v bloku je typicky obmedzený maximálnou veľkosťou bloku. Hlavička bloku obsahuje metadáta o bloku, kde najdôležitejšie a najbežnejšie sú nasledujúce [43]:

- Hash všetkých transakcií.
- Časové razítko vytvorenia bloku.
- Hash ukazovateľ na predošlý blok v blockchaine.

2.5.2 Transakcia

Transakcia je elementárna dátová jednotka na ukladanie digitálnych informácií v blockchaine. Bitcoin, prvý blockchain, použil transakciu na manipuláciu s elektronickými financiami. Takáto transakcia sa skladá z troch častí [30]:

- **Množina vstupov**: Každý vstup má uložený hash predošlej transakcie s ktorej vychádza. Ďalej definuje, ktoré výstupy s predošlej transakcie si nárokuje. Nakoniec obsahuje digitálny podpis, ktorý autorizuje tvorcu transakcie.
- **Množina výstupov**: Každý výstup má hodnotu, ktorá je uchovávaná v blockchaine (typicky minca nejakej kryptomeny). Suma hodnôt všetkých výstup transakcie musí byť menšia alebo rovná sume všetkých vstupov transakcie. Ak je menšia, tak tento rozdiel je použitý ako odmena pre toho, kto publikoval tento blok blockchainu.
- **Hlavička**: Obsahuje hash transakcie, ktorý je používaný ako unikátny identifikátor pomocou, ktorého sa na transakciu odkazujeme.

2.5.3 Markle tree

Binárny hashovací strom alebo tiež Merkle strom (anglicky *Merkle tree*) [12] je dátová štruktúra podobná binárnemu stromu, ktorá slúži na efektívne a rýchle vypočítanie hashu veľkého množstva dát. Blockchain používa tento strom na časovo efektívny výpočet hashu všetkých transakcií. Takto vypočítaný hash je uložený v hlavičke bloku.

Merkle strom je vyvážený binárny strom, kde listové uzly obsahujú jednotlivé transakcie uložené v danom bloku blockchainu. Každý nelistový uzol stromu obsahuje hash vypočítaný z jeho potomkov. Koreňový uzol teda obsahuje hash celého stromu a teda aj všetkých transakcií. Pridanie, odobranie, zmena obsahu, alebo zmena poradia transakcií bude teda viesť k zmene koreňového hashu. Konštrukcia stromu, inak povedané výpočet hashu všetkých transakcií, prebieha následovne:

1. Všetky transakcie sú uložené do listovej úrovne stromu. Ak je počet transakcií nepárny tak, je posledná vložená dvakrát.
2. Nad každým listovým uzlom je vypočítaný hash.
3. Každý nelistový uzol skonkatenuje hash ľavého a pravého syna, vypočíta nad nimi hash a uloží si ho.

Konštrukcia takéhoto stromu pre n transakcií má časovú zložitosť $O(log(n))$. Takýto spôsob výpočtu hashu je teda veľmi efektívny pre veľké množstvo transakcií (blok v blockchaine bežne obsahuje stovky transakcií).

Merkle strom umožňuje efektívne šetriť pamäťové nároky blockchainu. Do blockchainu sú neustále pridávané nové bloky, ktoré obsahujú aj rovnaké staré transakcie. Ak už sú transakcie zaznamenané v dostatočne veľkom množstve blokov tak sú z hľadiska bezpečnosti nemenné. V nových blokoch ich už preto nie je potrebné ukladať. Nový blok si preto uloží len hashe starých vetiev stromu, ale ich obsah už nepotrebuje. Takto je zachovaná integrita hashu všetkých transakcií. [29]

Kapitola 3

Konsenzus

Konsenzus v BC zabezpečuje, že skupina uzlov (peerov) sa zhodne na rovnakom stave BC. Tradične je táto vlastnosť zabezpečená centrálnou autoritou s ktorou musia byť všetky uzly spojené. Avšak BC je decentralizovaný a teda toto riešenie nie je možné. Konsenzus v decentralizovanej sieti BC je zabezpečený pomocou protokolu, ktorý sa snaží nájsť kompromis medzi nasledujúcimi troma vlastnosťami (anglicky *CAP theorem*) [20, 42, 26]:

- Konzistentnosť (*Consistency*): Všetci uživatelia majú rovnaký stav BC (rovnaké dáta).
- Dostupnosť (*Availability*): Pre každú požiadavku na dáta musí byť poskytnutá odpoveď v konečnom čase. Odpoveďou môže byť aj neúspech.
- Odolnosť voči prerušeniu (*Partial tolerance*): Sieť funguje aj v prípade, že v nej vznikajú chyby (výpadky uzlov, škodlivé správanie časti uzlov).

CAP theorem [20] deklaruje, že z týchto troch vlastností je súčasne možné dosiahnuť maximálne dve.

Technológia BC požaduje decentralizovanú distribuovanú sieť. Z vlastností takejto siete je zrejmé, že BC musí nevyhnutne riešiť problém odolnosť voči prerušeniu. Zo zvyšných dvoch vlastností sa môže zvoliť dostupnosť pred konzistenciou (napríklad Bitcoin). Bitcoin zaisťuje konzistenciu až dodatočne (anglicky *eventual consistancy*) pomocou procesu ťažby blokov a určenia najdlhšieho reťazca. Druhou možnosťou je zvoliť konzistenciu pred dostupnosťou. Táto skupina BC protokolov je založená na hlasovaní pomocou PBFT protokolu (viď 3.4.1) na zaručenie silnej konzistencie. [39]

3.1 Typy konsenzus protokolov

Existujú tri najbežnejšie techniky, ktoré sa používajú pre ustavenie konsenzu [42, 22]:

- **Lotéria**: Takéto protokoly náhodne zvolia uzol, ktorý vyprodukuje nový blok. Výhodou tohoto prístupu je jeho jednoduchosť keďže takýto proces nevyžaduje žiadnu interaktívnu komunikáciu. Nevýhodou tohoto prístupu je, že pripúšťa možnosť voľby viacerých uzlov súčasne. V takom prípade sa reťazec rozvetví (anglicky *fork*) a je potrebné určiť ktorá vetva je správna. Typicky sa za správnu vetvu volí tá najdlhšia. Avšak takéto správanie oslabuje konzistentnosť BC. Transakcie v posledných blokoch môžu byť potenciálne zahodené pretože nejde o správnu vetvu. Preto sa za konzistentné transakcie považujú až tie ktoré sú prekryté väčším množstvom nových blokov.

- **Hlasovanie**: Protokoly založené na hlasovaní dosahujú dohodu pomocou hlasovania všetkých zapojených uzlov. Môžeme použiť napríklad protokol Byzantskej chyby (anglicky *Byzant fault tolerance*), ktorý vyžaduje majoritu hlasov k uzavretiu konsenzu (typicky $\frac{2}{3}$). Výhodou je veľmi malá pravdepodobnosť vzniku vetiev reťazca. Na druhej strane, takéto protokoly majú nižšiu priepustnosť, ktorá klesá z narastajúcim počtom uzlov.

- **Kombinovaný prístup**: Tieto protokoly sa snažia kombinovať prístup lotérie a hlasovania s cieľom dosiahnuť výhody oboch prístupov. Napríklad je možné rozdeliť počet uzlov podieľajúcich sa na hlasovaní pomocou lotérie, čím sa zvýši priepustnosť.

Dnes sa hovorí najmä o dvoch rodinách konsenzus protokolov. Prvou a najbežnejšou rodinou je *Proof-of-Work*. Tento konsenzus mechanizmus je založený na princípe lotérie. Detailnejšie je popísaný v sekcii 3.2.

Táto práca sa však zaoberá predovšetkým konsenzus mechanizmom, ktorý sa všeobecne nazýva *Proof-of-Stake*. Táto rodina konsenzus protokolov je založená na hlasovaní. V poslednej dobe sa ponúka ako vhodná alternatíva k Proof-of-Work. Tento mechanizmus je detailnejšie popísaný v sekcii 3.3

3.2 Proof-of-Work

Dôkaz prácou, anglicky *Proof-of-Work* (ďalej PoW), je najbežnejšia stratégia konsenzus protokolu. Ak chce uzol publikovať nový blok, musí investovať svoj výpočtový výkon do riešenia netriviálneho kryptografického problému. Uzol, ktorý ako prvý vyrieši tento problém má najväčšiu pravdepodobnosť, že bude jeho blok pridaný do reťazca. Samozrejme, je tu možnosť, že problém vyrieši súčasne viacero uzlov. Konečná voľba je teda náhodná. PoW teda umožňuje, aj keď s oveľa menšou pravdepodobnosťou, že sa reťazec rozvetví. Bezpečnosť takéhoto konsenzu spočíva v tom, že majorita výpočtového výkonu siete (51 %) je vlastnená poctivými uzlami.

PoW konsenzus využíva dva typy uzlov. Prvý typ uzla je miner, ktorý vytvára nové bloky tak ako je popísané v sekcii 3.2.1. Druhý typ uzla je bežný vlastník zdrojov v danom BC, ktorý môže vytvárať transakcie a distribuovať ich do siete. Druhý typ uzla teda nehrá žiadnu rolu v ustanovovaní konsenzu. [26]

3.2.1 Ťažba blokov

Ťažba (anglicky *mining*) [30] bloku je proces pridania nové bloku na koniec blockchainu. Ťažba bloku zahŕňa validáciu transakcií a blokov. Preto je ťažba kritická pre správne a bezpečné fungovanie blockchainu. Každý uzol siete, ktorý ťaží nové bloky sa nazýva anglickým slovom *miner*. Tieto uzly umožňujú rozširovanie blockchainu. Aby takéto uzly existovali, musia byť motivované. Miner dostane za každý vyťažený blok ako odmenu zdroje uložené v blockchaine.

Ťažba všeobecne pozostáva z nasledujúcich krokov:

1. Miner prijíma požiadavky na transakcie z peer-to-peer siete. Každú transakciu si validuje pomocou kryptografie popísanej v sekcii 2.4. Miner musí taktiež udržiavať aktuálny stav blockchainu. Je potrebné sledovať či nevznikli nové bloky a udržiavať si validný blockchain.

2. Ak miner vlastní validnú a aktuálnu kópiu blockchainu a má dostatok transakcií, môže začať vytvárať nový blok. Do nového bloku vloží transakcie, ktoré prijal a boli validné. Následne investuje svoj výpočtový výkon do vyriešenia kryptografického problému (podrobnejšie popísané v sekcii 3.2.2). Ak úlohu vyrieši tak môže zostrojiť nový validný blok.

3. Novo vytvorený blok je potrebné distribuovať do siete. Ak väčšina siete blok získa a akceptuje, tak bol pridaný do blockchainu.

4. Ak sa podarilo úspešne blok pridať do blockchainu tak miner získava odmenu. Odmena za vyťažený blok je konštantná čiastka zdrojov poskytovaných daným blockchainom. Napríklad Bitcon poskytuje v roku 2021 ako odmenu 6,25 bitcoinov čo je približne 300 dolárov [1]. Avšak táto odmena môže byť zvýšená o poplatky, ktoré sú v transakciách. Ak teda chcete aby sa vaša transakcia dostala do blockchainu čo najrýchlejšie, poskytnete vyššiu odmenu v podobe poplatku za transakciu. Miner bude potom viac motivovaný pridať práve túto transakciu do bloku.

3.2.2 Bitcoin

Bitcoin je najznámejší BC protokol, ktorý používa PoW konsenzus. Samotný kryptografický problém, ktorý užívatelia tejto siete riešia spočíva v počítaní hashu (viď 2.4.1) z hlavičky nového bloku (viď 2.5.1). Hlavička obsahuje atribút nonce, ktorý môže miner ľubovolne nastaviť. Zmenou tohoto atribútu môže miner získať iný hash hlavičky bloku. Konsenzus vyžaduje aby výsledná hash hodnota bola menšia rovná určitej zvolenej hodnote. Miner môže túto podmienku dosiahnuť len tak, že bude inkrementovať hodnotu atribútu nonce až dokedy túto podmienku nesplní. Táto úloha sa teda dá riešiť len pomocou metódy útok hrubou silou (anglicky *brute force*). Miner môže svoju šancu na úspech zvýšiť len tým, že poskytne väčší výpočtový výkon do jej riešenia. Na druhej strane, ostatné uzly môžu overiť, že jeho riešenie je správne veľmi rýchlo a efektívne. [43]

3.2.3 Vlastnosti

PoW je overený konsenzus protokol, ktorý funguje a používa sa v blockchainoch ako je Bitcoin [29] alebo Ethereum [2]. Tento protokol má však jeden dlhodobý problém a to je spotreba energie. Uzly ktoré riešia kryptografický problém pre nové bloky spotrebujú veľké množstvo energie čo má nepriaznivý dopad na životné prostredie. Niektoré zdroje [3] napríklad hovoria, že v roku 2021 pokrýva ťažba Bitcoinu 0,5 % celkovej spotreby elektrickej energie na svete. Pre porovnanie, ide o sedemkrát väčšiu spotrebu energie ako má celá spoločnosť Google. [26]

Druhou veľkou nevýhodou Bitcoinu je jeho slabá priepustnosť transakcií (*TPS - transaction per second*). TPS je metrika kľúčová napríklad v oblasti kryptomien. Pre porovnanie, centralizovaný platobný systém VISA [4] má TPS približne 1 500 zatiaľ čo Bitcoin približne 5.

[1] https://www.investopedia.com/tech/how-does-bitcoin-mining-work/
[2] https://ethereum.org/en/whitepaper/
[3] https://www.businessinsider.com/bitcoin-mining-electricity-usage-more-than-google-2021-9
[4] https://towardsdatascience.com/the-blockchain-scalability-problem-the-race-for-visa-like-transaction-speed-5cce48f9d44

3.3 Proof-of-Stake

Dôkaz podielom na vlastníctve, anglicky *Proof-of-Stake* (ďalej PoS), je založený na technike lotérie, kde pravdepodobnosť výhry rastie s množstvom už vlastnených zdrojov. Základnou myšlienkou, je že vlastník veľkého množstva zdrojov v danom BC je veľmi nepravdepodobným útočníkom pretože svoje zdroje nechce ohroziť. Uzly sa teda musia preukázať vlastníctvom zdrojov v danom BC ak chcú publikovať nový blok. Pravdepodobnosť výberu uzlu rastie s množstvom zdrojov, ktoré v sieti vlastní. [22, 31]

3.3.1 Vlastnosti

Veľkou výhodou PoS oproti PoW je, že nevyžaduje také veľké množstvo energie. Uzly už nemusia súťažiť v riešení výpočtovo náročných úloh. Ďalšou veľkou výhodou je rýchlosť. PoS konsenzus umožňuje rádovo vyššie TPS, ktoré je v prípade PoW značne limitované. [26, 31]

PoS konsenzus však prináša aj viaceré problémy. Najväčším problémom je bezpečnosť. PoS prinieslo nové typy útokov, ktoré pri PoW nevznikali. Tieto útoky sú podrobnejšie popísané v sekcii 3.6. Bežným problémom väčšiny PoS protokolov je, že nedefinujú teoretický základ a formálne definovanú bezpečnosť. Existujú teda rôzne riešenia na tieto útoky avšak neexistuje formálny bezpečnostný model, ktorý by dokazoval, že tieto útoky sú zabezpečené.

Ďalším problémom je model motivácie. PoS konsenzus podporuje validné správanie odmeňovaním v podobe zdrojov. Naopak, škodlivé chovanie je trestané ich odoberaním. Problémom tohoto mechanizmu je jeho nejednoznačná definícia. Väčšina protokolov neobsahuje formálnu analýzu ich modelu motivácie. [31]

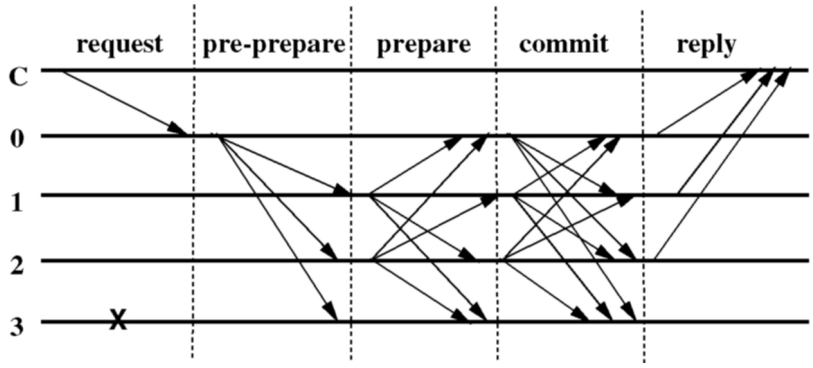
3.4 Proof-of-Authority

PoA [4] je rodina konsenzus protokolov založená na algoritmoch BFT (*Byzantian Fault Tolerance*). PoA predpokladá, že v sieti je N dôveryhodných uzlov. Tieto uzly (autority) sú jednoznačne identifikované a majorita autorít je poctivá (konkrétne $N/2 + 1$). Veľmi rozšírený PoA protokol je PBFT popísaný v sekcii 3.4.1.

3.4.1 Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance [14] (ďalej len PBFT) je protokol na uznesenie konsenzu v distribuovanej sieti pomocou hlasovania. Tento protokol patrí do rodiny proof-of-authority. Avšak je často používaný aj v rámci PoS konsenzu. PoS protokoly často vyžadujú interné hlasovanie ktoré sa často rieši práve pomocou PBFT, kde dôveryhodnosť uzlov je zabezpečená vlastníctvom zdrojov (PoS). PBFT funguje pod podmienkou, že v sieti je maximálne $\frac{1}{3}$ nepoctivých uzlov. Na komunikáciu v sieti sa používa všesmerové vysielanie (anglicky *broadcast*). Na zabezpečenie sa požíva kryptografia s digitálnym podpisom (viď sekcia 2.4.3). Jednotlivé uzly môžu posielať požiadavky do siete (vtedy hovoríme o tomto uzle ako o klientovi). Protokol definuje jeden uzol v sieti ako vodcu a ostatné uzly ako validátorov. Vodca hlasovania sa pravidelne mení podľa dohodnutého rozvrhu. Hlasovanie protokolu prebieha v štyroch fázach:

1. *Request*: Klient zašle vodcovi požiadavku na uznesenie konsenzu v sieti.
2. *Pre-prepare*: Vodca rozošle požiadavku všetkým validátorom.



Obr. 3.1: Priebeh hlasovania v PBFT (prevzaté z [14]).

3. *Prepare*: Validátori overia požiadavku, ak súhlasia, tak pošlú svoj hlas všetkým v sieti. Zároveň, každý uzol siete (validátori aj vodca) sledujú a overujú hlasy, ktoré prijali. Ak uzol zaznamenal viac ako $\frac{2}{3}$ hlasov tak zašle potvrdenie všetkým uzlom v sieti.

4. *Commit*: Každý uzol počíta potvrdenia od všetkých ostatných uzlov v sieti. Ak získa viac ako $\frac{1}{3}$ potvrdení, tak pošle potvrdenie klientovi, ktorý pôvodne inicializoval požiadavku na konsenzus. Klient vie, že jeho požiadavka bola prijatá sieťou, ak prijme viac ako $\frac{1}{3}$ potvrdení.

Obrázok 3.1 demonštruje tento protokol. **C** je klient, uzol 0 je vodca a uzly 1 až 3 sú validátori. Môžeme vidieť, že sieť funguje aj keď uzol 3 nereaguje na komunikáciu.

3.5 Všeobecné útoky na konsenzus

Nasledujúca sekcia vysvetľuje najbežnejšie typy útokov na bezpečnosť konsenzus protokolov. Ide o útoky, ktoré sú aplikovateľné na všetky rodiny konsenzus protokolov. Ďalej popísané útoky vychádzajú z nasledujúcej práce [22].

3.5.1 Ovládnutie konsenzu útočníkmi

Tento útok naruší decentralizovanosť siete tým, že útočníci dokážu utvoriť konsenzus aj bez poctivých uzlov. V takom prípade sa stáva sieť centralizovaná, kde centrálnou autoritou sú práve útočníci. Príkladom takéhoto útoku pre PoW a PoS konsenzus je ovládnutie 51 % siete. V prípade protokolov Byzantskej chyby dokáže $\frac{1}{3}$ uzlov spôsobiť, že bude protokol narušený alebo dokonca zastavený.

3.5.2 Porušenie synchronného doručovania

Ak útočník dokáže narušiť synchrónne doručovanie správ v protokole, ktorý synchronizáciu predpokladá tak takýto protokol prestane fungovať. Tento útok už nie je možné urobiť na protokole, ktorý umožňuje asynchronnú komunikáciu. Tento útok je možné vykonať napríklad pre protokoly Byzantskej chyby.

3.5.3 Útok na časovú synchronizáciu

Decentralizovaná sieť BC potrebuje časovú synchronizáciu medzi uzlami. Pri vytvorení nového bloku sa do jeho hlavičky zvyčajne vkladá aj časové razítko (anglicky *timestamp*).

Jednotlivé uzly v sieti majú vlastný čas, ktorý typicky vypočítavajú ako medián všetkých časov získaných od ostatných. Tvorca nového bloku potom vloží do hlavičky práve takto vypočítaný čas. Ostatné uzly v sieti budú pri distribúcii bloku overovať, že čas je dostatočne aktuálny aby bol akceptovaný. Ak útočník disponuje veľkým množstvom uzlov v sieti, tak môže narušiť synchronizáciu času, ktorý bloky získavajú mediánom. Tento útok následne spomaľuje sieť pretože bloky distribuujú bloky s časovou značkou, ktorá už nebude akceptovaná.

3.5.4 Zdvojnásobenie výdavkov

Zdvojnásobenie výdavkov (anglicky *double spending*) je útok, ktorý vzniká vytvorením dvoch alebo viac konfliktných blokov. Tieto bloky vytvárajú tzv. vetvy (anglicky *forks*). S tohoto dôvodu môžu byť niektoré krypto mince dočasne minuté v oboch konfliktných blokoch. Neskôr je síce len jeden z nich validný pretože druhá vetva bude zahodená. Avšak tento útok spomaľuje konzistentnosť blockchainu. Tento útok už bol popísaný aj v sekcii 2.3.2.

3.5.5 Útok na podskupiny uzlov

Niektoré konsenzus protokoly rozdeľujú celú sieť na podskupiny uzlov (anglicky *shards*). Sharding zvyšuje škálovateľnosť a priepustnosť siete pretože uzly validujú transakcie len v rámci svojej podskupiny. Na druhej strane, tento prístup môže viesť k zníženie bezpečnosti. Množstvo spolupracujúcich uzlov v jednej takejto podskupine je oveľa menší než v celej sieti. Pre útočníka môže byť preto jednoduchšie ovládnuť takúto podskupinu ako celú sieť.

3.6 Útoky na Proof-of-Stake

V tejto sekcii sa pozrieme na útoky špecifické pre PoS konsenzus protokoly. Vymenované útoky sú definované v tejto práci [22].

3.6.1 Vetvenie bez rizika straty zdrojov

Generovanie blokov v PoS nestojí žiadnu energiu v podobe výpočtového výkonu. Uzly preto môžu generovať viacero konfliktných uzlov súčasne a tým zvyšovať pravdepodobnosť, že bude práve ich uzol pridaný do reťazca. Takéto správanie nepredstavuje pre daný uzol žiaden risk v podobe straty zdrojov. Problémom tohoto správania je, že vzniká väčšie množstvo vetiev reťazca. Ako dôsledok sa potom zvyšuje čas do konzistentnosti BC.

3.6.2 Ovplyvnenie volieb

PoS protokoly často potrebujú aby podielnici hlasovali. Na to používajú hlasovacie protokoly z rodiny BFT. Ako bolo popísané v sekcii 3.4.1, tieto protokoly typicky fungujú tak, že jeden uzol (vodca) vedie voľby a ostatné uzly (validátori) ich overujú a potvrdzujú. Roľa vodcu má rôzne výhody (napríklad niektoré BC odmeňujú vodcu podielom z poplatkov za vložené transakcie do bloku). Útočník preto môže chcieť ovplyvniť voľbu vodcu v svoj prospech. Na určenie, ktorý uzol sa stane vodcom sa často používa nejaká forma náhodnosti, ktorá je distribuovane generovaná. Útočník môže ovplyvniť generovanie tejto náhodnosti vo svoj prospech, tak aby sa stal vodcom práve on.

3.6.3 Odmietnutie služby lídrovi/výboru

Ako už bolo spomenuté v sekcii 3.6.2, PoS protokoly často používajú BFT hlasovanie v ktorom je kľúčová roľa vodcu bez ktorého hlasovanie neprebehne. Niektoré PoS protokoly vytvárajú rozvrh vodcov na dlhšie časové obdobie. Takýto rozvrh je dopredu známi čo zjednodušuje a zrýchľuje celý konsenzus. Na druhej strane, útočníci dopredu vedia, ktorý uzol bude v konkrétnom čase vodcom. Útočník potom môže v danom čase vykonať DoS útok na vodcu a znemožniť uzatvárať konsenzus v BC pomocou hlasovania.

3.6.4 Neskoršia korupcia

Útočník sa snaží získať privátne kľúče uzlov, ktoré mali v minulosti vplyv na reťazec BC. Útočník ich môže ukradnúť, ale taktiež kúpiť. Tieto uzly môžu byť ochotné predať svoje privátne kľúče pretože tým nič neriskujú. Virtuálne zdroje, ktoré majú vložené do daného BC môžu kedykoľvek vymeniť za reálne peniaze. Ak útočník získa kľúče s dostatočným podielom zdrojov, tak môže pozmeniť históriu reťazca.

Kapitola 4

Harmony

Harmony je BC používajúci PoS konsenzus a *sharding*. Cieľom tohoto protokolu je poskytovať aplikácie, ktoré v minulosti nebolo možné uskutočňovať na technológii BC z dôvodu rýchlosti a škálovateľnosti. Harmony je BC, ktorý má poskytovať služby ako decentralizované zmenárne alebo platobný systém rozsahovo porovnateľný s Visa.

Ak nie je uvedené inak, tak všetky informácie o protokole Harmony popisované v tejto kapitole vychádzajú z oficiálnej dokumentácie publikovanej autormi tohoto protokolu [25, 37]. Sekcia 4.1 popisuje konsenzus protokol. Sekcia 4.2 vysvetľuje sharding používaný v tomto protokole a sekcia 4.3 definuje model motivácie. Na záver je v sekcii 4.4 tento protokol analyzovaný z hľadiska bezpečnosti a výkonnosti.

4.1 Konsenzus

Konsenzus v Harmony protokole je založený na algoritme PBFT, ktorý už bol popísaný v sekcii 3.4.1. Jeden z kľúčových problémov PBFT je, že jeho časová zložitosť je $O(n^2)$ pre n uzlov. Táto vlastnosť neumožňuje dobrú škálovatelnosť siete. Avšak Harmony používa úpravu PBFT, ktorá má lineárnu časovú zložitosť vzhľadom k počtu uzlov v sieti (viď sekcia 4.1.1).

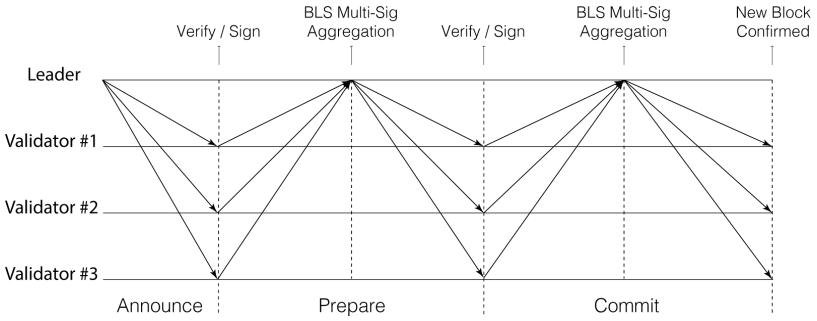
4.1.1 Protokol FBFT

Pre vygenerovanie nového bloku používa Harmony svoj vlastný protokol FBFT (*Fast Byzantine Fault Tolerance*). FBFT namiesto zasielania hlasov pomocou broadcastu používa prahový digitálny podpis (pozri sekciu 2.4.4). FBFT konsenzus prebieha následovne:

1. Vodca vytvorí blok a rozošle jeho hlavičku aj dátový obsah validátorom pomocou broadcastu.

2. Validátori príjmu nový blok, overia jeho hlavičku, podpíšu ju svojím digitálnym podpisu a pošlú späť vodcovi. Obsah bloku je zatiaľ ignorovaný.

3. Keď vodca prijme aspoň $\frac{2}{3}$ podpisov, tak ich agreguje do jediného prahového digitálneho podpisu (viď kryptografický prahový podpis popísaný v sekcii 2.4.4). Tento podpis rozošle pomocou broadcastu spolu s bitmapou indikujúcou validátorov ktorý podpísali.



Obr. 4.1: Priebeh hlasovania v FBFT (prevzaté z [25]).

4. Každý validátor overí, že prahový podpis obsahuje požadované $\frac{2}{3}$ hlasov. Až v tejto chvíli validátor overí transakcie v dátovom obsahu bloku, ktorý bol zasielaný už v kroku 1. Ak všetko súhlasí, tak podpíše správu s kroku 3 a pošle ju späť vodcovi.

5. Vodca čaká na $\frac{2}{3}$ podpisov validátorov s predošlého kroku (môžu sa líšiť od podpisov z kroku 3). Opäť ich agreguje do prahového podpisu a spolu s bitmapou účastníkov rozošle pomocou broadcastu nový blok na potvrdenie všetkým validátorom.

Obrázok 4.1 demonštruje FBFT protokol pre štyri uzly. Na tomto príklade môžeme vidieť, že celé hlasovanie má lineárnu časovú zložitosť, keďže broadcast vysiela len vodca (na rozdiel od PBFT, kde broadcast vysielajú aj validátori).

Je dôležité podotknúť, že Harmony je PoS konsenzus. Vodca v skutočnosti nečaká na $\frac{2}{3}$ podpisov, ale len na toľko aby ich vlastníci spoločne vlastnili $\frac{2}{3}$ celkového podielu.

4.2 Sharding

Harmony protokol používa *sharding* aby dosiahol lepšiu škálovateľnosť a priepustnosť transakcií. Niektoré zdroje[1] deklarujú, že Harmony má priepustnosť priemerne 2 000 TPS. Aktuálne používa 4 shardy a autori tvrdia, že pridaním každej ďalšej sa zvýši priepustnosť o 500 TPS. Táto sekcia vysvetľuje celý koncept shardingu v protokole Harmony.

4.2.1 Rozdelenie hlasovacím podielom

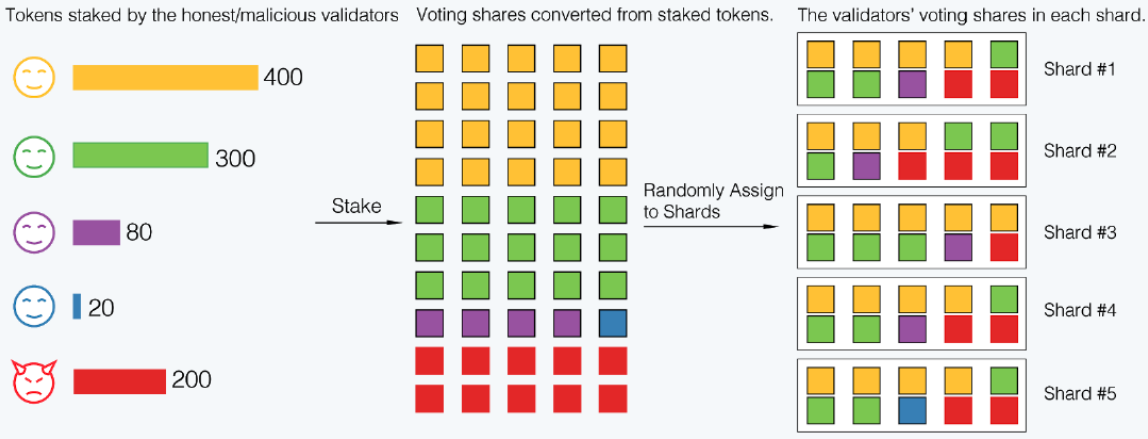
Ak sa chce uzol stať účastníkom Harmony blockchainu, musí vložiť určité množstvo zdrojov. Tento podiel určuje koľko bude mať hlasovacích lístkov pri validovaní. Tieto hlasovacie lístky sú následne náhodne rozdelené medzi všetky shardy (viď sekcia 4.2.4) po dobu jednej epochy (viď sekcia 4.2.2). Tento postup je demonštrovaný na obrázku 4.2.

4.2.2 Epocha

Epocha je časový interval počas, ktorého je štruktúra každého shardu nemenná. Epocha odpovedá času potrebnému na vygenerovanie 32 768 blokov v beacon sharde, čo je približne 18,2 hodiny[2] (koncept beacon shard bude vysvetlený v sekcii 4.2.6). Na začiatku epochy je vygenerované náhodné číslo (pozri sekciu 4.2.4) na základe ktorého sa vytvorí rozdelenie

[1] https://www.investing.com/news/cryptocurrency-news/harmony-one-slow-and-steady-wins-the-race-2609585

[2] https://docs.harmony.one/home/network/validators/definitions/epoch-transition



Obr. 4.2: Rozdelenie hlasovacieho podielu validátorov medzi všetky shardy (prevzraté z [37]).

do shardov. Účastníci, ktorý chcú validovať v nasledujúcej epoche musia v aktuálnej vložiť svoj podiel tokenov.

4.2.3 Hlasovacie lístky

Celkové množstvo hlasovacieho podielul je rozdelené na konštantne veľké tokeny (hlasovacie lístky). Hodnota jedného hlasovacieho lístku t v epoche e je určená rovnicou 4.1, kde S_{e-1} je celkový podiel vložený v epoche $e-1$, n je počet shardov a λ je bezpečnostný parameter. Ak $\lambda > 600$, tak pravdepodobnosť ovládnutia aspoň $\frac{1}{3}$ shardu jedným útočníkom je $0,000\,003\,\%$ (dôkaz viď [37] kapitola 3).

$$t = \frac{S_{e-1}}{n.\lambda} \tag{4.1}$$

4.2.4 Náhodné rozdelenie hlasovacieho podielu medzi shardy

Harmony rozdeľuje hlasovacie lístky do shardov pomocou náhodnej voľby (anglicky *randomness based sharding*). Na začiatku epochy je vykonaná náhodná permutácia hlasovacieho podielu všetkých podielnikov. Získaná permutácia je rozdelená na n rovnakých častí, kde n je počet shardov. Časť i, kde $1 \leq i \leq n$, predstavuje validátorov a ich hlasovací podiel v sharde i. Permutácia je vykonaná na základe náhodného čísla *rnd*, ktoré je generované distribuovane.

Na vygenerovanie *rnd* sa používa algoritmus VRF (*Verifiable Random Function [19]*) a VDF (*Verifiable Delay Function [9]*). Tento výpočet vykonáva beacon shard (koncept beacon shard bude vysvetlený v sekcii 4.2.6):

1. Vodca pošle hash posledného bloku všetkým validátorom.
2. Každý validátor vypočíta s prijatého hashu pomocou VRF náhodné číslo, ktoré pošle späť vodcovi.
3. Keď vodca príjme $\frac{1}{3}$ náhodných čísel, tak nad nimi urobí XOR. Výslednú hodnotu *pRand* vloží do nového bloku pomocou FBFT konsenzu popísaného v sekcii 4.1.1.

4. Keď je *pRand* potvrdená v bloku, vypočíta vodca z tejto hodnoty finálne náhodne číslo *rnd* pomocou VDF.

5. VDF garantuje, že výpočet *rnd* zaberie toľko času aby sa stihlo vyprodukovať špecifické množstvo nových blokov. Keď vodca vypočíta *rnd* tak ho pomocou FBFT vloží do nového bloku.

4.2.5 Voľba vodcu

V jednej epoche sa vždy vygeneruje počet blokov n. Inak povedané, prebehne práve n kôl protokolu FBFT. Počas celej epochy je rovnaký vodca FBFT. Za vodca shardy v epoche je určený podielnik, ktorý vlastní prvý hlasovací lístok v diele hlasovacích lístkov určených pre túto shardu (viď 4.2.4). Pravdepodobnosť, že podielnik bude zvolený za vodcu shardy je priamo úmerná množstvu podielu, ktorý vložil (princíp PoS).

4.2.6 Komunikácie medzi shardami

Každý shard spravuje vlastný reťazec (anglicky *shard chain*), ktorý spracováva vlastné transakcie. Jeden shard má špeciálnu úlohu a nazývame ho *beacon shard*. Tento shard generuje náhodné číslo a rozdeľuje na základe neho hlasovacie lístky do shardov (viď 4.2.4). Beacon shard taktiež prijíma tokeny na hlasovanie v ďalšej epoche (viď 4.2.2).

Harmony umožňuje komunikáciu medzi shardami. Každá taká komunikácie predstavuje broadcast na úrovni celej siete. Pomocou tejto komunikácie si môžu užívatelia presúvať svoje zdroje medzi shardami.

Vždy keď niektorý shard potvrdí nový blok, zašle jeho hlavičku na beacon chain. Ten ju validuje a uloží do svôjho nového bloku. Nový blok v beacon chaine je následne pomocou broadcastu zaslaný všetkým ostatným shardom, ktoré si ho uložia. Vďaka tomu vedia jednotlivé shardy validovať transakcie z iných shardov.

4.3 Model motivácie

Za každý nový blok v reťazci sú odmenený všetci validátori v podobe protokolom definovaného počtu tokenov. Každý validátor dostane podiel z celkovej odmeny priamo úmerný množstvu jeho hlasovacích lístkov v danom sharde. Úplne rovnakým spôsobom sú rozdelené aj poplatky za transakcie, ktoré boli v danom bloku vložené.

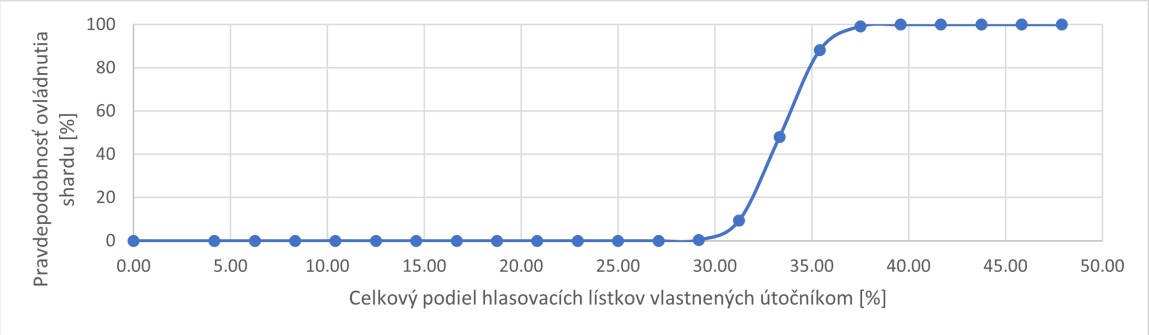
Na druhej strane, zlomyseľné chovanie je potrestané odobraním časti vlastnených tokenov. Ak bude dokázané, že validátor podpísal neplatný blok (podpis na dvoch konfliktných blokoch súčasne), budú mu odobrané všetky zdroje v danom sharde.

4.4 Teoretická analýza

V tejto sekcii teoreticky zanalyzujeme protokol Harmony z hľadiska bezpečnosti a výkonnosti.

4.4.1 FBFT konsenzus

FBFT konsenzus je obdobou PBFT a teda platí, že aby sieť fungovala, musí v nej byť menej ako $\frac{1}{3}$ škodlivých uzlov. Namiesto hlasovania broadcastom sa používa prahový digitálny podpis čo je kryptograficky bezpečný spôsob hlasovania. Tento konsenzus protokol teda



Obr. 4.3: Pravdepodobnosť ovládnutia jedného shardu útočníkom.

neznižuje bezpečnosť a zároveň zvyšuje výkonnosť keďže znižuje časovú zložitosť z $O(n^2)$ na $O(n)$.

4.4.2 Sharding

Na rozdelenie do shardov sa používa lotéria, ktorá je postavená na kryptograficky bezpečnom algoritme VRF. Náhodne rozdelenie do shardov je všeobecne najbezpečnejší spôsob, ktorý efektívne bráni útoku na podskupinu uzlov (viď sekcia 3.5.5). Navyše sa do shardov nerozdeľujú uzly ale samotné hlasovacie lístky. Vďaka zaručenej náhodnosti rozdelenia hlasovacích lístkov do shardov je možné určiť pravdepodobnosť ovládnutia shardu pomocou distribučnej funkcie kumulatívneho hypergeometrického rozdelenia $H(N, K, n, k)$ [33, 25], kde:

- N je celkové množstvo hlasovacích lístkov,
- $K = \frac{N}{3}$ je maximálne množstvo škodlivých hlasovacích lístkov
- $n = \frac{N}{n_{shard}}$ je množstvo lístkov v každom sharde, kde $n_{shard} = 4$ je počet shardov,
- $k = \frac{K}{n_{shard}}$ je množstvo škodlivých lístkov v jednom sharde.

Obrázok 4.3 ukazuje pravdepodobnosť ovládnutia jedného shardu v závislosti na celkovom podiele hlasovacích lístkov vlastnených útočníkom. Tento výpočet je založený na už spomínanom hypergeometrickom rozdelení a parametroch BC Harmony [25]. Môžeme vidieť, že útočník má zanedbateľnú pravdepodobnosť ovládnuť shard pokiaľ nevlastní približne $\frac{1}{3}$ z celkového množstva hlasovacích lístkov. S toho plynie, že shardingu v tomto protokole neumožňuje útočníkovi využiť útok na podskupinu uzlov (viď sekcia 3.5.5).

Harmony sharding používa algoritmus VDF na oneskorenie odhalenia náhodného čísla.

4.4.3 Komunikácie medzi shardami

Autori deklarujú vysokú škálovateľnosť pomocou shardingu. Avšak samotné shardy sú všetky úzko závislé na beacon sharde z dôvodu medzishardovej komunikácie. Každý nový blok v ľubovoľnom sharde musí byť zaslaný do beacon shardu. Ten musí jeho hlavičku pomocou konsenzu uložiť do svôjho nového bloku a tento blok broadcastom poskytnúť všetkým ostatným shardom. Táto závislosť na beacon sharde môže potenciálne predstavovať výkonnostné úzke hrdlo. Aktuálne Harmony používa len 4 shardy a teda nie je overené ani jasné nakoľko bude tento protokol efektívny ak by pracoval s desiatkami až stovkami shardov.

4.4.4 Voľba vodcu FBFT

Roľa vodcu v protokole FBFT je dôležitá pretože, ak vodca nespolupracuje tak nie je možné vytvárať nové bloky. Voľba vodcu je skutočne náhodná a zároveň vodcom sa pravdepodobnejšie stávajú najväčší podielnici čo je z pohľadu PoS bezpečné (viď útok 3.6.2). Avšak, vodca shardy sa nemení počas celej epochy (približne 18,2 hodiny) a všetci útočníci vedia, ktorý uzol je vodcom. S tohoto hľadiska je možné počas celej epochy vykonávať DoS útok na vodcu, čím sa znemožní uzatvárať konsenzus (viď útok 3.6.3). Samotný protokol Harmony tento útok nerieši a ochrana vodcu by preto musela byť vykonaná na sieťovej vrstve napríklad pomocou firewallu.

Kapitola 5

Solana

Solana je BC technológia zameraná na vysokú priepustnosť transakcií. Klasická centralizovaná databáza dokáže mať priepustnosť až 710 000 TPS na gigabitovej sieti s priemernou veľkosťou transakcie 176 B [40]. Decentralizovaná databáza BC má dramaticky nižšiu priepustnosť. Tradičné PoW protokoly ako je Bitcoin (5 TPS) alebo Ethereum (15 TPS) dosahujú veľmi nízku priepustnosť [1]. V kapitole 4 bol popísaný PoS protokol Harmony, ktorý má už značne vyššiu priepustnosť (2 000 TPS) dosiahnutú pomocou konceptu sharding. Autori Solana projektu deklarujú, že ich architektúra môže teoreticky dosiahnuť rovnakú priepustnosť ako centralizovaná sieť (710 000 TPS) a to bez použitia konceptu sharding.

Solana poukazuje na to, že zníženie priepustnosti BC voči centralizovanej sieti je zapríčinené tým, že uzly v sieti potrebujú synchronizovať čas a súčasne si navzájom nemôžu dôverovať. Solana projekt navrhol kryptografický algoritmus *Proof-of-History* (viď sekcia 5.1) pomocou ktorého môže rýchlo a efektívne synchronizovať čas aj decentralizovaná sieť.

Solana konsenzus je založený na mechanizme PoS (sekcia 5.2) na ktorom je funguje aj model motivácie tohoto protokolu (sekcia 5.3). Sekcia 5.4 na záver teoreticky zhodnocuje tento protokol z hľadiska bezpečnosti a výkonnosti.

Ak nie je uvedené inak, všetky informácie v tejto kapitole vychádzajú s pôvodnej publikácie Solana [41] a oficiálnej dokumentácie tohoto projektu [40].

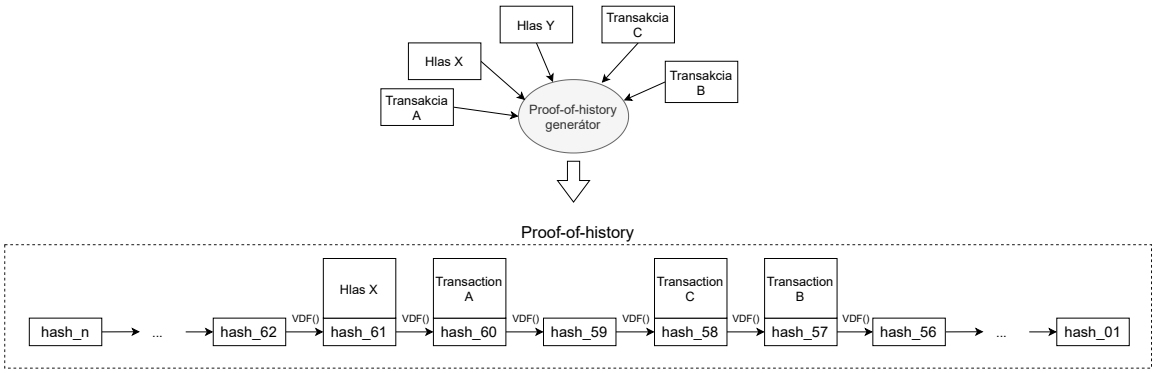
5.1 Proof-of-History

Proof-of-History (ďalej len PoH) je kryptografický algoritmus, ktorý usporiada dátové záznamy v čase tak ako vznikajú. Takýto algoritmus pomáha jednoznačne a nepopierateľne usporiadať transakcie a konsenzus hlasovania v čase pre decentralizovanú sieť ako je BC. Dôkaz o usporiadaní je založený na kryptografickej hash funkcii (pozri sekciu 2.4.1), ktorá spĺňa kritérium *collision resistant*.

Obrázok 5.1 demonštruje princíp fungovania PoH. V sieti vždy existuje jeden uzol, ktorý nazývame *PoH generátor*. Ten neustále (najrýchlejšie ako dokáže) pridáva nové záznamy do hash pointer zoznamu (pozri sekciu 2.4.2). Pridávanie nových hashov do zoznamu je sekvenčné a podmienené existenciou predchádzajúceho hashu. Tento princíp sa nazýva *Verifiable Delay Function [9]*. Každý hash teda predstavuje časový bod a vzdialenosť dvoch hashov udáva časový interval.

Generátor je aktuálny vodca v konsenzus protokole. Všetky transakcie a hlasy od rôznych klientov sú zasielané na generátor. Ten ich serializuje do PoH reťazca. Vždy keď

[1] https://academy.binance.com/en/glossary/transactions-per-second-tps



Obr. 5.1: Reprezentácia času v decentralizovanej sieti pomocou proof-of-history.

generátor prijme dáta od klienta (transakciu alebo hlas) pridá ich do zoznamu. Ak práve nemá dáta tak generuje prázdny záznam. Na obrázku môžeme vidieť v ktorých hashoch, alebo tiež časových momentoch, bola transakcia pridaná do reťazca.

Generátor priebežne rozosiela PoH sekvenciu do celej siete. Každý uzol potom môže jednoznačne určiť v akom čase boli vygenerované transakcie a hlasy konsenzu. Synchronizácia v BC Solana je založená na tomto koncepte.

Existuje teoretický útok pri ktorom útočník zmení poradie v reťazci. Predpokladajme, že útočník pozná všetky transakcie a hlasy a má väčší výpočtový výkon než aktuálny generátor. Potom by dokázal vypočítať reťazec s alternatívnym usporiadaním a publikovať ho skôr. Práve kvôli tomuto generátor reťazec vždy podpíše a až potom distribuuje (pozri sekciu 2.4.3).

5.2 Konsenzus

Solana navrhla PoS mechanizmus, ktorý vychádza s protokolu PBFT (pozri sekcii 3.4.1). Konsenzus sa používa na zvolenie PoH generátoru (sekcia 5.2.1), hlasovanie o validnosti PoH sekvencie (sekcia 5.2.2) a je na ňom založný aj model motivácie (odmeňovanie a tresty v sekcii 5.3).

5.2.1 Voľba PoH generátoru (vodcu)

Solana sieť sa rozdeľuje do tzv. zhlukov. Zhluk je množina validátorov, ktoré spolupracujú na konsenze. Zhluky môžu mať prieniky. Každý zhluk má vodcu, ktorý generuje záznamy v PoH. Vodca sa mení po fixnom počte vygenerovaných záznamov. Tento interval voláme *slot*. Autori protokolu tvrdia, že jeden slot je približne 400 ms.

Nový vodca sa určí na základe už dopredu stanoveného rozvrhu. Rozvrh je stanovený na fixný počet slotov, ktorý voláme *epocha*. Rozvrh na aktuálnu epochu sa vypočíta už v predchádzajúcej. Samotný výpočet rozvrhu prebieha tak, že sa zoberú všetci validátori, ktorý hlasovali v posledných n záznamoch PoH sekvencie, kde n je hodnota stanovená zhlukom. Títo validátori dostanú v rozvrhu pridelené vodcovské sloty na základe váhy ich podielu. Inak povedané, validátor s najväčším podielom bude v nasledujúcej epoche najčastejšie vodcom (princíp PoS).

5.2.2 Tower BFT

Solana navrhla konsenzus protokol *Tower BFT*, ktorý je založený na PBFT protokole:

1. Klienti siete zasielajú transakcie na aktuálneho vodcu (PoH generátor).
2. Vodca pridáva transakcie do PoH sekvencie a túto dátovú štruktúru distribuuje validátorom.
3. Validátori overia transakcie v PoH sekvencii a vykonajú ich v stanovenom poradí. Takto získajú nový stav.
4. V špecifikovanom čase, po uplynutí určitého počtu PoH záznamov, pošlú validátori svoje hlasy vodcovi. Hlas je hash stavu validátora po vykonaní transakcií s danej PoH sekvencie. Tento hash je navyše podpísaný privátnym kľúčom validátora. Takto validátor potvrdzuje, že súhlasí s PoH sekvenciou ktorú podpísal. Ak zo sekveciou nesúhlasí, jednoducho svôj hlas nepošle.
5. Vodca pridáva prijaté hlasy do PoH sekvencie.
6. Validátori sledujú hlasy, ktoré sa následne vyskytujú v PoH sekvencii. Každý validátor počíta hlasy a ak dosahujú $\frac{2}{3}$ celkového počtu validátorov, tak bol dosiahnutý konsenzus.

Solana používa PoS a teda váha hlasu validátora je priamo úmerná jeho podielu. Pre konsenzus nie je nutné dosiahnuť $\frac{2}{3}$ celkového počtu hlasov validátorov, ale len toľko hlasov aby ich vlastníci vlastnili $\frac{2}{3}$ celkového podielu.

5.2.3 Vetvenie

Uzly v Solana sieti tolerujú dočasnú stratu spojenia s aktuálnym vodcom (PoH generátorom). Pri strate spojenia predpokladajú len prázdne záznamy, ktoré neobsahujú žiadne dáta. Takýto PoH reťazec si dokážu generovať aj sami, bez komunikácie zo sieťou, na základe posledného hashu minulého slotu.

Vetvenie v Solana konsenze vzniká pri zmene vodcu. Nový vodca nemusel zachytiť hlasy uložené v poslednom slote. Tento slot teda nahradí prázdnymi záznamami. Vetvenie slotu je teda binárne (vetva s dátami a prázdna vetva).

Každý validátor musí sledovať celý binárny strom možných vetiev a uchovávať si ich stav. Voľbu vetvy urobí validátor tak, že vodcovi v danej vetve pošle svoj hlas čím vetvu potvrdí. Validátor potom nemôže hlasovať v žiadnej inej vetve po dobu (pevný počet slotov) ktorá sa nazýva *lockout*. Lockout perióda sa navyše zdvojnásobuje každým ďalším pridaným hlasom v danej vetve až do maximálnej hodnoty (aktuálne je maximum 32 hlasov). Ak chce validátor hlasovať, za inú vetvu, musí počkať do konca periódy lockout. Následovne vykoná návrat späť (anglicky *rollback*) do posledného spoločného stavu medzi starou a novou vetvou. Fungovanie a význam lockoutu a rollbacku je podrobnejšie popísaný v sekcii 5.3.1.

5.3 Model motivácie

Model motivácie je založený na mechanizme PoS. Validátori sú odmeňovaný podielom na zdrojoch BC za správanie, ktoré je pre sieť vhodné. Naopak, nevhodné správanie je penalizované odobraním zdrojov v sieti.

5.3.1 Vetvenie

Hlasy v každej vetve sa ukladajú do pomyselného zoznamu s obmedzenou kapacitou na 32 položiek. Každý nový pridaný hlas do zoznamu zdvojnásobí lockout všetkých predošlých hlasov v zozname. Keď zoznam dosiahne maximálnu kapacitu tak je najstarší hlas odstránený (FIFO) a majiteľ hlasu je odmenený.

Ešte pred tým ako je pridaný nový hlas do zoznamu tak je vykonaný *rollback*. Rollback porovná časový slot pridania nového hlasu s časovými slotmi v ktorých končí lockout každého hlasu v zozname. Ak niektorému hlasu skončil lockout v slote, ktorý je skôr ako slot pridania nového hlasu tak je tento starší hlas odstránený zo zoznamu (LIFO).

S toho vyplýva, že čím viac je vetva používaná, tím pravdepodobnejšie dosiahne ľubovoľný hlas koniec FIFO zoznamu a s tým spojenú odmenu. Validátor ktorý chce maximalizovať svoje zisky je takýmto odmeňovaním motivovaný vybrať vetvu, ktorá je najpoužívanejšia a najstabilnejšia.

Validátor je potrestaný za súbežné hlasovanie v rôznych vetvách. Ak hlasoval v jednej vetve, tak je zaviazaný na hlasovanie len v tejto vetve po dobu lockoutu. Ak validátor poruší lockout a bude mu to preukázané, budú mu odobrané jeho zdroje.

5.4 Teoretická analýza

5.4.1 Proof-of-history

Why Proof-of-History is not (yet) the consensus of the future? [2]

5.4.2 Konsenzus

Autori tvrdia, že sa nepoužíva sharding. Avšak v dokumentácii je veľmi vágne povedané, že uzly nepracujú globálne ale v nejakých zhlukoch. Ako takéto zhluky fungujú nie je jasné. Očividne však ide o sharding.

[2] https://anycoindirect.eu/en/blog/what-is-proof-of-history

Kapitola 6

Ouroboros

Ouroboros je PoS protokol, ktorý je používaný v kryptomene Cardano. Autori Cardano deklarujú [1], že všetky technológie, protokoly a algoritmy použité v ich BC sú vyberané koncepciou recenzného hodnotenia. Inak povedané, Cardano používa technológie, ktoré boli vedecky prezentované a kladne prijaté komunitou okolo technológie BC. V zmysle tohoto konceptu bol ako PoS protokol zvolený práve Ouroboros. V tejto kapitole je vysvetlený princíp fungovania protokolu Ouroboros na základe oficiálnej publikácie tohoto projektu [24].

6.1 Konsenzus

Ouroboros konsenzus je uznesený ak sa viac ako 50 % (majorita vlastníkov podielu) zhodne na stave BC.

6.1.1 Synchronizácia v čase

Protokol Ouroboros pracuje s časom v dvoch jednotkách. Najmenšia časová jednotka je *slot*. Za jeden slot je vytvorený maximálne jeden blok (čas potrebný na komunikáciu v decentralizovanej peer-to-peer sieti). Slot je najmenšia časová jednotka na synchronizáciu uzlov v sieti.

Druhou (väčšou) časovou jednotkou je *epocha*, ktorú tvorí pevný počet slotov R. Pre Cardano [2] je R rovné 432 000 slotom (približne päť dní).

6.1.2 Protokol

Pre každý slot je určený vodca podľa dopredu známeho rozvrhu. Proces určenia rozvrhu vodcov je popísaný v sekcii 6.2. Vodca môže ako jediný vytvoriť blok v danom slote. Blok obsahuje sekvenčné číslo slotu a je podpísaný privátnym kľúčom jeho tvorcu. Overenie validnosti bloku prebieha na základe čísla slotu zapísaného v hlavičke tohoto bloku. Každý validátor sa pozrie do rozvrhu a určí kto má byť vodca pre daný slot. Následne overí, že blok je podpísaný práve týmto vodcom.

Každý uživateľ vykonáva nasledujúce činnosti:

- Zber validných transakcií zo siete.

[1] https://www.youtube.com/watch?v=Ja9D0kpksxw

[2] https://developers.cardano.org/docs/stake-pool-course/introduction-to-cardano/

- Zber všetkých BC, ktoré sú distribuované sieťou a kontrola ich validity (sekvencia blokov má len rastúce sekvenčné čísla slotov, transakcie sú validné, blok je vytvorený správnym vodcom). Každý udržiava najdlhší validný BC (rovnaký princíp ako Bitcoin).

- Ak je užívateľ aktuálny vodca, tak vytvorí nový blok z transakcií ktoré získal. Blok pridá na koniec BC a distribuuje ho. Je dôležité podotknúť, že vodca nemusí vždy publikovať nový blok a teda slot môže byť "prázdny". Avšak jeden slot môže publikovať maximálne jeden blok.

6.1.3 Schéma delegovania

Ouroboros protokol vyžaduje aby bol vlastník podielu neustále online, ak sa chce podieľať na tvorbe nových blokov. Táto požiadavka je príliš nepraktická a nerealistická. Preto Ouroboros poskytuje schému delegovania práva na generovanie blokov. Ak je vlastník zvolený za vodcu slotu, môže tento post delegovať inému užívateľovi (delegátovi). Tento užívateľ teda vytvorí nový blok namiesto skutočného vodcu slotu. Avšak, vlastník podielu deleguje len svoje právo na tvorbu bloku. Delegát nemá možnosť manipulovať zo zdrojmi, ktoré nie sú jeho. Koncept delegovania umožňuje zoskupovať podielnikov do väčších celkov (anglicky *stake pools*).

Delegovanie je zabezpečené pomocou kryptografického konceptu *Proxy Signatures* [8]. Skutočný vodca slotu si vytvorí kľúč na delegovanie digitálneho podpisu (anglicky *proxy signing key*), ktorým delegát podpíše vytvorený blok. Proxy kľúč kryptograficky zabezpečuje, že ktokoľvek môže overiť nasledujúce jeho vlastnosti:

- Verifikácia: Identita tvorcu proxy kľúča a identita delegáta, ktorému bol vystavený.

- Prevencia pred zneužitím: Tento kľúč je časovo limitovaný pretože jeho tvorca definuje presný slot v ktorom platnosť kľúča expiruje.

6.2 Vodca slotu

Rozvrh vodcov pre jednotlivé sloty sa vytvára na obdobie jednej epochy a je známy už na jej začiatku. Každý užívateľ si môže vypočítať vodcov $(U_1, U_2, ..., U_R)$ pre jednotlivé sloty $(1, 2, ..., R)$ v aktuálnej epoche pomocou funkcie $L(stake, rnd)$, kde:

- *stake*: Rozdelenie podielu (anglicky *stake distribution*) v predchádzajúcej epoche. Rozdelenie podielu sa vezme z k-tého slotu v predchádzajúcej epoche, kde k je parametricky nastaviteľná konštanta.

- *rnd*: Dostatočne dlhý a skutočne náhodný reťazec $rnd \in \{0, 1\}^*$ vytvorený pomocou distribuovanej generácie náhodnosti v minulej epoche. Aby bol reťazec považovaný za skutočne náhodný, musí byť produktom kryptograficky bezpečného výpočtu všetkých zainteresovaných strán. Ouroboros používa pre tento účel kombináciu algoritmov *Coin Tossing* (viď 6.2.1) a PVSS (viď 6.2.2).

Funkcia L z náhodného reťazca *rnd* pomocou deterministického algoritmu určí rozvrh pričom pravdepodobnosť zvolenia za vodcu je priamo úmerná podielu daného užívateľa. Pri voľbe vodcu sa teda uplatní PoS princíp (napríklad, ak v predchádzajúcej epoche vlastní užívateľ 2 % podielu tak má v aktuálnej epoche práve 2 % šancu stať sa vodcom slotu).

6.2.1 Coin Tossing protokol

Coin Tossing [7] protokol umožňuje dvom stranám (uvažujme Alicu a Boba) vytvoriť rovnomerne náhodný reťazec. Protokol funguje nasledovne:

1. Alica vygeneruje náhodný reťazec $u_1 \in \{0,1\}^*$ a navzorkuje z neho náhodnosť r. Následne pošle Bobovi dôkaz $Commit(r, u_1)$ tejto hodnoty, ktorý ju však neodhaľuje.

2. Bob vygeneruje svoj náhodný reťazec $u_2 \in \{0,1\}^*$ a pošle ho Alici.

3. Alica odhalí u_1 zaslaním kryptografického dôkazu $Open(r, u_1)$. Bob si môže overiť, že $Open(r, u_1)$ odpovedá $Commit(r, u_1)$ a teda Alica určite nepozmenila pôvodné u_1.

4. Obe strany vypočítajú výstupnú náhodnosť $u = u_1 \oplus u_2$.

Ouroboros používa pre vytvorenie náhodného reťazca rozšírenie protokolu *Coin Tossing*, ktoré uvažuje viac ako dve strany a to pomocou algoritmu PVSS (viď sekcia 6.2.2).

6.2.2 PVSS

PVSS (*Publicly Verifiable Secret Sharing*) [17] je kryptografická schéma, ktorá umožňuje rozdeliť tajomstvo σ na n dielov. Pôvodné σ je možné zrekonštruovať pokiaľ je poškodených maximálne t dielov (nedostupných alebo úmyselne pozmenených útočníkom), kde t je nastaviteľný parameter. Schéma definuje nasledujúce dve funkcie:

- $Deal(n, t, \sigma) = (\sigma_1, \sigma_2, ..., \sigma_n)$, kde vstupy n, t a σ sú počet dielov na vygenerovanie, maximálny počet poškodených dielov a vstupné tajomstvo. Výstupom funkcie sú diely tajomstva.

- $Rec(\sigma_1, \sigma_2, ..., \sigma_n) = \sigma$, kde vstupom sú diely vytvorené funkciou *Deal* a výstupom je rekonštrukcia pôvodného tajomstva σ ak platí, že maximálne t vstupných dielov je poškodených.

6.2.3 Protokol pre generovanie distribuovanej náhodnosti

Nech dĺžka epochy e_j je $R = 10k$ slotov a vodcovia slotov $1, 2, ..., R$ sú $U_1, U_2, ..., U_R$ (vodcovia nemusia byť nutne rôzny). Potom protokol pre distribuované generovanie náhodnosti pre epochu e_{j+1} je definovaný následovne:

1. Fáza záväzku: Ide o prvých $4k$ slotov. Každý vlastník podielu U_i, pre $1 \leq i \leq R$, vygeneruje náhodný reťazec u_i a navzorkuje z neho náhodnosť r_i. Následne rozdelí tajomstvo u_i na diely $\sigma_1, \sigma_2, ..., \sigma_R$ pomocou $Deal(R, R/2, u_i)$. Každé σ_j, pre $1 \leq j \leq R$, zašifruje verejným kľúčom užívateľa U_j. Vlastník podielu U_i uloží do BC zašifrované diely tajomstva a zároveň kryptografický dôkaz $Commit(r_i, u_i)$. Všimnime si, že t vo funkcii *Deal* je nastavené na $R/2$. To znamená, že na rekonštrukciu tajomstva je potrebná majorita vlastníkov podielu.

2. Fáza odhalenia: Po uplynutí $4k$ slotov, každý účastník odstráni posledných k slotov a vo zvyšku identifikuje držiteľov podielu. Ak majorita držiteľov podielu publikovala *Commit*, tak začína fáza odhalenia. V opačnom prípade protokol zastaví. Následne každý vlastník podielu U_i, pre $1 < i \leq R$, odhalí tajomstvo u_i tým, že vloží do BC $Open(r_i, u_i)$.

3. Fáza obnovy: Po uplynutí $8k$ slotov, každý účastník odstráni posledných k slotov a vo zvyšku identifikuje všetkých držiteľov podielu, ktorý odhalili ich tajomstvo u_i (overenie $Commit(r_i, u_i)$ voči $Open(r_i, u_i)$). Ak nejaký nepoctivý účastník U_x nezverejnil $Open$ k svojmu $Commit$, tak všetci poctivý účastníci zverejnia $\sigma_1, \sigma_2, ..., \sigma_n$, ktoré patria k tajomstvu u_x. Ak bude poctivých účastníkov viac ako t (majorita), tak bude možné použiť $Rec(\sigma_1, \sigma_2, ..., \sigma_n)$ na rekonštrukciu tajomstva u_x.

4. Nová epocha: Každé tajomstvo u_i, pre $1 \leq i \leq R$, je teraz už verejne známe. Výsledná náhodnosť pre epochu e_{j+1} vypočíta každý účastník ako $u_1 \oplus u_2 \oplus ... \oplus u_R$.

6.3 Model motivácie

Ouroboros motivuje účastníkov protokolu chovať sa poctivo pomocou modelu odmeňovania. Transakcie zahrnuté do blokov obsahujú poplatky (anglicky *transaction fee*), ktoré tvorcovia transakcií musia zaplatiť podielnikom. Samotné odmeny sa počítajú a vyplácajú za obdobie jednej epochy.

Celkový fond odmien T_{all} pre epochu e_j je daný rovnicou 6.1, kde R je počet slotov v epoche a T_i sú všetky poplatky za transakcie v slote i. Ak sa ľubovoľná transakcia vyskytne vo viacerých blokoch, tak je poplatok uvažovaný len pri jej prvom výskyte.

$$T_{all} = \sum_{i=1}^{R} \sum_{k \in T_i} t_k \tag{6.1}$$

V epoche e_j uvažujme vodcov $U_1, U_2, ..., U_R$ pre sloty $1, 2, ..., R$ a všetkých podielnikov označme ako množinu $\mathbf{P}$. Potom i-tý podielnik $p_i \in \mathbf{P}$ dostane za epochu e_j odmenu α_i určenú rovnicou 6.2.

$$\alpha_i = \frac{|\{k \mid p_i = U_k\}|}{R} T_{all} \tag{6.2}$$

Odmena podielnika teda nevychádza z transakcií, ktoré sú pridané do BC v jeho vodcovských slotoch, ale je vždy priamo úmerná jeho podielu. Tento model motivácie teda odmeňuje podielnikov nie za vytváranie blokov, ale za samotné vlastníctvo podielu.

6.4 Teoretická analýza

6.4.1 Vetvenie

Kapitola 7

Prehľad existujúcich simulátorov

Pre simuláciu zvolených PoS protokolov nebude vytvorený úplne nový simulátor. Simulácia celej technológie BC s dostatočne presným model je rozsiahla úloha nad rámec tejto práce. Navyše už boli vytvorené simulačné nástroje, ktoré simulujú rôzne BC protokoly. K týmto simulátorom existujú práce ktoré experimentálne vyhodnotili ich presnosť (viď[32, 16]). Preto bude použitý jeden s týchto nástrojov. Zvolený nástroj bude následne upravený a rozšírený tak aby bolo možné simulovať zvolené PoS protokoly.

Táto kapitola analyzuje niekoľko aktuálne dostupných simulátorov BC. V závere kapitoly je vykonané porovnanie najvhodnejších nástrojov a je zvolený jeden pre ďalšiu prácu.

7.1 SimBlock

SimBlock je simulátor založený na diskrétnej simulácii. Je implementovaný v programovacom jazyku Java a jeho zdrojový kód je voľne dostupný (anglicky *open source*) [1]. Simulátor podporuje PoW protokoly Bitcoin, Litecoin, a Dogecoin. Avšak umožňuje aj modulárne zmeniť konsenzus protokol. Posledná stabilná verzia pridala jednoduchú implementáciu PoS konsenzu [2].

Autori tohoto projektu v rámci vyhodnocovania presnosti simulátora vykonali experiment ktorý porovnal ich prácu s podobným už existujúcim simulátorom. Obe simulácie spustili s rovnakými parametrami a to pre protokoly Bitcoin, Litecoin, a Dogecoin. Výsledky oboch simulátorov boli veľmi podobné. Na základe tohoto experimentu autori zhodnotili, že ich simulátor napodobuje reálne BC systémy s porovnateľnou presnosťou ako podobné simulátory.

Autori ďalej navrhli úpravu algoritmu na voľbu susedných uzlov (anglicky *neighbor node selection algorithm*) v protokole Bitcoin. Navrhnutú úpravu odsimulovali a vyhodnotili, že ich vylepšenie algoritmu zvyšuje priepustnosť transakcií. Touto simuláciou bol demonštrovaný význam tohoto simulačného nástroja a to je zlepšovanie BC protokolov z hľadiska výkonnosti a bezpečnosti pomocou simulácie. [5]

Výhodou tohoto simulačného nástroj je schopnosť simulovať aj rozsiahle siete s viac ako 10 000 uzlami. Ďalšou výhodou je, že simulácia ustanovuje medzi uzlami priame spojenie (anglicky *point-to-point*) uvažujúce geografickú lokalitu jednotlivých uzlov. Simulácia teda umožnuje zohľadňovať sieťové oneskorenie (anglicky *latency*) a šírku pásma (anglicky *bandwidth*).

[1] Apache License, Version 2.0

[2] https://github.com/dsg-titech/simblock/releases/tag/v0.8.0

Na druhú stranu, simulátor predpokladá, že všetky uzly sú poctivé. V aktuálnej implementácii teda neumožňuje experimentovanie so zlomyseľným správaním niektorých uzlov. Pre analýzu útokov ako je double-spending alebo selfish-mining by bolo potrebné tento simulačný nástroj rozšíriť. Ďalšou možnou nevýhodou je, že simulácia prebieha na úrovni blokov a propagácia transakcií zatiaľ nie je uvažovaná. [16]

7.2 Bitcoin Simulator

Bitcoin Simulator je open source [3] simulačný nástroj implementovaný v programovacom jazyku C++. Simulátor je postavený nad platformou NS-3 [4], ktorá slúži na diskrétnu simuláciu Internetových systémov. Na tejto platforme je teda postavená simulácia peer-to-peer siete BC. Tento simulátor je určený a bol vyvinutý na experimentovanie s PoW protokolmi. Aktuálne podporuje protokoly Bitcoin, Litecoin, a Dogecoin. Simulátor teda nepodporuje PoS konsenzus. Avšak v minulosti už bol tento nástroj použitý treťou stranou, ktorá rozšírila implementáciu o PoS protokoly Algorand, Casper FFG a Gasper (viď [11]).

Autori simulátoru vykonali experiment na vyhodnotenie presnosti ich nástroja. Tri vyššie spomenuté protokoly boli simulované na rozsiahlej sieti a boli namerané mediány propagačného času bloku. Mediány získané zo simulácie boli relatívne podobné mediánom skutočných sietí postavených na týchto troch protokoloch. S tohoto hľadiska bol simulátor vyhodnotený ako pomerne presný. [18]

Medzi výhody tohoto simulačného nástroja patrí jeho rozsiahlosť, ktorá poskytuje veľkú škálu vstupných parametrov a taktiež veľké množstvo nameraných výstupných metrík. Z hľadiska simulácie PoW protokolov umožňuje simulátor rozlišovať rôzne typy uzlov (bežný uzol a miner uzol). Rozlišovanie rôznych uzlov umožňuje tomuto nástroju simulovať aj zlomyseľné chovanie niektorých uzlov. Tento simulátor teda umožňuje analyzovať aj bezpečnosť PoW konsenzu z hľadiska ťažby blokov (útoky selfish-mining a double-spending). [16]

7.3 BlockSim

BlockSim je open source [5] simulátor vytvorený v programovacom jazyku Python3 určený na diskrétnu simuláciu BC. Autori definujú tri základné ciele tohoto simulátora: všeobecnosť, rozširovateľnosť a jednoduchosť. Všeobecnosť architektúry simulátora umožňuje simuláciu veľkého množstva BC systémov. Simulátor by malo byť možné jednoducho rozšíriť o ďalšie protokoly. Oba predošlé ciele majú viesť k tomu aby bol tento nástroj jednoduchý na použitie.

Simulátor poskytuje všeobecnú abstrakciu BC, ktorú autori nazvali *base model*. Base model je zdieľaná vrstva pre všetky konkrétne protokoly pretože pokrýva všetky základné prvky BC (uzly, transakcie, bloky, reťazec blokov, fork reťazca). Nad touto vrstvou je potom možné vytvoriť implementáciu konkrétnych protokolov. Autori demonštrovali túto flexibilnú architektúru tak, že vytvorili nad base modelom simuláciu dvoch PoW protokolov (Bitcoin a Ethereum). Simulátor by teda mal byť priamočiaro rozšíriteľný aj o simuláciu PoS protokolov. Avšak takáto simulácia by neumožnila analýzu špecifickej sekvencie správ PoS konsenzu. [3]

[3] https://arthurgervais.github.io/Bitcoin-Simulator/index.html
[4] https://www.nsnam.org/
[5] https://github.com/maher243/BlockSim

Najväčšou výhodou tohoto simulátoru je jeho jednoduchá architektúra, ktorá umožňuje rozšíriteľnosť o ďalšie BC protokoly. Simulátor pokrýva sieťovú, dátovú a konsenzus vrstvu. Z hľadiska sieťovej vrstvy je možné simulovať oneskorenie a šírku pásma pomocou geografickej distribúcie uzlov siete. Ďalšou výhodou je pomerne veľké množstvo konfigurovateľných vstupných parametrov simulácie.

Naopak nevýhodu simulátoru je, že nedokáže efektívne simulovať rozsiahle siete. Tento simulátor sice podporuje simulácie transakcií avšak tento proces je limitovaný, keďže implementácia neobsahuje žiaden model účtov (napríklad UTXO [6] pre Bitcoin). [16]

7.4 VIBES

VIBES je open source [7] nástroj pre simuláciu BC technológie. Simulátor implementuje výhradne PoW konsenzus. Podľa autora je možná implementáciu rozšíriť o PS konsenzus. Tento simulátor je zameraný na všeobecnejšiu simuláciu BC v peer-to-peer sieti a neobsahuje implementáciu žiadneho konkrétneho protokolu. [38]

Výhodu tohoto nástroja je možnosť simulácie škodlivých uzlov. Simulátor je teda možné použiť na bezpečnostnú analýzu konsenzus protokolov. Ďalej, je podporovaná simulácia na úrovni dát (transakcie). Avšak rovnako ako pri nástroji BlockSim (viď sekcia 7.3), transakcie neobsahujú žiaden model účtov.

Nevýhodou tohoto nástroja je, že používa konštantné oneskorenie (anglicky *delay*) propagácie blokov a transakcií v sieti. Dôsledkom je nepresnosť simulácie z hľadiska výkonnostnej analýzy BC. Autor tvrdí, že simulátor umožňuje simuláciu rozsiahlych sietí s viac ako 10 000 uzlami. Avšak v skutočnosti je takáto simulácia pomerne náročná pretože tento nástroj používa centrálny koordinátor, ktorý tvorí úzke hrdlo (anglicky *botleneck*) simulácie.[16]

7.5 Shadow

Shadow [8] je paralelný diskrétny simulátor, ktorý umožňuje beh reálnych aplikácií ako doplnkov. Tento nástroj teda simuluje vrstvu sieťovej komunikácie a na samotných uzloch siete spúšťa reálnu aplikáciu. Samotnú aplikáciu je pritom nutné modifikovať len minimálne. Takýmto spôsobom je pomocou tohoto nástroja simulovaná napríklad sieť Tor. [23]

Nad týmto simulátorom bol vytvorený doplnok, ktorý umožňuje priamo spustiť implementáciu referenčného klienta Bitcoinu. Vďaka rôznym optimalizáciám umožňuje tento doplnok spustiť tisíce takýchto Bitcoin uzlov na jedinom stroji.

Nespornou výhodou tohoto nástroja je presná simulácia, keďže implementácia nevytvára žiadnu abstrakciu Bitcoinu, ale spúšťa jeho natívnu implementáciu. Tento nástroj je vhodný na štúdium jemných rozdielov softvéru distribuovaného systému.

Nevýhodou je nízka flexibilita tohoto nástroja. Spomínaný doplnok je použiteľný jedine pre simuláciu protokolu Bitcoin. Na simuláciu ľubovoľného iného protokolu by bolo potrebné vytvoriť nový doplnok. [28]

[6] https://river.com/learn/terms/u/unspent-transaction-output-utxo/
[7] https://github.com/i13-msrg/vibes
[8] https://shadow.github.io/

7.6 FoBSim

FoBSim je open source [9] simulátor, ktorého model pokrýva dve oblasti:

- Fog Computing: Je fyzický lebo virtuálny zdroj vložený medzi koncového užívateľa a tradičné dátové centrum (anglicky *cloud*). Toto rozšírenie cloudu má zvýšiť efektivitu a bezpečnosť. Táto služba využíva tradične centrálnu autoritu.
- Blockchain: Technológia BC integrovaná s fog computing by mala viesť k decentralizácii tejto služby.

Rozoberať prečo je snaha integrovať tieto dve služby do jednej nie je predmetom tejto práca. Avšak tento simulačný nástroj je vyvíjaný práve za účelom experimentovania v tejto oblasti. Pre našu prácu je zaujímavá len časť, ktorá simuluje BC.

Simulátor poskytuje model BC, ktorý obsahuje nasledujúcu funkcionalitu: sieťová vrstva, konsenzus algoritmy, odmeňovanie užívateľov za vytváranie nových blokov (anglicky *incentive mechanism*), paralelná ťažba blokov, distribúcia správ v sieti. [6]

Autori deklarujú, že simulátor implementuje tri konsenzus protokoly: PoW, PoS a Proof-of-Authority. Avšak nikde nedefinujú o aké konkrétne protokoly ide. Presnosť simulácie je nejasná keďže ide o nový projekt a neexistuje zatiaľ žiadna práca, ktorá by sa zaoberala jeho vyhodnocovaním. Samotná implementácia predstavuje asi 3 000 riadkov kódu v pythone [10] čo je dramaticky menej voči iným nástrojom.

7.7 Zhodnotenie

Táto sekcia zhodnocuje simulačné nástroje popísané v rámci tejto kapitoly. Z týchto nástrojov sú porovnané vlastnosti troch najvhodnejší. Na základe týchto vlastností je na záver zvolený simulátor, ktorý bude v rámci tejto práce rozšírený o simuláciu požadovaných PoS protokolov.

7.7.1 Porovnanie najvhodnejších nástrojov

Pre potreby tejto práce definujeme niekoľko požiadaviek na simulačný nastroj ktorý použijeme. V prvom rade požadujeme dostupnosť zdrojového kódu s licenciou open source. Ďalej požadujeme aby simulátor vykazoval relatívne dostatočnú presnosť v porovnaní s reálnou BC sieťou. Dostatočnou presnosťou z hľadiska našej práce sa myslí, že simulátor vierohodne napodobuje sieťovú a konsenzuálnu vrstvu. Zaoberáme sa analýzou konsenzus protokolov a teda dátová vrstva už nie je pre simuláciu kľúčová.

Tieto požiadavky spĺňajú v najväčšej miere tri simulačné nástroje: SimBlock, Bitcoin Simulator a BlockSim. Tabuľka 7.1 zhrňuje vlastnosti týchto troch simulátorov na sieťovej, konsenzus a dátovej vrstve. Na základe tohoto zhodnotenia môžeme povedať, že tieto tri nástroje poskytuje podobne rozsiahle modely.

7.7.2 Výsledná voľba

Predošlá klasifikácia určila simulátory, ktoré majú podporu požadovaných vlastností pre potreby tejto práce. Avšak táto klasifikácia bola na vysokej úrovni a ignorovala komplexnosť

[9] GNU General Public License v3.0
[10] https://github.com/sed-szeged/FobSim

	Popis vlastnosti	Protokol		
		SimBlock	**Bitcoin Simulator**	**BlockSim**
Sieťová vrstva	Velkosť bloku	1	1	1
	Veľkosť transakcie	0	0	1
	Počet uzlov	1	1	1
	Geografická distribúcia uzlov	1	1	1
	Bandwith	1	1	1
	Latency	1	1	1
Konsenzus vrstva	PoW	1	1	1
	PoS	0	0	1
	Množstvo vygenerovaných blokov	1	1	1
Dátavá vrstva	Generovanie transakcií	0	0	1
	Vzťah transakcie k uzlu	0	0	0

Tabuľka 7.1: Matica podporovaných vlastností zvolených simulátorov na sieťovej, konsenzus a dátovej vrstve.

jednotlivých vrstiev. Napríklad sieťová vrstva nástroja SimBlock je značne komplexnejšia než simulácie sieťovej vrstvi nástroja BlockSim. Tabuľka 7.2 zhrnuje a porovnáva vlastnosti týchto troch simulátorov z hľadiska náročnosti rozšíriteľnosti pre potreby tejto práce. Ako môžeme vydieť, nástroj BlockSim má problém simulovať rozsiahle siete čo je veľkým nedostatkom. Bitcoin Simulator je zase projekt už 5 rokov nevyvýjaný. V tejto práci preto bude použitý nástroj SimBlock, ktorý bude rozšírený o požadovanú funkcionalitu.

	SimBlock	**Bitcoin Simulator**	**BlockSim**
Veľkosť simulovaných sietí	rozsiahle	rozsiahle	malé
Programovací jazyk	Java	C++	Python
Dostupnosť	open source	open source	open source
Vytvorenie projektu	Jún 2019	Apríl 2016	Apríl 2019
Posledná zmena v repozitári	Február 2021	Október 2016	Máj 2021
Podpora PoS	áno (s ukážkou)	nie	áno (bez ukážky)
Simulácia útočiacich uzlov	nie	áno	nie je jasné
Podporované protokoly	Bitcoin Litecoin Dogecoin	Bitcoin Litecoin Dogecoin	Bitcoin Ethereum

Tabuľka 7.2: Porovnanie vlastností troch najrozsiahlejších simulátorov.

Literatúra

[1] *Horizen Academy - Blockchain as a data structure* [https://academy.horizen.io/technology/expert/blockchain-as-a-data-structure/]. Accessed: 2021-06-03.

[2] Aberer, K., Alima, L., Ghodsi, A., Girdzijauskas, S., Haridi, S. et al. The Essence of P2P: A Reference Architecture for Overlay Networks. In:. Január 2005, s. 11– 20. DOI: 10.1109/P2P.2005.38. ISBN 0-7695-2376-5.

[3] Alharby, M. a Moorsel, A. van. BlockSim: An Extensible Simulation Tool for Blockchain Systems. *Frontiers in Blockchain*. 2020, zv. 3, s. 28. DOI: 10.3389/fbloc.2020.00028. ISSN 2624-7852. Dostupné z: https://www.frontiersin.org/article/10.3389/fbloc.2020.00028.

[4] Angelis, S. D., Aniello, L., Baldoni, R., Lombardi, F., Margheri, A. et al. PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain. In: *ITASEC*. 2018.

[5] Aoki, Y., Otsuki, K., Kaneko, T., Banno, R. a Shudo, K. SimBlock: A Blockchain Network Simulator. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2019, s. 325–329. DOI: 10.1109/INFCOMW.2019.8845253.

[6] Baniata, H. a Kertész, A. FoBSim: An extensible open-source simulation tool for integrated Fog-Blockchain systems. In:. Október 2020. DOI: 10.36227/techrxiv.13148390.v1.

[7] Blum, M. Coin Flipping by Telephone. In: *Advances in Cryptology: A Report on CRYPTO 81*. 1981, s. 11–15. Dostupné z: /archive/crypto81/11_blum.pdf.

[8] Boldyreva, A., Palacio, A. a Warinschi, B. Secure Proxy Signature Schemes for Delegation of Signing Rights. *Journal of Cryptology*. Jún 2003, zv. 25. DOI: 10.1007/s00145-010-9082-x.

[9] Boneh, D., Bonneau, J., Bünz, B. a Fisch, B. Verifiable delay functions. In: Springer. *Annual international cryptology conference*. 2018, s. 757–788.

[10] Boneh, D., Lynn, B. a Shacham, H. Short Signatures from the Weil Pairing. In: *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*. Berlin, Heidelberg: Springer-Verlag, 2001, s. 514–532. ASIACRYPT '01. ISBN 3540429875.

[11] Borčík, F. *Testování bezpečnosti a výkonu Proof-of-Stake Protokolů pomocí simulace*. Brno, CZ, 2021. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22846/.

[12] Bosamia, M. a Patel, D. Current Trends and Future Implementation Possibilities of the Merkel Tree. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING*. August 2018, zv. 6, s. 294–301. DOI: 10.26438/ijcse/v6i8.294301.

[13] Buford, J., Yu, H. a Lua, E. P2P Networking and Applications. *P2P Networking and Applications*. Január 2009. DOI: 10.1016/B978-0-12-374214-8.X0001-3.

[14] Castro, M. a Liskov, B. Practical Byzantine Fault Tolerance. In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. USA: USENIX Association, 1999, s. 173–186. OSDI '99. ISBN 1880446391.

[15] Dworzanski, P. A note on committee random number generation, commit-reveal, and last-revealer attacks. In:. Dostupné z: http://paul.oemm.org/commit_reveal_subcommittees.pdf.

[16] Fan, C., Ghaemi, S., Khazaei, H. a Musilek, P. Performance Evaluation of Blockchain Systems: A Systematic Survey. *IEEE Access*. 2020, zv. 8, s. 126927–126950. DOI: 10.1109/ACCESS.2020.3006078.

[17] Feldman, P. A practical scheme for non-interactive verifiable secret sharing. In: *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*. 1987, s. 427–438. DOI: 10.1109/SFCS.1987.4.

[18] Gervais, A., Karame, G. O., Wust, K., Glykantzis, V., Ritzdorf, H. et al. On the Security and Performance of Proof of Work Blockchains. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2016, s. 3–16. CCS '16. DOI: 10.1145/2976749.2978341. ISBN 9781450341394. Dostupné z: https://doi.org/10.1145/2976749.2978341.

[19] Gilad, Y., Hemo, R., Micali, S., Vlachos, G. a Zeldovich, N. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In:. New York, NY, USA: Association for Computing Machinery, 2017, s. 51–68. SOSP '17. DOI: 10.1145/3132747.3132757. ISBN 9781450350853. Dostupné z: https://doi.org/10.1145/3132747.3132757.

[20] Gilbert, S. a Lynch, N. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. *SIGACT News*. New York, NY, USA: Association for Computing Machinery. jún 2002, zv. 33, č. 2, s. 51–59. DOI: 10.1145/564585.564601. ISSN 0163-5700. Dostupné z: https://doi.org/10.1145/564585.564601.

[21] Hoepman, J.-H. Distributed Double Spending Prevention. Marec 2008.

[22] Homoliak, I., Venugopalan, S., Hum, Q., Reijsbergen, D., Schumi, R. et al. The Security Reference Architecture for Blockchains: Towards a Standardized Model for Studying Vulnerabilities, Threats, and Defenses. Október 2019.

[23] Jansen, R. a Hopper, N. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In: *Proceedings of the 19th Symposium on Network and Distributed System Security (NDSS)*. Internet Society, February 2012.

[24] Kiayias, A., Russell, A., David, B. a Oliynykov, R. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In:. Júl 2017, s. 357–388. ISBN 978-3-319-63687-0.

[25] Lan, R. et al. Harmony: Technical Whitepaper. In:. Dostupné z: https://harmony.one/whitepaper.pdf.

[26] Lepore, C., Ceria, M., Visconti, A., Rao, U. P., Shah, K. A. et al. A Survey on Blockchain Consensus with a Performance Comparison of PoW, PoS and Pure PoS. *Mathematics*. 2020, zv. 8, č. 10. DOI: 10.3390/math8101782. ISSN 2227-7390. Dostupné z: https://www.mdpi.com/2227-7390/8/10/1782.

[27] Menezes, A. J. *Handbook of Applied Cryptography*. Taylor & Francis Inc, 1996. ISBN 0849385237ID.

[28] Miller, A. a Jansen, R. Shadow-Bitcoin: Scalable Simulation via Direct Execution of Multi-Threaded Applications. In: *Proceedings of the 8th USENIX Conference on Cyber Security Experimentation and Test*. USA: USENIX Association, 2015, s. 7. CSET'15.

[29] Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. *Cryptography Mailing list at https://metzdowd.com*. Marec 2009.

[30] Narayanan, A., Bonneau, J., Felten, E., Miller, A. a Goldfeder, S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016. ISBN 9780691171692.

[31] Nguyen, C., Dinh Thai, H., Nguyen, D., Niyato, D., Nguyen, H. et al. Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities. *IEEE Access*. Jún 2019, PP, s. 1–1. DOI: 10.1109/ACCESS.2019.2925010.

[32] Paulavicius, R., Grigaitis, S. a Filatovas, E. A Systematic Review and Empirical Analysis of Blockchain Simulators. *IEEE Access*. 2021, zv. 9, s. 38010–38028.

[33] Rice, J. *Mathematical Statistics and Data Analysis*. Cengage Learning, 2006. Advanced series. ISBN 9780534399429. Dostupné z: https://books.google.sk/books?id=EKA-yeX2GVgC.

[34] Robleh Ali, R. B. et al. *Distributed Ledger Technology: beyond block chain*. London, 1 Victoria Street, 2016.

[35] Schollmeier, R. A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications. In:. September 2001, s. 101 – 102. DOI: 10.1109/P2P.2001.990434. ISBN 0-7695-1503-7.

[36] Smart, N. *Cryptography: An Introduction*. McGraw-Hill, 2003.

[37] Stephen Tse, S. D. L. C. et al. Harmony: Documentation. In:. Mountain View, CA: Harmony. Dostupné z: https://docs.harmony.one/home/.

[38] Stoykov, L. VIBES: fast blockchain simulations for large-scale peer-to-peer networks. In:. Technische Universität München, Marec 2018.

[39] Viswanath, P. Bridging BFT protocols with the longest chain protocol. In:. 2021.

[40] Yakovenko, A. Solana: Documentation. In:. San Francisco, California: Solana Labs. Dostupné z: https://docs.solana.com/.

[41] Yakovenko, A. Solana: A new architecture for a high performance blockchain v0.8.13. In:. 2017. Dostupné z: https://solana.com/solana-whitepaper.pdf.

[42] Zhang, S. a Lee, J.-H. Analysis of the main consensus protocols of blockchain. *ICT Express*. 2020, zv. 6, č. 2, s. 93–97. DOI: https://doi.org/10.1016/j.icte.2019.08.001. ISSN 2405-9595. Dostupné z: https://www.sciencedirect.com/science/article/pii/S240595951930164X.

[43] Zheng, Z., Xie, S., Dai, H.-N., Chen, X. a Wang, H. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In:. Jún 2017. DOI: 10.1109/BigDataCongress.2017.85.