VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

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 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání diplomové práce

Student:	**Holub Juraj, Bc.**
Program:	Informační technologie
Obor:	Kybernetická bezpečnost
Název:	**Testování bezpečnosti a výkonu Proof-of-Stake Protokolů pomocí simulace** **Security and Performance Testbed for Simulation of Proof-of-Stake Protocols**
Kategorie:	Bezpečnost

Zadání:

1. Seznamte se s existujícími proof-of-stake protokoly a jejich populárními hybridními variantami. Vytvořte si přehled o existujících simulačních nástrojích, které jsou orientovány na simulaci konsenzuálních protokolů pro blockchain.
2. Vytvořte teoretické srovnání těchto protokolů z hlediska propustnosti, škálovatelnosti, bezpečnosti, soukromí, odolnosti vůči selhání, dostupnosti a dalších vlastností. V oblasti bezpečnosti zohledněte všechny známé existující zranitelnosti proof-of-stake protokolů, jakož i ty obecné.
3. Zvolte aspoň tři protokoly (včetně Harmony a Solana) a implementujte je jako součást simulačního nástroje.
4. Porovnejte vybrané protokoly pomocí výsledků získaných ze simulací. Experimentujte se simulací různých hrozeb.
5. Na základě výsledků simulace a teoretické analýzy navrhněte několik vylepšení, které můžete ověřit pomocí vytvořeného simulačního nástroje.

Literatura:

- Homoliak, Ivan, et al. "The security reference architecture for blockchains: Towards a standardized model for studying vulnerabilities, threats, and defenses." *arXiv preprint arXiv:1910.09775* (2019).
- Yakovenko, Anatoly. "Solana: a new architecture for a high performance blockchain v0. 8.14." (2021).
- Harmony Team. "Harmony Technical Whitepaper", https://harmony.one/whitepaper.pdf
- Aoki, Y., Otsuki, K., Kaneko, T., Banno, R. and Shudo, K., 2019. SimBlock: a blockchain network simulator. arXiv preprint arXiv:1901.09777.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Homoliak Ivan, Ing., Ph.D.**
Vedoucí ústavu:	Hanáček Petr, doc. Dr. Ing.
Datum zadání:	1. listopadu 2021
Datum odevzdání:	18. května 2022
Datum schválení:	3. listopadu 2021

Abstrakt

Cieľom tejto práce je analýza bezpečnosti a výkonnosti troch Proof-of-Stake konsenzus protokolov pre blockchain (Harmony, Solana a Ouroboros). Pre tento účel je v rámci práce vytvorený simulačný nástroj, ktorý s týmito protokolmi experimentuje. Súčasťou riešenia je aj porovnanie aktuálne dostupných simulátorov blockchainu. Výsledky simulácie ukazujú, že všetky tri protokoly môžu fungovať efektívne aj v rozsiahlych sieťach. Z hľadiska bezpečnosti simulácia poukazuje na zraniteľnosť v podobe DoS útoku. Na základe zistených výsledkov boli navrhnuté modifikácie protokolov, ktoré spomínanú bezpečnostnú zraniteľnosť minimalizujú. Vytvorený simulátor je voľne dostupný a určený pre potenciálny ďalší výskum podobných konsenzus protokolov.

Abstract

This work aims to analyze the security and performance of three Proof-of-Stake consensus protocols for blockchain (Harmony, Solana and Ouroboros). For this purpose, a simulation tool is created, which experiments with these protocols. The solution also includes a comparison of currently available blockchain simulators. The simulation results show that all three protocols can work efficiently even in large networks. In terms of security, the simulation points to a vulnerability in the form of a DoS attack. Based on the results, modifications to the protocols were proposed that minimize the mentioned security vulnerability. The created simulator is freely available and intended for potential further research of similar consensus protocols.

Kľúčové slová

blockchain, bezpečnosť, konsenzus, simulácia, Proof-of-Stake, Harmony, Solana, Ouroboros

Keywords

blockchain, security, consensus, Proof-of-Stake, Harmony, Solana, Ouroboros, simulation

Citácia

HOLUB, Juraj. *Testování bezpečnosti a výkonu Proof-of-Stake Protokolů pomocí simulace.* Brno, 2022. 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

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ivana Homoliaka. Uviedol sem všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Juraj Holub
9. mája 2022

Poďakovanie

Chcem poďakovať môjmu vedúcemu Ivanovi Homoliakovi od ktorého mi bola poskytnutá odborná pomoc.

Obsah

Kapitola 1

Úvod

V súčasnosti je blockchain populárnou technológiou používanou v distribuovaných aplikáciach, ktoré kladú dôraz na bezpečnosť. Ide napríklad o kryptomeny, zmenárne, obchodovanie či priemysel Internet-of-Things. Tradičné systémy pre tieto typy aplikácií poskytujú bezpečnosť založenú na centrálnej dôveryhodnej autorite. Naproti tomu, blockchain je decentralizovaný a jeho bezpečnosť je vystavaná na kryptografickom dôkaze. Kapitola 2 popisuje podrobnejšie túto technológiu a jej vlastnosti. Detailnejšie je popísaná konsenzus vrstva blockchainu, ktorá zabezpečuje, že sa všetci užívatelia bezpečne zhodnú na rovnakom stave v decentralizovanom systéme. Pre tento účel sa v mnohých blockchain protokoloch používa mechanizmus Proof-of-Work, ktorý však má nevýhodu v podobe veľkej spotreby energie. Táto práca sa zaoberá alternatívnym mechanizmom nazývaným Proof-of-Stake, ktorý týmto energetickým problémom netrpí. Tento nový prístup súčasne prináša aj nové bezpečnostné zraniteľnosti. Na záver kapitoly sú preto popísané všeobecne známe útoky na konsenzus v blockchaine.

Predmetom tejto práce je analýza bezpečnosti a výkonnosti troch Proof-of-Stake konsenzus protokolov: Harmony, Solana a Ouroboros. V kapitole 3 sú zvolené konsenzus protokoly popísané a teoreticky analyzované. Na záver kapitoly sú porovnané vlastnosti týchto troch protokolov a súčasne je vyhodnotená ich odolnosť voči rôznym útokom na bezpečnosť.

Simulácia blockchainu sa ponúka ako vhodný spôsob pre analýzu vlastností zvolených protokolov v rozsiahlych sieťach. V kapitole 4 sú popísané a porovnané aktuálne voľne dostupné simulátory blockchain technológie. Ďalej je prezentovaný simulačný nástroj vytvorený v rámci tejto práce, ktorý slúži na modelovanie konsenzus protokolov Harmony, Solana a Ouroboros.

Kapitola 5 prezentuje výsledky simulácie zvolených troch protokolov pomocou vytvoreného nástroja. Simulácia experimentuje z výkonnosťou analyzovaných konsenzus protokolov v rozsiahlych sieťach. Ďalej sú simulované možné zraniteľnosti. Na základe výsledkov experimentov sú navrhnuté modifikácie týchto protokolov, ktorých cieľom je zvýšiť bezpečnosť. Navrhované vylepšenie je následne opätovne simulované a zhodnotené.

Na záver sú v kapitole 6 zhrnuté výsledky tejto práce vzhľadom k stanoveným cieľom.

Kapitola 2

Blockchain

Účtovný systém (anglicky *accounting system*) slúži na záznam transakcií do účtovnej knihy (anglicky *ledger*). V dnešnej dobe predstavuje ledger dôležitú časť riadenia organizácií. Tento mechanizmus slúži na správu akýkoľvek digitálnych či fyzických zdrojov. Z hľadiska kontroly obsahu rozlišujeme dva základné typy technológie ledger: centralizovaný a distribuovaný prístup. Tradičným riešením je centralizovaný ledger, v ktorom existuje jedna entita ovládajúca celý systém. Táto entita spravuje všetky záznamy ledgeru a predstavuje dôveryhodnú centrálnu autoritu (napríklad banka). Naopak distribuovaný ledger je technológia, ktorá poskytuje dôveryhodné a bezpečné úložisko záznamov zdieľané naprieč viacerými inštitúciami, krajinami, a to typicky verejne. Na rozdiel od centralizovaného prístupu, distribuovaný ledger nie je ovládaný jedinou entitou. Systém je decentralizovaný a pridávanie nových záznamov do ledgeru je predmetom dohody medzi všetkými participujúcimi stranami (tzv. konsenzus) [2, 5].

V roku 2008 bola publikovaná práca (viď [35]), ktorá navrhla konkrétne technické riešenie distribuovaného ledgeru. Práca navrhla koncept elektronického platobného systému, ktorého bezpečnosť je založená na kryptografickom dôkaze namiesto dôvery v centralizovanú autoritu. Tento systém sa nazýva **blockchain**. Blockchain je dátová štruktúra, ktorá má nasledujúce vlastnosti [48]:

- **Decentralizácia** (anglicky *decentralization*): Blockchain funguje v P2P sieti, ktorá nepotrebuje centralizovanú dôveryhodnú autoritu. Na zabezpečenie konzistencie používa konsenzus protokol.
- **Auditovateľnosť** (anglicky *auditability*): Blockchain v sebe nesie celú histórie zmien dátového obsahu. Každú zmenu stavu dát uložených v blockchaine je teda možné sledovať.
- **Nemennosť** (anglicky *persistency*): Nie je možné modifikovať alebo zmazať už existujúci záznam v blockchaine.
- **Anonymita** (anglicky *anonymity*): Užívatelia pracujúci s blockchainom používajú na identifikáciu asymetrickú kryptografiu s digitálnym podpisom. Takýto kryptografický identifikátor neodhaľuje skutočnú identitu užívateľa, a pritom umožňuje nepopierateľne určiť vlastníka elektronického zdroja.

Aplikačné využitie

Blockchain bol navrhnutý, a po prvýkrát implementovaný, za účelom poskytnúť elektronickú peňažnú menu nezávislú od centralizovaného bankovníctva. Tento prvý a najznámejší blockchain sa nazýva Bitcoin [35]. Avšak, vlastnosti blockchain technológie nachádzajú uplatnenie vo veľkom množstve odvetví. Nasledujúci zoznam vymenúva niekoľko aplikácií, ktoré blockchain môže poskytovať [28, 3]:

- **Financie**: Blockchain môže všeobecne slúžiť ako dôveryhodná tretia strana používaná na finančné aktivity medzi rôznymi subjektami. Ide napríklad o potvrdzovanie obchodov, ochrana pred duplicitou finančných transakcií alebo registrácia a validácia finančných aktivít. Konkrétne môže byť blockchain použitý napríklad pre kryptomeny, obchodovanie s akciami, poisťovníctvo či zmenárne.

- **Zdravotníctvo**: Kľúčovou a veľmi citlivou zložkou zdravotníctva sú dáta popisujúce zdravotný stav pacientov. Blockchain môže byť použitý ako bezpečné úložisko týchto dát, ale taktiež môže poskytovať médium na zdieľanie týchto dát medzi rôznymi inštitúciami.

- **Logistika**: Priemysel logistiky slúži na efektívne riadenie prepravy produktov medzi zákazníkmi a klientmi. Blockchain môže poskytnúť bezpečný nástroj pre logistické operácie naprieč rôznymi organizáciami.

- **Výroba**: V dnešnej dobe je kladený veľký dôraz na automatizáciu výrobných procesov. Blockchain môže v tomto odvetví poslúžiť podobným spôsobom ako v prípade logistiky. Napríklad, môže umožniť bezpečnejšiu komunikáciu medzi rôznymi zariadeniami v rámci technológie *Internet-of-Things*. Ďalším priemyselným využitím je ochrana proti falšovaniu výrobkov (autorské práva).

- **Energetika**: V odvetví energetiky sa používa systém nazývaný *microgrids*. Táto technológia spravuje energetické zdroje rôznych poskytovateľov a efektívne ich distribuuje spotrebiteľom. Ďalej umožňuje predávať a poskytovať prebytočnú energiu koncových užívateľov späť do systému. Blockchain by v tomto systéme mohol poskytovať systém pre správu predaja a nákupu týchto zdrojov na globálnej úrovni.

- **Robotika**: Koordinácia veľkého množstva robotických zariadení sa nazýva *swarm robotics*. Takéto systému nachádzajú uplatnenie napríklad vo výrobných linkách, armáde či zdravotníctve. Použitie blockchainu v tomto priemysle by prinieslo bezpečnejšou komunikáciu medzi jednotlivými zariadeniami (blockchain umožňuje aj výskyt nepoctivých zariadení). Blockchain taktiež prináša väčšiu kontrolu, ktorá by zaručovala, že roboty budú vykonávať len činnosti, ktoré majú schválené.

- **Zábava**: Blockchain môže byť použitý ako médium pre obchodovanie s virtuálnymi zdrojmi v online video hrách. Ďalšou možnosťou využitia sú hazardné hry (toto odvetvie ale prestavuje aktuálne problém, pretože umožňuje klientom hrať mimo prostredie regulované zákonom konkrétnej krajiny). Blockchain taktiež môže poslúžiť, ako prostriedok na obchodovanie s elektronickými zdrojmi (hudba, filmy, hry a podobné).

- **Iné**: V dnešnej dobe už existujú decentralizované súborové systémy založené na P2P sieťach. Implementácia takéhoto decentralizovaného súborového systému v podobe

blockchainu by umožnila nepopierateľne sledovať históriu zmien obsahu. Ďalšou potenciálnou aplikáciou je 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. Blockchain by v tomto prípade opäť umožnil náhradu tejto centralizovanej autority za decentralizovanú sieť. Elektronické hlasovanie (voľby) je ďalším vhodným príkladom, kde je možné efektívne využiť blockchain. Hlasujúce entity predstavujú decentralizovanú sieť a vlastnosti blockchainu zase poskytujú transparentnosť a verejnú overiteľnosť. Podobným príkladom sú reputačné systémy. Tie merajú úroveň dôvery v určité entity. Typickým príkladom je reputácia rôznych predajcov na základe hlasovania zákazníkov. Transparentnosť a nemennosť blockchain histórie by znížila možnosť manipulácie s reputáciu v prospech niektorej entity.

Architektúra

Existuje veľké množstvo blockchain protokolov, ktoré majú rôzne využitie a technologické riešenie. Z hľadiska architektúry môžeme všetky konkrétne riešenia technológie blockchain popísať nasledujúcim modelom abstrakcie so štyrmi vrstvami [28]:

1. **Sieťová vrstva** predstavuje najnižšiu vrstvu blockchainu a zaoberá sa komunikáciu na úrovni P2P siete. Sieť rieši pripájanie nových uzlov či šírenie transakcií a dát medzi nimi. Táto vrstva má kritický dosah na výkonnosť blockchainu. Napríklad, Bitcoin [35] 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 systému [20]. Sieťová vrstva je popísaná v sekcii 2.2.

2. **Konsenzus vrstva** definuje protokol pomocou, ktorého sa ustanovuje dohoda na stave blockchainu. Konsenzus umožňuje uskutočniť globálne akceptované rozhodnutia v distribuovanom systéme bez centrálnej autority. Konsenzus v blockchaine je popísaný v sekcii 2.4.

3. **Dátová vrstva**,alebo tiež úložisko, definuje model transakcií a s ním spojenú kryptografiu. Sekcia 2.1 popisuje kryptografické primitíva používané v technológii blockchain a sekcia 2.3 popisuje blockchain ako dátovú štruktúru.

4. **Aplikačná vrstva** definuje využitie v konkrétnej službe (napríklad kryptomena).

Táto práca sa primárne zaoberá konsenzus vrstvou blockchainu. Do určitej miery je však potrebné uvažovať aj dopad ostatných vrstiev blockchainu. Dôvodom je, že všetky spomenuté vrstvy sa navzájom ovplyvňujú. Napríklad, aplikačná vrstva určuje množstvo transakcií, ich veľkosť a veľkosť siete. Sieťová a dátová vrstva zase kladie fyzické hranice na priepustnosť systému. Tieto vlastnosti priamo ovplyvňujú výkonnosť a bezpečnosť konsenzus vrstvy.

2.1 Kryptografia v blockchaine

Technológia blockchain svoje vlastnosti zabezpečuje pomocou rôznych kryptografických mechanizmov. V tejto sekcii je popísaná kryptografická hashovacia funkcia (pozri 2.1.1) a jej využitie na tvorbu dátových štruktúr zabezpečených proti modifikácii obsahu (viď sekcia 2.1.2). Ďalej je vysvetlený koncept asymetrickej kryptografie a digitálneho podpisu (viď

sekcia 2.1.3), ktoré umožňujú anonymne vlastniť zdroje v blockchaine. Tieto kryptografické primitíva sú základom, na ktorom stojí nemennosť, auditovateľnosť a anonymita blockchainu.

2.1.1 Hashovacia funkcia

Hashovacia funkcia je taká funkcia h, ktorá má ako parameter x reťazec bitov ľubovoľnej 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 [33].

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

Kryptografická hashovacia funkcia, alebo tiež jednocestná funkcia (anglicky *one-way function*), je taká hashovacia funkcia pre ktorú platia nasledujúce tri vlastnosti [41]:

1. Pre konkrétny hash x je výpočtovo nezvládnuteľné nájsť správu takú, že $h(x) = y$. Anglicky voláme túto vlastnosť *first preimage resistant*.

2. Pre konkrétnu správu je výpočtovo nezvládnuteľné nájsť inú správu s rovnakým hashom. Anglicky voláme túto vlastnosť *second preimage resistant*.

3. Pre ľubovoľnú správu je výpočtovo nezvládnuteľné nájsť inú správu s rovnakým hashom. Anglicky voláme túto vlastnosť *collision resistant*.

Hashovacia funkcia má v oblasti počítačovej bezpečnosti dôležité využitie. Slúži napríklad pre bezpečné ukladanie hesiel. Digitálna služba neukladá v databáze heslo, ale len jeho hash. Pri ukradnutí databázy potom nedochádza k odhaleniu hesiel užívateľov. Ďalším možným využitím je ochrana integrity dát. Akákoľvek zmena dátového obsahu pozmení výsledný hash. Ten je vždy konštantne veľký, a preto nie je problém ho k ľubovoľným dátam pridať. Hashovacia funkcia je aj 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) [33, 41].

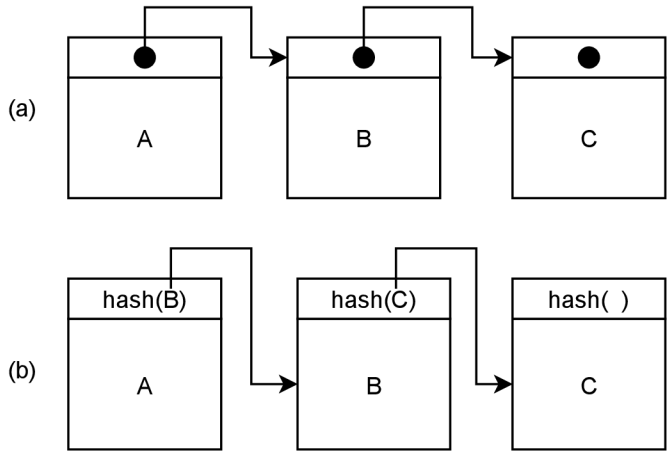
2.1.2 Hash ukazovateľ

Hash ukazovateľ (anglicky *hash pointer* [36]) tvorí základ dátových štruktúr, ktoré poskytujú kryptografické zabezpečenie proti manipulácii s obsahom (anglicky *tamper-evident*). Hash ukazovateľ funguje ako klasický ukazovateľ v zozname či strome. Jeho pridanou hodnotou je, že neukazuje na položku v dátovej štruktúre, ale na dáta uložené v tejto položke. Vďaka tomu dôjde pri pozmenení dát k narušeniu celej dátovej štruktúry. Mechanizmus teda neumožňuje meniť už pridané prvky. Jediná povolená operácia je pridanie ďalšieho prvku na koniec dátovej štruktúry.

Obrázok 2.1 demonštruje rozdiel medzi zoznamom vytvoreným pomocou klasických ukazovateľov a pomocou hash ukazovateľov. V bežnom zozname môžeme nahradiť dátový obsah prvku B bez ovplyvnenia štruktúry zoznamu. V prípade zoznamu s hash ukazovateľom by zmena prvku B narušila referenciu v prvku A.

2.1.3 Digitálny podpis

Digitálny podpis (anglicky *digital signature* [33, 35]) je kryptografický koncept používaný na autentifikáciu, autorizáciu a nepopierateľnosť. Digitálny podpis jednoznačne prepojí určitú



Obr. 2.1: (a) Klasický zoznam (b) Zoznam vytvorený pomocou hash ukazovateľov

entitu s digitálnym dokumentom. V technológii blockchain slúži digitálny podpis na určenie vlastníctva zdrojov, ktoré blockchain uchováva.

Moderná kryptografia používa pre zaistenie dôvernosti šifrovanie pomocou tajného kľúča (tzv. symetrická kryptografia). Pre zašifrovanie a dešifrovanie tajnej správy je potrebná znalosť tajného kľúča. Tento mechanizmus zaisťuje dôvernosť, ale 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:

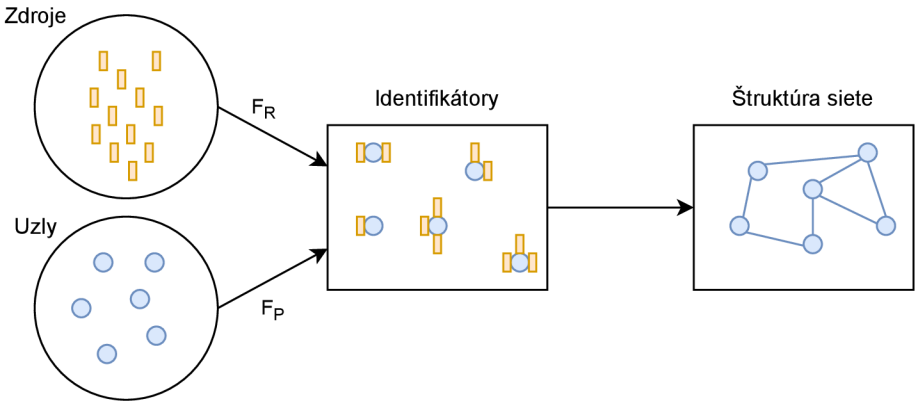
- **Privátny kľúč** je tajný a pozná ho len odosielateľ správy. Odosielateľ používa tento kľúč na zašifrovanie správy.
- **Verejný kľúč** je dostupný všetkým. Ktokoľvek s týmto kľúčom dokáže dešifrovať správu.

Tieto dva kľúče tvoria dvojicu prepojenú matematickým dôkazom. Zo znalosti verejného kľúča je výpočtovo nezvládnuteľné zistiť privátny kľúč v reálnom čase. Zašifrovaná správa nie je dôverná, pretože ktokoľvek môže použiť verejný kľúč na jej dešifrovanie. Zašifrovaná správa ale 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 2.1.1) 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 či ECDSA [33].

2.1.4 Prahový digitálny podpis

Prahový digitálny podpis (anglicky *threshold signature*) 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.



Obr. 2.2: Referenčný model P2P siete (prevzaté z [1]).

Potom hovoríme o *t*-z-*n* prahovom podpise. Takýto koncept sa dá využiť pri kryptografickom hlasovaní. Konkrétnym príkladom prahového digitálneho podpisuje je *Boneh Lynn Shacham* (BLS [11]).

2.2 Sieťová vrstva

Peer-to-peer sieť (ďalej len P2P) je dynamický súbor nezávislých uzlov (anglicky *peers*), ktoré sú prepojené do grafovej štruktúry. Každý uzol obsahuje zdroje, ktoré zdieľa všetkým ostatným uzlom v sieti [15, 40]. P2P siete teda vo všeobecnosti slúžia ako prostriedok na zdieľania zdrojov ako sú súbory, fyzické zariadenia, výpočtový výkon alebo aj elektronické finančné zdroje. Technológia blockchain je vždy postavená nad P2P sieťou. Sieť sa podieľa na decentralizovanosti, nemennosti a auditovateľnosti blockchainu. Dnes existuje množstvo konkrétnych riešení P2P sietí. Veľmi známe sú napríklad Gnutella, Kazaa alebo BitTorrent [1].

Najbežnejšie technické riešenie P2P siete je navrstvenie siete (anglicky *overlay network* [1]) na už existujúcu sieť, ktorou je internet. Takúto sieť potom môžeme definovať ako päticu (P, R, I, F_P, F_R), kde:

- P je množina uzlov v systéme,
- R je množinu zdrojov, ktoré uzly v systéme zdieľajú,
- I je priestor identifikátorov,
- $F_P : P \to I$ je funkcia, ktorá mapuje uzly na identifikátory a
- $F_R : R \to I$ je funkcia, ktorá mapuje zdroje na identifikátory

Priestor identifikátorov je aplikačne špecifický a spolu s mapovacími funkciami slúžia na priraďovanie zdrojov k uzlom na základe nejakej metriky blízkosti. Na základe priestoru identifikátorov je potom vytvorená štruktúra siete, ktorá ľubovoľnému uzlu umožní pristúpiť k ľubovoľnému zdroju. Obrázok 2.2 ukazuje architektúru takto definovanej siete. Konkrétne riešenie P2P siete je závislé od požiadaviek na efektivitu, škálovateľnosť, samoorganizovanosť, odolnosť voči chybám či kooperáciu systému [1].

2.2.1 Decentralizácia

P2P sieť zabezpečuje z veľkej časti nemennosť histórie blockchainu. Blockchain, na rozdiel od tradičnej centrálnej databázy, predstavuje decentralizované úložisko. Klasickú databázu spravuje jediná centrálna autorita. Ak táto autorita podľahne korupcii, tak môže dáta modifikovať a pozmeniť záznamy reprezentujúce históriu. Blockchain je ale postavený na P2P sieti. V takejto sieti je jediný zdroj na zdieľanie, a to je dátová štruktúra blockchain. Všetky uzly v sieti vlastnia kópiu blockchainu. V samotnom blockchaine je uložená celá história transakcií. Kópia blockchainu vlastnená väčšinou siete reprezentuje skutočný stav. Útočník by teda musel vlastniť aspoň 51 % uzlov v sieti, aby mohol pozmeniť dátový obsah blockchainu [35, 27].

2.3 Dátová vrstva

Blockchain je dátová štruktúra založená na hash ukazovateli (viď sekcia 2.1.2). Dátová štruktúra je podobná zoznamu, ale v prípade technológie blockchain hovoríme o tzv. reťazci (anglicky *ledger*). Rozdiel medzi zoznamom a blockchain ledgerom je v tom, že dátový obsah blockchainu je zabezpečený proti manipulácia (anglicky *tamper-evident*) pomocou hashovania. Jediná povolená operácia v blockchaine je pridanie ďalšieho bloku dát na koniec zoznamu [36].

2.3.1 Blok

Blockchain organizuje dáta do podmnožín, ktoré sa volajú bloky [36]. Blok reprezentuje jednu položku v hash zozname. Blok sa skladá z hlavičky a tela. Telo bloku obsahuje dáta vo forme transakcií. Transakcie sú popísané v sekcii 2.3.2. Počet transakcií v bloku je typicky obmedzený maximálnou veľkosťou bloku. Hlavička bloku obsahuje metadáta o bloku, kde najdôležitejšie a najbežnejšie sú nasledujúce [48]:

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

2.3.2 Transakcia

Transakcia je elementárna dátová jednotka na ukladanie digitálnych informácií v blockchaine. 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 uložená množina 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 2.3.3).

Transakcia sa typicky skladá z troch častí [36]:

- **Množina vstupov**: Každý vstup má uložený hash predošlej transakcie z ktorej vychádza. Ďalej definuje, ktoré výstupy z predošlej transakcie si nárokuje. Nakoniec obsahuje digitálny podpis, ktorý autorizuje tvorcu transakcie.

- **Množina výstupov**: Každý výstup reprezentuje digitálny zdroj, ktorý je uchovávaný v blockchaine. Suma hodnôt všetkých výstupov transakcie musí byť menšia alebo rovná sume všetkých vstupov transakcie. Ak je menšia, tak tento rozdiel je použitý ako odmena pre toho, kto publikoval blok ktorého je transakcia súčasťou.
- **Hlavička**: Obsahuje hash transakcie, ktorý je používaný ako unikátny identifikátor pomocou, ktorého sa na transakciu odkazujeme.

2.3.3 Markle tree

Merkle tree [13] je binárny hashovací strom. Ide o dátovú štruktúru podobnú 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 potom 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 tisícky 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 [35].

2.4 Konsenzus vrstva

Konsenzus zabezpečuje, že skupina uzlov v P2P sieti sa zhodne na rovnakom stave blockchainu. Konsenzus v decentralizovanej sieti je zabezpečený pomocou protokolu, ktorý sa snaží nájsť kompromis medzi nasledujúcimi troma vlastnosťami (tzv. *CAP theorem* [24, 47, 32]):

- Konzistentnosť (*Consistency*): Po uznesení konsenzu majú všetci užívatelia rovnaký stav blockchainu (rovnaké dáta).
- Dostupnosť (*Availability*): Konsenzus musí byť dokončený v konečnom čase. Výsledkom konsenzu môže byť aj neúspech.

- Odolnosť voči prerušeniu (*Partial tolerance*): Konsenzus protokol funguje aj v prípade, že v systéme vznikajú chyby (výpadky uzlov, škodlivé správanie časti uzlov).

CAP teorém [24] deklaruje, že z týchto troch vlastností je súčasne možné dosiahnuť maximálne dve (tretiu vlastnosť je možné dosiahnuť dodatočne, ale nie súčasne). P2P sieť, na ktorej blockchain funguje, nedokáže garantovať vysokú spoľahlivosť. Blockchain teda musí nevyhnutne riešiť problém odolnosť voči prerušeniu [44].

Zo zvyšných dvoch vlastností sa môže zvoliť dostupnosť pred konzistenciou. Bitcoin používa tento prístup a konzistenciu zaisťuje až dodatočne (anglicky *eventual consistancy*). Transakcie v Bitcoin blockchaine sú konzistentné, ak sa nachádzajú v bloku, ktorý je prekrytý väčším množstvom ďalších blokov.

Druhou možnosťou je zvoliť konzistenciu pred dostupnosťou. Táto skupina blockchain protokolov je založená na hlasovaní. Po ukončení hlasovania je jednoznačne rozhodnuté, ktoré transakcie sú konzistentné. Samotné hlasovanie v P2P sieti, ale zaberá nezanedbateľný čas, ktorý navyše rastie úmerne veľkosti siete.

2.4.1 Typy konsenzus protokolov

Existuje množstvo konkrétnych konsenzus protokolov. Všeobecne ale môžeme všetky tieto protokoly rozdeliť na tri skupiny [47, 28]:

- **Lotéria**: Takéto protokoly náhodne zvolia uzol, ktorý vyprodukuje nový blok. Výhodou tohto prístupu je jeho jednoduchosť, keďže takýto proces nevyžaduje žiadnu interaktívnu komunikáciu. Nevýhodou tohto 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. Toto správanie spomaľuje konzistentnosť blockchainu. 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. Lotéria s pohľadu CAP teorému volí dostupnosť pred konzistenciou.

- **Hlasovanie**: Tieto protokoly dosahujú dohodu pomocou hlasovania všetkých zapojených uzlov. Používa sa 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. Z pohľadu CAP teorému, volí hlasovanie konzistentnosť pred dostupnosťou.

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

2.5 Proof-of-Work

Dôkaz prácou, anglicky *Proof-of-Work*, je najdlhšie používaná stratégia konsenzus protokolu v blockchaine [35]. Proof-of-Work je založený na náhodnej voľbe (viď sekcia 2.4.1). 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. Vždy je tu možnosť, že problém

vyrieši súčasne viacero uzlov. Konečná voľba je teda náhodná. Proof-of-Work z tohto dôvodu principiálne umožňuje vetvenie reťazca. Bezpečnosť takéhoto konsenzu spočíva v tom, že majorita výpočtového výkonu siete (51 %) je vlastnená poctivými uzlami.

Proof-of-Work konsenzus využíva dva typy uzlov. Prvý typ uzla vytvára nové bloky, tak ako je popísané v sekcii 2.5.1. Druhý typ uzla je bežný vlastník zdrojov v danom blockchaine, ktorý môže vytvárať transakcie a distribuovať ich do siete. Druhý typ uzla teda nehrá žiadnu rolu v ustanovovaní konsenzu [32].

2.5.1 Ťažba blokov

Ťažba bloku (anglicky *mining*) [36] 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 P2P siete. Každú transakciu si validuje pomocou kryptografie popísanej v sekcii 2.1. 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 2.5.2). 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]. Táto odmena môže byť ďalej zvýšená o poplatky, ktoré sú v transakciách. Ak teda majiteľ transakcie chce, aby sa jeho transakcia dostala do blockchainu čo najrýchlejšie, poskytne vyššiu odmenu v podobe poplatku za transakciu. Miner bude potom viac motivovaný pridať práve túto transakciu do bloku.

2.5.2 Bitcoin

Bitcoin je najznámejší blockchain protokol, ktorý používa Proof-of-Work konsenzus. Samotný kryptografický problém, ktorý užívatelia tejto siete riešia spočíva v počítaní hashu (viď 2.1.1) z hlavičky nového bloku (viď 2.3.1). Hlavička obsahuje atribút *nonce*, ktorý môže miner ľubovoľne nastaviť. Zmenou tohto 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. Úlohu je teda možné riešiť len pomocou metódy útoku hrubou silou (anglicky *brute force*). Miner môže svoju šancu

[1] https://www.investopedia.com/tech/how-does-bitcoin-mining-work/

na úspech zvýšiť len tým, že poskytne väčší výpočtový výkon na jej vyriešenie. Na druhej strane, ostatné uzly môžu overiť, že jeho riešenie je správne veľmi rýchlo a efektívne [48].

2.5.3 Vlastnosti

Proof-of-Work je overený konsenzus protokol, ktorý funguje a používa sa v blockchainoch ako je Bitcoin [35] 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 [32].

Druhou veľkou nevýhodou Bitcoinu je jeho nízka priepustnosť transakcií a pomalá konzistencia. Priepustnosť transakcií je zvyčajne reprezentovaná metrikou TPS (*Transactions Per Second*), ktorá definuje koľko transakcií spracuje systém za sekundu. V prípade Proof-of-Work blockchainu je ale TPS pomerne nejasná metrika, keďže samotný výskyt transakcie v blockchaine ešte nezaručuje jej konzistenciu. Pre porovnanie, centralizovaný platobný systém Visa[4] spracuje približne 1 500 transakcií za sekundu, zatiaľ čo Bitcoin spracuje za rovnaký čas približne päť transakcií. Dôvodom je, že ťažba bloku trvá pomerne dlhý čas (približne 10 minút[5]). Navyše novo pridaný blok nie je hneď možné považovať za konzistentný.

2.6 Proof-of-Stake

Dôkaz podielom na vlastníctve (anglicky *Proof-of-Stake*) je konsenzus mechanizmus, ktorý predstavuje alternatívu k Proof-of-Work (popísané v sekcii 2.5). Základnou myšlienkou je, že vlastník veľkého množstva zdrojov v danom blockchaine je veľmi nepravdepodobným útočníkom, pretože svoje zdroje nechce ohroziť. Uzly sa teda musia preukázať vlastníctvom zdrojov v danom blockchaine, ak chcú publikovať nový blok. Proof-of-Stake sa používa v konsenzus protokoloch založených na lotérii, ale aj hlasovaní [28, 37].

2.6.1 Ovládnutie podielu

Proof-of-Stake konsenzus je bezpečný do tej doby, dokedy neexistuje entita, ktorá vlastní majoritu podielu. Z pohľadu teórie hrania[6] hier teda môže existovať mocná entita, ktorá sa rozhodne nakúpiť obrovské množstvo zdrojov v Proof-of-Stake blockchaine, čím ovládne konsenzus. Tento argument je ale rovnako aplikovateľný aj na Proof-of-Work konsenzus. Rovnaký vplyvný subjekt môže podobným spôsobom nakúpiť majoritu výpočtového výkonu a ovládnuť proces ťažby blokov. Z tohto pohľadu teda Proof-of-Stake nie je menej bezpečný ako Proof-of-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

[5] https://scholar.smu.edu/cgi/viewcontent.cgi?article=1185&context=datasciencereview

[6] https://www.youtube.com/watch?v=MHYSOxgZ9dQ

2.6.2 Vlastnosti

Veľkou výhodou Proof-of-Stake, oproti Proof-of-Work, je jeho energetická nenáročnosť. Uzly už nemusia súťažiť v riešení výpočtovo náročných úloh. Ďalšou veľkou výhodou je rýchlosť. Uzly svoju dôveryhodnosť preukážu vlastníctvom svojich zdrojov. Takéto overenie predstavuje konštantne dlhý čas. Proof-of-Work svoju dôveryhodnosť dokazujú riešením úlohy, ktorá trvá rádovo minúty [32, 37].

Proof-of-Stake konsenzus ale prináša aj mnohé nové výzvy, ktoré v prípade Proof-of-Work nepredstavovali problém. Najväčším problémom je bezpečnosť. Proof-of-Stake prinieslo nové typy útokov. Tieto útoky sú podrobnejšie popísané v sekcii 2.8. Ďalším problémom je, že konkrétne protokoly často nedefinujú svoju bezpečnosť formálne. Zabezpečenie týchto protokolov je potom nejasné [37].

2.6.3 Pripojenie nového uzlu

Z hľadiska pripojenia nového uzlu do blockchain konsenzu rozlišujeme dva prístupy, ktoré sa v literatúre [28] označujú ako:

- *Permissionless*: Ktokoľvek sa môže pripojiť do konsenzu.
- *Permissioned*: Na pripojenie do konsenzu je potrebné získať povolenie od autority.

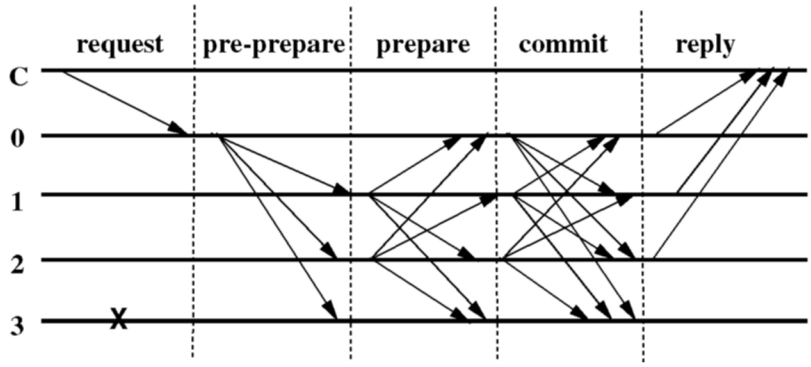
Permissionless prístup umožňuje väčšiu decentralizáciu, pretože nepotrebuje žiadnu centrálnu autoritu. Z hľadiska decentralizovanej bezpečnosti ide teda o výhodnejší prístup. Problémom tohto mechanizmu je útok nazývaný *Sybil Attack* [19]. Ten spočíva v tom, že útočníkovi teoreticky nič nebráni vytvoriť veľkého množstva falošných identít. Permissionless konsenzus na zabránenie tomuto útoku vyžaduje od každého nového uzlu dôkaz o vlastníctve zdrojov (anglicky *Proof-of-Resources*). Ak ide o skutočne cenné zdroje, tak bude mať útočník len konečné prostriedky na ich získanie a nebude môcť neobmedzene vytvárať falošné identity. Proof-of-Work je konsenzus založený na preukazovaní zdrojov v podobe výpočtového výkonu. Ide o čistú podobu permissionless konsenzus. Proof-of-Stake taktiež teoreticky umožňuje permissionless konsenzus. V tomto prípade uzly preukazujú vlastníctvo zdrojov uložených v samotnom blockchaine. Proof-of-Stake už ale nie je čistá podoba permissionless konsenzus, pretože nové uzly musia najprv tieto zdroje niekde kúpiť. Teoreticky by entita predávajúce tieto zdroje mohla niekomu odoprieť predaj (zneužitie moci centrálnou autoritou). Preto Proof-of-Stake označujeme ako tzv. *semi-permissionless*.

2.7 Proof-of-Authority

Proof-of-Authority je rodina konsenzus protokolov založená na hlasovaní pomocou algoritmov byzantskej chyby *Byzantian Fault Tolerance* (BFT [17]). Proof-of-Authority predpokladá, že v sieti je množina dôveryhodných uzlov. Tieto uzly (autority) sú jednoznačne identifikované a majorita autorít je poctivá. Veľmi rozšírený Proof-of-Authority protokol je PBFT popísaný v sekcii 2.7.1.

2.7.1 Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance (ďalej len PBFT [16]) je protokol na uznesenie konsenzu v distribuovanej sieti pomocou hlasovania. Tento protokol patrí do rodiny Proof-of-Authority, ale je často používaný aj v rámci Proof-of-Stake konsenzu. Proof-of-Stake



Obr. 2.3: Priebeh hlasovania v PBFT (prevzaté z [16]).

protokoly často vyžadujú interné hlasovanie, ktoré zabezpečuje práve PBFT. Dôveryhodnosť uzlov je potom zabezpečená vlastníctvom zdrojov (Prooof-of-Stake). PBFT funguje pod podmienkou, že v sieti je maximálne $\frac{1}{3}$ nepoctivých uzlov. Na komunikáciu v sieti sa používa *broadcast*. Na zabezpečenie sa požíva kryptografia s digitálnym podpisom (viď sekcia 2.1.3). 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.
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 2.3 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.

2.8 Útoky na konsenzus

Nasledujúca sekcia vysvetľuje najbežnejšie typy útokov na bezpečnosť konsenzus protokolov. Niektoré útoky sú aplikovateľné na všetky rodiny konsenzus protokolov (napríklad, ovládnutie konsenzus, alebo zdvojnásobovanie výdavkov). Iné útoky prestavujú problém špeciálne pre Proof-of-Stake konsenzus (ovplyvnenie volieb, neskoršia korupcia a ďalšie). Popísané útoky vychádzajú z nasledujúcej práce [28].

2.8.1 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. Tento útok je aplikovateľný na ľubovoľný konsenzus mechanizmus. V prípade Proof-of-Work blockchainu ako je Bitcoin ide o ovládnutie 51 % výpočtového výkonu. Ak útočník vlastní takto veľký podiel výpočtového výkonu, tak dokonca môže modifikovať históriu blockchainu. V prípade Proof-of-Stake ide o úplne rovnaký problém (útočník potrebuje vlastniť majoritu podielu). Ak je konsenzus založený na hlasovaní pomocou protokolov Byzantskej chyby, tak dokáže $\frac{1}{3}$ celkového podielu spôsobiť, že bude protokol narušený alebo dokonca zastavený. Na ovládnutie konsenzu je ale v tomto prípade potrebné viac ako $\frac{2}{3}$ podielu.

2.8.2 Porušenie synchrónneho 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 asynchrónnu komunikáciu. Útok je možné vykonať napríklad pre protokoly založené na hlasovaní pomocou byzantskej chyby.

Uvažujme konsenzus protokol založený na hlasovaní. Takýto konsenzus považuje nový blok za konzistentný ak majorita siete poslala svoj hlas do určitého času. Útok synchrónneho doručovania by v takomto protokole mohli nepoctivé uzly dosiahnuť pozdržaním svojich hlasov. Vďaka tomu by nemusel byť konsenzus úspešný a blok pridaný do blockchain. Neskôr tieto uzly môžu zverejniť svoje hlasy a spochybniť dodatočne výsledok hlasovania čím narušia konzistenciu blockchainu.

2.8.3 Útok na časovú synchronizáciu

Decentralizovaná sieť blockchainu 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 známkou, ktorá už nebude akceptovaná.

2.8.4 Zdvojnásobenie výdavkov

Zdvojnásobenie výdavkov (anglicky *double spending*) je útok, ktorý vzniká vytvorením dvoch alebo viac konfliktných blokov. Tieto bloky vytvárajú tzv. vetvy (anglicky *forks*). Z tohto dôvodu môžu byť v týchto blokoch konfliktné transakcie. Neskôr je síce len jedna z nich validný, pretože ostatné vetvy budú zahodené. Avšak, tento útok spomaľuje konzistenciu konsenzu. V prípade mechanizmu náhodnej voľby je kvôli tomuto útoku považovaný blok za konzistentný až keď je prekrytý väčším množstvom ďalších blokov. V prípade hlasovacieho mechanizmu je konzistencia finálna po ukončení hlasovania. Hlasovacie konsenzus protokoly teda neumožňujú ani krátkodobé zdvojnásobenie výdavkov.

2.8.5 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.

Uvažujme Proof-of-Stake konsenzus založený na hlasovaní v podskupinách. Podiel jednotlivých uzlov v tomto protokole môže byť vzhľadom k celej sieti malý a potenciálny útočník nedokáže svojím podielom ovplyvňovať blockchain. Vo vhodne zvolenej podskupine môže ale rovnaký podiel predstavovať majoritu. Útočník sa preto môže snažiť ovplyvniť rozdelenie do podskupín vo svoj prospech.

2.8.6 Vetvenie bez rizika straty zdrojov

Generovanie blokov v Proof-of-Stake 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 tohto správania je, že vzniká väčšie množstvo vetiev reťazca. Ako dôsledok sa potom zvyšuje čas do konzistentnosti konsenzu.

2.8.7 Ovplyvnenie volieb

V blockchaine je vždy priebežne určený uzol, ktorý ako ďalší pridá nový blok. Táto roľa má rôzne výhody. Napríklad, niektoré blockchainy odmeňujú tvorcu bloku podielom z poplatkov za vložené transakcie do bloku. Ďalšou výhodou je, že tvorca bloku má možnosť určiť, ktoré transakcie budú do bloku pridané. Útočník preto môže chcieť ovplyvniť mechanizmus voľby tejto role vo svoj prospech. Pre Proof-of-Stake prebieha táto voľba typicky na základe neobjektívnej náhodnosti. Pritom platí, že uzol s veľkým podielom bude pravdepodobnejšie tvorcom nového bloku. Pre tento účel sa typicky používa nejaký algoritmus na distribuované generovanie náhodnosti, ktorý vygeneruje náhodné číslo. Toto číslo je potom použité pre deterministické určenie tvorcu. Útočník sa potom snaží ovplyvniť samotné generovanie náhodnosti tak, aby bol na jej základe zvolený práve on.

2.8.8 Útok na vodcu hlasovania

Konsenzus protokol založený na hlasovaní je vždy náchylný na dostupnosť. Proof-of-Stake protokoly založené na hlasovaní často používajú konsenzus z rodiny byzantskej chyby (viď sekcia 2.7.1). Pre tieto hlasovacie protokoly je kľúčová roľa vodcu bez ktorého hlasovanie neprebehne. Vodca je typicky uzol, ktorý zároveň publikuje blok. Ak útočník pozná aktuálneho vodcu, môže sa ho pokúsiť zneškodniť nejakou formou DoS (anglicky *Denial of Service*) útoku. Bez funkčného vodcu hlasovanie nemôže prebehnúť úspešne a nový blok nebude finalizovaný. Niektoré Proof-of-Stake protokoly vytvárajú dokonca rozvrh vodcov na dlhé časové obdobie. Takýto rozvrh je dopredu známy, čo zjednodušuje a zrýchľuje celý konsenzus. Na druhej strane, útočníci dopredu vedia, ktorý uzol bude v konkrétnom čase vodcom. Útočník potom môže pomerne jednoducho v danom čase vykonať DoS útok na vodcu.

2.8.9 Neskoršia korupcia

Útočník sa snaží získať privátne kľúče uzlov, ktoré mali v minulosti vplyv na reťazec blockchainu. 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 blockchainu môžu kedykoľvek vymeniť za reálne peniaze. Ak útočník získa kľúče s dostatočným podielom zdrojov, tak môže pozmeniť históriu reťazca.

Kapitola 3

Analýza zvolených konsenzus protokolov

Nasledujúca kapitola popisuje Proof-of-Stake protokoly Harmony (sekcia 3.1), Solana (sekcia 3.2) a Ouroboros (sekcia 3.3). Harmony a Solana sú protokoly založené na hlasovaní a Ouroboros na náhodnej voľbe. Na záver sú v sekcii 3.4 všetky tri protokoly teoreticky porovnané. Porovnávajú sa všeobecné vlastnosti (synchronizácia v čase, spôsob tvorby bloku, model motivácie, výkonnosť či škálovateľnosť siete), ako aj zabezpečenie protokolov voči konkrétnym útokom.

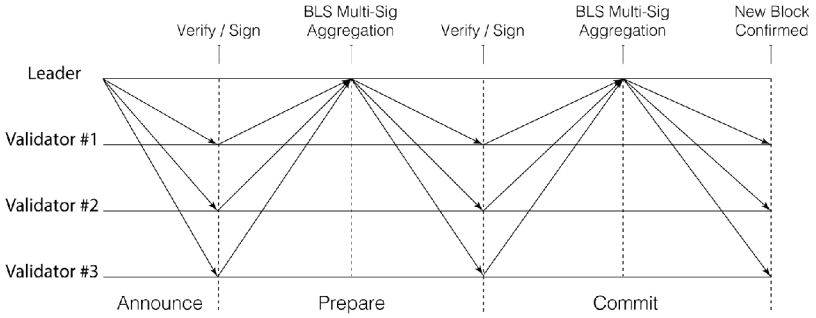
3.1 Harmony

Harmony je blockchain používajúci Proof-of-Stake konsenzus a *sharding*. Cieľom tohto protokolu je poskytovať aplikácie, ktoré v minulosti nebolo možné uskutočňovať na technológii blockchain z dôvodu rýchlosti a škálovateľnosti. Na aplikačnej úrovni je Harmony určené pre služby ako decentralizované zmenárne alebo platobný systém rozsahovo porovnateľný s Visa.

Ak nie je uvedené inak, všetky informácie o protokole Harmony popisované v tejto sekcii vychádzajú z oficiálnej dokumentácie publikovanej autormi tohto protokolu [31, 43]. Sekcia 3.1.1 popisuje konsenzus protokol založený na hlasovaní. V sekcii 3.1.2 je vysvetlený mechanizmus sharding, ktorý slúži na optimalizáciu škálovateľnosti. Bezpečnosť shardingu je úzko spojená s distribuovaným generovaním náhodnosti popísaným v sekcii 3.1.3. Ďalej je v sekcii 3.1.4 popísaný princíp komunikácie medzi shardami. Sekcia 3.1.5 definuje model motivácie. Na záver je v sekcii 3.1.6 tento protokol teoretický analyzovaný z hľadiska bezpečnosti a výkonnosti.

3.1.1 Protokol FBFT

Konsenzus v Harmony protokole je založený na algoritme PBFT, ktorý už bol popísaný v sekcii 2.7.1. Jeden z kľúčových problémov PBFT je, že jeho časová zložitosť je $O(n^2)$ pre n uzlov. Táto vlastnosť neumožňuje dobrú škálovateľnosť siete. Harmony preto používa úpravu PBFT, ktorá má lineárnu časovú zložitosť vzhľadom k počtu uzlov v sieti (viď sekcia 3.1.1). Harmony svoj protokol nazvala FBFT (*Fast Byzantine Fault Tolerance*). FBFT namiesto zasielania hlasov pomocou broadcastu používa prahový digitálny podpis (pozri sekciu 2.1.4). FBFT konsenzus prebieha následovne:



Obr. 3.1: Priebeh hlasovania v FBFT (prevzaté z [31]).

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 podpisom 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 2.1.4). Tento podpis rozošle pomocou broadcastu spolu s bitmapou indikujúcou validátorov ktorý podpísali.

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.

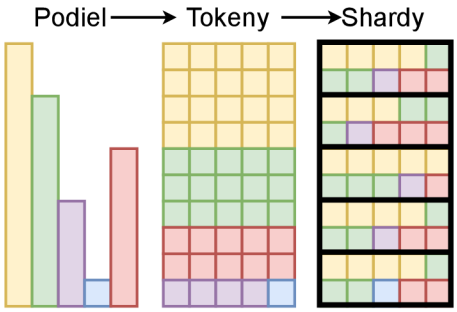
Obrázok 3.1 demonštruje FBFT protokol pre štyri uzly. Na tomto príklade môžeme vidieť, že celé hlasovanie má lineárnu časovú zložitosť, keďže broadcast vysiela len vodca (na rozdiel od PBFT, kde broadcast vysielajú aj validátori).

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

3.1.2 Sharding

Harmony protokol používa *sharding*, aby dosiahol lepšiu škálovateľnosť siete. Čas potrebný na uznesenie konsenzu pomocou hlasovania v distribuovanej sieti rastie úmerne veľkosti danej siete. Harmony preto rozdeľuje sieť na podskupiny uzlov, ktoré hlasujú separátne. Tieto skupiny sa volajú shardy. Shardy sú vždy približne rovnako veľké a ich počet je odvodený od celkovej veľkosti siete. Hlasovanie by vďaka tomuto mechanizmu malo trvať približne rovnaký čas, a to nezávislo na veľkosti siete.

Sharding ale komplikuje Proof-of-Stake mechanizmus. Podiel jednotlivých uzlov sa mení v čase, a preto je potrebné priebežne meniť rozloženie uzlov v shardoch. Harmony pre tento účel používa časové obdobie epochy. Epocha je časový interval počas, ktorého je štruktúra každého shardu nemenná. Epocha odpovedá času potrebnému na vygenerovanie 32 768



Obr. 3.2: Rozdelenie hlasovacieho podielu validátorov medzi všetky shardy (prevzraté z [43]).

blokov, čo je približne 18,2 hodiny[1]. Na začiatku každej epochy sú uzly rozdelené medzi shardy.

Sharding taktiež predstavuje problém z hľadiska bezpečnosti Proof-of-Stake, pretože hlasovacie právo v jednotlivých shardoch už neodpovedá pôvodnému rozdeleniu podielu. Uzol môže vlastniť malý hlasovací podiel vzhľadom k celej sieti, ale v sharde môže byť tento podiel oveľa väčší vzhľadom k ostatným podielnikom v rovnakej sharde. Harmony tento problém minimalizuje rozdeľovaním hlasovacieho podielu rovnomerne medzi všetky shardy. Obrázok 3.2 demonštruje mechanizmus rozdeľovania hlasovacieho podielu uzlov medzi shardy. Jednotlivé uzly majú rôzne veľký podiel. Celkové množstvo hlasovacieho podielu je rozdelené na konštantne veľké tokeny (hlasovacie lístky). Hodnota jedného hlasovacieho lístku t v epoche e je určená rovnicou 3.1, kde S_{e-1} je celkový podiel 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 [43] kapitola 3). Tieto hlasovacie lístky sú následne náhodne rozdelené medzi všetky shardy po dobu jednej epochy. Obrázok ukazuje, že rozdelenie hlasovacieho podielu v jednotlivých shardoch približne odpovedá pôvodnému celkovému rozloženiu podielu. Náhodné rozdelenie lístkov medzi shardy je založené na náhodnom čísle, ktoré je vygenerované pomocou distribuovaného generovania náhodnosti (viď sekcia 3.1.3).

$$t = \frac{S_{e-1}}{n.\lambda} \tag{3.1}$$

3.1.3 Distribuované generovanie náhodnosti

Harmony rozdeľuje hlasovacie lístky do shardov pomocou náhodnej voľby (anglicky *randomness based sharding*). Na začiatku epochy je vykonaná náhodná permutácia všetkých hlasovacích tokenov (viď sekcia 3.1.2). 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 [23]*) a VDF (*Verifiable Delay Function [10]*):

1. Vodca pošle hash posledného bloku všetkým validátorom.

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

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 3.1.1.

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.

V jednej epoche sa vždy vygeneruje n blokov (zvlášť v každom sharde). Inak povedané, prebehne práve n kôl protokolu FBFT. Počas celej epochy je v každom sharde 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 3.1.3). Pravdepodobnosť, že podielnik bude zvolený za vodcu shardy je priamo úmerná množstvu podielu, ktorý vložil (princíp Proof-of-Stake).

3.1.4 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 (ako bolo popísané v sekcii 3.1.3) a rozdeľuje na základe neho hlasovacie lístky do shardov. 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 svojho nového bloku. Nový blok v beacon sharde je následne pomocou broadcastu zaslaný všetkým ostatným shardom, ktoré si ho uložia. Vďaka tomu vedia jednotlivé shardy validovať transakcie z iných shardov.

3.1.5 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.

3.1.6 Teoretická analýza

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

Obr. 3.3: Pravdepodobnosť ovládnutia jedného shardu útočníkom.

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 2.8.5). 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 $H(N, K, n, k)$ [39, 31], 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 3.3 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 blockchainu Harmony [31]. Môžeme vidieť, že útočník má zanedbateľnú pravdepodobnosť ovládnuť shard pokiaľ nevlastní približne $\frac{1}{3}$ z celkového množstva hlasovacích lístkov. Z toho plynie, že shardingu v tomto protokole neumožňuje útočníkovi využiť útok na podskupinu uzlov (viď sekcia 2.8.5).

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 svojho 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.

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 Proof-of-Stake bezpečné (viď útok 2.8.7). Avšak, vodca shardy sa nemení počas celej epochy (približne 18,2 hodiny) a všetci útočníci vedia, ktorý uzol je vodcom. Z tohto hľadiska je možné počas celej epochy vykonávať DoS útok na vodcu, čím sa znemožní uzatvárať konsenzus (viď útok 2.8.8). Samotný protokol Harmony tento útok nerieši a ochrana vodcu by preto musela byť vykonaná na sieťovej vrstve napríklad pomocou firewallu.

3.2 Solana

Solana je blockchain technológia zameraná na vysokú priepustnosť transakcií. Klasická centralizovaná databáza dokáže mať priepustnosť až 710 000 TPS (anglicky *Transaction Per Second*) na gigabitovej sieti s priemernou veľkosťou transakcie 176 B [45]. Decentralizovaná databáza blockchain má dramaticky nižšiu priepustnosť. Tradičné Proof-of-Work protokoly ako je Bitcoin (5 TPS) alebo Ethereum (15 TPS) dosahujú veľmi nízku priepustnosť[2]. V sekcii 3.1 bol popísaný Proof-of-Stake 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 blockchainu 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 3.2.1) pomocou ktorého môže rýchlo a efektívne synchronizovať čas aj decentralizovaná sieť. Solana používa na uznesenie konsenzu Proof-of-Stake hlasovanie popísané v sekcii 3.2.2. Sekcia 3.2.6 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 [46] a oficiálnej dokumentácie tohto projektu [45].

3.2.1 Proof-of-History

Proof-of-History 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 blockchain. Dôkaz o usporiadaní je založený na kryptografickej hash funkcii (pozri sekciu 2.1.1), ktorá spĺňa kritérium *collision resistant*.

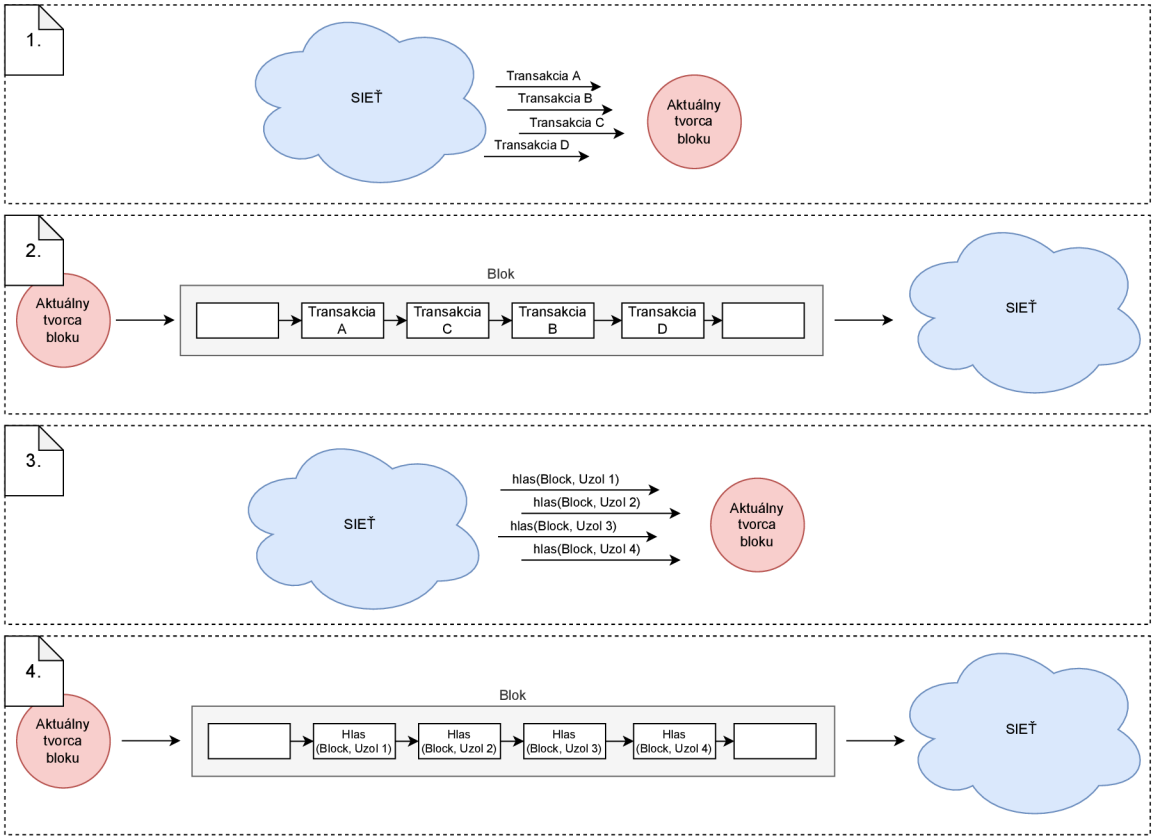
V sieti vždy existuje jeden uzol, ktorý nazývame generátor. Generátor je uzol, ktorý aktuálne vytvára nový blok. Ten neustále (najrýchlejšie ako dokáže) pridáva nové záznamy do hash zoznamu (pozri sekciu 2.1.2). Pridávanie nových hashov do zoznamu je sekvenčné a podmienené existenciou predchádzajúceho hashu. Na samotné vytvorenie hashu sa používa algoritmus VDF (*Verifiable Delay Function [10]*). VDF garantuje, že vytvorenie nového hashu zaberie k sekvenčných krokov pre nejaké k. Každý hash teda predstavuje časový bod a vzdialenosť dvoch hashov udáva časový interval.

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 2.1.3).

3.2.2 Tower BFT

Solana navrhla konsenzus protokol, ktorý kombinuje Proof-of-History a Proof-of-Stake spolu s hlasovaním. Solana blockchain je tvorený Proof-of-History sekvenciou. Na uznesenie o validnom stave Proof-of-History sekvencie sa používa hlasovací konsenzus protokol, ktorý vychádza s protokolu PBFT (pozri sekcii 2.7.1).

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



Obr. 3.4: Synchronizácia hlasovania pomocou Proof-of-History.

Obrázok 3.4 demonštruje princíp fungovania konsenzus protokolu *Tower BFT*:

1. Klienti rozosielajú svoje transakcie do siete. Uzly ktoré budú v blízkej dobe vytvárať bloky ich zbierajú.

2. Keď tvorca bloku prijme zo siete validnú transakciu, pridá ju do Proof-of-History sekvencie. Ak generátor práve nemá dáta, tak generuje prázdny záznam. Celá Proof-of-History sekvencia, ktorú vytvorí potom predstavuje dátový obsah bloku. Po určitom čase (tj. počte hashov) rozošle nový blok do siete.

3. Validátori v sieti overia transakcie v Proof-of-History sekvencii a vykonajú ich v stanovenom poradí. Takto získajú nový stav. V špecifikovanom čase, po uplynutí určitého počtu Proof-of-History záznamov, pošlú validátori svoje hlasy vodcovi. Hlas je hash stavu validátora po vykonaní transakcií z danej Proof-of-History sekvencie. Tento hash je navyše podpísaný privátnym kľúčom validátora. Takto validátor potvrdzuje, že súhlasí s Proof-of-History sekvenciou ktorú podpísal. Ak zo sekvenciou nesúhlasí, jednoducho svoj hlas nepošle.

4. Keď generátor (už nie nutne ten istý uzol ako predtým) prijme validný hlas, pridá ho do Proof-of-History sekvencie ako akúkoľvek inú transakciu. Po uplynutí stanovenej doby blok rozošle do siete. Validátori sledujú hlasy, ktoré sa následne vyskytujú v Proof-of-History sekvencii. Ak pre niektorý blok z minulosti napočítali viac ako $\frac{2}{3}$ hlasov od ostatných, tak je tento blok finalizovaný.

Solana používa Proof-of-Stake, 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.

3.2.3 Rozvrh vodcov

Solana má obtiažnosť VDF nastavenú tak, aby bolo možné vygenerovať Proof-of-History sekvenciu dlhú 160 hashov[3]. Jeden vygenerovaný hash nazývame *tick*. Sekvencia 64 tickov sa nazýva *slot* a predstavuje jeden blok (ide o približne 400 ms). Počas jedného slotu vygeneruje vodca jeden alebo žiaden blok. Ak nevygeneruje blok, tak si všetky uzly nahradia tento slot prázdnou sekvenciou tickov, ktorú si vždy paralelne počítajú. Nový vodca sa určí na základe už dopredu stanoveného rozvrhu. Rozvrh je stanovený na fixný počet slotov, ktorý voláme *epocha*. Epocha trvá 420 000 blokov, čo sú približne dva dni. Rozvrh na aktuálnu epochu sa vypočíta už v predchádzajúcej.

Samotný výpočet rozvrhu prebieha tak, že sa zoberú všetci vlastníci podielu, ktorý hlasovali v posledných n záznamoch Proof-of-History sekvencie, kde n je konfigurovateľná konštanta. Pridelenie slotov je určené pseudo náhodným generátorom, ktorý je inicializovaný číslom aktuálneho slotu. Toto číslo je monotónne rastúci čítač v čase. Algoritmus pseudo náhodne vyberie z množiny možných kandidátov jedného vodcu pre každý slot nasledujúcej epochy pričom pravdepodobnosť voľby je priamo úmerná podielu daného validátora (Proof-of-Stake princíp). Rozvrh Proof-of-History generátorov na celú epochu si vypočíta každý uzol lokálne. Tento algoritmus je deterministický, a teda všetky uzly získajú rovnaký rozvrh.

3.2.4 Nedostupnosť vodcu a vetvenie

Uzly v Solana sieti tolerujú dočasnú stratu spojenia s aktuálnym vodcom (Proof-of-History generátorom). Pri strate spojenia predpokladajú len prázdne záznamy, ktoré neobsahujú žiadne dáta. Takýto Proof-of-History 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.

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 kaž-

[3] https://github.com/solana-labs/solana/blob/master/sdk/program/src/clock.rs

dého 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).

Z toho vyplýva, že čím viac je vetva používaná, tým 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.

3.2.5 Model motivácie

Hlasy v konsenzus protokole sú zasielané v podobe normálnych transakcií. Validátori teda musia za svoje hlasy v konsenze platiť poplatky za transakcie. Všetky transakčné poplatky v novom bloku sú rozdelené na dva fixne veľké diely (α, β), ktoré sa môžu vývojom blockchainu meniť a na počiatku sú nastavené na hodnotu $(\frac{1}{2}, \frac{1}{2})$. Tvorca nového bloku (Proof-of-History generátor) získava za odmenu diel α a diel β je zničený (anglicky *burned*).

Validátori dostanú za validovanie globálneho stavu blockchainu fixnú kompenzáciu. Táto odmena kompenzuje poskytnutie ich výpočtových zdrojov použitých na overovanie a hlasovanie. Kompenzácia je umelo vytvorená a spôsobuje infláciu hodnoty blockchainu. Vyplácanie týchto odmien prebieha vždy za obdobie jednej epochy.

3.2.6 Teoretická analýza

Synchronizácia v čase pomocou Proof-of-History umožňuje veľkú efektivitu a rýchlosť finalizácie transakcií. Solana používa na uznesenie konsenzu obdobu PBFT protokolu, ktorú nazvala Tower BFT. PBFT má časovú zložitosť $O(n^2)$ pre n uzlov, pretože broadcast vysiela vodca a následne aj validátori. Tower BFT má časovú zložitosť lineárnu pretože broadcast vysiela vždy len jediný uzol siete (Proof-of-Hisory generátor). Na druhej strane, Solana je závislá od vodcu pretože on jediný pridáva v aktuálnom slote obsah do blockchainu. Vodca sa každý slot mení, ale rozvrh vodcov je známy na viac ako dva dni dopredu. Útočník teda môže vykonať DoS útok na vodcu (viď útok 2.8.8).

Útočník nemôže ovplyvniť voľbu vodcu v svoj prospech inak ako vložením väčšieho množstva zdrojov pretože inicializačný vektor pre pseudo náhodnú voľbu je preňho neovplyvniteľný (viď sekcia 3.2.3). Potenciálnou nevýhodou tohto blockchainu je aj fakt, že každý validátor musí byť neustále dostupný.

3.3 Ouroboros

Ouroboros je Proof-of-Stake protokol, ktorý je používaný v kryptomene Cardano. Ouroboros je konsenzus založený na náhodnej voľbe. Protokoly Harmony (sekcia 3.1) a Solana (sekcia 3.2) používajú hlasovací konsenzus. Naproti tomu, Ouroboros konsenzus funguje podobne ako známy blockchain Bitcoin (sekcia 2.5.2). Nový blok pridá uzol zvolený neobjektívnou náhodnosťou. Ouroboros, na rozdiel od Bitcoinu, je ale Proof-of-Stake protokol. Voľba teda nie je založená na riešení náročného výpočtového problému, ale závisí od podielu jednotlivých uzlov. Uzly s veľkým podielom teda budu najpravdepodobnejšie zvolené za tvorcu bloku. Ouroboros konsenzus je uznesený, ak sa viac ako 50 % (majorita vlastní-

kov podielu) zhodne na stave blockchainu. V tejto sekcii je vysvetlený princíp fungovania protokolu Ouroboros na základe oficiálnej publikácie tohto projektu [30].

3.3.1 Synchronizácia v čase

Protokol Ouroboros pracuje s časom v dvoch jednotkách. Najmenšia časová jednotka je *slot*. Za jeden slot je vytvorený maximálne jeden blok (približná dĺka 1 000 ms). Slot je najmenšia časová jednotka na synchronizáciu uzlov v sieti. Druhou a väčšou časovou jednotkou je *epocha*, ktorú tvorí pevný počet slotov R. Pre Cardano[4] je R rovné 432 000 slotom (približne päť dní).

3.3.2 Protokol

Pre každý slot je určený vodca podľa dopredu známeho rozvrhu. Proces určenia rozvrhu vodcov je popísaný v sekcii 3.3.4. 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 tohto 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 užívateľ vykonáva nasledujúce činnosti:

- Zber validných transakcií zo siete.
- Zber všetkých vetiev blockchainu, 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 blockchain (rovnaký princíp ako Bitcoin).
- Ak je užívateľ aktuálny vodca, tak vytvorí nový blok z transakcií ktoré získal. Blok pridá na koniec blockchainu a distribuuje ho. Vodca nemusí vždy publikovať nový blok, a teda slot môže byť prázdny. Jeden slot ale môže publikovať maximálne jeden blok.

3.3.3 Schéma delegovania

Ouroboros protokol vyžaduje, aby bol vlastník podielu neustále online, ak sa chce podieľať na tvorbe nových blokov. Táto požiadavka môže byť pre niektorých užívateľov 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* [9]. 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.

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

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

3.3.4 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 3.3.5) a PVSS (viď 3.3.6).

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 Proof-of-Stake princíp. Napríklad, ak v predchádzajúcej epoche vlastní užívateľ 2 % podielu, tak má v aktuálnej epoche práve 2 % šancu stať sa vodcom slotu.

3.3.5 Coin Tossing protokol

Coin Tossing [8] 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 3.3.6).

3.3.6 PVSS

PVSS (*Publicly Verifiable Secret Sharing*) [21] 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.

3.3.7 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 blockchainu 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 blockchainu $Open(r_i, u_i)$.

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 $u_1 \oplus u_2 \oplus ... \oplus u_R$.

3.3.8 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 3.2, 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.

$$T_{all} = \sum_{i=1}^{R} \sum_{k \in T_i} t_k \tag{3.2}$$

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 3.3.

$$\alpha_i = \frac{|\{k \,|\, p_i = U_k\}|}{R} T_{all} \tag{3.3}$$

Odmena podielnika teda nevychádza z transakcií, ktoré sú pridané do blockchainu 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.

3.3.9 Teoretická analýza

Ouroboros používa pravidlo najdlhšieho reťazca (anglicky *longest chain rule*) na voľbu správneho reťazca v prípade vetvenia. Útočník nemôže efektívne vytvoriť dostatočne dlhý alternatívny reťazec pretože jednotlivé bloky musia byť podpísané správnym vodcom daného slotu. Útočník by teda potreboval privátne kľúče všetkých vodcov slotov od bodu, v ktorom by chcel vytvoriť vetvenie.

Voľba vodcu slotu je skutočne náhodná. Voľbu vodcu nie je možné ovplyvniť inak ako investovaním väčšieho množstva zdrojov do blockchainu (viď sekcia 3.3.7). Tento princíp je z hľadiska Proof-of-Stake správny.

Útok zdvojenia výdavkov nie je možné uskutočniť (dôkaz viď [30], kapitola 5, teorém 5.5).

3.4 Teoretické porovnanie protokolov

Teoretická analýza protokolov Harmony (seckia 3.1), Solana (sekcia 3.2) a Ouroboros (sekcia 3.3) je zhrnutá v tabuľke 3.5. Tabuľka porovnáva tieto tri protokoly z hľadiska shardingu, synchronizácie v čase, tvorby nového bloku, modelu motivácie, vetvenia a odolnosti voči rôznym útokom.

Môžeme vidieť, že len protokol Harmony používa škálovateľnosť pomocou shardingu. Pre ovládnutie jedinej shardy útočník potrebuje viac ako $\frac{1}{3}$ celkového podielu.

Na synchronizáciu v čase používajú všetky tri protokoly koncept epochy a slotu. Solana je hlasovací protokol, ktorý dokáže uzniesť konsenzus za 400 ms. Za hodinu potom vygeneruje približne 8 750 blokov. Protokol Harmony za hodinu vygeneruje približne 1 820 blokov v každom sharde. Aktuálne má štyri shardy, a teda vygeneruje za hodinu spoločne 7 280 blokov. Ouroboros za hodinu vygeneruje 3 600 blokov. Z tohto hľadiska poskytuje Harmony najväčšiu priepustnosť dát, pretože nepotrebuje skracovať dobu slotu na zvýšenie priepustnosti (postačuje navýšiť počet shardov).

Pre vytvorenie nového bloku potrebujú všetky tri protokoly určiť uzol, ktorý blok vytvorí (tzv. vodcu). Túto dôležitú roľu prideľujú všetky protokoly náhodnou voľbou. Harmony a Ouroboros používajú algoritmus na vytvorenie distribuovanej náhodnosti, čím zabraňujú útočníkom ovplyvňovať výsledok volieb. Solana používa pseudonáhodný generátor, ktorý je inicializovaný číslom aktuálneho slotu (neovplyvniteľná monotónne rastúca hodnota). Pravdepodobnosť zvolenia za vodcu je vždy priamo úmerná podielu (princíp Proof-of-Stake). Ouroboros používa pravidlo *longest chain rule* na určenie správneho reťazca. Uznesenie konsenzu má teda konštantnú časovú zložitosť vzhľadom k počtu uzlov siete. Harmony a Solana používajú hlasovacie protokoly na uznesenie konsenzu pre novo publikovaný blok. Časová zložitosť hlasovania je potom lineárna vzhľadom k počtu uzlov.

Solana umožňuje potenciálne binárne vetvenie pre každý nový blok. Každý validátor si musí udržiavať všetky možné vetvy, čo kladie nemalé nároky na zdroje validátorov. Solana ale zamedzuje podielnikom hlasovať v rôznych vetvách súčasne pod trestom straty všetkých zdrojov.

Z hľadiska potenciálnych útokov môže pre všetky tri protokoly predstavovať problém DoS útoku na vodcu. Všetky tri protokoly vytvoria zoznam vodcov na celú epochu dopredu. Útočník potom môže vykonať DoS útok na vodcu, bez ktorého protokol nepublikuje žiadny

nový obsah do blockchainu. V prípade Harmony je dokonca rovnaký vodca v každom sharde po dobu 18 hodín. Ouroboros je konsenzus protokol založený na náhodnej voľbe a umožnuje potenciálne vetvenie blockchainu, ktoré spomaľuje finalizáciu transakcií.

Na celkové ovládnutie Harmony alebo Solana blockchainu je potrebné vlastniť viac ako $\frac{2}{3}$ podielu. Na prekazenie konsenzu postačuje viac ako $\frac{1}{3}$. Ouroboros požaduje na ovládnutie aj prekazenie konsenzu majoritu podielu (viac ako polovicu).

		Harmony	**Solana**	**Ouroboros**
Sharding		ano	nie	nie
Synchronizácia v čase	Časové jednotky	epochy a sloty	epochy a sloty	epochy a sloty
	Dĺžka epochy (pocet slotov)	32 768	420 000	432 000
	Približná dĺžka slotu (1 vygenerovaný blok)	2 sec	400 ms	1 sec
	Približná dĺžka epochy	18,2 hod	2 dni	5 dní
Tvorba bloku	Určenie tvorcu bloku (vodcu)	jeden vodca na epochu v každom sharde	rozvrh vodcov na epochu	rozrh vodcov na epochu
	Spôsob určenia vodcu	náhodná voľba (distribuované generovanie náhodnosti)	voľba pre každý slot pomocou pseudonáhodného generátoru s neovplivniteľným IV	náhodná voľba (distribuované generovanie náhodnosti)
	Pravdepodobnosť podielnika stať sa vodcom je priamo úmerná jeho podielu v minulej epoche (Proof-of-Stake)	áno	áno	áno
	Konsenzus na bloku publikovanom vodcom	FBFT (BFT hlasovanie)	TowerBFT (BFT hlasovanie)	longest chain rule (majorita)
	Časová zložitosť uznesenia konsenzu vzhľadom k počtu uzlov N	O(N)	O(N)	O(1)
Model motivácie	Odmeny za tvorbu bloku	poplatky za transakcie	poplatky za transakcie + inflácia	poplatky za transakcie
	Zisk z transakčných poplatkov	validátori úmerne ich podielu	1/2 vodcovi, 1/2 je zničená	podielnici úmerne ich podielu
Vetvenie		nevzniká (potreba ovládnutia majority podielu)	potenciálne binárne každým blokom	možné
Odolnosť voči útokom	Ovplyvnenie volieb vodcu	jedine zvýšením podielu (PoS)	jedine zvýšením podielu (PoS)	jedine zvýšením podielu (PoS)
	DoS útok na vodcu	možný a jednoducho uskotočniteľný	možný	možný
	Prekazenie konsenzu útočníkmi	viac ako 1/3 podielu	viac ako 1/3 podielu	viac ako 1/2 podielu
	Ovládnutie konsenzu útočníkmi	viac ako 2/3 podielu	viac ako 2/3 podielu	viac ako 1/2 podielu
	Long range útok	-	-	nie je možné vytvoriť alternatívny dlhší reťazec
	Double spending útok	potreba ovládnutia konsenzu	trestané odobratim zdrojov	potreba ovládnutia konsenzu
	Útok na podskupinu uzlov	viac ako 1/3 celkového podielu	-	-
CAP teorém		CP	CP	AP

Obr. 3.5: Porovnanie vlastností zvolených troch Proof-of-Stake protokolov.

Kapitola 4

Implementácia simulátoru blockchainu

S narastajúcou používanosťou blockchainu narastá aj množstvo bezpečnostných incidentov spojených s touto technológiou[1]. Medzi rokmi 2011 až 2021 bolo zaznamenaných 120 útokov na bezpečnosť blockchainu. Súčasne bolo objavených 73 zraniteľností v rôznych blockchain technológiách. Za toto obdobie bolo ukradnutých približne 12 miliárd dolárov z rôznych kryptomien, ktoré používajú blockchain ako svoj základ. Tieto čísla poukazujú na potrebu preverovať túto technológiu v oblasti bezpečnosti a výkonnosti. Prirodzene sa pre tento účel naskytá simulácia. Táto kapitola sa zaoberá simuláciou blockchain technológie.

Sekcia 4.1 popisuje a porovnáva aktuálne dostupné simulátory blockchain technológie. Pre simuláciu protokolov Harmony, Solana a Ouroboros nie je vytvorený úplne nový simulátor. Simulácia celej technológie blockchain s dostatočne presným modelom je rozsiahla úloha nad rámec tejto práce. Pre analýzu Proof-of-Stake konsenzu je použitý a rozšírený už existujúci simulačný nástroj Wittgenstein.

V sekcii 4.2 je následne prezentovaný vytvorený simulačný nástroj, jeho štruktúru, postup a abstrakciu simulácie. Na záver je poskytnuté porovnanie veľkosti siete a rozloženia podielu v skutočne nasadených blockchain systémoch, ktoré používajú protokoly Harmony, Solana a Ouroboros. Výsledky porovnania sú použité v simulačných experimentoch popísaných v kapitole 5.

4.1 Prehľad existujúcich simulátorov

Táto sekcia analyzuje sedem simulátorov blockchainu. V závere je vykonané porovnanie najvhodnejších nástrojov a je zvolený jeden pre ďalšiu prácu. Porovnanie je založené na publikáciách autorov jednotlivých nástrojov a analýze zdrojového kódu. K niektorým s týchto simulátorov existujú práce, ktoré experimentálne vyhodnotili ich presnosť (viď [38, 20]). Tieto výsledky sú v porovnaní taktiež zhrnuté.

4.1.1 SimBlock

SimBlock je simulátor založený na diskrétnej simulácii. Je implementovaný v programovacom jazyku Java a jeho zdrojový kód je voľne dostupný[2]. Simulátor podporuje Proof-of-

[1] Bezpečnostný report blockchain technológie za roky 2011 až 2021: https://crystalblockchain.com/security-breaches-and-fraud-involving-crypto/

[2] Apache License, Version 2.0

Work protokoly Bitcoin, Litecoin, a Dogecoin. Nástroj ďalej umožňuje modulárne zmeniť konsenzus protokol. Posledná stabilná verzia pridala jednoduchú implementáciu Proof-of-Stake konsenzu[3].

Autori tohto 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 tohto experimentu autori zhodnotili, že ich simulátor napodobuje reálne blockchain systémy s porovnateľnou presnosťou ako aktuálne dostupné simulátory.

Autori ďalej navrhli úpravu algoritmu na voľbu susedných uzlov (anglicky *neighbor node selection algorithm*) v protokole Bitcoin. Navrhnutú úpravu simulovali a vyhodnotili, že ich vylepšenie algoritmu zvyšuje priepustnosť transakcií. Touto simuláciou bol demonštrovaný význam tohto simulačného nástroja. Simulátor umožňuje experimentálne modifikovať protokol s cieľom zlepšiť výkonnosť a bezpečnosť [6].

Výhodou tohto simulačného nástroja 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*).

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 [20].

4.1.2 Bitcoin Simulator

Bitcoin Simulator je *open source*[4] simulačný nástroj implementovaný v programovacom jazyku C++. Simulátor je postavený nad platformou NS-3[5], ktorá slúži na diskrétnu simuláciu internetových systémov. Na tejto platforme je teda postavená simulácia P2P siete blockchainu. Tento simulátor je určený a bol vyvinutý na experimentovanie s Proof-of-Work protokolmi. Aktuálne podporuje protokoly Bitcoin, Litecoin, a Dogecoin. Simulátor teda nepodporuje Proof-of-Stake konsenzus. Avšak, v minulosti už bol tento nástroj použitý treťou stranou, ktorá rozšírila implementáciu o Proof-of-Stake protokoly Algorand, Casper FFG a Gasper (viď [12]).

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. Z tohto hľadiska bol simulátor vyhodnotený ako pomerne presný [22].

Medzi výhody tohto simulačného nástroja patrí jeho rozsiahlosť, ktorá poskytuje veľkú škálu vstupných parametrov a taktiež veľké množstvo nameraných výstupných metrík. Z hľadiska simulácie Proof-of-Work protokolov umožňuje simulátor rozlišovať rôzne typy uzlov (bežný uzol a miner uzol). Rozlišovanie rôznych uzlov umožňuje tomuto nástroju simu-

[3] https://github.com/dsg-titech/simblock/releases/tag/v0.8.0
[4] https://arthurgervais.github.io/Bitcoin-Simulator/index.html
[5] https://www.nsnam.org/

lovať aj zlomyseľné chovanie niektorých uzlov. Tento simulátor teda umožňuje analyzovať aj bezpečnosť konsenzu z hľadiska ťažby blokov (útoky *selfish-mining* a *double-spending*) [20].

4.1.3 BlockSim

BlockSim je *open source* [6] simulátor vytvorený v programovacom jazyku Python a je určený na diskrétnu simuláciu blockchainu. Autori definujú tri základné ciele tohto simulátora: všeobecnosť, rozširovateľnosť a jednoduchosť. Všeobecnosť architektúry simulátora umožňuje simuláciu veľkého množstva blockchain 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 blockchainu, ktorú autori nazvali *base model*. Base model je zdieľaná vrstva pre všetky konkrétne protokoly pretože pokrýva všetky základné prvky blockchainu (uzly, transakcie, bloky, reťazec blokov, vetvenie). Nad touto vrstvou je potom možné vytvoriť implementáciu konkrétnych protokolov. Autori demonštrovali túto flexibilnú architektúru tak, že vytvorili nad base modelom simuláciu dvoch Proof-of-Work protokolov (Bitcoin a Ethereum). Simulátor by teda mal byť priamočiaro rozšíriteľný aj o simuláciu Proof-of-Stake protokolov. Avšak, simulátor neumožňuje analýzu špecifickej sekvencie správ v prípade hlasovacieho konsenzu [4].

Najväčšou výhodou tohto simulátoru je jeho jednoduchá architektúra, ktorá umožňuje rozšíriteľnosť o ďalšie blockchain protokoly. Simulátor pokrýva sieťovú, dátovú a konsenzus vrstvu. Z hľadiska sieťovej vrstvy je možné simulovať latenciu a *bandwidth* 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. Simulátor síce podporuje dátovú vrstvu, ale len limitovane. Implementácia transakcií neobsahuje žiaden model účtov (napríklad UTXO[7] pre Bitcoin) [20].

4.1.4 VIBES

VIBES je *open source*[8] simulátor blockchainu s architektúrou klient-server. Server je aplikácia vytvorená v programovacom jazyku Scala. Klient tvorí webové rozhranie, ktoré poskytuje množstvo grafických výstupov simulácie (analýza *double-spending* útoku, štatistiky propagácie transakcií a ďalšie). Simulátor implementuje výhradne Proof-of-Work konsenzus. Podľa autora je možná implementáciu rozšíriť o Proof-of-Stake konsenzus. Tento simulátor je zameraný na všeobecnejšiu simuláciu blockchainu v P2P sieti a neobsahuje implementáciu žiadneho konkrétneho protokolu [42].

Výhodu tohto 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 4.1.3), transakcie neobsahujú žiaden model účtov.

Nevýhodou tohto nástroja je aplikácia konštantného oneskorenia pre propagácie blokov a transakcií v sieti. Dôsledkom je nepresnosť simulácie z hľadiska výkonnostnej analýzy blockchainu. Autor tvrdí, že simulátor umožňuje simuláciu rozsiahlych sietí s viac ako 10 000 uzlami. V skutočnosti je takáto simulácia časovo pomerne náročná pretože tento nástroj používa centrálny koordinátor, ktorý tvorí úzke hrdlo simulácie [20].

[6] https://github.com/maher243/BlockSim
[7] https://river.com/learn/terms/u/unspent-transaction-output-utxo/
[8] https://github.com/i13-msrg/vibes

4.1.5 Shadow

Shadow[9] 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 (v našom prípade ide o konkrétny blockchain protokol). Skutočnú implementáciu blockchainu je pritom nutné modifikovať len minimálne. Takýmto spôsobom je pomocou tohto nástroja simulovaná napríklad sieť Tor [29].

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 tohto 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 tohto nástroja. Spomínaný doplnok je použiteľný jedine pre simuláciu protokolu Bitcoin. Na simuláciu ľubovoľného iného protokolu by bolo potrebné vytvoriť nový doplnok [34].

4.1.6 FoBSim

FoBSim je *open source*[10] simulátor, ktorého model pokrýva dve oblasti:

- *Fog Computing* je fyzický alebo 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 integrovaná s fog computing by mala viesť k decentralizácii tejto služby.

Tento simulačný nástroj je vyvíjaný za účelom experimentovania v oblasti integrácie týchto dvoch služieb do jednej. Pre našu prácu je zaujímavá len časť, ktorá simuluje blockchain.

Simulátor poskytuje model blockchainu, ktorý obsahuje nasledujúcu funkcionalitu: sieťová vrstva, konsenzus algoritmy, odmeňovanie užívateľov za vytváranie nových blokov (anglicky *incentive mechanism*), paralelná ťažba blokov a distribúcia správ v sieti [7].

Autori deklarujú, že simulátor implementuje tri konsenzus protokoly: Proof-of-Work, Proof-of-Stake a Proof-of-Authority. Nikde ale nie je definované o aké konkrétne protokoly ide. Nie je teda možbné vyhodnotiť presnosť simulátora z hľadiska reálnych blockchain protokolov. Presnosť simulácie je ďalej nejasná aj preto, že ide o nový projekt a neexistuje zatiaľ žiadna práca, ktorá by sa zaoberala jeho vyhodnocovaním. Samotná implementácia predstavuje približne 3 000 riadkov kódu v programovacom jazyku Python[11], čo je dramaticky menej voči iným nástrojom.

4.1.7 Wittgenstein

Wittgenstein [14] je nástroj určený priamo na simuláciu konsenzus protokolov alebo všeobecne distribuovaných algoritmov. Nástroj umožňuje definovať sieť s vlastným chovaním uzlov. Uzly je možné geograficky rozmiestniť, a tak ovplyvňovať ich latenciu. Simulátor

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

		Simulátor			
		SimBlock	**Bitcoin Simulator**	**BlockSim**	**Wittgenstein**
Dátová vrstva	Generovanie transakcií	✓	✓	✓	✓
Konsenzus vrstva	Proof-of-Work	✓	✓	✓	✓
	Proof-of-Stake	×	×	✓	✓
	Simulácia útočiacich uzlov	×	✓	×	✓
	Interval pre distribúciu bloku	✓	✓	✓	✓
Sieťová vrstva	Veľké siete (>1000 uzlov)	✓	✓	×	✓
	Geografická distribúcia uzlov	✓	✓	✓	✓
	Bandwith	✓	✓	✓	✓
	Latency	✓	✓	✓	✓
	Veľkosť transakcie	✓	✓	✓	✓
Výstup simulácie	Proof-of-Work metriky	✓	✓	✓	×
	Proof-of-Stake metriky	×	×	×	✓
	Througput (TPS)	×	×	×	×
	Throughput (bytes)	×	×	×	✓
Iné vlastnosti	Programovací jazyk	Java	C++	Python	Java
	Vytvorenie projektu	06/2019	04/2016	04/2019	10/2018
	Posledná zmena v repozitári	02/2021	10/2016	05/2021	01/2020
	Podporované protokoly	Bitcoin Litecoin Dogecoin	Bitcoin Litecoin Dogecoin	Bitcoin Ethereum	Ethereum CasperIMD Dfinity Handel

Tabuľka 4.1: Porovnanie blockchain simulátorov.

obsahuje databázu geografických lokalít pre uzly. Taktiež je možné definovať časť uzlov ako škodlivých a určiť pre ne vlastné chovanie. Projekt poskytuje jednoduché rozhranie pre definíciu ľubovoľného distribuovaného algoritmu založeného na zasielaní správ. Simulátor obsahuje množstvo už naimplementovaných distribuovaných algoritmov a pre každý poskytuje sadu testov (napríklad, Handel, CapserIMD, Snowflake, Dfinity a ďalšie). Simulácia ako výstup poskytuje rôzne metriky relevantné pre hlasovací konsenzus, a to zvlášť pre každý uzol (množstvo zaslaných správ, prenesených dát a podobné). Ďalej je vstavaná podpora pre rôzne grafické výstupy (vykreslenie distribúcie uzlov do geografickej mapy, grafy priepustnosti dát a podobné).

4.1.8 Porovnanie dostupných simulátorov

Táto sekcia zhodnocuje simulačné nástroje popísané v tejto kapitole. Z týchto nástrojov sú porovnané vlastnosti štyroch najvhodnejší. Na základe ich vlastností je na záver zvolený simulátor, ktorý je ďalej rozšírený o simuláciu Proof-of-Stake protokolov Harmony, Solana a Ouroboros.

Pre potreby 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 blockchain sieťou. Dostatočnou presnosťou z hľadiska našej práce sa myslí, že simulátor vierohodne napodobuje sieťovú a konsenzus vrstvu. Zaoberáme sa analýzou konsenzus protokolov, a teda dátová vrstva už nie je pre simuláciu kľúčová.

Tieto požiadavky spĺňajú v najväčšej miere štyri simulačné nástroje: SimBlock, Bitcoin Simulator, BlockSim a Wittgenstein. Tabuľka 4.1 sumarizuje vlastnosti týchto štyroch simulátorov na sieťovej, konsenzus a dátovej vrstve. Ďalej sú simulátory porovnané z hľadiska výstupov a ostatných vlastností. Na základe tohto zhodnotenia môžeme povedať, že tieto štyri nástroje poskytujú podobne rozsiahle možnosti. Ale porovnanie z hľadiska dátovej, konsenzus a sieťovej vrstvy je na pomerne vysokej úrovni a zanedbáva komplexnosť jednotlivých vrstiev. Napríklad, sieťová vrstva nástroja SimBlock je značne komplexnejšia, než simulácie sieťovej vrstvy nástroja BlockSim a Wittgenstein. Dátová vrstva nie je pre túto prácu podstatná, a postačuje ak je možné periodicky generovať transakcie z určeného rozdelenia (napríklad normálneho). Túto vlastnosť všetky simulátory podporujú, prípadne je ich pomerne jednoduché modifikovať, aby požadovanú funkcionalitu podporovali. Naopak, najdôležitejšia je podpora simulácie konsenzus vrstvy. Na tejto vrstve je jedine Wittgenstein zameraný na Proof-of-Stake simuláciu. Ostatné nástroje sú určené na Proof-of-Work a jedine BlockSim poskytuje veľmi jednoduchú podporu Proof-of-Stake. Simulátory SimBlock a BlockSim neumožňujú definovať škodlivé uzly. Vďaka tomu sú pre účely analýzy zraniteľností nevhodnou voľbou. Táto práca nemá vysoké nároky na simuláciu sieťovej vrstvy. Nástroj BlockSim je ako jediný nevhodný z hľadiska sieťovej vrstvy. Dôvodom je, že neumožňuje efektívne simulovať dostatočne veľké siete (1 000 a viac uzlov). Z hľadiska ostatných vlastností poskytuje Wittgenstein najvhodnejšie výstupy simulácie (podpora vykresľovať grafy a štatistiky spojených s hlasovaním). Ďalším aspektom je stav vývoja jednotlivých nástrojov. Bitcoin Simulator je projekt už päť rokov neudržiavaný. Ostatné nástroje sú pomerne aktuálne.

V tejto práci je použitý nástroj Wittgenstein, ktorý je rozšírený o požadovanú funkcionalitu. Tento nástroj je ako jediný priamo určený pre simulácií hlasovacích protokolov s podporov Proof-of-Stake. Najväčšou slabinou tohto nástroja je jeho sieťová vrstva. Simulátor poskytuje pomerne jednoduchú implementáciu P2P siete (napríklad v porovnaní s Bitcoin simulátorom). Na druhej strane, simulátor umožňuje veľmi efektívne zadefinovať rôzne škodlivé scenáre v ktorých sa podmnožina uzlov chová nepoctivo.

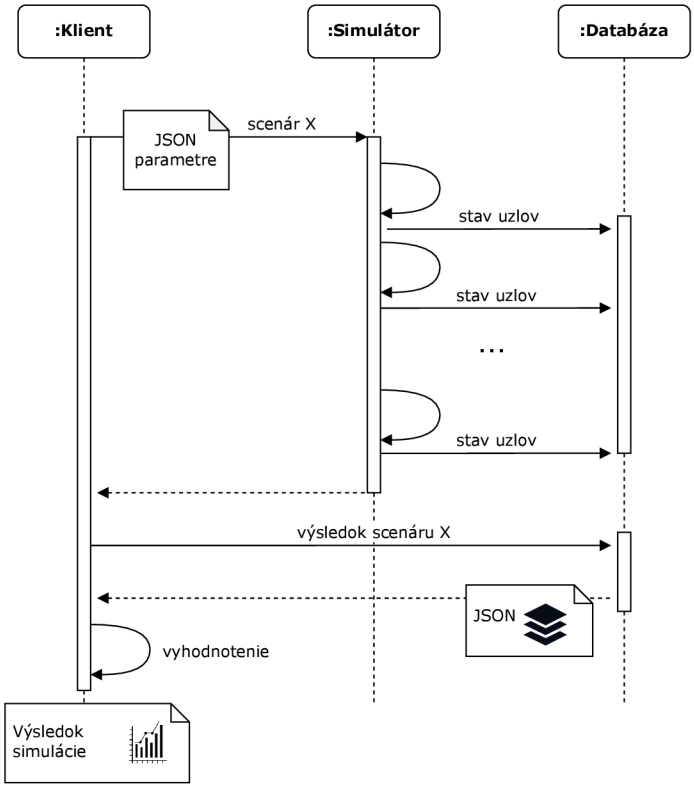
4.2 Vytvorené riešenie

Vytvorený nástroj obsahuje okrem samotného rozšírenia simulátora Wittgenstein o zvolené protokoly aj modul pre spracovanie a vizuálne zobrazenie výsledkov jednotlivých simulačných scenárov. Vytvorené riešenie má architektúru klient-server:

- Na strane serveru beží samotný simulátor. Ten má definované REST API, pomocou ktorého sa dá spustiť simulácia s definovanou vstupnou konfiguráciou. Jednotlivé uzly simulovanej siete priebežne zaznamenávajú svoj stav. Dáta produkované simuláciou rozsiahlych sietí v dostatočne dlhom časovom intervale sa objemovo pohybujú v jednotkách gigabajtov. Preto si ich server priebežne ukladá do databázy[12].
- Klientská aplikácia spúšťa simuláciu na serveri a po jej dokončení analyzuje dáta uložené v databáze nástrojmi určenými na dolovanie znalostí[13] z rozsiahlych dátových sád.

Sekvenčný diagram na obrázku 4.1 ukazuje proces simulácie. Z užívateľského hľadiska nie je potrebné serverovú časť nijako konfigurovať. Server (simulátor aj databáza) je kon-

[12] https://www.mongodb.com/
[13] https://pandas.pydata.org/



Obr. 4.1: Sekvenčný diagram zobrazuje proces simulácie a vyhodnotenie jej výsledkov.

tajnerizovaný technológiou *docker*[14], ktorá si sama nainštaluje všetky potrebné závislosti do svojho virtuálneho prostredia. Následne užívateľ inicializuje simuláciu zvoleného experimentu pomocou klienta. Po dokončení simulácie sú výsledné grafy dostupné na strane klienta.

Klient definuje protokol, ktorý bude simulovaný. Ďalej pre zvolený protokol definuje simulačný scenár prostredníctvom sady konfigurovateľných parametrov. Podporované parametre sú zhrnuté v tabuľke 4.3. Tieto parametre zašle klient serveru pomocou definovaného API vo formáte JSON. Časť parametrov je nezávislá od zvoleného protokolu a definuje veľkosť siete, dĺžku aj počet slotov a epoch, rozloženie podielu medzi uzly, množstvo transakcií, či veľkosť bloku. Harmony protokol navyše definuje parametre spojené s shardingom. Ďalšie parametre definujú scenár DoS útoku na vodcov hlasovania. V prípade protokolu Ouroboros je možné vetvenie parametrizovať.

4.2.1 Implementácia

Simulátor poskytuje nasledujúce moduly, ktoré poskytujú základnú funkcionalitu potrebnú pre simuláciu konsenzu v blockchaine:

- *Protokol* je vstupné rozhranie pomocou, ktorého sa definuje konkrétny distribuovaný algoritmus (v našom prípade Harmony, Solana a Ouroboros). Každý protokol sa skladá zo siete, uzlov a definície správ.

[14] https://www.docker.com/

Kategória podielu	Interval [%]		
	Solana	**Harmony**	**Ouroboros**
Veľmi malý	$(0.000; 0.017\rangle$	$(0.000; 0.100\rangle$	$(0.000; 0.010\rangle$
Malý	$(0.017; 0.025\rangle$	$(0.100; 1.000\rangle$	$(0.010; 0.050\rangle$
Stredný	$(0.025; 0.035\rangle$	$(1.000; 2.000\rangle$	$(0.050; 0.100\rangle$
Veľký	$(0.035; 1.000\rangle$	$(2.000; 16.00\rangle$	$(0.100; 0.250\rangle$
Veľmi veľký	> 1.000	> 16.000	> 0.250

Tabuľka 4.2: Rozdelenie uzlov na päť kategórií z hľadiska veľkosti vlastneného podielu (zvlášť pre každý protokol).

- *Sieť* sa skladá z množiny uzlov. Simulátor umožňuje definovať štruktúru P2P siete. Ďalej je možné definovať geografické rozloženie uzlov, ktoré bude mať dopad na latenciu zasielaných správ. V sieti je možné zasielať broadcastové aj unicastové správy.
- *Uzol* definuje zasielanie správ, vlastný stav a identifikácia. Ďalej je možné rozšíriť funkcionalitu uzlu o ľubovoľné ďalšie vlastnosti. Pre konkrétny protokol je možné definovať rôzne uzly s rozličným chovaním. Samotný simulátor uchováva štatistiky o každom uzle (napríklad počet odoslaných a prijatých správ, bytov).
- *Správa* je rozhranie, ktoré umožňuje definovať formát a veľkosť zasielaných správ. Toto rozhranie umožňuje definovať protokol zasielania správ (inak povedané postupnosť správ v konkrétnom algoritme).

Pre konkrétne protokoly bola vždy rozšírená implementácia uzlu o ďalšiu špecifickú funkcionalitu. Napríklad, Solana uzol musí rozlišovať medzi vodcom a validátorom. U protokolu Harmony bola pridaná podpora pre sharding. Samotný Harmony sharding je súčasťou konsenzus vrstvy, a teda štruktúru shardov nezabezpečuje sieť. Každý uzol si vo svojom stave pamätá ku ktorým shardom náleží. Pre simuláciu Proof-of-Stake bola pridaná reprezentácia podielu v sieti. Kvôli pamäťovým nárokom simulácia uchováva distribúciu podielu globálne a jednotlivé uzly k nej len pristupujú.

Wittgenstein simulátor umožňuje definovať komunikáciu konsenzus protokolu na sieťovej vrstve funkcionálne. Pre sieťovú komunikáciu postačuje určiť komunikujúce strany a zasielanú správu. Samotný simulátor potom zaručí zaslanie správy v zvolenom čase s príslušnou latenciou. Kód 4.1 zobrazuje rozhranie Wittgenstein simulátoru pre definície komunikácie medzi uzlami. Na riadku 2 je ukážka definície správy v komunikačnom protokole pre uzol v Solana sieti. Definícia správy sa vždy vzťahuje ku konkrétnej triede uzlu (`SolanaNode`). Správa zvyčajne prenáša dáta (viď riadok 3). Pre každú správu je potrebné definovať dve metódy: `action` a `size`. Odosielateľ môže správu poslať všetkým, konkrétnej skupine či jedinému uzlu (na riadku 20 je príklad zaslania správy všetkým). V čase prijatia správy príjemcom sa vyvolá metóda `action` (riadok 7), ktorá zavolá obslužnú metódu pre danú triedu správy na strane príjemcu (riadok 24). Metóda `size` definuje veľkosť správy v bytoch a slúži na záznam štatistík o prenesených dátach zvlášť pre každý uzol v sieti.

4.2.2 Namerané vlastnosti protokolov

Každý z troch simulovaných protokolov má množstvo parametrov, ktoré budú mať veľký vplyv na jeho výsledné vlastnosti. Parametre s najväčším vplyvom sú veľkosť siete, rýchlosť generovania blokov a distribúcia podielu. Simulácia používa v niektorých scenároch

```java
/** Specific message definition for SolanaNode. */
public class ExampleMessage extends Message<SolanaNode> {
    private Data data; // message content
    public ExampleMessage(Data data) { this.data = data; }

    @Override
    public void action(Network<SolanaNode> network, Node from, Node to){
        to.onExampleMessageReceive(from, data);
    }

    /** for statistics about received / send messages */
    @Override
    public int size() { return data.size(); }
}

/** SolanaNode inherits from general network Node. */
public class SolanaNode extends Node {

    ... // Specific node sends ExampleMessage through broadcast.
    network.sendAll(new ExampleMessage(data), this);
    ...

    /** Handler method for ExampleMessage receive. */
    public void onExampleMessageReceive(SolanaNode from, Data data) { }
}
```

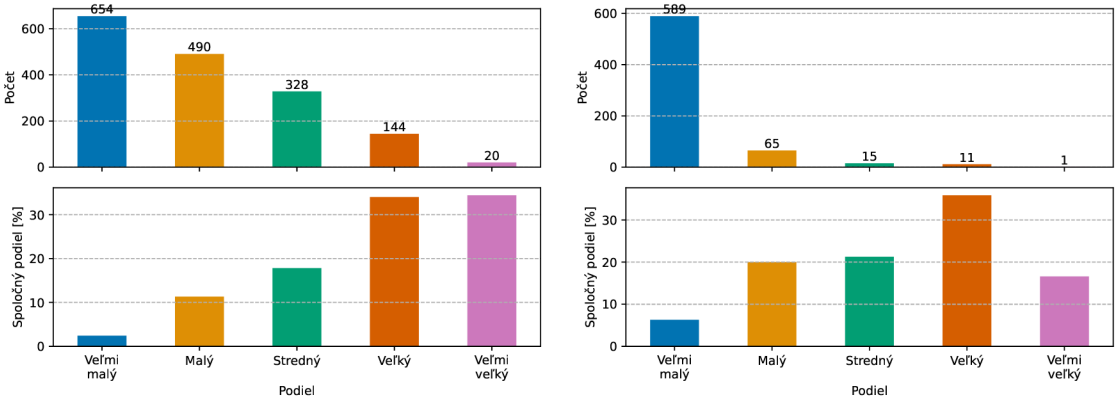
Kód 4.1: Príklad sieťovej komunikácie pre Solana konsenzus v zdrojovom kóde simulátoru Wittgenstein (programovací jazyk Java).

stav odpovedajúci skutočným blockchain sieťam v ktorých sú tieto protokoly používané. V iných experimentoch sú tieto parametre experimentálne menené. Veľkosť siete je menená predovšetkým z dôvodu analýzu výkonnosti a škálovateľnosti. Rýchlosť generovania blokov je vo všetkých experimentoch rovná skutočnej. Distribúcia podielu je zase menená z dôvodu analýzy bezpečnosti. Vo všetkých troch protokoloch nie je distribúcia podielu rovnomerná, čo má značne nepriaznivý dopad na Proof-of-Stake bezpečnosť. Preto niektoré simulačné experimenty uvažujú rovnomerné rozdelenie podielu.

Všetky tri protokoly sú nasadené v reálnych blockchainoch. Parametre a dostupné štatistiky týchto blockchainov boli prevzaté z nasledujúcich zdrojov:

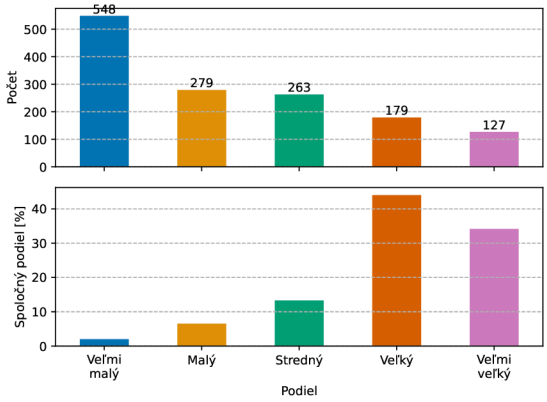
- Solana[15] (ku dňu 22.2.2022): Veľkosť siete je 1 637 uzlov. Jeden blok je generovaný približne každých 400 ms.
- Harmony[16] (ku dňu 24.2.2022): Sieť tvorí 682 uzlov a blok je generovaný každé 2 sekundy.

[15] https://solanabeach.io/
[16] https://harmony.smartstake.io/



(a) Solana

(b) Harmony



(c) Cardano (Ouroboros)

Obr. 4.2: Distribúcia skutočného podielu rozdelená na päť kategórií podľa veľkosti.

- Cardano[17] je blockchain používajúci Ouroboros (ku dňu 4.4.2022): Sieť pozostáva z 1 397 uzlov a blok sa generuje každú sekundu.

Obrázok 4.2 sumarizuje skutočnú distribúciu podielu zvlášť vo všetkých troch blockchain protokoloch. Uzly sú rozdelené do piatich kategórií podľa veľkosti podielu (veľmi malý, malý, stredný, veľký a veľmi veľký podiel). Pre každý protokol sú hranice jednotlivých kategórií rozdielne. Význam kategórií pre každý protokol popisuje tabuľka 4.2. V Solane vlastní len 20 uzlov $\frac{1}{3}$ podielu. V prípade Harmony dokonca jediný uzol vlastní viac ako 16 % podielu. Jedine v prípade Cardano (Ouroboros) nie je veľká časť podielu držaná veľmi malým počtom uzlov. Vo všetkých troch blockchainoch je kategória podielnikov s veľmi malým podielom. Títo podielnici tvoria približne polovicu siete ale jednotlivo vlastnia veľmi malý podiel. Pre tieto uzly je vysoko nepravdepodobné, aby sa stali vodcami slotu. Napríklad, v Cardano sieti je 548 uzlov, ktoré spoločne vlastnia len približne 2 % celkového podielu. Zvyšných 98 % vlastní 849 uzlov. Inak povedané, tretinu siete tvoria uzly, ktoré nedokážu konsenzus takmer vôbec ovplyvniť.

[17] https://adapools.org/

Parameter	Typ	Význam
Všetky protokoly		
`networkSize`	int	Počet uzlov v sieti.
`slotDurationInMs`	int	Dĺžka slotu v ms.
`epochDurationInSlots`	int	Počet slotov v epoche.
`numberOfEpochs`	int	Počet epoch.
`expectedTps`	int	V sieti bude vznikať dostatok transakcií na to, aby bolo možné dosiahnuť TPS stanovené touto hodnotou.
`uniformStakeDistribution`	bool	Ak `true`, tak všetky uzly vlastnia podiel pridelený z rovnomerného rozdelenia. V opačnom prípade je podiel rozdelený tak ako v skutočnej sieti (viď obrázok 4.2).
`txSizeInBytes`	int	Priemerná veľkosť transakcie v bytoch.
`blockHeaderSizeInBytes`	int	Veľkosť hlavičky bloku v bytoch.
`vrfLeaderSelection`	bool	Ak `true`, tak je namiesto rozvrhu vodcov použitá priebežná voľba vodcu pomocou VRF (viď sekcia 5.4). Inak je použitý rozvrh vodcov na epochu.
`mongoServerAddress`	string	Adresa databázy MongoDB.
Harmony		
`numberOfShards`	int	Počet shardov.
`ddosAttack`	bool	DoS útok na vodcov shardov. Útok postupne narastá (počet shardov pod útokom).
`lambda`	int	Nastavenie bezpečnostného parametru λ.
`vdfInSlot`	int	Číslo slotu, v ktorom bude inicializované distribuované generovanie náhodnosti používané pre určenie rozvrhu vodcov (viď sekcia 5.4.2).
`byzantineStake`	float	Interval v rozmedzí $\langle 0, 1\rangle$, ktorý určuje percentuálny podiel byzantských uzol. V sieti sa vyberie taká množina uzlov, aby spoločne vlastnila práve takýto podiel.
Solana		
`numberOfNodesUnderAttack`	int	Počet uzlov v rozvrhu vodcov pod útokom.
`validatorReliability`	float	Priemerná spoľahlivosť uzlu. Ak $1,0$, tak $100\,\%$, ak $0,0$ tak $0\,\%$.
Ouroboros		
`p2pConnectionCount`	int	Počet spojení zo susednými uzlami pre každý uzol v P2P sieti.
`p2pMinimum`	bool	Ak `true`, tak je `p2pConnectionCount` minimálny počet spojení pre uzol. Inak ide o priemerný počet spojení.
`numberOfNodesUnderDos`	int	Počet uzlov v rozvrhu vodcov pod útokom.
`byzantineStake`	float	Rovnako ako pri Harmony.
`forkRatio`	int	Priemerný počet vetiev v prípade byzantského slotu.

Tabuľka 4.3: Prehľad podporovaných vstupných parametrov simulácie pre jednotlivé protokoly.

Kapitola 5

Simulačné experimenty

V nasledujúcej kapitole sú zhrnuté výsledky simulácie protokolov Harmony, Solana a Ouroboros pomocou vytvoreného simulátoru. Záujemca si môže ľubovoľný experiment zreprodukovať. Príloha A obsahuje návod, ktorý vysvetľuje ako simuláciu spustiť (simulácia však môže trvať pre niektoré experimenty aj niekoľko hodín). Ak nie je uvedené inak, experimenty primárne uvažujú veľkosť blockchain siete a rozloženie podielu medzi uzly zhodné zo skutočným (viď sekcia 4.2.2).

V prípade protokolu Solana simulácia experimentuje s hlasovaním, priepustnosťou a škálovateľnosťou (sekcia 5.1). Protokol Harmony je simulovaný z hľadiska shardingu a jeho bezpečnosti a škálovateľnosti (sekcia 5.2). Pre Ouroboros je simulovaný útok na vetvenie blockchainu, ktorý spomaľuje konzistenciu (sekcia 5.3). Ďalej je pre všetky tri protokoly pomocou simulácie demonštrovaná zraniteľnosť v podobe DoS útoku. V sekcii 5.4 je navrhnutá a odsimulovaná modifikácia týchto protokolov, ktorá znižuje ich zraniteľnosť voči spomínanému útoku. Predovšetkým, v prípade protokolu Harmony simulácia tohto návrhu poukazuje na výrazné zlepšenie bezpečnosti oproti pôvodnému riešeniu. Na záver je porovnaná škálovateľnosť protokolov Harmony a Solana vzhľadom k veľkosti siete.

5.1 Solana

Solana je blockchain zameraný na vysokú priepustnosť transakcií. Aktuálna Solana sieť má približne 1 500 uzlov (viď 4.2.2). Simulácia experimentuje aj so sieťami o veľkosti 10 000 uzlov s cieľom preveriť škálovateľnosť Solana konsenzu. Z bezpečnostného hľadiska simulácia poukazuje na zraniteľnosť v podobe DoS útoku. Experiment ďalej prezentuje aktuálnu centralizáciu podielu v skutočnom Solana blockchaine. V aktuálnom Solana blockchaine vlastní len 20 uzlov s 1 500 viac ako $\frac{1}{3}$ podielu. DoS útok je v tomto prípade ešte účinnejší, pretože malý počet uzlov vlastní veľkú časť celkového podielu.

5.1.1 Hlasovanie

Hlasy sú v Solana konsenze zasielané v podobe bežných transakcií. To prináša dodatočnú Proof-of-Stake bezpečnosť, pretože za každý hlas musí podielnik zaplatiť poplatok. Na druhej strane, tieto transakcie sú nezávislé od aplikačnej vrstvy blockchainu. Solana blockchain vo svojich oficiálnych štatistikách[1] zverejňuje priepustnosť TPS (*Transactions Per Second*), do ktorej započítava aj hlasovacie transakcie. Toto je skresľujúci údaj, pretože hlasovacie

[1] https://solanabeach.io/blocks

Veľkosť siete v uzloch	Množstvo aplikačných transakcií v bloku	TPS
1 000	1 000	2 500
1 000	10 000	25 000
1 000	20 000	50 000
1 000	50 000	125 000
10 000	1 000	2 500
10 000	10 000	25 000
10 000	20 000	50 000
10 000	50 000	125 000

Tabuľka 5.1: Veľkosti simulovaných sietí pre Solana hlasovanie.

transakcie vznikajú na úrovni konsenzus vrstvy a ich množstvo je priamo úmerné veľkosti siete. Priepustnosť blockchainu je potom umelo zvýšená o hlasovacie transakcie, ktoré neprinášajú žiadnu aplikačnú funkcionalitu.

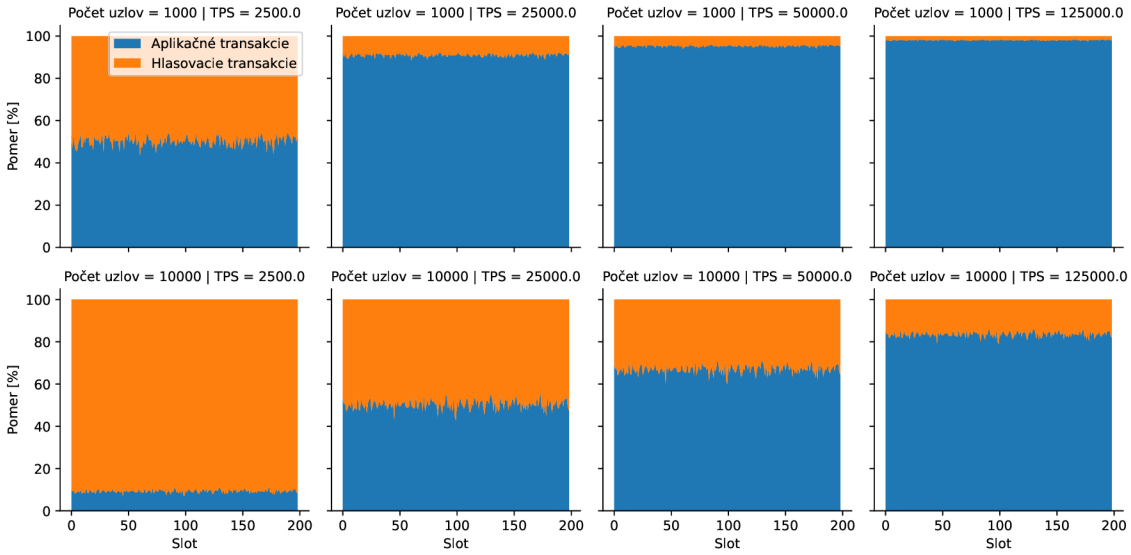
Následujúci simulačný experiment poukazuje na pomer hlasovacích a aplikačných transakcií. Simulácia je opakovaná pre rôzne rozsiahle siete s rôznym množstvom priebežne vznikajúcich aplikačných transakcií. Všetky simulované scenáre sú zhrnuté v tabuľke 5.1. Simulácia uvažuje aplikácie s vysokou produkciou transakcií (od 2 500 do 125 000 TPS).

Výsledky experimentu sumarizuje obrázok 5.1. Grafy zobrazujú pomer hlasovacích (oranžová plocha) a aplikačných transakcií (modrá) po dobu 200 slotov pre rôzne veľké siete. Graf poukazuje na nezanedbateľnú komunikačnú záťaž v podobe hlasovania. Ak bude v blockchaine vznikať rádovo viac transakcií ako je uzlov, tak bude hlasovacia záťaž takmer zanedbateľná. Napríklad, pre 1 000 uzlov a 125 000 TPS predstavuje hlasovanie necelé 2 %. Naopak, ak bude vznikať menej transakcií ako je uzlov, tak bude hlasovanie predstavovať majoritu transakcií v systéme (viď konfigurácia 10 000 uzlov a 1 000 TPS). Ak je veľkosť siete rádovo podobná množstvu generovaných transakcií za blok (napríklad 10 000 uzlov a 25 000 TPS), tak predstavuje hlasovanie približne polovicu všetkých transakcií.

Okrem pomeru početnosti hlasovacích a aplikačných transakcií je dôležité poukázať aj na ich dátový pomer. Priepustnosť z tohto hľadiska skresľuje ešte viac, pretože hlasovacie transakcie majú fixnú veľkosť. Najväčšie zložky hlasovacej transakcie sú digitálny podpis (pre RSA približne 256 B) a hash bloku, za ktorý sa hlasuje (32 B). Každý uzol v rámci hlasovacieho protokolu TowerBFT (viď sekcia 3.2.2) zašle svoj hlas dvakrát. Potom teda každý hlasujúci uzol spôsobuje fixný prenos približne 0,6 KB za slot. Aplikačné transakcie ale môžu predstavovať väčší a premenlivejší dátový prenos. Napríklad, blockchain Ethereum umožňuje, aby niektoré transakcie boli veľké až 24 KB (ide o tzv. *smart contract* transakcie[2]). Z tohto uhlu pohľadu teda hlasovacie transakcie predstavujú typicky menšiu časť prenášaného dátového objemu. O to viac ale skresľujú priepustnosť, ak sú započítavané do štatistík priepustnosti blockchainu.

Výsledky simulačného experimentu poukazujú na to, že Solana konsenzus predstavuje neprimerane veľkú komunikačnú záťaž, ak v systéme vzniká pomerne malé množstvo transakcií. Solana konsenzus navyše prebieha veľmi často (približne každých 400 ms). Z toho vyplýva záver, že Solana je určená pre aplikácie, ktoré dokážu produkovať veľké množstvo transakcií v krátkom čase. Tento konsenzus je preto vhodný napríklad pre aplikáciu ako je finančný platobný systém globálnej úrovne. Ak by aplikácia neprodukovala dostatočné

[2] https://ethereum.org/en/developers/docs/smart-contracts/



Obr. 5.1: Výsledky simulácie Solana hlasovania zobrazujú pomer aplikačných a hlasovacích transakcií pre rôzne veľkú sieť v čase.

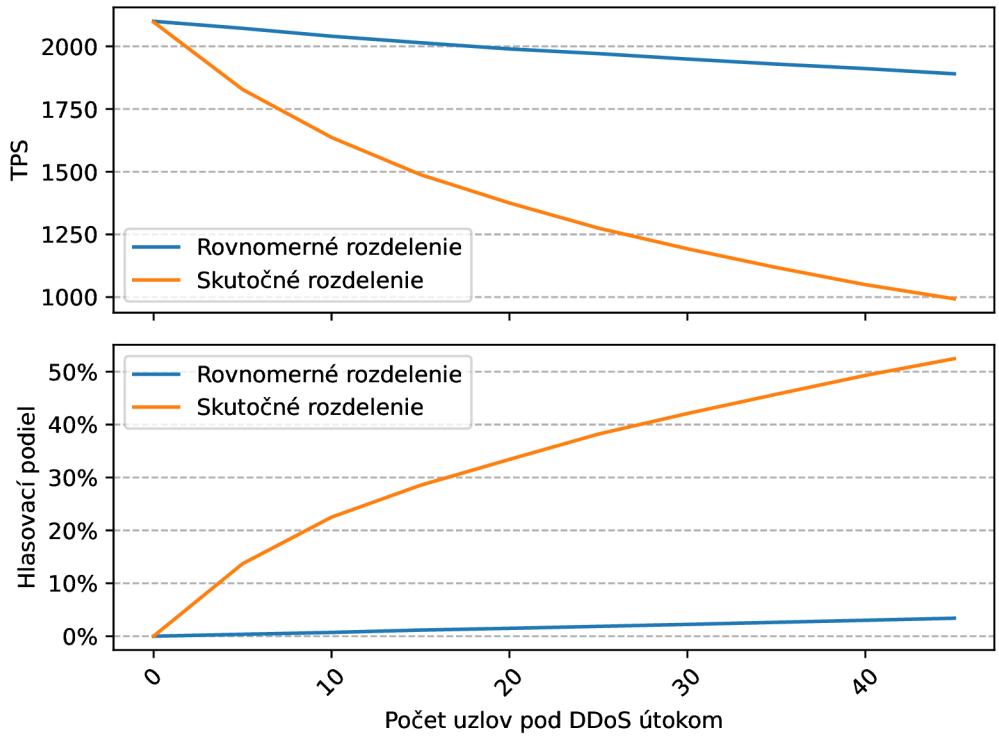
množstvo transakcií, Solana by mohla nastaviť dlhšiu periódu generovania bloku. Toto riešenie by však predĺžilo čas do finalizácie transakcií.

5.1.2 DoS útok na vodcu

Následujúci experiment simuluje DoS útok na vodcu hlasovania Solana konsenzu. Pre útočníka je najvýhodnejšie v každom slote útočiť na aktuálneho vodcu. Takýto útok je teoreticky možný, keďže rozvrh vodcov je dopredu známy. Útok by mal za následok nulovú priepustnosť transakcií po dobu celej epochy. Vodcovia sa ale menia každých 400 ms a sieť má aktuálne viac ako 1 500 uzlov. Je preto nepravdepodobné, že by útočník dokázal tak efektívne útočiť na ľubovoľný uzol v tak krátkych intervaloch. Útočník si ale môže zvoliť len niekoľko uzlov, ktoré budú v danej epoche najčastejšie vodcami. Aktuálne v Solana blockchaine vlastní viac ako $\frac{1}{3}$ podielu len 20 uzlov. Tento stav povedie k tomu, že väčšinu epochy bude vodca práve jeden s týchto uzlov.

Uvažujme simulačný scenár, v ktorom je 1 500 uzlov. Simulácia je najprv vykonaná bez DoS útoku na vodcu. Následne je opakovaná s narastajúcim počtom uzlov, na ktoré sa útočí (5, 10, ..., 50). Útočník sa zameriava na uzly, ktoré sú najčastejšie vodcami. Celý experiment je simulovaný pre rovnomerné rozdelenie podielu medzi uzlami, ale aj pre skutočné rozdelenie, v ktorom minorita uzlov vlastní veľkú časť podielu.

Výsledok simulácie zachytáva obrázok 5.2. Osa x reprezentuje počet uzlov, ktoré sú pod DoS útokom. Spodný graf zobrazuje celkový podiel, ktorý vlastnia tieto uzly. Horný graf ukazuje závislosť priepustnosti transakcií (TPS) na počte uzlov pod útokom. S narastajúcim počtom uzlov pod DoS útokom klesá priepustnosť blockchainu. V prípade rovnomerného rozdelenia podielu je táto závislosť lineárna. Pri útoku na 45 uzlov (teda 3 % siete) poklesla priepustnosť približne o 10 %. Pri skutočnom rozdelení podielu je ale prepad priepustnosti výrazný aj pre malý počet uzlov pod DoS útokom. Pri útoku na 45 uzlov klesla priepustnosť o viac ako 50 %.



Obr. 5.2: Solana pod DoS útokom: Simulácia útočila na rôzny počet uzlov (osa x) a sledovala vplyv útoku na priepustnosť (horný graf). Spodný graf ukazuje veľkosť podielu vlastneného uzlami, ktoré sú pod útokom (zvlášť pre rovnomerné a skutočné rozdelenie podielu v sieti).

Experiment poukazuje na bezpečnostnú slabinu Solana konsenzu v podobe DoS útoku. Tento útok je navyše výrazne efektívnejší ak je rozdelenie podielu v sieti nevyvážené.

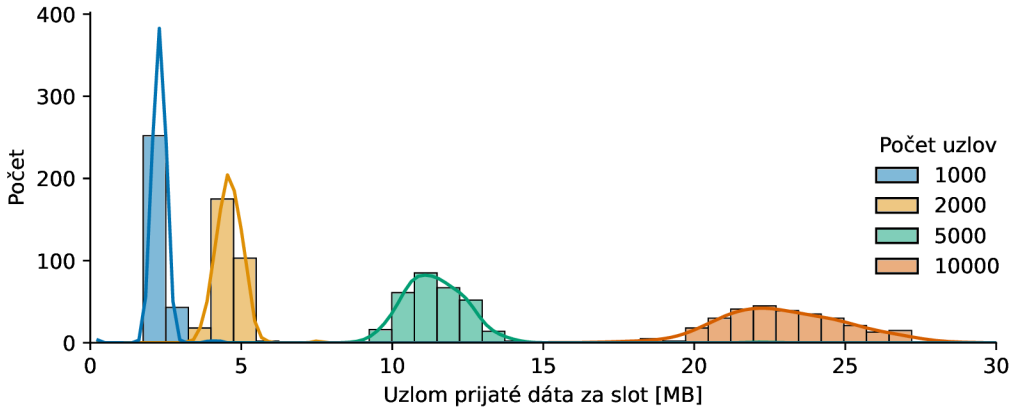
5.1.3 Priepustnosť a škálovateľnosť

Simulačný experiment popísaný v tejto sekcii sa zameriava na škálovateľnosť Solana konsenzu vzhľadom k veľkosti siete. V rámci experimentu je simulovaná rôzne veľká sieť (od 1 000 po 10 000 uzlov). Experiment predpokladá, že podľa veľkosti siete vzniká v systéme množstvo transakcií. Simulácia ďalej uvažuje, že blok má veľkosť hlavičky 80 B, priemerná aplikačná transakcie je 670 B a hlasovacia transakcia predstavuje 256 B (veľkosť RSA digitálneho podpisu). Jednotlivé behy simulácie sú porovnané z hľadiska škálovateľnosti a priepustnosti. Tabuľka 5.2 zhŕňa vstupné parametre pre jednotlivé simulácie.

Výsledok simulačného experimentu prezentuje obrázok 5.3, ktorý zobrazuje histogram prijatých dát v MB pre rôzne veľké siete. Ide o dáta priemerne prijaté jedným uzlom za dobu jedného slotu. Pre sieť 1 000 uzlov je medián prijatých dát 2,2 MB (maximálne 5,3 MB). Pre sieť 10 000 uzol je medián 22,8 MB (maximálne 53 MB). Nárast prenesených dát teda rastie lineárne vzhľadom k veľkosti siete. Solana sieť teda umožňuje lineárnu škálovateľnosť. Jeden slot má dĺžku 400 ms, a teda pre najväčšiu simulovanú sieť potrebuje uzol prijať približne 57 MB/sec. Na 1 Gbps linke by takýto dátový prenos uzol zvládol bez problémov (1 Gbps má priepustnosť približne 125 MB/s). Avšak, na vrchole vyťaženosti potrebuje uzol priepustnosť až 132 MB/s. V týchto kritických momentoch by finalita konsenzu nedokázala zachovať 400 ms.

Uzly	TPS	Veľkosť bloku [MB]
1 000	3 000	2
2 000	6 000	4
5 000	15 000	10
10 000	30 000	20

Tabuľka 5.2: Solana priepustnosť: Vstupné parametre pre jednotlivé behy simulačného experimentu.



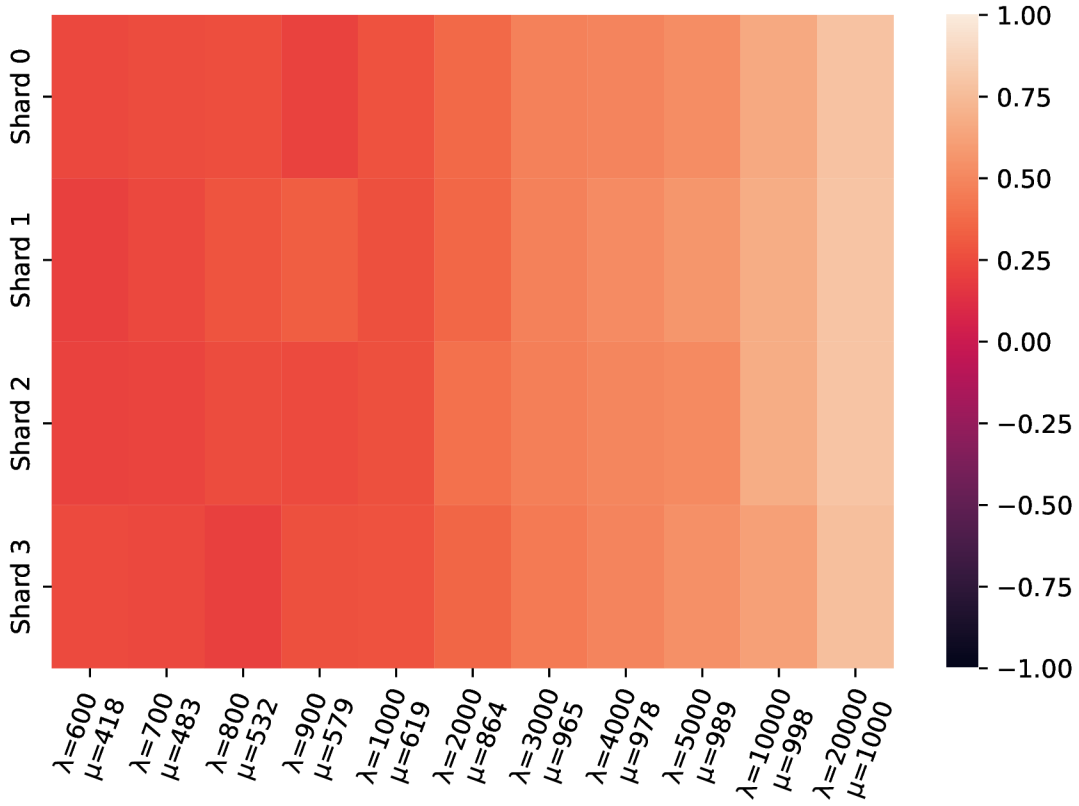
Obr. 5.3: Solana škálovateľnosť: Histogram prijatých dát za slot pre priemerný uzol v rôzne veľkej sieti.

5.2 Harmony

Blockchain protokoly založené na hlasovaní zvyčajne predstavujú problém v prípade veľkých sietí. Uzniesť konsenzus pomocou volieb vo veľkej decentralizovanej sieti zaberie nezanedbateľný čas, ktorý porastie s veľkosťou siete. Harmony tento problém minimalizuje pomocou shardingu, ktorý rozdelí sieť na menšie skupiny uzlov. Simulácia Harmony experimentuje s shardingom a jeho škálovateľnosťou a priepustnosťou. Simulácia sa taktiež zameriava na bezpečnosť Harmony shardingu. Rozdelenie siete na podskupiny môže znížiť bezpečnosť Proof-of-Stake mechanizmu. Z tohto hľadiska je vykonaný simulačný experiment, ktorý sa snaží ovládnuť podskupinu uzlov. Výsledky experimentu poukazujú na dobrú odolnosť Harmony shardingu voči tomuto typu zraniteľnosti. Na záver je prezentovaný DoS útok na vodcu hlasovania. Simulácia poukazuje na veľkú zraniteľnosť Harmony konsenzu voči tomuto útoku.

5.2.1 Sharding

Harmony sharding distribuuje celkové hlasovacie právo približne rovnomerne do všetkých shardov. Z toho vyplýva, že jeden uzol môže byť pridelený do viacerých shardov. Presnosť distribúcie medzi shardy je určené parametrom λ (viď sekcia 3.1.2). Čím je hodnota λ väčšia, tým je distribúcia podielu v shardoch viac podobná skutočnej celkovej distribúcii. Väčšia λ teda predstavuje väčšiu bezpečnosť, pretože je nepravdepodobnejšie ovládnutie podskupiny uzlov. Súčasne ale rastúca λ zvyšuje redundanciu uzlov v shardoch. S narastajúcou redundanciou sa zvyšuje aj komunikačnú záťaž. To vedie k zníženiu efektívnosti



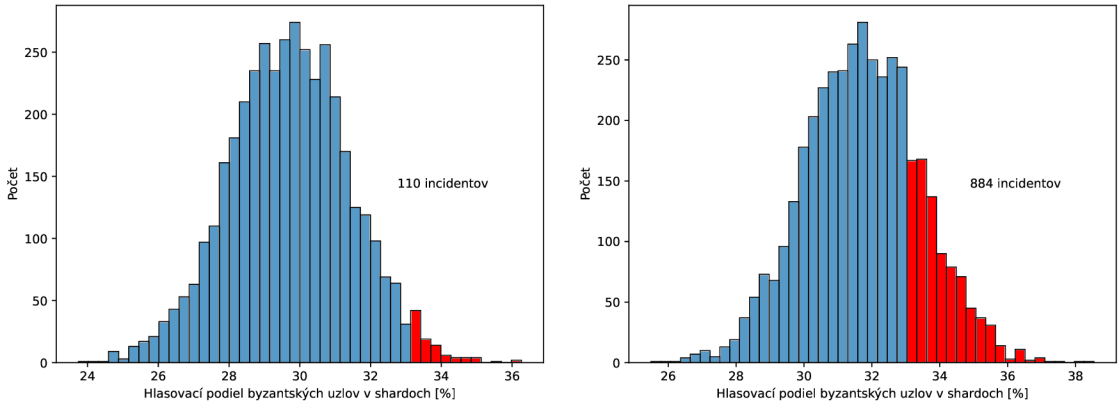
Obr. 5.4: Harmony sharding: Teplotná mapa podobnosti medzi skutočným podielom uzlov a hlasovacím právom v jednotlivých shardoch. Shardy predstavujú riadky. Stĺpce ukazujú výsledky simulácie pre rôzne λ. Hodnota μ reprezentuje priemerný počet uzlov v jednom sharde pre dané λ.

Harmony shardingu. Nastavenie parametru λ teda predstavuje kompromis medzi bezpečnosťou a škálovateľnosťou.

Simulácia rôznych hodnôt λ experimentuje s bezpečnosťou shardingu pre sieť s veľkosťou 1 000 uzlov. Experiment uvažuje, že podiel uzlov v sieti je pre každú epochu z rovnomerného rozloženia. Takéto rovnomerné rozdelenie hlasovacieho práva je ideálny stav z hľadiska decentralizovanosti Proof-of-Stake konsenzu. Simulácia porovnáva celkový podiel s tým, ktorý bol skutočne pridelený jednotlivým uzlom v každom sharde.

Výsledok simulácie zhŕňa teplotná mapa na obrázku 5.4, ktorá zobrazuje v jednotlivých stĺpcoch výsledok simulácie pre konkrétne λ. Hodnota μ reprezentuje priemerný počet uzlov v jednom sharde pre dané λ (môžeme vidieť, že μ rastie úmerne k veľkosti λ). Jednotlivé riadky ukazujú nameranú koreláciu medzi celkovým hlasovacím podielom a tým, ktorý bol skutočne pridelený uzlom v danom sharde. Na koreláciu bol použitý Spearmanov[3] korelačný koeficient. Môžeme vidieť, že rastúce λ zvyšuje podobnosť hlasovacieho podielu v shardoch voči skutočnému. Z teplotnej mapy je vidieť, že hlasovací podiel v jednotlivých shardoch začína byť významne podobný skutočnému až pre $\lambda > 3000$. Pri takto vysokej λ sú v každom sharde takmer všetky uzly (pre $\lambda > 3000$ je $\mu \approx 1000$). To však úplne odstraňuje potenciál

[3] https://www.sciencedirect.com/topics/mathematics/spearman-correlation



(a) 30 % byzantský podiel

(b) 32 % byzantský podiel

Obr. 5.5: Simulácia Harmony útoku na ovládnutie podskupiny uzlov. Histogramy hlasovacieho podielu byzantských uzlov za 1000 epoch. Červená časť predstavuje incidenty, pri ktorých bol pridelený podiel v sharde neprávom väčší ako $\frac{1}{3}$.

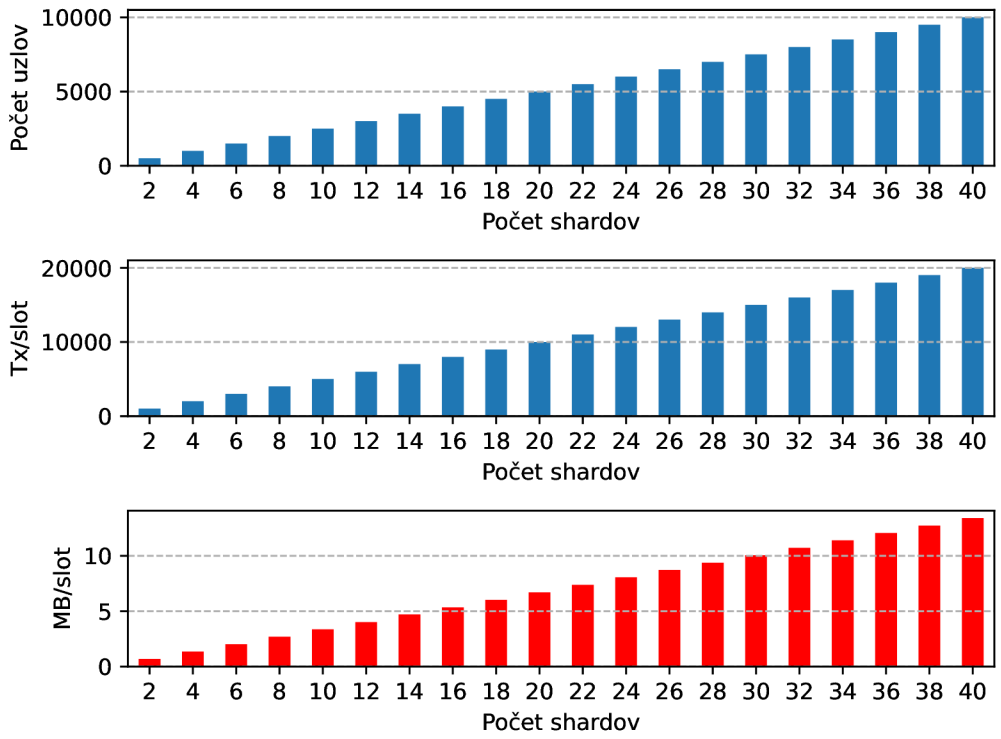
škálovatelnosti, a naopak ešte zvyšuje komunikačnú záťaž. Harmony doporučuje nastaviť $\lambda = 600$. V takejto konfigurácii je podobnosť distribúcie podielu nižšia a v každom sharde je približne 400 uzlov. Priemerne sa v tomto prípade uzol vyskytuje v 1,6 shardoch (teda väčšina uzlov je práve v dvoch shardoch a niektoré len v jednej). Výsledky experimentu ukazujú, že distribúcia podielu medzi shardy je nepresná a teoreticky umožňuje nepoctivému podielnikovi získať v niektorej sharde väčší podiel ako v skutočnosti vlastní. S pravdepodobnosťou výskytu takéhoto incidentu experimentuje simulácia popísaná v sekcii 5.2.2.

5.2.2 Útok na podskupinu uzlov

Uvažujme incident, že nepoctivé uzly (ďalej len byzantské) vlastnia menej ako $\frac{1}{3}$ podielu. Ak by Harmony nepoužíval sharding, útočníci by nemali dostatočný podiel na manipuláciu blockchainu. Ale vďaka nedokonalosti prerozdeľovania hlasovacieho práva získajú neprávom v niektorom sharde dostatočný podiel (teda $\frac{1}{3}$) na prekazenie konsenzu po dobu celej epochy.

Simulačný experiment popísaný v tejto sekcii analyzuje ako často by mohol nastať tento incident. Experiment simuloval 1000 epoch, a teda 1000 redistribúcii do shardov. Pre simuláciu je nastavené λ na odporúčanú hodnotu 600. Celá simulácia je viackrát opakovaná s rôznym množstvom byzantského podielu (postupne 28, 30 až 32 %). Simulácia uvažuje sieť s veľkosťou 1 000 uzlov a štyri shardy. Rozdelenie podielu je rovnomerné.

Výsledok simulácie zhŕňajú histogramy na obrázku 5.5, ktoré zachytávajú distribúciu byzantského hlasovacieho podielu v shardoch. V prípade 5.5a je 30 % celkového podielu byzantského a v prípade 5.5b ide až o 32 %. V oboch scenároch má distribúcia byzantského hlasovacieho podielu očakávanú strednú hodnotu. Avšak, môžeme vidieť, že rozptyl je pomerne vysoký. Čím viac sa celkový byzantský podiel približuje $\frac{1}{3}$, tým častejšie tieto incidenty nastávajú. Za dobu 1 000 epoch boli uzly rozdelené do 4 000 shardov. Ak byzantský podiel predstavoval 30 %, tak nastalo 110 incidentov kedy byzantské uzly v niektorej sharde získali viac ako $\frac{1}{3}$ podielu. V tejto konfigurácii je teda pravdepodobnosť popísaného incidentu 2,8 %. Útočníci so získaným podielom nemôžu ovládnuť hlasovanie (na ovládnutie hlasovania je potrebné viac ako $\frac{2}{3}$ podielu). Útočníci ale môžu prerušiť konsenzus. V daných



Obr. 5.6: Simulácia Harmony škálovateľnosti: Priemerné množstvo dát, ktoré uzol prijme za jeden slot v závislosti na veľkosti siete.

prípadoch teda teoreticky môže konsenzus po dobu celej epochy (viac ako 18 hodín) prestať fungovať. V ovládnutej sharde potom bude po dobu celej epochy nulová priepustnosť transakcií.

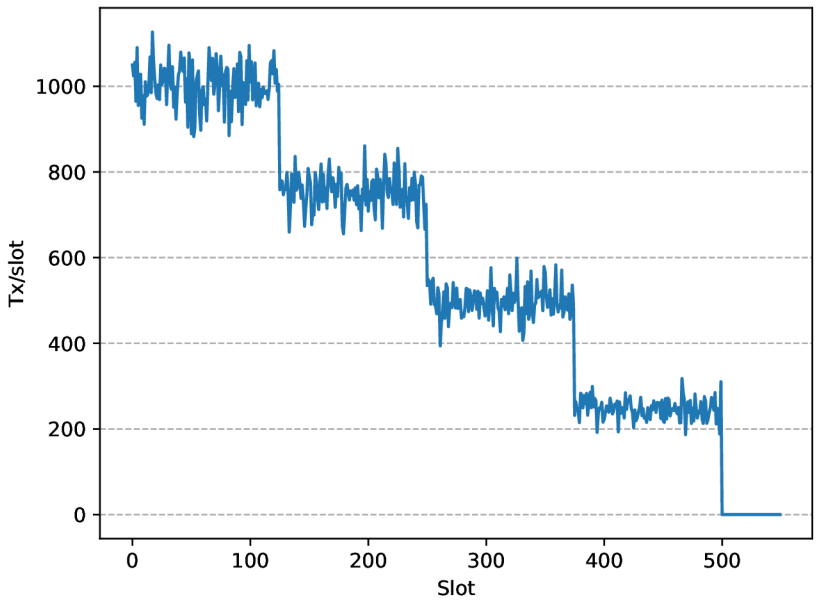
Ak bol byzantský podiel menší ako 28 %, tak takéto incidenty vôbec nenastávali. Ak bol byzantský podiel 32 %, tak tieto incidenty nastávali s pravdepodobnosťou 22 %. Je ale potrebné podotknúť, že ak byzantské uzly vlastnia tak veľkú časť podielu, ide o útok priamo na Proof-of-Stake a nie len sharding. Ak je v Proof-of-Stake konsenze vlastnená tak veľká časť celkového podielu byzantskou skupinou, tak ide o problém aj keby Harmony sharding vôbec nepoužívalo.

Výsledky simulačných experimentov ukazujú, že Harmony sharding poskytuje pomerne vysokú bezpečnosť. Útočníci potrebujú pre ovládnutie shardy v podstate rovnaký podiel ako aj pre ovládnutie celej siete (viac ako $\frac{1}{3}$).

5.2.3 Priepustnosť a škálovateľnosť

Následujúci simulačný scenár vyhodnocuje priepustnosť transakcií v Harmony vzhľadom na veľkosť siete. V simulácii je uvažované, že veľkosť hlavičky každého bloku je 80 B. Veľkosť transakcie závisí na aplikácii, ktorú blockchain poskytuje. Simulácia predpokladá kryptomenu podobnú Bitcoinu. Transakcia je preto priemerne veľká 670 B (priemerná veľkosť Bitcoin transakcie v roku 2021 vypočítaná z dostupných štatistík[4]). Simulácia ďalej predpokladá, že v sieti vzniká dostatok transakcií na to, aby bolo možné do každého bloku

[4] https://www.blockchain.com/charts



Obr. 5.7: Harmony DoS útok: Simulácia postupne narastajúceho útoku na vodcov hlasovania.

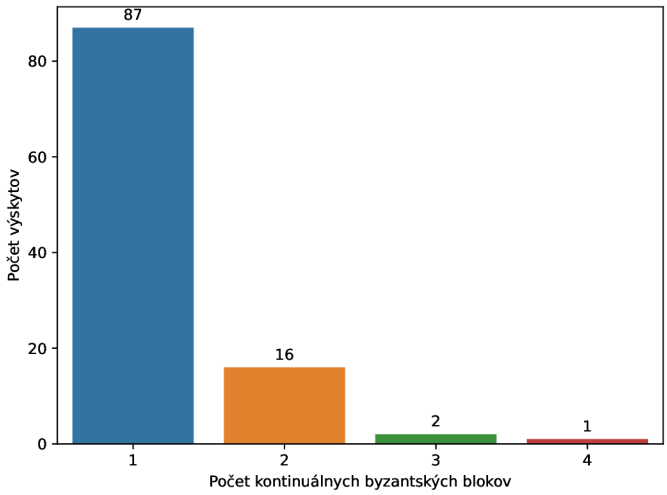
v každom sharde pridať približne 600 nových transakcií. Každý blok by teda mal mať približne 400 KB. Hodnota λ bola nastavená na odporúčanú hodnotu 600. Harmony doporučuje, aby počet shardov v sieti bol určený podľa vzorca $\frac{n}{250}$, kde n je počet uzlov v sieti. Simulácia bola preto opakovaná postupne pre narastajúci počet uzlov a shardov (2 shardy a 500 uzlov, 4 shardy a 1 000 uzlov, ..., 40 shardov a 10 000 uzlov).

Výsledok simulácie sumarizujú grafy na obrázku 5.6. Môžeme vidieť, že priepustnosť transakcií za slot sa lineárne zvyšuje s narastajúcim počtom shardov (stredný graf). Z tohto hľadiska sharding skutočne umožňuje efektívne škálovať veľkosť siete. Na druhej strane, graf taktiež ukazuje množstvo dát v MB, ktoré každý uzol priemerne prijal za dobu jedného slotu (spodný graf). Komunikačná záťaž konsenzu rastie s narastajúcou veľkosťou siete aj napriek shardingu. Ak má sieť 40 shardov, tak uzol počas jedného slotu prijme približne 15 MB (teda 7,5 MB/s). Uvažujme, že celá sieť používa minimálne 1 Gbps linku. Každý uzol potom môže prijať približne 125 MB/s. Komunikačná záťaž spojená s hlasovaním je teda v takejto konfigurácii jednoznačne zvládnuteľná.

5.2.4 DoS útok na vodcu

Následujúci experiment ukazuje náchylnosť Harmony na DoS útok. Simulovaná sieť má 1000 uzlov a štyri shardy. Počas epochy je pomocou DoS útoku dočasne znemožnená komunikácie pre vodcu každého shardu. Bez vodcu ostatné uzly nemôžu uzniesť konsenzus. Útok je stupňovaný: postupne sa útočí na 1,2,3 až 4 vodcov súčasne.

Výsledky experimentu zobrazuje obrázok 5.7, ktorý ukazuje množstvo transakcií finalizovaných v jednotlivých slotoch. V slote 125 začal útok na jedného vodcu. Od slotu 250 boli pod útokom dvaja vodcovia. V slote 375 už traja a od slotu 500 všetci štyria. Z grafu je vidieť, že v týchto momentoch vždy došlo k poklesu finalizovaných transakcií približne



Obr. 5.8: Ouroboros forking: Histogram množstva byzantských blokov, ktoré boli vygenerované kontinuálne za sebou po dobu 1000 slotov.

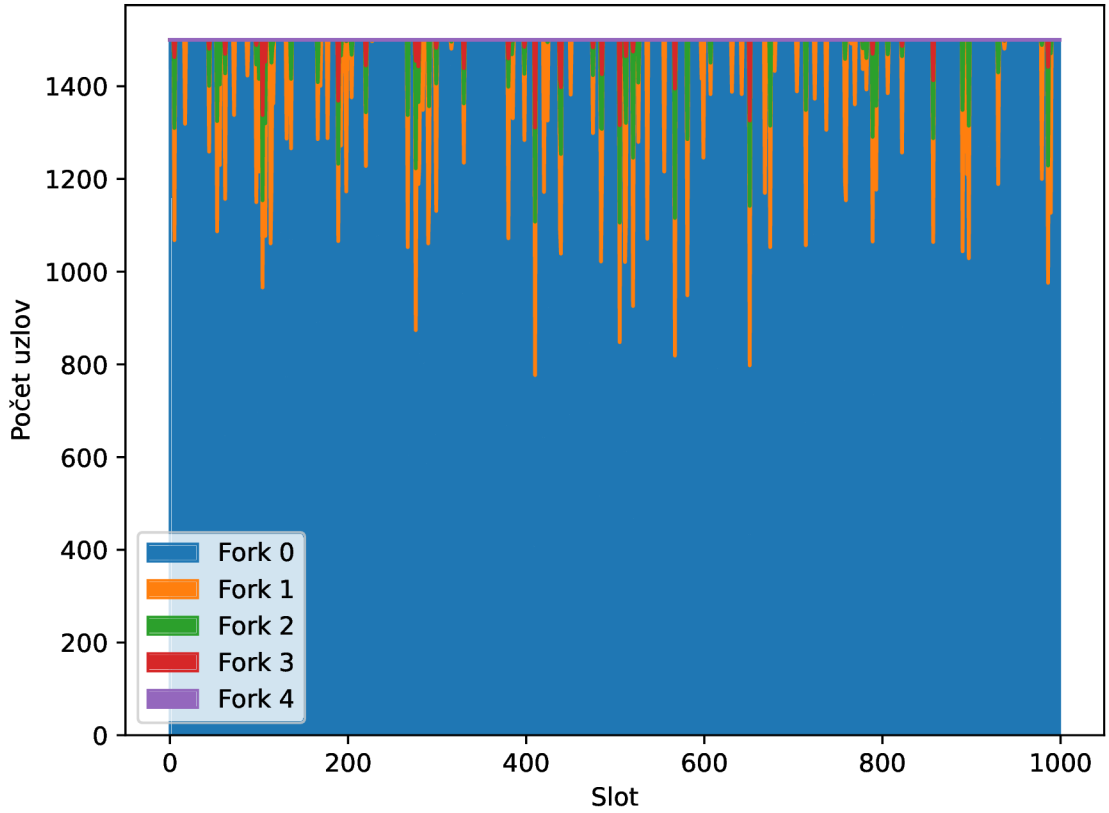
o $\frac{1}{4}$. Z experimentu je vidieť, že stačí intenzívny útok len na 4 uzly v sieti s 1000 uzlami a celý blockchain prestáva fungovať. Navyše je tento útok možné vykonať bez akéhokoľvek podielu v blockchaine. Útočníci teda neriskujú žiadnu stratu zdrojov.

5.3 Ouroboros

Ouroboros konsenzus je založený na náhodnej voľbe. V systéme neprebieha hlasovanie. Simulačné experimenty sa preto zameriavajú na útok vetvenia, ktoré je pri náhodnej voľbe bez hlasovania vždy možné. Výsledky simulácie ukazujú, že Ouroboros dovoľuje len krátkodobé vetvenie u ktorého nie je možné vykonať skutočný *double-spending* útok. Na druhej strane, aj krátkodobé vetvenie predlžuje čas do finality bloku. Ďalej simulácia experimentuje s DoS útokom na zoznam vodcov, ktorý Ouroboros používa.

5.3.1 Útok zdvojenia výdavkov (vetvenie)

Ouroboros konsenzus je založený na náhodnej voľbe a nie na hlasovaní (na rozdiel od Harmony a Solana konsenzu). Náhodná voľba pri výbere vodcu principiálne umožňuje vetvenie. Ouroboros poskytuje zabezpečenie proti modifikácii histórie už finalizovaného reťazca (nahradenie časti reťazca alternatívnou vetvou). Avšak, dočasné vetvenie pred finalizáciou je v tomto protokole možné. Ak sa stane vodcom byzantský uzol, môže vygenerovať niekoľko rôznych blokov a tie rozposlať do P2P siete. Každý uzol v sieti bude považovať za správny blok ten, ktorý mu príde ako prvý. Vetva sa ukončí hneď v ďalšom slote, pretože ďalší vodca si vyberie jeden s týchto blokov. Zvolený blok sa použije ako predchodcu nového bloku a jeho vetva sa tak stane najdlhšou. Pravdepodobnosť, že byzantský uzol získa dva vodcovské sloty po sebe je zanedbateľná, ak nevlastní veľmi veľký podiel. Blok v hĺbke jedna by sa teda mohol považovať za finalizovaný. Teoreticky by však mohla nastať situácia, že byzantské uzly tvoria koalíciu, ktorá spoločne vlastní pomerne veľký podiel. Potom môže nastať incident, že niekoľko po sebe nasledujúcich slotov budú vždy vodcovia práve uzly z tejto koalície. V takomto scenári vetvenie nekončí po jednom slote, ale môže byť udržia-



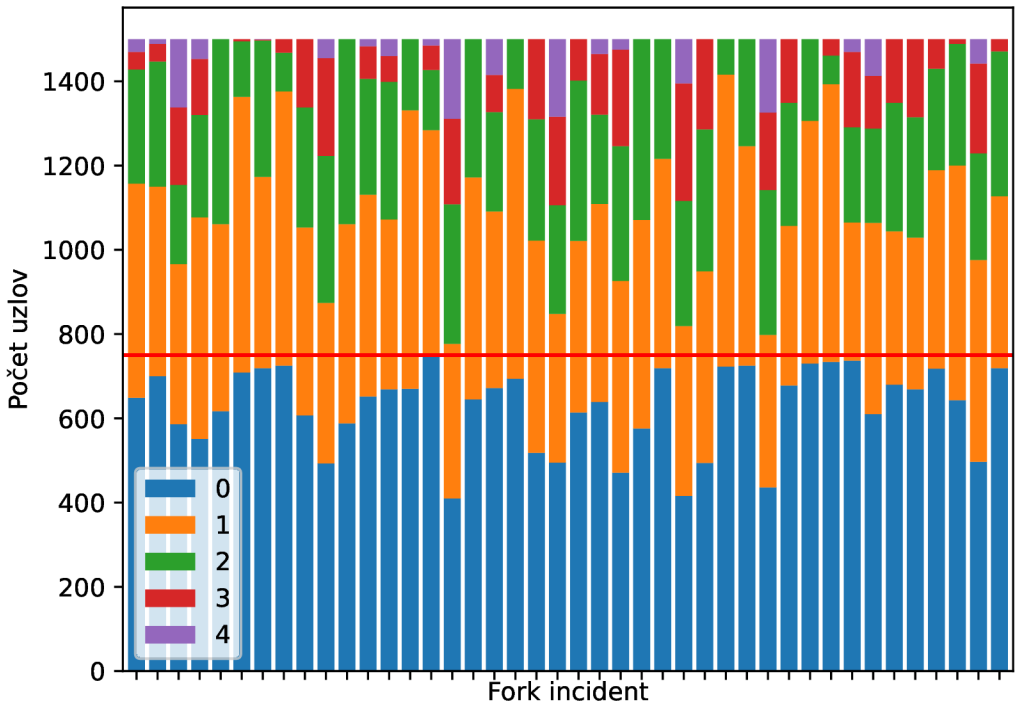
Obr. 5.9: Ouroboros vetvenie: Priebežný stav blockchainu z pohľadu jednotlivých uzlov v čase.

vané po všetky tieto sloty. Tým sa predlžuje čas do finalizácie bloku. Popísaná situácia je predmetom ďalšieho simulačného experimentu.

Simulácia uvažuje sieť z veľkosťou 1 500 uzlov, ktoré majú rozdelenie podielu rovnaké ako skutočná Cardano sieť (viď sekcia 4.2.2). V sieti je 157 uzlov byzantských a spoločne vlastnia 15 % podielu. Obrázok 5.8 ukazuje histogram fork incidentov za 1 000 simulovaných slotov. Byzantský vodcovia mali 87 príležitostí vygenerovať jeden blok. Ďalej mohli v 16 prípadoch vytvoriť dva bloky v rade. Dokonca nastala aj situácia kedy vygenerovali tri alebo štyri bloky v rade. V simulovanom experimente teda za 1 000 slotov (približne 15 minút) nastala jedenkrát situácia, že blok je finalizovaný až po štyroch slotoch (približne 4 sekundy).

Obrázok 5.9 zobrazuje priebežný stav blockchainu. Osa x ukazuje jednotlivé sloty a osa y stav blockchainu pre každý uzol v sieti. Jednotlivé farby reprezentujú vetvy blockchainu v danom slote. Simulácia uvažovala, že byzantský vodca vytvorí súčasne maximálne päť vetiev. Vidíme, že vo väčšine slotov vzniká vetvenie. Na druhej strane, majorita siete vlastní rovnakú vetvu (modrá farba), a to vo väčšine slotov.

Obrázok 5.10 zobrazuje zvlášť len sloty kedy v sieti už neexistuje majorita. V týchto slotoch najviac uzlov vlastní modrá vetva (anglicky *fork*). Červená čiara zobrazuje polovicu uzlov siete. Vidíme, že v týchto incidentoch v sieti neexistuje majorita. Byzantská koalície s celkovým podielom 15 % teda v simulácii dokázala až 42-krát dostať sieť do takého stavu, že rovnaký blockchain vlastnilo menej ako 50 % uzlov.



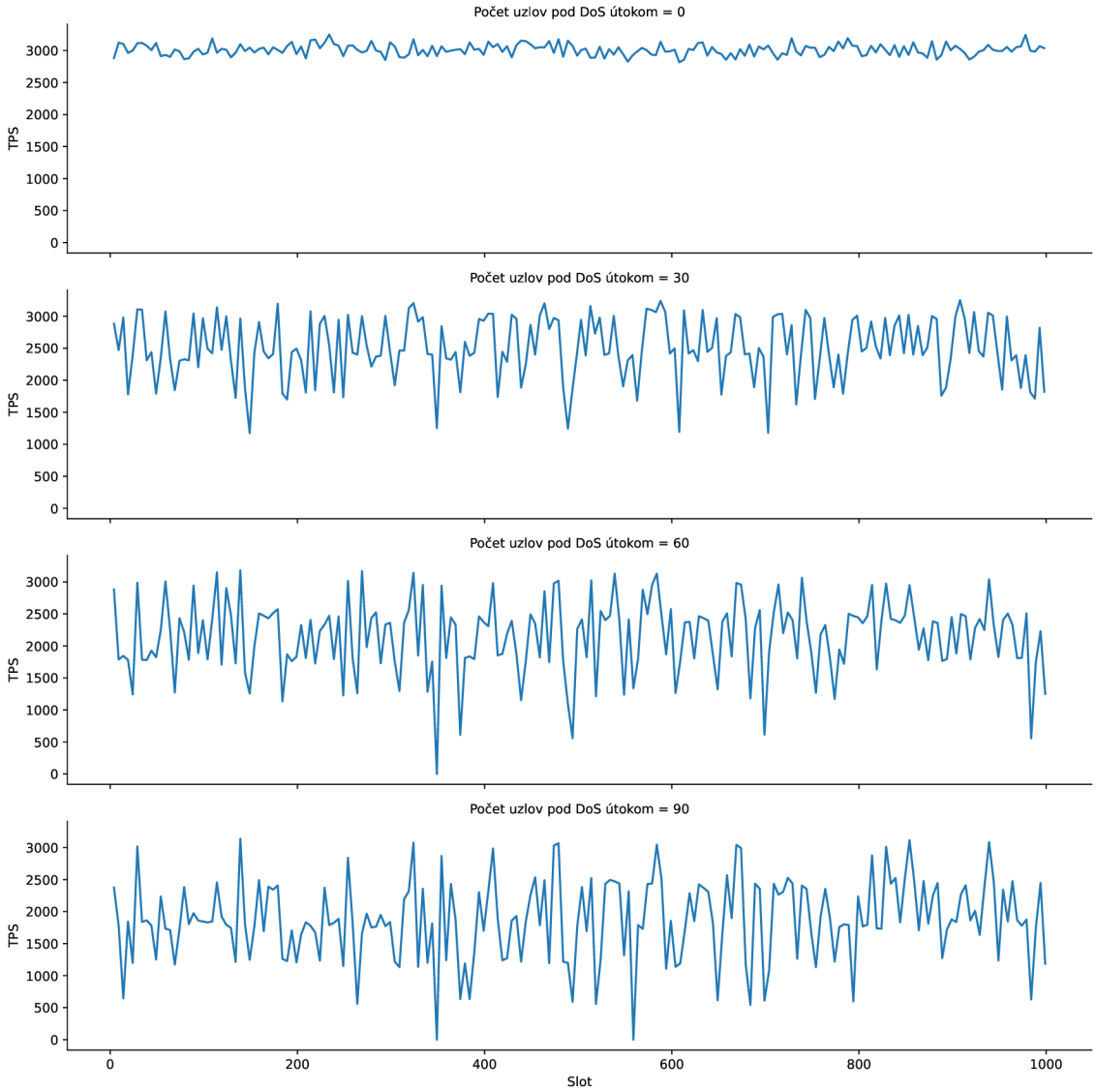
Obr. 5.10: Ouroboros vetvenie: Výber slotov v ktorých neexistuje žiaden stav, ktorý by vlastnila majorita uzlov siete.

5.3.2 DoS útok na vodcu

Ouroboros používa rozvrh vodcov, ktorý určuje uzly vytvárajúce bloky v jednotlivých slotoch. V princípe ide o rovnakú techniku akú používa protokol Solana. V sekcii 5.1.2 bola popísaná náchylnosť Solany na DoS útok na vodcu. Ouroboros umožňuje podobnú zraniteľnosť. Vodca slotu v protokole Ouroboros neslúži ako vodca hlasovania (na rozdiel od Solany). Táto roľa udeľuje danému uzlu právo ako jedinému vytvoriť nový blok v určenom slote. Ouroboros nepoužíva hlasovanie, a preto DoS útok nebude cieliť na prekazenie hlasovania. Zámerom útočníka bude znemožniť distribúciu nového bloku. Toho dosiahne napríklad tým, že sa mu podarí izolovať časť siete v ktorej je vodca. Túto situáciu nazývame *partitioning*. Principiálne je však útok podobný ako v prípade Solany. Útočník si zvolí uzly, ktoré sú v rozvrhu vodcov najčastejšie, a práve na tie útočí.

Simulácia DoS útoku uvažuje sieť podobnú skutočnej sieti Cardano (viď sekcia 4.2.2). Systém má 1 500 uzlov a rozdelenie podielu ako v Cardano blockchaine. Simulácia bola opakovaná s rôznym počtom uzlov, na ktoré bol vykonaný DoS útok (postupne 0, 30, 60 a 90 najčastejších vodcov). Obrázok 5.11 sumarizuje výsledok simulácie. Pre prehľadnosť je priepustnosť transakcií priemerovaná do časových intervalov po päť slotov. Sieť bez útoku produkuje stabilne priemerne 3 000 transakcií za slot. Ak je pod útokom 30 uzlov, priepustnosť klesne približne na 2 500 transakcií. Pri ešte väčšom počte uzlov pod útokom priepustnosť klesá ešte viac. Môžeme vidieť, že v mnohých slotoch je stále propagovaných

približne 3 000 transakcií. Ide o sloty vlastnené uzlami, ktoré nie sú pod útokom. Naopak, v prípade 60 a viac uzlov pod útokom nastávajú incident kedy aj päť slotov za sebou je vodca pod útokom. V týchto momentoch sieť v podstate po dobu piatich sekúnd neprepúšťa žiadne transakcie. Výsledky simulácie poukazujú na zraniteľnosť rozvrhu vodcov. Útočníci navyše ani nemusia vlastniť žiadne zdroje v blockchaine, pretože rozvrh vodcov je verejná informácia.



Obr. 5.11: Ourobors DoS útok: Priepustnosť transakcií pre rôzny počet uzlov pod DoS útokom.

5.4 Navrhované vylepšenie bezpečnosti

Výsledky simulačných experimentov poukázali na bezpečnostnú slabinu v podobe DoS útoku na vodcu pre protokoly Solana (viď 5.1.2), Harmony (viď 5.2.4) aj Ouroboros (viď 5.3.2). Hlavným dôvodom je, že všetky tri protokoly používajú rozvrh vodcov. Ten je verejne do-

stupný a umožňuje plánovať útok na dlhé časové obdobie (hodiny až dni). Jedným riešením tohto problému je ochrana na úrovni sieťovej vrstvy. Sieť môže poskytovať firewall, ktorý DoS útoky neumožní. Iným riešením je, že sieť môže poskytovať anonymizačnú vrstvu (napríklad známa anonymizačná sieť Tor [18]). Anonymizácia by odstránila potenciál DoS útoku úplne. Na druhej strane, anonymizácia predstavuje nezanedbateľné zvýšenie komunikačnej záťaže. Priepustnosť blockchainu by pridaním tejto vrstvy klesla. Obe riešenia odstraňujú nedostatok až dodatočne na sieťovej vrstve. Príčinou samotnej zraniteľnosti je ale konsenzus protokol, ktorý používa rozvrh vodcov. Odstránením rozvrhu vodcov by útočníci z väčšej miery stratili možnosť určiť kľúčové uzly pre DoS útok. Existujú aj konsenzus protokoly, ktoré rozvrh vodcov pre svoje fungovanie nepotrebujú. Algorand [23] je blockchain založený na hlasovaní s vodcom (rovnako ako Harmony alebo Solana). Algorand však určuje vodcu nepredvídateľne. Útočníci teda nemôžu dopredu určiť cieľ DoS útoku (iba čiastočne na základe rozdelenia podielu v sieti, pretože najbohatšie uzly budú najpravdepodobnejšie vodcami). Nahradenie rozvrhu vodcov za schému použitú v Algorande, by mohlo odstrániť túto bezpečnostnú slabinu priamo na konsenzus vrstve.

5.4.1 Verifiable Random Function

Algorand je Proof-of-Stake konsenzus protokol založený na byzantskom hlasovaní. Pre každé kolo hlasovania potrebuje Algorand určiť vodcu, ktorý bude publikovať nový blok. V danom kole hlasovania nie je dopredu známe kto bude tento vodca. Algorand zabezpečuje voľbu vodcu pomocou kryptografickej schémy *Verifiable Random Function* (VRF [26]), ktorá pozostáva z troch algoritmov [25]:

- $Keygen() = (Vk, Sk)$: Každý uzol si pre seba pomocou algoritmu $Keygen$ vygeneruje dvojicu kľúčov, kde Sk je tajný kľúč a Vk je verejný kľúč. Verejné kľúče sa zverejnia pre všetky uzly v sieti.

- $Evaluate(Sk, X) = (Y, \rho)$: Pre každé kolo hlasovania je verejne známy náhodný reťazec X. Každý uzol použije funkciu $Evaluate$ so svojím tajným kľúčom Sk a reťazcom X. Výstupom je dvojica (Y, ρ), kde Y je pseudonáhodný reťazec a ρ je dôkaz. Uzol si následne overí podmienku $Y \in P$, kde P je interval s veľkosťou priamo úmernou podielu daného uzlu. Ak podmienka platí, uzol je potenciálnym vodcom. V takom prípade sa začne uzol chovať ako vodca. Napríklad publikuje nový blok. Súčasťou tohto bloku bude aj dvojica (Y, ρ).

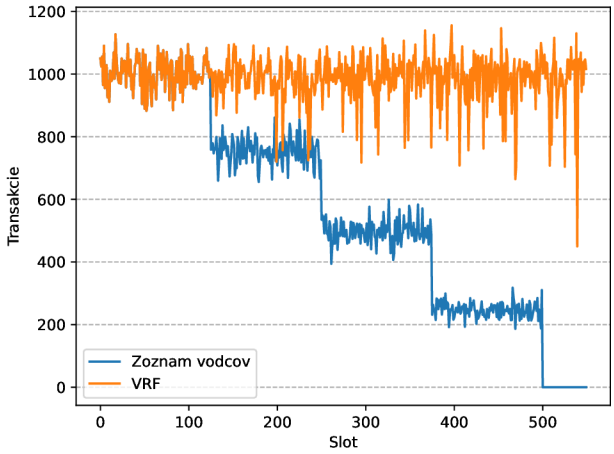
- $Verify(Vk, X, Y, \rho) = 0/1$: Keď ľubovoľný uzol prijme navrhovaný blok od potenciálneho vodcu, použije funkciu $Verify$. Vk je verejný kľúč potenciálneho vodcu. Dvojicu (Y, ρ) získal ako súčasť návrhu. Ak Y prináleží ρ, tak výstupom $Verify$ je 1, inak 0. V prípade, že verifikácia prebehla úspešne, overí uzol podmienku $Y \in P$ (rovnako ako bolo popísané v predchádzajúcom kroku).

Kľúčovou vlastnosťou tejto schémy je, že vodca nie je dopredu známy. Zároveň je pravdepodobnosť voľby vodcu priamo úmerná jeho podielu (princíp Proof-of-Stake).

Jednotlivé protokoly experimentálne upravme tak, aby používali VRF schému:

- **Harmony**: Po rozdelení uzlov do shardov sa nebude používať jeden uzol ako vodca po dobu celej epochy. Namiesto toho, sa v danej epoche pre každý slot použije VRF na určenie vodcu hlasovania.

- **Solana**: Rozvrh vodcov na epochu sa nahradí VRF pre každý slot.



Obr. 5.12: Harmony s VRF: Porovnanie priepustnosti transakcií pri DoS útoku v prípade pôvodného rozvrhu vodcov a voľby vodcu pomocou VRF.

- **Ouroboros**: Rozvrh vodcov po dobu epochy, rovnako ako v prípade Solana, bude nahradený priebežnou voľbou pomocou VRF.

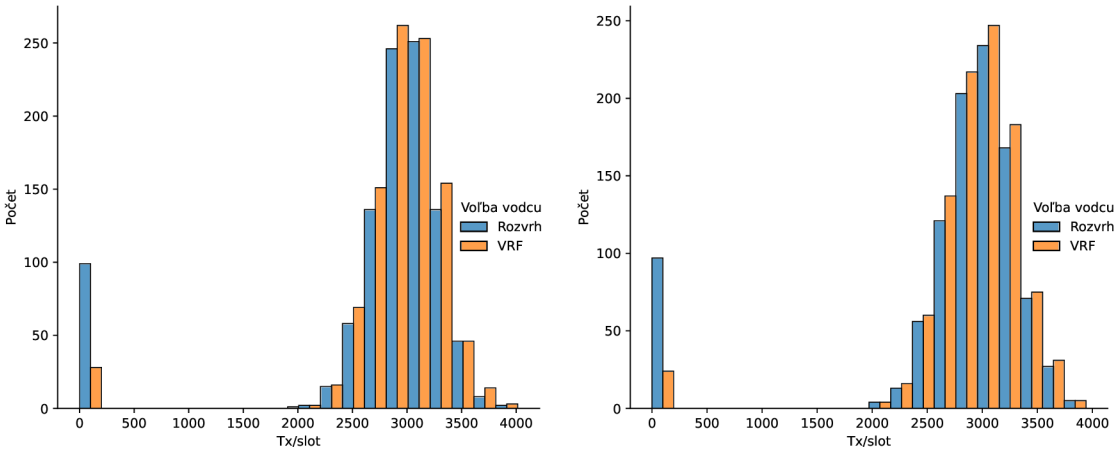
5.4.2 Harmony s VRF

Nasledujúci simulačný scenár opakuje DoS útok popísaný v sekcii 5.2.4. Simulácia stále uvažuje rovnaký postup pre rozdelenie uzlov do shardov. Ale v jednotlivých shardoch už nie je jediný vodca po celú dobu epochy. Namiesto toho je pre každý slot vždy použitá nepredvídateľná voľba vodcu pomocou VRF. Útočníci teda už nemôžu útočiť na konkrétny uzol. Najúčinnejší spôsob útoku je preto zvoliť niekoľko uzlov s najväčším podielom v zvolenom sharde. Simulácia uvažuje, že útočníci si zvolia vždy päť najbohatších uzlov v každom sharde. Výsledok simulácie zobrazuje obrázok 5.12. V pôvodnom protokole stačil útok na štyri uzly a sieť prestala propagovať transakcie. V pozmenenom protokole sa v poslednej fáze útočí na 20 uzlov súčasne a priepustnosť transakcií nie je takmer vôbec ovplyvnená. Občasný pokles transakcií je zapríčinený tým, že útokom sa náhodou podarilo zasiahnuť aktuálneho vodcu v danom sharde. Výsledok experimentu ukazuje, že nahradenie rozvrhu vodcov za VRF voľbu výrazne zvýši odolnosť Harmony voči DoS útoku.

5.4.3 Ouroboros a Solana s VRF

Následujúci experiment vyhodnocuje odolnosť protokolov Ouroboros a Solana voči DoS útoku pri použití VRF voľby vodcu. V oboch prípadoch je simulovaná sieť veľká 1 500 uzlov a DoS útok je aplikovaný na 30 najbohatších uzlov. Pre objektívnejšie porovnanie je rozdelenie podielu v oboch sieťach generované z rovnomerného rozdelenia. Simulácia je nastavená tak, aby priepustnosť bez útoku bola približne 3 000 transakcií za slot.

Výsledok experimentu sumarizuje obrázok 5.13, ktorý ukazuje histogram propagovaných transakcií v čase zvlášť pre oba protokoly. Výsledky simulácie sú veľmi podobné v obidvoch prípadoch. Toto je očakávané, pretože oba protokoly používajú principiálne rovnaký spôsob voľby vodcu. Modrý histogram ukazuje priepustnosť transakcií v prípade použitia rozvrhu vodcov. Za 1 000 simulovaných slotov došlo v oboch protokoloch k približne 100 incidentom,



(a) Ouroboros

(b) Solana

Obr. 5.13: Solana a Ouroboros s VRF: Histogram priepustnosti transakcií pre DoS útok na rozvrh vodcov (modrá) a VRF voľbu (oranžová).

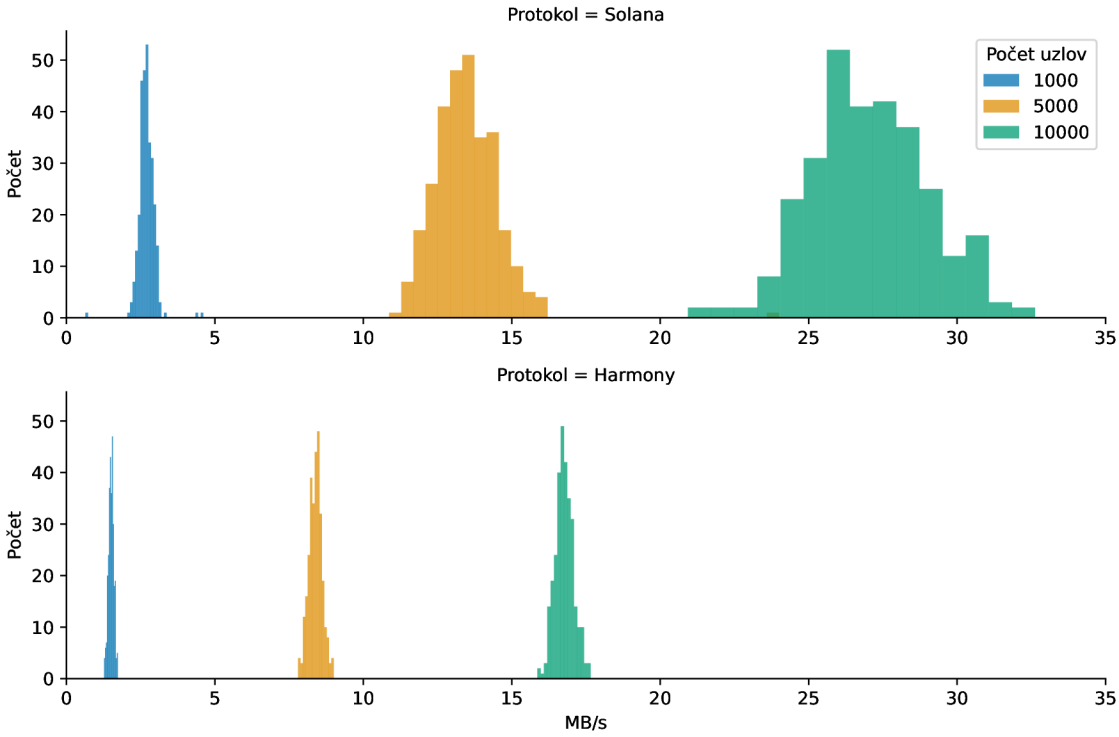
kedy DoS útok zasiahol aktuálneho vodcu slotu. V týchto prípadoch je nulová priepustnosť transakcií. Z dlhodobého hľadiska potom priepustnosť klesne priemerne o 10 %. Oranžové stĺpce zase ukazujú simuláciu rovnakého útoku, ale s použitím nepredvídateľnej voľby vodcu pomocou VRF. V oboch protokoloch je opäť podobný trend. Množstvo úspešných DoS útokov kleslo zo 100 na asi 20. Priemerne teda priepustnosť klesne len o 2 %. Pre simulovanú konfiguráciu teda VRF znížilo v oboch protokoloch približne päťnásobne pravdepodobnosť úspešného DoS útoku na vodcu.

5.5 Porovnanie škálovateľnosti

Následujúca sekcia zhodnocuje a porovnáva výkonnosť analyzovaných protokolov z hľadiska škálovateľnosti veľkosti siete. Pre objektívnosť porovnania sú simulované siete vždy rovnako veľké a rozdelenie podielu je rovnomerné. Vo všetkých sieťach vzniká rovnaké množstvo aplikačných transakcií. V prípade Harmony je počet shardov odvodený ako $\frac{n}{250}$, kde n je počet uzlov siete.

Solana a Harmony sú protokoly založené na hlasovaní a Ouroboros na náhodnej voľbe. Komunikačná záťaž je v prípade hlasovania nezanedbateľne väčšia. Porovnanie vyťaženosti sieťovej komunikácie medzi Harmony a Solana vzhľadom na veľkosť siete poukáže na škálovateľnosť protokolov. Naopak, porovnanie s protokolom Ouroboros je neobjektívne, pretože protokoly náhodnej voľby nemajú žiadnu nadbytočnú komunikačnú záťaž okrem samotnej distribúcie nového bloku. Tento simulačný experiment preto porovnáva množstvo prijatých dát za sekundu pre priemerný uzol v protokoloch Harmony a Solana. Experiment postupne simuluje siete s veľkosťami 1 000, 5 000 a 10 000 uzlov. Množstvo transakcií je pre oba protokoly rovnaké a je vždy priamo úmerne vzhľadom k veľkosti siete.

Výsledok experimentu ukazuje obrázok 5.14, ktorý zobrazuje histogram prijatých dát za sekundu pre priemerný uzol. Solana komunikačná záťaž rastie lineárne vzhľadom k veľkosti siete. Pre sieť 1 000 uzlov je medián prenesených dát 2,7 MB/s. Pre sieť 5 000 uzlov ide o 13,4 MB/s a pre 10 000 uzlov je to 29,9MB/s. Lineárny rast komunikačnej záťaže je dobrý výsledok. Navyše aj pre sieť s veľkosťou 10 000 uzlov je komunikácia pomerne



Obr. 5.14: Histogram sieťovej vyťaženosti protokolov Harmony a Solana pri hlasovaní v rôzne veľkých sieťach.

malá (30 MB/s je bezproblémový prenos pre 1 Gbps linku). Harmony komunikačná záťaž je vďaka shardingu ešte optimálnejšia. Pre sieť 1 000 uzlov je medián 1,5 MB/s, pre 5 000 ide o 8,4 MB/s a pre 10 000 predstavuje medián prenesených dát 16,8 MB/s. V priemere prenesie jeden uzol v Harmony o takmer polovicu menej dát, než Solana, a to nezávisle na veľkosti siete.

Výsledky experimentu ukazujú, že oba protokoly umožňuje efektívne škálovať veľkosť siete. Komunikačná záťaž rastie lineárne vzhľadom k veľkosti siete, čo je stále dobrý výsledok (napríklad klasický hlasovací algoritmus PBFT škáluje až kvadraticky). Protokol Harmony navyše, vďaka shardingu, dosahuje takmer o polovicu nižší dátový prenos ako Solana.

5.6 Zhrnutie a diskusia

Simulácia Solana konsenzu ukázala na jeho nezanedbateľnú komunikačnú záťaž spojenú s Proof-of-Stake hlasovaním. Na základe výsledkov experimentov doporučujem používať Solana konsenzus len v takých aplikáciách, ktoré generujú veľké množstvo transakcií za sekundu. Ak je veľkosť siete rádovo menšia ako množstvo transakcií, tak je hlasovanie neakceptovateľne drahé. Simulácia ďalej ukázala, že s rastúcou veľkosťou siete bude úmerne rásť aj finalita bloku (aj naďalej však ide o stovky milisekúnd).

Simulácia Harmony poukazuje na problémy spojené s shardingom. Táto technika predstavuje kompromis medzi výkonnosťou a bezpečnosťou. Simulácia ukázala, že ak nepoctivé uzly vlastnia 30 % podielu, tak majú vďaka shardingu možnosť prekaziť hlasovanie v sharde s pravdepodobnosťou 2 %. Útočník teda musí vlastniť veľmi veľkú časť podielu, aby mohol

využiť sharding vo svoj prospech. Na druhej strane, sharding neumožňuje až tak efektívnu škálovateľnosť, pretože každý uzol je približne v dvoch shardoch súčasne. Sieť sa teda nerozdelí na disjunktné množiny, ale na podskupiny, ktoré sa čiastočne prekrývajú.

Simulácia Ouroboros konsenzu poukazuje na problém vetvenia. Experimenty ukázali, že Ouroboros nedokáže garantovať okamžitú konzistentnosť bloku. Okrem zníženia komunikačnej záťaže spôsobenej absenciou hlasovania tento protokol nepriniesol žiadne bezpečnostné výhody oproti hlasovacím protokolom Solana a Harmony. Na základe týchto výsledkov doporučujem hlasovanie ako efektívnejší spôsob pre rýchle uznesenie konsenzu. Náhodná voľba použitá v protokole Ouroboros nemôže garantovať tak rýchlu konzistenciu, aby bol možný tento mechanizmus použiť v platobnom systéme porovnateľnom s Visa.

Simulácia poukázala na principiálnu slabinu Proof-of-Stake protokolov, ktoré používajú rozvrh tvorcov bloku. Solana aj Harmony hlasovanie je založené na BFT (anglicky *Byzantian Fault Tolerance*) mechanizme, ktorý po dobu hlasovania potrebuje jeden uzol v sieti označiť za vodcu. Tento uzol predstavuje po krátku dobu centrálnu autoritu. Pre správne fungovanie hlasovania musia všetci dopredu vedieť, o ktorý uzol pôjde. Toto však principiálne umožňuje DoS útok, pretože bez tohto uzlu nebude vytvorený nový blok. Ide o zraniteľnosť, ktorá v prípade Proof-of-Work nepredstavoval problém, pretože tam je tvorca bloku nepredvídateľný. Simulácia potvrdila túto zraniteľnosť vo všetkých troch protokoloch. Experimenty ale taktiež ukázali, že táto zraniteľnosť sa môže významne minimalizovať použitím VRF (*Verifiable Random Function*). VRF zabezpečí, že voľba vodcu bude dopredu nepredvídateľná. Experimenty ukázali, že pravdepodobnosť úspešného DoS útoku s použitím VRF klesne približne päťnásobne. Na základe týchto výsledkov doporučujem nahradiť problematický rozvrh vodcov za VRF nepredvídateľnú voľbu.

Kapitola 6

Záver

Táto práca sa zaoberala analýzou bezpečnosti a výkonnosti troch Proof-of-Stake protokolov pre uznesenie konsenzu v blockchaine: Harmony, Solana a Ouroboros. V prvej časti práce boli tieto protokoly teoreticky analyzované a porovnané z hľadiska bezpečnosti a výkonnosti. V rámci teoretickej analýzy sa podarilo odhaliť bezpečnostnú zraniteľnosť v podobe DoS útoku na tvorcu nového bloku. Tomuto problému čelia všetky tri protokoly z dôvodu predvídateľného určenia budúcich tvorcov blokov v čase. V druhej časti práce bol vytvorený simulačný nástroj, ktorý experimentoval s vlastnosťami analyzovaných protokolov. Simulovaná bola výkonnosť v rozsiahlych sieťach ako aj konkrétne útoky na bezpečnosť. Simulácia potvrdila zraniteľnosť na DoS útok. Hlavným úspechom simulačných experimentov je návrh a otestovanie modifikácie protokolov, ktorá túto zraniteľnosť minimalizuje.

Vytvorený simulačný nástroj je verejne dostupný a určený pre potenciálny ďalší vývoj. V budúcej práci by mohlo byť zaujímavé rozšíriť simulátor o ďalšie Proof-of-Stake protokoly a vytvoriť porovnanie väčšieho počtu protokolov. Najväčší prínos by pritom mala analýza protokolov založených na hlasovaní (podobne ako Harmony a Solana). Obzvlášť zaujímavé by bolo zamerať sa na protokoly, ktoré používajú pre zvýšenie výkonnosti sharding a porovnať ich efektivitu a bezpečnosť s protokolom Harmony, ktorý túto techniku používa tiež. Naopak, voľba protokolov podobných Ouroborosu sa spätne javí ako menej vhodná. Tento protokol spadá do inej rodiny konsenzu, a preto ho v mnohých oblastiach nebolo moc praktické porovnávať s protokolmi Harmony a Solana.

Pre naplnenie stanovených cieľov práce bolo potrebné hlbšie pochopenie komplexnej technológie blockchain. Veľká časť práce sa preto venuje teoretickej problematike Proof-of-Stake konsenzu a možností jeho simulácie. Získané poznatky boli použité na teoretické porovnanie zvolených protokolov, ktoré sú zhrnuté v sekcii 3.4. Na základe získaných teoretických znalostí bol zvyšok práce venovaný implementácii simulátoru zvolených protokolov a následnému simulačnému experimentovaniu. Na záver sekcia 5.6 diskutuje výsledky simulačných experimentov a navrhované modifikácie protokolov.

Literatúra

[1] Aberer, K., Alima, L., Ghodsi, A. et al. The Essence of P2P: A Reference Architecture for Overlay Networks. In: *Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*. Január 2005, s. 11– 20 [cit. 2022-05-06]. DOI: 10.1109/P2P.2005.38.

[2] Acharya, V., Yerrapati, A. a Prakash, N. *Oracle Blockchain Services Quick Start Guide*. Packt Publishing, september 2019 [cit. 2022-05-06]. 350 s. ISBN 1789804167.

[3] Al Jaroodi, J. a Mohamed, N. Blockchain in Industries: A Survey. *IEEE Access*. 2019, zv. 7, s. 36500–36515, [cit. 2022-05-06]. DOI: 10.1109/ACCESS.2019.2903554.

[4] Alharby, M. a Moorsel, A. van. BlockSim: An Extensible Simulation Tool for Blockchain Systems. *Frontiers in Blockchain*. 2020, zv. 3, s. 28, [cit. 2022-05-06]. DOI: 10.3389/fbloc.2020.00028. ISSN 2624-7852. Dostupné z: https://www.frontiersin.org/article/10.3389/fbloc.2020.00028.

[5] Ali, R., Brown, R. et al. *Distributed Ledger Technology: beyond block chain*. London, 1 Victoria Street, 2016 [cit. 2022-05-06]. Dostupné z: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf.

[6] Aoki, Y., Otsuki, K., Kaneko, T. et al. SimBlock: A Blockchain Network Simulator. In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops*. 2019, s. 325–329 [cit. 2022-05-06]. DOI: 10.1109/INFCOMW.2019.8845253.

[7] Baniata, H. a Kertész, A. FoBSim: An extensible open-source simulation tool for integrated Fog-Blockchain systems. *PeerJ Computer Science*. Október 2020, zv. 7, [cit. 2022-05-06].

[8] Blum, M. Coin Flipping by Telephone. In: *Advances in Cryptology: A Report on CRYPTO 81*. 1981, s. 11–15 [cit. 2022-05-06]. Dostupné z: https://www.iacr.org/archive/crypto81/11_blum.pdf.

[9] Boldyreva, A., Palacio, A. a Warinschi, B. Secure Proxy Signature Schemes for Delegation of Signing Rights. *Journal of Cryptology*. Jún 2003, zv. 25, [cit. 2022-05-06]. DOI: 10.1007/s00145-010-9082-x.

[10] Boneh, D., Bonneau, J., Bünz, B. et al. Verifiable delay functions. In: Springer. *Annual international cryptology conference*. 2018, s. 757–788 [cit. 2022-05-06].

[11] 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 [cit. 2022-05-06]. ISBN 3540429875.

[12] Borčík, F. *Testování bezpečnosti a výkonu Proof-of-Stake Protokolů pomocí simulace*. Brno, CZ, 2021. [cit. 2022-05-06]. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/22846/.

[13] 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, [cit. 2022-05-06]. DOI: 10.26438/ijcse/v6i8.294301.

[14] Bridge, V. a Nicolas, L. *Wittgenstein simulator* [online]. New York, USA [cit. 2022-05-06]. Dostupné z: https://github.com/ConsenSys/wittgenstein.

[15] Buford, J., Yu, H. a Lua, E. *P2P Networking and Applications*. December 2008 [cit. 2022-05-06]. 408 s. ISBN 0123742145. Dostupné z: https://www.amazon.com/Networking-Applications-Morgan-Kaufmann-Hardcover/dp/0123742145.

[16] Castro, M. a Liskov, B. Practical Byzantine Fault Tolerance. In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. New Orleans, Louisiana, USA: USENIX Association, 1999, s. 173–186 [cit. 2022-05-06]. ISBN 1880446391.

[17] De Angelis, S., Aniello, L., Lombardi, F. et al. PBFT vs Proof-of-Authority: Applying the CAP Theorem to Permissioned Blockchain. In: *Italian Conference on Cybersecurity*. Január 2017 [cit. 2022-05-06]. Dostupné z: https://www.researchgate.net/publication/320619309.

[18] Dingledine, R., Mathewson, N. a Syverson, P. Tor: The Second-Generation Onion Router. *Paul Syverson*. Jún 2004, zv. 13, [cit. 2022-05-06].

[19] Douceur, J. The Sybil Attack. In: *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*. Február 2002 [cit. 2022-05-06]. Dostupné z: https://www.microsoft.com/en-us/research/publication/the-sybil-attack/.

[20] Fan, C., Ghaemi, S., Khazaei, H. et al. Performance Evaluation of Blockchain Systems: A Systematic Survey. *IEEE Access*. 2020, zv. 8, s. 126927–126950, [cit. 2022-05-06]. DOI: 10.1109/ACCESS.2020.3006078.

[21] 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 [cit. 2022-05-06]. DOI: 10.1109/SFCS.1987.4.

[22] Gervais, A., Karame, G., Wust, K. 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 [cit. 2022-05-06]. DOI: 10.1145/2976749.2978341. ISBN 9781450341394. Dostupné z: https://doi.org/10.1145/2976749.2978341.

[23] Gilad, Y., Hemo, R., Micali, S. et al. In: *Algorand: Scaling Byzantine Agreements for Cryptocurrencies*. New York, NY, USA: Association for Computing Machinery, 2017, s. 51–68 [cit. 2022-05-06]. DOI: 10.1145/3132747.3132757. ISBN 9781450350853. Dostupné z: https://doi.org/10.1145/3132747.3132757.

[24] Gilbert, S. a Lynch, N. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. *ACM SIGACT News*. New York, NY, USA: Association for Computing Machinery. November 2002, zv. 33, č. 2, s. 51–59, [cit. 2022-05-06]. DOI: 10.1145/564585.564601.

[25] Gorbunov, S. *Algorand Releases First Open-Source Code: Verifiable Random Function* [online]. Medium, 2018 [cit. 2022-05-06]. Dostupné z: https://medium.com/algorand/algorand-releases-first-open-source-code-of-verifiable-random-function-93c2960abd61.

[26] Gouget, A., Patarin, J. a Toulemonde, A. Unpredictability properties in Algorand consensus protocol. In: *2021 IEEE International Conference on Blockchain and Cryptocurrency*. 2021, s. 1–3 [cit. 2022-05-06]. DOI: 10.1109/ICBC51069.2021.9461057.

[27] Hoepman, J.-H. *Distributed Double Spending Prevention* [online]. arXiv, marec 2008 [cit. 2022-05-06]. DOI: 10.48550/ARXIV.0802.0832. Dostupné z: https://arxiv.org/abs/0802.0832.

[28] Homoliak, I., Venugopalan, S., Reijsbergen, D. et al. The Security Reference Architecture for Blockchains: Toward a Standardized Model for Studying Vulnerabilities, Threats, and Defenses. *IEEE Communications Surveys Tutorials*. 2021, zv. 23, č. 1, s. 341–390, [cit. 2022-05-06]. DOI: 10.1109/COMST.2020.3033665.

[29] 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*. Internet Society, Február 2012 [cit. 2022-05-06].

[30] Kiayias, A., Russell, A., David, B. et al. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In: *Advances in Cryptology – CRYPTO 2017*. Cham: Springer International Publishing, Júl 2017, s. 357–388 [cit. 2022-05-06]. ISBN 978-3-319-63688-7.

[31] Lan, R. et al. *Harmony: Technical Whitepaper* [online]. Mountain View, CA [cit. 2022-05-06]. 22 s. Dostupné z: https://harmony.one/whitepaper.pdf.

[32] Lepore, C., Ceria, M., Visconti, A. et al. A Survey on Blockchain Consensus with a Performance Comparison of PoW, PoS and Pure PoS. *Mathematics*. Október 2020, zv. 8, s. 1782, [cit. 2022-05-06]. DOI: 10.3390/math8101782. Dostupné z: https://www.mdpi.com/2227-7390/8/10/1782.

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

[34] 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*. Washington, D.C., USA: USENIX Association, 2015, s. 7 [cit. 2022-05-06].

[35] Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System* [online]. Cryptography Mailing list at https://metzdowd.com, Marec 2009, s. 9 [cit. 2022-05-06]. Dostupné z: https://bitcoin.org/bitcoin.pdf.

[36] Narayanan, A., Bonneau, J., Felten, E. et al. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016 [cit. 2022-05-06]. 336 s. ISBN 0691171696.

[37] Nguyen, C., Hoang, D. T., Nguyen, D. et al. Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities. *IEEE Access*. Jún 2019, zv. 7, s. 1–1, [cit. 2022-05-06]. DOI: 10.1109/ACCESS.2019.2925010.

[38] Paulavičius, R., Grigaitis, S. a Filatovas, E. A Systematic Review and Empirical Analysis of Blockchain Simulators. *IEEE Access*. 2021, zv. 9, s. 38010–38028, [cit. 2022-05-06]. DOI: 10.1109/ACCESS.2021.3063324.

[39] Rice, J. *Mathematical Statistics and Data Analysis*. Brooks/Cole, apríl 2006 [cit. 2022-05-06]. ISBN 0495110892.

[40] Schollmeier, R. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In: *Proceedings First International Conference on Peer-to-Peer Computing*. 2001, s. 101–102. DOI: 10.1109/P2P.2001.990434.

[41] Smart, N. *Cryptography: An Introduction*. McGraw-Hill, 2003. ISBN 0077099877.

[42] Stoykov, L., Zhang, K. a Jacobsen, H.-A. VIBES: Fast Blockchain Simulations for Large-Scale Peer-to-Peer Networks: Demo. In: *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Posters and Demos*. New York, NY, USA: Association for Computing Machinery, 2017, s. 19–20 [cit. 2022-05-06]. DOI: 10.1145/3155016.3155020. ISBN 9781450352017. Dostupné z: https://doi.org/10.1145/3155016.3155020.

[43] Tse, S., Lan, R., Dewan, S. et al. *Harmony: Documentation* [online]. Mountain View, CA [cit. 2022-05-06]. Dostupné z: https://docs.harmony.one/home/.

[44] Viswanath, P. a Sankagiri, S. Bridging BFT protocols with the longest chain protocol. In: *Principles of Blockchain* [online]. University of Illinois, Marec 2021 [cit. 2022-05-06].

[45] Yakovenko, A. *Solana: Documentation* [online]. San Francisco, California [cit. 2022-05-06]. Dostupné z: https://docs.solana.com/.

[46] Yakovenko, A. *Solana: A new architecture for a high performance blockchain v0.8.13* [online]. San Francisco, California, 2017 [cit. 2022-05-06]. Dostupné z: https://solana.com/solana-whitepaper.pdf.

[47] Zhang, S. a Lee, J.-H. Analysis of the main consensus protocols of blockchain. *ICT Express*. 2020, zv. 6, č. 2, s. 93–97, [cit. 2022-05-06]. DOI: https://doi.org/10.1016/j.icte.2019.08.001. Dostupné z: https://www.sciencedirect.com/science/article/pii/S240595951930164X.

[48] Zheng, Z., Xie, S., Dai, H.-N. et al. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In: *2017 IEEE International Congress on Big Data (BigData Congress)*. Jún 2017, s. 557–564 [cit. 2022-05-06]. DOI: 10.1109/BigDataCongress.2017.85.

Príloha A

Návod pre prácu zo simulátorom

Nasledujúca kapitola poskytuje postup inštalácie simulátoru. Ďalej je popísané ako jednotlivo spustiť každý z experimentov popísaných v kapitole 5. Vytvorený simulátor je súčasťou pamäťového média priloženého k tejto práci, ale je taktiež dostupný z internetu[1].

Simulátor má nasledujúcu adresárovú štruktúru:

```
simulation/
|-- simulator-client
‘-- simulator-server
```

Pre jeho spustenie je potrebné spustiť serverovú a klientskú časť v nasledujúcom poradí:

1. *Server*: Serverová časť simulátoru je kontajnerizovaná technológiu Docker. Postup inštalácie a spustenia je popísaný v sekcii A.1. Celý postup je potrebné vykonať v adresári `simulation`.

2. *Klient*: Klientská aplikácia je Python konzolová aplikácia. Postup inštalácie potrebných závislostí je popísaný v sekcii A.2. Inštaláciu je potrebné vykonať v zložke `simulation/simulator-client`.

3. *Simulácia*: Po nainštalovaní serverovej aj klientskej časti aplikácie je možné spúšťať jednotlivé simulačné scenáre z klientskej aplikácie (adresár `simulation/simulator-client`). Postup spúšťania simulácie je popísaný v sekcii A.3.

A.1 Server

Pre spustenie simulátoru vo virtualizovanom prostredí je najprv potrebné nainštalovať Docker. Postupovať je možné podľa oficiálneho tutoriálu[2], alebo nasledovať postup popísaný v tejto sekcii:

- Postup inštalácie pre Windows A.1.1.
- Postup inštalácie pre Linux A.1.2.

Po nainštalovaní dockeru, je možné spustiť simulátor ako virtualizovanú serverovú aplikáciu. Spustiť aplikáciu je možné pomocou dvojice príkazov cez konzolu[3]:

[1] Verejný GitHub repozitár vytvoreného simulátoru: https://github.com/JurajHolub/wittgenstein

[2] https://docs.docker.com/get-docker/

[3] Tento proces nainštaluje všetky potrebné závislosti a preto vyžaduje internetové pripojenie a môže trvať niekoľko minút.

```
$ docker-compose build
$ docker-compose up
```

Po dokončení práce je možné simulátor zastaviť a zmazať pomocou príkazu:

```
$ docker-compose down -v
```

Prepínač `-v` zaručí, že bude zmazaný obsah MongoDB databázy prezistentne.

A.1.1 Inštalácia Dockeru pre Windows

Ak na zariadení ešte nie je nainštalovaný Docker, je možné použiť nasledujúci postup[4]:

1. Stiahnite inštalačný program z oficiálnej webovej stránky: https://desktop.docker.com/win/main/amd64/Docker%20Desktop%20Installer.exe
2. Spustite stiahnutý program a nainštaluj (vyžaduje administrátorský prístup).
3. Pomocou *Windows Search Bar* vyhľadajte aplikáciu *Docker Desktop* a spustite.
4. Po spustení bude *Docker Desktop* ponúkať používateľský tutoriál, ktorý nie je potrebné absolvovať (postačí preskočiť).

A.1.2 Inštalácia Dockeru pre Linux

Nasledujúci postup je určený pre prostredie Debian (pre iné linuxové prostredia je možné použiť nasledujúci postup[5]):

1. Odinštalujte staré verzie Dockeru, ak sú nainštalované:

   ```
   $ sudo apt-get remove docker docker-engine docker.io containerd runc
   ```

2. Aktualizujte a nainštalujte potrebné balíčky pomocou `apt`:

   ```
   $ sudo apt-get update
   $ sudo apt-get install ca-certificates curl gnupg lsb-release
   ```

3. Pridajte oficiálny Docker GPG kľúč:

   ```
   $ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg
       --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
   ```

4. Nastavte stabilnú verziu repozitára pre Docker:

   ```
   $ echo "deb [arch=$(dpkg --print-architecture)
       signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
       https://download.docker.com/linux/debian $(lsb_release -cs)
       stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
   ```

5. Nainštalujte Docker pomocou `apt`:

   ```
   $ sudo apt-get update
   $ sudo apt-get install docker-ce docker-ce-cli containerd.io
   ```

[4] Prevzaté z https://docs.docker.com/desktop/windows/install/
[5] Prevzaté z https://docs.docker.com/engine/install/

A.2 Klient

Klientská aplikácia je konzolový skript v programovacom jazyku Python. Pre jej spustenie je potrebné mať nainštalovaný `Python3.9`[6] spolu s nasledujúcimi knižnicami:

```
requests~=2.25.1
pandas~=1.3.3
matplotlib~=3.4.3
seaborn~=0.11.2
pymongo~=3.12.0
```

Knižnice je možné nainštalovať pomocou priloženého `requirements.txt` súboru:

```
$ pip install -r requirements.txt
```

A.3 Simulačné scenáre

Pre spustenie nápovedy ako používať simulačného klienta:

- Všeobecná nápoveda: `python client.py -h`
- Solana experimenty nápoveda: `python client.py solana -h`
- Harmony experimenty nápoveda: `python client.py harmony -h`
- Ouroboros experimenty nápoveda: `python client.py ouroboros -h`
- Nápoveda experimentov porovnávajúcich protokoly:
  `python client.py comparison -h`

Pre spustenie simulačných experimentov:

- Harmony viď A.3.1
- Solana viď A.3.2
- Ouroboros viď A.3.3
- Porovnanie protokolov viď A.3.4

Samotné simulačné scenáre môžu bežať aj niekoľko hodín. Výsledky simulácie sa uložia v podobe grafov a obrázkov uložených v adresári `output`. Výstup experimentu je pomenovaný rovnako ako samotný experiment. Napríklad, experiment `harmony --scenario01` vygeneruje výstup: `harmony-scenario01.png` a `harmony-scenario01.pdf`.

A.3.1 Harmony

1. Bezpečnosť distribúcia podielu medzi shardy (viď 5.2.1):

   ```
   $ python client.py --mongoserver mongodb:27017 harmony --scenario01
   ```

2. Útok na ovládnutie shardy (viď 5.2.2):

[6] Nainštalovať napríklad z https://www.python.org/downloads/

```
$ python client.py --mongoserver mongodb:27017 harmony --scenario02
```

3. Sharding priepustnosť a škálovateľnosť (viď 5.2.3):

```
$ python client.py --mongoserver mongodb:27017 harmony --scenario03
```

4. DoS útok na vodcu (viď 5.2.4):

```
$ python client.py --mongoserver mongodb:27017 harmony --scenario04
```

5. DoS útok na vodcu s použitím VRF (viď 5.4.2):

```
$ python client.py --mongoserver mongodb:27017 harmony --scenario05
```

A.3.2 Solana

1. Pomer hlasovacích a aplikačných transakcií (viď 5.1.1):

```
$ python client.py --mongoserver mongodb:27017 solana --scenario01
```

2. DoS útok na vodcu (viď 5.1.2):

```
$ python client.py --mongoserver mongodb:27017 solana --scenario02
```

3. Priepustnosť a škálovateľnosť (viď 5.1.3):

```
$ python client.py --mongoserver mongodb:27017 solana --scenario03
```

4. DoS útok na vodcu s VRF (viď 5.4.3):

```
$ python client.py --mongoserver mongodb:27017 solana --scenario04
```

A.3.3 Ouroboros

1. DoS útok na vodcu (viď 5.3.1):

```
$ python client.py --mongoserver mongodb:27017 solana --scenario01
```

2. Útok vetvením (krátkodobý double-spending) (viď 5.3.2):

```
$ python client.py --mongoserver mongodb:27017 solana --scenario02
```

3. DoS útok na vodcu s VRF (viď 5.4.3):

```
$ python client.py --mongoserver mongodb:27017 solana --scenario03
```

A.3.4 Porovnanie

1. Škálovateľnosť (viď 5.5):

```
$ python client.py --mongoserver mongodb:27017 comparison --scenario01
```