Usporedba radnih karakteristika konsenzus algoritama računalnom simulacijom

Šajina, Robert

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

DIPLOMSKI RAD

Usporedba radnih karakteristika konsenzus algoritama računalnom simulacijom

Robert Šajina

Pula, srpanj 2019.

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

DIPLOMSKI RAD

Usporedba radnih karakteristika konsenzus algoritama računalnom simulacijom

Robert Šajina

JMBAG: 0303054675, redovni student
Studijski smjer: Informatika

Predmet: Blockchain aplikacije
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc.dr.sc. Siniša Sovilj
Komentor: dr.sc. Nikola Tanković

Pula, srpanj 2019.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Robert Šajina, kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____, _____ godine

IZJAVA

o korištenju autorskog djela

Ja, Robert Šajina dajem odobrenje Sveučilištu Jurja Dobrile

u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom „Usporedba radnih karakteristika konsenzus algoritama računalnom simulacijom“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____ (datum)

Potpis

_____

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE

Pula, 21. ožujka 2019.

DIPLOMSKI ZADATAK

Pristupnik:	**Šajina Robert (0303054675)**
Studij:	Sveučilišni diplomski studij Informatike
Naslov (hrv.):	**Usporedba radnih karakteristika konsenzus algoritama računalnom simulacijom**
Naslov (eng.):	Performance comparison of consensus algorithms with computer simulation
Opis zadatka:	Zadatak ovog diplomskog je proučiti, objasniti i simulirati način na koji rade konsenzus algoritmi vrste Proof-of-Authority (skraćeno PoA). U svrhu razumijevanja i simulacije rada takvih algoritama rad će pojasniti postupak priključivanja čvorova na mrežu, te problem postizanja konsenzusa u okolini koja ne pretpostavlja povjerenje između čvorova. Princip PoA pretpostavlja da su čvorovi zaduženi za ostvarivanje konsenzusa poznati i rezultat su glasovanja svih čvorova. Rad će navesti i usporediti najpoznatije takve algoritme u vidu radnih karakteristika i karakteristika postizanja konsenzusa. Radne karakteristike utvrdit će se metodom računalne simulacije. Simulacijom će se dobiti uvid u radne karakteristike različitih konfiguracija pojedinog algoritma što omogućava ranu evaluaciju prilikom njegovog odabira u projektiranju budućih sustava.

Mentor:

Siniša Sovilj

_____

doc.dr.sc. Siniša Sovilj

Komentor:

_____

dr.sc. Nikola Tanković

ZAHVALA

Zahvaljujem svojem komentoru, dr.sc. Nikoli Tankoviću za njegovu potporu tijekom rada, pomaganju u poboljšanju kvalitete rada podjelom znanja i dostupnosti za raspravu u kratkom roku. Također želim zahvaliti mentoru, doc.dr.sc. Siniši Sovilju, na prilici da napišem diplomski rad na fakultetu Informatike u Puli. Konačno, želio bih se zahvaliti svojoj obitelji što su mi pružili beskrajnu podršku i poticanje tijekom mojih godina školovanja i kroz proces istraživanja i pisanja ovog rada. Takvo postignuće ne bi bilo moguće bez njih.

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

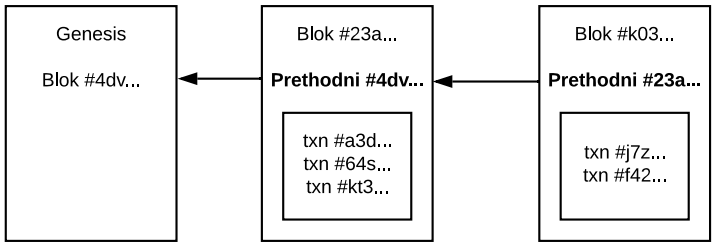
1. Uvod

Blockchain je jedna od najpopularnijih tehnologija posljednjih godina. Svoju primjenu najčešće nalazi u kripto valutama kao što su Bitcoin [32] i Ethereum [6]. Blockchain svojstva nepromjenjivosti podataka, integriteta i potpune decentralizacije, ključni su za njegovo široko korištenje, od računalstva u oblaku do poslovnih aplikacija. Svaki čvor unutar blockchain mreže je samoupravljajući i ponaša se neovisno o drugim čvorovima. Unatoč tome, sveukupno stanje blockchain sustava je identično. Ova suglasnost postiže se uz pomoć konsenzus algoritama. Početni konsenzus algoritmi temeljeni su na dokazivanju rada (Proof of Work), to jest, rješavanjem matematičkog problema koje zahtjeva veliku računalnu snagu. Zbog loših performansi i velike potrošnje električne energije koje ovaj konsenzus nudi, predloženi su novi konsenzus algoritmi, kao što je Proof of Stake. Također, za privatne blockchain sustave, gdje su sudionici poznati, predloženi su konsenzus algoritmi Proof of Authority (PoA). Testiranje takvih konsenzus algoritama s većim brojem čvorova zahtjeva dobru hardversku podlogu. Jeftiniji i brži način analize ponašanja konsenzus algoritama je kroz računalnu simulaciju. Simulator koji oponaša rad sustava može dati dobru, brzu i jeftinu aproksimaciju svojstava sustava za određene početne parametre sustava. Organizacija rada podijeljena je na sljedeći način: 2. poglavlje započinje s teorijskim znanjem potrebnim za razumijevanje sustava. Poglavlje 3 pruža uvid u odnos konzistentnosti i dostupnosti između dva PoA konsenzus algoritma Clique i Aura. Četvrto poglavlje prikazuje dizajn i način rada simulatora, zajedno sa simulacijama i rezultatima. Peto poglavlje pruža uvid u povezan rad, gdje su razmotreni različiti alati u peer-to-peer i blockchain sustavu. U posljednjem poglavlju pružamo svoja opažanja, zaključke i mogući budući rad. Simulator je dostupan na https://github.com/rosaj/poasim.

2. Pozadina

2.1. Blockchain

Blockchain (lanac blokova) je skup povezanih podatkovnih struktura koje organiziraju uređeni skup transakcija u blokove. To je razlog zašto se često blockchain naziva i distribuirana knjiga transakcija (DTL - Distributed Transaction Ledger). Blockchain započinje s inicijalnim blokom koji se naziva *genesis* blok. Prilikom dodavanja novih blokova u blockchain, povezuju se s prethodnim blokom pokazivanjem na hash prethodnog bloka [33].



Slika 2.1: Reprezentacija blockchain-a. Svaki blok ima referencu na hash prošlog bloka

Proces kreiranja bloka naziva se rudarenje (eng. mining) i provodi se na čvorovima koji se nazivaju rudari (eng. miners), koji periodično predlažu nove blokove suglasno s distribuiranim konsenzus protokolom. Na taj način sudionici postižu sporazum o procesuiranim transakcijama, dijele jednako stanje i garantiraju konzistentnost. Zato što je decentraliziran i potpuno replicirajući, blockchain se također nosi s nužnošću povjerenja u središnju vlast kao što to čine najopćenitiji moderni sustavi. Ne postoji potreba za pouzdanom trećom stranom koja kontrolira sve transakcije i upravlja podacima. Svaki čvor može neovisno provjeriti konzistentnost dijeljenog stanja blockchain-a. Ove važne karakteristike osiguravaju integritet i nepromjenjivost podataka, budući da bi svaka nepravilna promjena (npr. neovlašteno mijenjanje stanja) bila odmah otkrivena i odbačena od strane blockchain mreže [17].

Nakon što miner kreira blok, šalje taj blok po blockchain mreži. Tada se taj blok smatra kao najnoviji blok lanca. Svi miner-i tada kreću u rudarenje novih blokova koji će biti uključeni na blockchain. Ako više miner-a istodobno dodaju blok u blockchain, stvara se nestalno račvanje (eng. fork) koje se obično rješava s vremenom spajanjem drugih blokova, ovisno o algoritmu rješavanja račvanja blockchain-a. Miner-i su potaknuti na ispravno ponašanje,

putem nagrada za uspješno kreiranje blokova i naknada od obrade svake od transakcija [17].

Kada je blok dio lanca, to znači da su se svi miner-i suglasili o njegovom sadržaju, što znači da je praktički nepromjenjiv i ustrajan, osim ako napadač pripada napadačkoj grupi koja posjeduje više od polovice blockchain mreže. U tom slučaju napadač je u stanju stvoriti račvanje lanaca ili poništiti transakciju. Pretpostavljajući da je u rudarenju većina poštenih miner-a, vjerojatnost račvanja, s dubinom n je $O(2^{-n})$ [17].

2.1.1. Bitcoin

Bitcoin je poznat kao prva implementacija digitalne valute na blockchain tehnologiji [32]. To je elektronički sustav plaćanja koji se temeljni na kriptografskim dokazima, koji izbjegava potrebu za centraliziranom vlasti (npr. bankom). Trenutni sustavi plaćanja zahtijevaju pouzdanu treću stranu za obradu transakcija. Ovo posredovanje često podliježe transakcijskim troškovima, vremenskim čekanjima obrade, time ograničavajući minimalnu praktičnu veličinu transakcije i mogućnost za male povremene transakcije. Bitcoin kao konsenzus protokol koristi Proof-of-Work(PoW), koji je pobliže objašnjen u poglavlju 2.4.1. Prosječni razmak između kreiranja novih blokova je 10 minuta, čime se dozvoljava dovoljno vremena kako bi se novo kreirani blok propagirao po mreži [17]. Kao princip rješavanja račvanja lanaca, Bitcoin koristi princip najduljeg lanca, gdje se kao trenutni lanac uzima lanac koji ima najviše uzastopnih blokova, odnosno lanac na kojem je napravljeno najviše rudarskog posla.

2.1.2. Ethereum

Ethereum se često navodi kao kripto konkurent Bitcoin-u, ali zapravo je puno više od toga. Kao što navodi [6], token unutar Ethereum-a, Ether, nije zamišljena kao kripto valuta, već kao gorivo za izvršavanja programa na Ethereum blockchain-u. Ethereum omogućuje decentraliziranu logiku „svjetskog računala" pružajući platformu za pametne ugovore (eng. smart contract) temeljene na blockchain-u, a to su programi koji provode pravila koja postavljaju kreatori. Izvršenje ugovora potvrđuju svi sudionici blockchain-a, što znači da se ti programi izvode bez zastoja, prijevare ili smetnji trećih strana. Iako Bitcoin također sadržava koncept pametnih ugovora, njihova je upotreba ograničena na valutne transakcije. Bitna razlika Ethereum-a u odnosu na Bitcoin je ugrađeni Turing-kompletan programski jezik koji omogućuje bilo kome stvaranje ugovora za bilo koju uporabu izvan valutne transakcije [25].

EVM

Izvršavanje pametnih ugovora događa se unutar Ethereum Virtual Machine (EVM) [38]. Pozivanje pametnih ugovora izvršava se kroz transakcije. Stog svaka transakcija ima dva dodatna polja: *Gas Price* - skalarna vrijednost koja predstavlja broj Ether-a koji se plaća

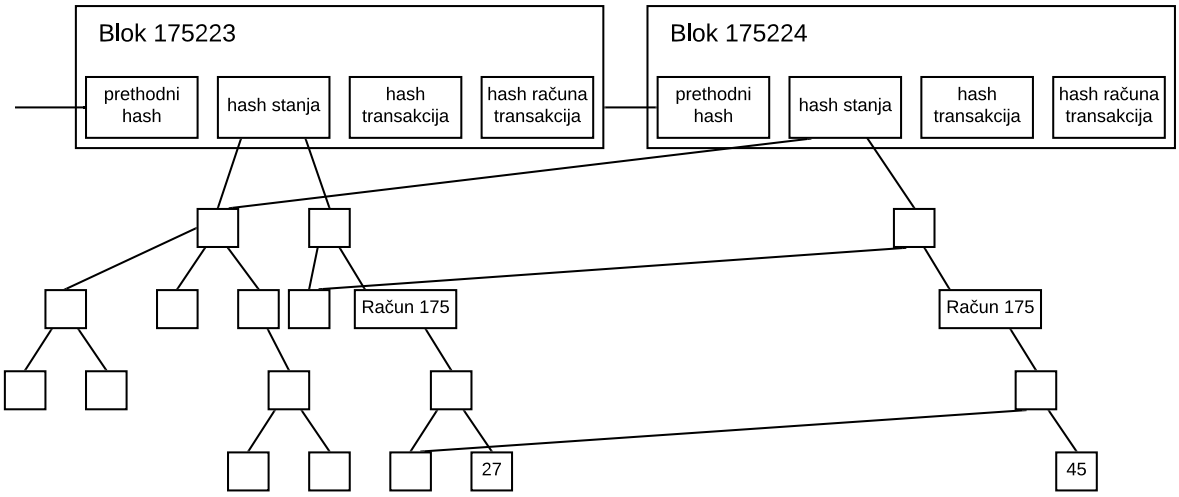
za svaki komputacijski korak izvršavanja transakcije; *Gas Limit* - skalarna vrijednost koja predstavlja maksimalnu količinu goriva se može iskoristiti za izvršenje transakcije. Kako bi se spriječile slučajne ili namjerne beskonačne petlje ili drugi komputacijski gubitak u kodu, svaka transakcija mora imati komputacijski limit izvršavanja koda. Jedinica goriva postavlja sustav provizija te na takav način zahtjeva od napadača da plaćaju, proporcionalno svakom resursu kojeg potroše, uključujući računanje, propusnost i pohranu. Stoga, bilo kakva transakcija koja vodi do toga da mreža troši veću količinu bilo kojeg od tih resursa mora imati proviziju goriva približno proporcionalno povećanju [17].

Merkle Patricia stablo

Blok u Ethereum blockchain-u se sastoji od zaglavlja, liste transakcija i liste *uncle*-a. U zaglavlje je uključen hash korijena transakcija, koji se koristi za validaciju transakcija uključenih u blok. Transakcije bloka stvaraju posebnu podatkovnu strukturu nazvanu Merkle Patricia stablo, odnosno trie. Ta podatkovna struktura koristi se kod verifikacije bloka, kao i dokazivanja uključenosti transakcije u blok, dobivanja trenutnog salda računa, provjere postojanja računa i sl. Uz transakcijski trie, u zaglavlju se nalaze i trie računa transakcija i trie stanja. Trie računa transakcija prati račune generirane nakon izvođenja transakcije. Trie stanja sadrži vrijednosti svih računa s kojima se klijent susreo do trenutnog datuma. To se stanje sastoji od ključ-vrijednost mape u kojoj su ključevi adrese računa korisnika, a vrijednosti deklaracije računa, vrijednost salda, *nonce*, kôd i pohrana (pohrana je također stablo). Za razliku od trie-a transakcija i trie-a računa transakcija, koji se ne mogu mijenjati, trie stanja se često mijenja: saldo i *nonce* računa se čestom mijenjaju, novi računi se često dodaju, ključevi u pohrani se često dodaju ili brišu. U tim slučajevima je trie implementacija iznimno brza pri kalkulaciji novog hash-a korijena. Prilikom operacije dodavanja, uređivanja ili brisanja novo izračunati korijen računa se bez ponovnog računanja cijelog stabla. Trie stablo također karakteriziraju dva važna svojstva [15, 16]:

- Dubina stabla je ograničena. Napadač bi mogao izvršiti napad uskraćivanja usluge (DOS - Denial of service) manipuliranjem stabla kako bi postalo što dublje kako bi svako pojedinačno ažuriranje postalo iznimno sporo
- Korijen stabla ovisi samo o podacima, a ne o redoslijedu kojim se vrši ažuriranje stabla [16]

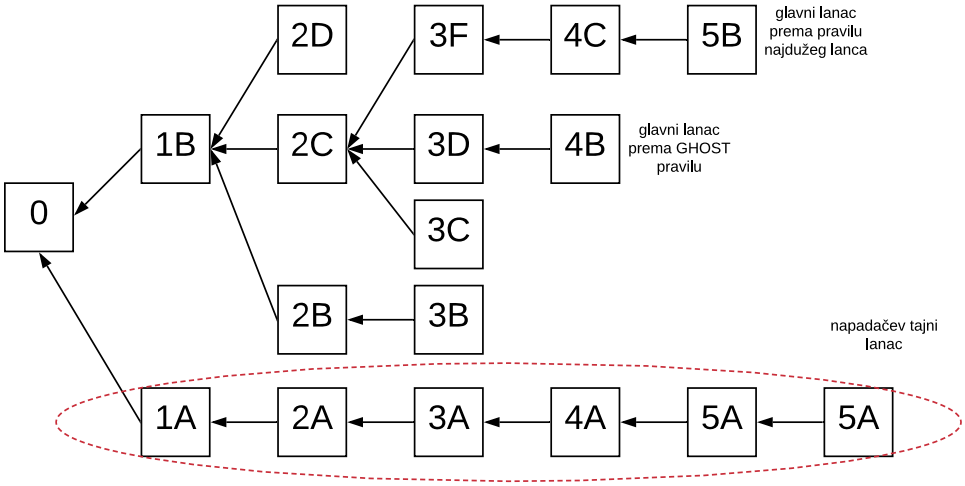
Pronalazak podatka u stablu ostvaruje se pomoću ključa. Ključ pod kojim je vrijednost pohranjena, pretvoren je u putanju kojom se dolazi do podatka. Svaki čvor ima 16 djece, tako da je putanja određena heksadecimalnim šifratom (npr. ključ „dog" u heksimalnoj reprezentaciji izgleda kao 6 5 6 15 6 7, što znači da se krećući od korijena, pretraživanje spušta na 6. dijete, tada na 4. dijete, i tako dalje sve dok ne dođe do kraja) [16].



Slika 2.2: Merkle Patricia stablo stanja [16]

GHOST Protokol

Vrijeme kreiranja blokova u Ethereum blockchain mreži je 15 sekundi. S tako kratkim vremenom kreiranja blokova, blockchain mreža pati od niske sigurnosti zbog visoke stope istodobnog kreiranja novih blokova. Za propagaciju bloka kroz mrežu potrebno je određeno vrijeme, te se u međuvremenu može dogoditi da neki od ostalih miner-a također kreira blok, prije nego što zaprimi već kreirani blok. Umjesto da se ti kreirani duplicirani blokovi odbace, uključuju se u blok kao potomci pretka bloka, to jest *uncles*. Svaki blok može imati maksimalno 2 *uncle*-a. Za razliku od prihvaćanja najduljeg lanca kao principa rješavanja račvanja blockchain-a, Ethereum koristi GHOST (Greedy Heaviest Observed Subtree) princip prikazano na slici 2.3, u kojeg se uključuju i *uncle*-ovi, čime se dokazuje kako je za određeni lanac odrađeno najviše rudarenja [17].



Slika 2.3: Stablo grananja blockchain-a gdje najduži lanac nije odabran prema GHOST protokolu [17]

2.2. Vrste blockchain-a

Blockchain sustavi mogu se klasificirati na temelju pristupa u različitim modelima ovlasti. Sudionici blockchain mreže imaju pravo na:

- pristup podacima na blockchain-u (čitanje)
- slanje transakcija (pisanje)
- ažuriranje stanja s novim blokovima (spremanje) [17]

Blockchain se može podijeliti na dvije glavne klase:

- javni blockchain: nema ograničenja na operacije čitanja
- privatni blockchain: samo je unaprijed definiranom popisu entiteta dopuštene operacije čitanja [17]

Stoga su za operacije pisanja i spremanja identificirane dvije klase blockchain-a:

- permissionless (bez dozvole): nema ograničena na operacije pisanja i spremanja
- permissioned (potrebna dozvola): samo su unaprijed definiranoj listi entiteta dopuštene operacije pisanja i spremanja [17]

U permissionless blockchain-u svatko se može pridružiti mreži, te izvršavati operacije pisanja i čitanja te sudjelovati u konsenzusu, odnosno predlaganju novih blokova. Nasuprot tome, u permissioned blockchian-u čvorovi su poznati unaprijed, te samo ti čvorovi mogu kontrolirati blockchain. Javni permissionless blockchain, kao što je Bitcoin ili Ethereum, djeluje u neprijateljskom okruženju, te zahtijeva korištenje kriptografskih tehnika za poticanje sudionika na pošteno ponašanje. Nasuprot tome, privatni permissioned blockchain-ovi djeluju u okruženju u kojem su sudionici već poznati. U tom scenariju čvorovi su potaknuti djelovati pošteno, zato što su autentificirani, tako da se svako loše ponašanje može otkriti i osuditi [17].

2.3. Peer-to-peer

Peer-to-peer (veza ravnopravnih računala) mreža je mreža formirana spajanjem dvaju ili više uređaja bez potrebe za središnjem koordinatorom. Svi uređaju u mreži imaju istu razinu privilegija te jednako sudjeluju u radu mreže [34]. Za razliku od tradicionalnog modela klijent-poslužitelj, gdje postoji strogo odvajanje uloga, potrošača i proizvođača, u peer-to-peer sustavima ovo odvajanje se izbjegava jer uređaj može biti i potrošač i proizvođač [20]. U peer-to-peer mreži sustav radi neometano i stabilno ukoliko se čvor pridruži ili isključi iz mreže, dok se u standardnom klijent-server modelu sustav usporava ili uopće ne radi.

2.3.1. Otkrivanje čvorova

Kako bi se čvorovi mreže povezali međusobno, moraju proći kroz fazu otkrivanja ostalih čvorova mreže. Otkrivanje čvorova mreže bazira se na Kademlia protokolu.

Kademlia

Kademlia je protokol baziran na UDP-u za pohranu i dohvaćanje podataka u distribuiranom okruženju. Nasumično generirani ID-evi čvorova i ključevi pohranjenih podataka (generirani hashiranjem podatka) su oboje predstavljeni kao 160-bitne vrijednosti, što omogućuje izravnu usporedbu ID-a čvora i podatkovnih ključeva. Kademlia koristi bitnu XOR operaciju za računanje udaljenosti d(a,b) = a $\oplus$ b između dvije 160-bitne vrijednosti, a zatim, koristeći cjelobrojnu vrijednost udaljenosti, pohranjuje podatke u čvorovima čiji su ID-evi u blizini ključa podatka. Rad Kademlie ovisi o tom lokalnom, deterministčkom mapiranju između podataka i mrežnih čvorova, kako bi otkrila ostale čvorove mreže s velikom brzinom [27].

Svaki Kadmelia čvor održava vlastitu tablicu usmjerivanja za praćenje međusobno povezanih čvorova te određivanja susjednih čvorova koji pohranjuju podatke za određeni ključ. Tablica usmjeravanja podijeljena je na 160 redaka baziranih na XOR udaljenosti između vlastitog ID-a i ID-a susjednog čvora. Svaki redak sadrži listu čvorova koji su udaljeni između 2^i i 2^{i+1} u XOR udaljenosti. Svaki redak je ograničen s maksimalno *k* čvorova, stoga se naziva *k*-redak. Čvorovi unutar *k*-retka sortirani su po uvjetu aktivnosti, odnosno na početku nalaze se čvorovi s kojima čvor nije dugo vremena komunicirao, dok se na kraju retka nalaze čvorovi s kojima je čvor nedavno komunicirao. Prilikom otkrivanja susjednog čvora, Kademlia dodaje novo otkrivenog čvora u odgovarajući *k*-redak. Ukoliko je *k*-redak popunjen s maksimalnim brojem čvorova, Kademlia pogoduje starim poznatim čvorovima te dodaje novi čvor u tablicu, samo ukoliko najstariji postojeći aktivni čvor ne odgovara na *PING* poruku sa *PONG* porukom [27].

Kako bi se novo inicijalizirani čvor priključio mreži, odnosno povezao sa svojim susjedima, najprije dodaje statički set opće poznatih čvorova (BOOT čvor) u svoju tablicu usmjeravanja. Prilikom pokušaja lociranja određenog čvora na mreži, čvor pretražuje svoju tablicu usmjeravanja za *k* najbližih čvorova ciljanom čvoru. Prilikom pretraživanja tablice, najprije se pretražuje *k*-redak u kojem se nalaze najbliži čvorovi traženom čvoru. Ukoliko je *k*-redak prazan, rekurzivno pretražuje se sljedeći *k+1*-redak, sve dok se ne prikupi *k* čvorova, odnosno dok se ne pretraži cijelu tablicu usmjeravanja. Čvor tada odabranim susjednim čvorovima šalje *FIND_NODE* poruku u kojoj specificira ciljani ID čvora. Svaki od odabranih čvorova odgovara s listom *k* najbližih čvorova iz vlastite tablice usmjerivanja, pretražujući tablicu na prije spomenut način. Ukoliko je broj svih čvorova u tablici usmjeravanja manji od *k*, čvor odgovara s listom svih čvorova za koje on zna. Nakon zaprimanja odgovora na

upit, čvor u svoju tablicu usmjeravanja sprema sve novootkrivene čvorove, a zatim ponavlja postupak pretraživanja mreže sve dok ne pronađe traženi čvor ili u svojim upitima ne sazna za niti jedan novi čvor [31].

RLPx

To je TCP baziran transportni protokol koji se koristi za komunikaciju između Ethereum čvorova. Sve su poruke kriptirane. Nazvan je po RLP serijalizacijskom formatu kojeg Ethereum protokol koristi [29].

RLPx protokol baziran je na Kademlia protokolu. Ipak, ova dva protokola nisu jednaka. Za razliku od Kademlie, RLPx ne podržava pohranu/dohvaćanje podataka, već podržava samo otkrivanje čvorova i usmjeravanje. Nadalje, RLPx koristi 512-bitni ID čvora, umjesto 160-bitnog ID-a čvora kojeg koristi Kademlia. ID čvora također služi kao javni ključ istog, te se koristi u RLPx kako bi se uspostavila provjerena TCP veza koja pruža sigurnosne značajke, uključujući potpisane pakete i enkripciju, nakon ECIES (eng. Elliptic Curve Integrated Encryption Scheme) razmjene ključeva. Za računanje udaljenosti između čvorova, RLPx ne koristi računanje XOR udaljenosti izravno na ID-evima kao što nalaže Kademlia, već računa logaritam XOR udaljenosti Keccak-256 hash-a [13] ID-eva čvorova, što odgovara 257 različitih redaka udaljenosti [27]. Prilikom osvježavanja tablice usmjeravanja, RLPx čvor svojim susjedima, najprije šalje *FIND_NODE* poruku sa svojim ID-om kao ciljem kako bi pronašao sebi bliske susjede. Tada šalje još 3 *FIND_NODE* poruke s nasumičnim ID-om, kako bi pokušao popuniti i ostale retke tablice usmjeravanja. Faktor paralelnih *FIND_NODE* zahtjeva ograničen je sa a. RLPx kao parametre koristi $k = 16$, $a = 3$.

$$distance(ID_1, ID_2) = floor(log(Keccak256(ID_1) \oplus Keccak256(ID_2)))$$

2.3.2. Komunikacija između čvorova

DEVp2p

Nakon što se susjedni čvorovi pronađu kroz RLPx otkrivanje, i nakon što se uspostavi sigurna TCP konekcija, DEVp2p uspostavlja aplikacijsku sesiju između dva povezana čvora mreže. Svaki od čvorova, svojem susjedu najprije šalje *HELLO* poruku, koja sadrži ID čvora, DEVp2p verziju, ime klijenta, podržane aplikacijske protokole, odnosno verzije, te broj porta (početno zadana vrijednost je 30303) koju čvor osluškuje. Na temelju informacija *HELLO* poruke, čvorovi uspostavljaju ili raskidaju sigurnu vezu [27]. Veze između čvorova uspostavlja se ukoliko imaju barem jedan zajednički protokol, te ukoliko nijedan od čvorova već nije dostigao maksimalni broj povezanih čvorova mreže. Nakon uspostave veze između čvorova, svaki od čvorova pokreće protokole koji su zajednički za oba čvora. DEVp2p periodično

(svakih 15 sekundi) šalje *DEVp2p PING* poruku (nije isto što i *RLPx PING* poruka) kako bi provjerio da li su njegovi povezani čvorovi još uvijek aktivni. Ukoliko DEVp2p čvor ne odgovori *DEVp2p PONG* porukom, pošiljatelj *DEVp2p PING* poruke šalje *DISCONNECT* poruku, koja može sadržavati i razlog prekida veze, te prekida vezu s čvorom.

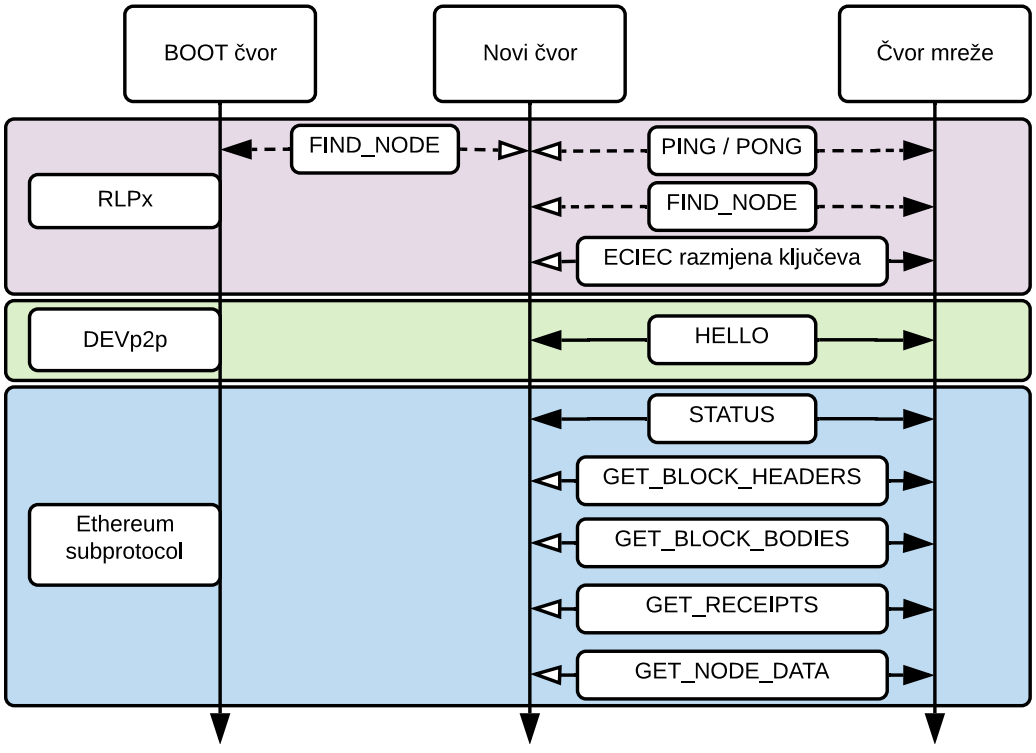
Ethereum Subprotocol

Ethereum subprotokol radi nad DEVp2p protokolom, te se označava kao „eth" tijekom *DEVp2p HELLO* razmjene poruka. Ethereum subprotokol se koristi za dohvaćanje i pohranjivanje informacija na Ethereum blockchain. Kao što je prije spomenuto, DEVp2p nakon uspostavljanja veze s čvorom mreže, pokreće sve protokole koji su zajednički za oba čvora mreže. Prilikom pokretanja Ethereum subprotokola, oba čvora dužni su poslati *STATUS* poruku, koja sadrži informacije o blockchain mreži na kojoj se čvor nalazi (NetworkId), Keccak-256 hash prvog (Genesis) bloka, verziju protokola, Keccak-256 hash trenutnog bloka u lokalnom blockchain-u čvora, te trenutnu težinu generiranja blokova [27]. Ukoliko se dva čvora nalaze na različitim blockchain mrežama, imaju različit Genesis blok, ili podržavaju različite verzije protokola, nemoguće je uspostaviti vezu između ta dva čvora, te se veza prekida i na DEVp2p razini.

Čvorovi koji uspješno uspostave vezu započinju sinkronizaciju. Sinkronizacija se sastoji od dva djela: sinkronizacija transakcija, te sinkronizacija blockchain-a. Odmah nakon razmjenjivanja *STATUS* poruka, oba čvora razmjenjuju međusobno sve transakcije koje još nisu uključene u blok. Oba čvora, neovisno o lokalnom stanju blockchain-a, propagiraju novoprimljene transakcije i blokove koje prime od svojih povezanih susjednih čvorova. Ukoliko se čvor prvi puta priključuje na blockchain mrežu ili je zbog nekog razloga bio nedostupan (rušenje sustava, problemi s mrežom), mora se sinkronizirati s ostatkom blockchain mreže. Za vrijeme sinkronizacije, čvor ne može predlagati nove blokove. Sinkronizacija čvora se pokreće tek kada se čvor spoji s barem *minDesiredPeerCount* (u geth implementaciji je to 5) čvorova blockchain mreže, ili kada se pokrene sinkronizacijski proces koji se pokreće svakih 10 sekunda, te se tada kao sinkronizacijski čvor odabire čvor koji ima najnovije stanje blockchain-a [5, 7]. Sinkronizacija započinje slanjem *GET_BLOCK_HEADERS* porukom kako bi se dobio popis zaglavlja, koji sadrže meta podatke bloka, kao što su hash prethodnog bloka, adresa miner-a, vrijeme i sl. Nakon što čvor zaprimi cijeli popis zaglavlja, šalje *GET_BLOCK_BODIES* poruku za dohvaćanje punog sadržaja blokova. Pri preuzimanju blockchain-a, validiraju se zaglavlja i blokovi. Nakon što se blok validira dodaje se u blockchain. Dodavanjem bloka u blockchain pokreće se blockchain validacija stanja. Ona se sastoji od sekvencijalnog izvođenja svih transakcija, prateći svaku promjenu računa u globalnoj bazi podataka, te spremanjem svakog stanja kao čvor unutar globalnog Merkle Particia stabla stanja [27].

Kako bi se smanjilo vrijeme sinkronizacije i validacije cijelog blockchain-a, Ethereum verzije 63 uvela je brzu sinkronizaciju, način rada koji smanjuje radno opterećenje validacije stanja blockchaina, te poboljšava vrijeme sinkronizacije na način da ne izvršava sve transakcije unutar blokova, već odabere blok pri samom vrhu blockchain-a mreže, te počne preuzimati cijelo Merkle Particia stablo i kôd za pametne ugovore, inkrementalno počevši od korijena stabla, njegove djece, unučadi itd., dohvaćajući tako sve čvorove dok cijelo stablo nije sinkronizirano [5, 7].

Nakon što je čvor u potpunosti sinkroniziran s mrežom, počinje aktivno sudjelovati u blockchain mreži generiranjem blokova. Prilikom zaprimanja novih transakcija, transakcije se validiraju lokalno kako bi se osiguralo da su ispravno potpisane, ne prelaze granice veličina/goriva (Gas), ne prenose negativnu vrijednost, te imaju dovoljno sredstava (Ether) za prijenos iznosa i plaćanja izvođenja transakcije. Nakon što su transakcije validirane, čvor šalje *TRANSACTIONS* poruku s novim transakcijama, svim svojim povezanim susjednim čvorovima, osim onima za koje zna da za tu transakciju već znaju. Čvorovi po primitku *TRANSACTIONS* poruke ponavljaju isti postupak. Propagacija novih blokova obavlja se na sličan način koristeći *NEW_BLOCK_HASHES* i *NEW_BLOCK* poruke.



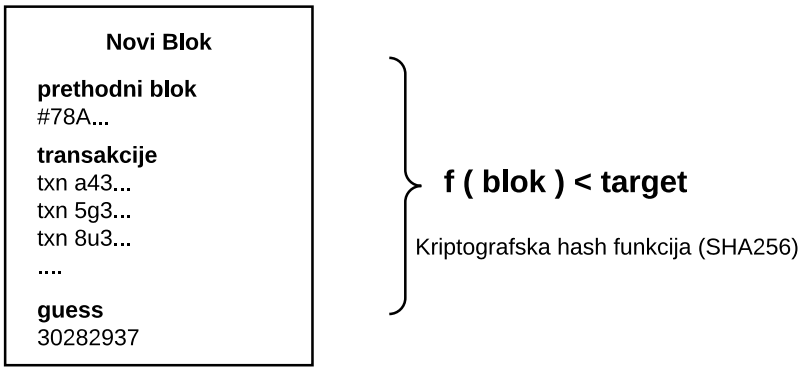
Slika 2.4: Ethereum mrežni protokoli— pregled RLPx, DEVp2p i Ethereum subprotokola. Ovi protokoli komuniciraju koristeći UDP (iscrtkana linija) i TCP (puna linija), kroz niz zahtjeva (puna strelica) i odgovora (prazna strelica). Preuzeto sa [27] uz nadopunu autora

2.4. Konsenzus protokol

Unutar blockchain konteksta, uloga konsenzus protokola je definiranje algoritma pomoću kojeg se svi čvorovi na blockchain mreži mogu dogovoriti o kanoničnom slijedu blokova. Većina ti protokola ima neku razinu tolerantnosti na bizantske pogreške (BFT - Byzantine-fault-tolerance) [28]. To znači da se ti protokoli mogu pretrpjeti neispravno ponašanje dijela čvorova mreže koji se nazivaju bizantski (Byzantine) čvorovi. U to neispravno ponašanje ubraja se bilo kakav proizvoljan rad čvora, uključujući odgađanje slanja poruka poštenim čvorovima i kreiranja tajnog sporazuma između čvorova te izoliranja dijela mreže [33].

2.4.1. Proof of Work

Proof of Work je trenutno najraširenija oblik konsenzusa blockchain-a. PoW je prvi uveo Bitcoin [32] koji prisiljava čvora blockchain mreže na riješavanje kriptografkog koputacijskog problema kao dokaza poštenosti [23]. Kako bi riješili problem, miner-i moraju pronaći nasumičnu vrijednost *guess*. Takav broj konkateniran s transakcijama sakupljenih unutar bloka, hash-om prethodnog bloka, mora dati cjelokupni hash manji od ciljanog broja. Ciljani broj je reguliran s blockchain *difficulty* vrijednosti koja regulira prosječno vrijeme rudarenja potrošenog od strane rudara kako bi riješili zadatak i predložili blok. Kako bi se pružio dokaz o rješavanju zadatka, u posebnu varijabla *nonce* zapisuje se broj koraka potrebnih za rješavanje problema. Na taj način svi ostali čvorovi blockchain mreže jednostavno validiraju ispravnost dokaza [17].



Slika 2.5: Proof of Work komputacijski problem za predlaganje bloka [17]

2.4.2. Proof of Stake

Proof of Stake koristi deterministično (pseudo-slučajno) rješenje za definiranje kreatora sljedećeg bloka, a vjerojatnost da se čvor odabere kao predlagatelj ovisi o njegovom bogatstvu - ulogu (eng. stake). Proces kreiranja blokova unutar PoS konsenzusa naziva se *minting*.

Uglavnom su svi kripto valutni novci kreirani na početku blockchain-a, tako da se ukupan broj nova ne mijenja nakon toga. Dakle, ne postoje nagrade za kreiranje blokova, već samo provizije od obrade transakcija koje su uključene u kreirani blok [17].

PoS konsenzus ima mnogo različitih implementacija, koje se razlikuju po načinu glasanja, mehanizmima nagrađivanja itd. Ipak, mogu se podijeliti na dva glavna tipa:

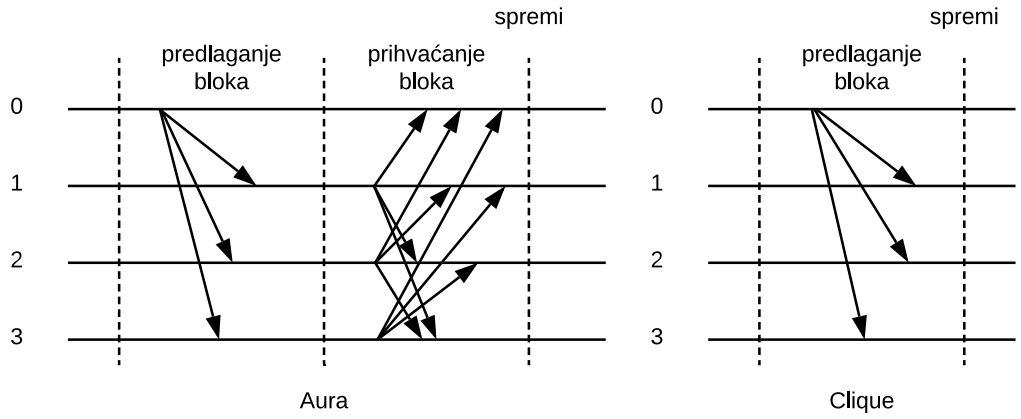
- bazirani na lancu (eng. Chain-based): kreator bloka je odabran pseudo-slučajno s pravom dodavanja bloka na blockchain tijekom rezerviranog vremenskog razdoblja (npr. svakih 15 sekundi može biti vremensko razdoblje)
- BFT bazirani: predlagači bloka su nasumično odabrani. Odabrani predlagači tada sudjeluju u postizanju konsenzusa kroz višestruki proces gdje svaki validator šalje svoj „glas" za neki određeni blok tijekom svake iteracije procesa, te se na kraju svi predlagatelji slože oko toga je li blok dio lanaca [17, 26].

2.4.3. Proof of Authority

Proof of Authority je vrsta konsenzus algoritama za premissioned blockchain mreže koji se ističu zbog svojih povećanja performansi u odnosu na tipične BFT algoritme. PoA je izvorno predložen kao dio Ethereum ekosustava za privatne mreže, implementirajući dva takva algoritma: Aura i Clique. PoA algoritmi oslanjaju se na skup *N* poznatih čvorova nazvanih autoritetima. Svaki je autoritet identificiran jedinstvenim identifikacijskim brojem (ID) te se za većinu njih, najmanje $N/2+1$, smatra poštenima. Autoriteti sudjeluju u konsenzusu kako bi prihvatili transakcije zaprimljene od klijenata. Konsenzus u PoA algoritmima zasniva se na rudarskoj rotacijskoj shemi, široko korišten pristup za pravednu raspodjelu odgovornosti kreiranja blokova između autoriteta. Vrijeme je podijeljeno na korake, u kojem je jedan od autoriteta odabran kao rudarski predvodnik. Budući da ne postoji komputacijski težak proces, kao što je PoW, proces rudarenja unutar PoA algoritama naziva se *minting*. Ove dvije PoA implementacije djeluju na različite načine: obje imaju prvi krug gdje novi blok predlaže trenutni predvodnik; tada Aura zahtijeva daljnji krug prihvaćanja bloka, dok Clique to ne zahtijeva [18]. Na slici 2.6 prikazan je proces razmjene poruka jedne runde predlaganja blokova.

Aura

Aura, Authority Round [1], je PoA algoritam implementiran u Parity [8], Etherum klijent baziran na Rust programskom jeziku [4]. Pretpostavlja se da je mreža sinkrona kao i da su svi autoriteti sinkronizirani s jednakim UNIX vremenom. Vrijeme je podjeljeno na diskretne korake trajanja t (vrijeme između kreiranja novog bloka). Korak s računa se prema formuli $s = \frac{UNIX}{t}$. Svaki korak ima predodređenog autoriteta koji ima ovlasti predložiti novi blok.



Slika 2.6: Proces kreiranja i prihvaćanja novog bloka sa 4 autoriteta, gdje autoritet 0 predlaže blok [18]

Predlaganje više od jednog bloka ili izvanredno predlaganje bloka smatra se nedopuštenim. Autoritet koji predlaže blok u koraku s računa se kao čvor s indeksom $i = s \bmod N$ [1, 18].

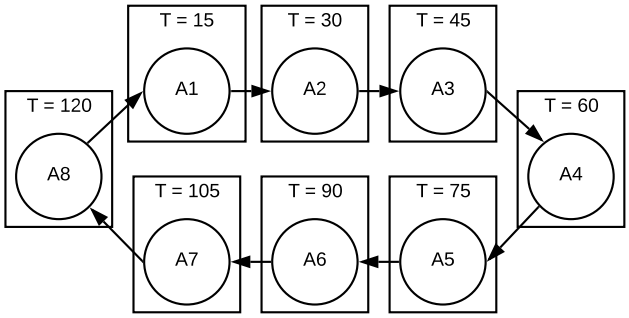
Svaki od autoriteta lokalno održava dvije liste realizirane kao red čekanja. Jedna lista služi za ne uključene transakcije $Qtxn$, dok druga služi za trenutne nepotvrđene blokove Qb. Svaka zaprimljena transakcija dodaje se u $Qtxn$ listu. Za svaki korak s, predvodnik i uključuje transakcije iz $Qtxn$ reda u blok b, te ga propagira ostalim autoritetima (na slici 2.6 označeno sa *predlaganje bloka*). Tada svaki autoritet šalje zaprimljeni blok ostalim autoritetima (na slici 2.6 označeno sa *prihvaćanje bloka*). Ukoliko svi autoriteti zaprime isti blok b, prihvaćaju blok b dodavajući ga u listu Qb. Blokovi zaprimljeni od bilo kojeg autoriteta koji nije očekivan autoritet za trenutni korak s su odbačeni. Od predvodnika koraka s se uvijek očekuje slanje bloka. Ukoliko se nije nijedna transakcija dodala u listu $Qtxn$, predvodnik je dužan poslati prazan blok, blok bez transakcija [4, 18].

Ako se autoriteti ne slože oko predloženog bloka tijekom faze prihvaćanja bloka, pokreće se glasanje kako bi se odlučilo o tome je li trenutni vođa zlonamjeran, odnosno treba li ga izbaciti iz liste autoriteta. Postoji više razloga za izbacivanje autoriteta iz liste autoriteta:

- autoritet nije predložio blok za korak s u kojem je on predvodnik
- autoritet je predložio više blokova od očekivanog
- autoritet je predložio različite blokove različitim autoritetima [18]

Mehanizam glasanja ostvaren je kroz pametan ugovor, te je za izbacivanje autoriteta iz liste legitimnih autoriteta potrebna većina glasova ostalih autoriteta. Ukoliko dođe do izbacivanja autoriteta, svi blokovi predloženi od autoriteta i se izbacuju iz liste blokova Qb. Razlozi nepoštenog ponašanja predvodnika mogu biti blage pogreške kao što su mrežna asinhronija, pad sustava, odstupanje sata u odnosnu na ostale autoritete, ili bizantske pogreške gdje se vođa ponaša na zlonamjeran način [18].

Blok b ostaje u listi Qb dok većina autoriteta ne predloži svoje blokove, te se tada blok b sprema na blockchain. S većinom poštenih autoriteta, ovaj bi mehanizam trebao spriječiti bilo koju manjinu (čak i u uzastopnim koracima) bizantskih predvodnika spremanje bloka koje su oni predložili na blockchain, što znači da bilo kakvo sumnjivo ponašanje autoriteta može odbaciti blokove koje jef on predložio prije nego se uopće spreme na blockchian. [1, 4, 18]. Na slici 2.7 prikazano je predlaganje blokova za 8 autoriteta i vremenom $T = 15$ kao razmakom između predlaganja blokova.



Slika 2.7: Redosljed mogućnosti predlaganja blokova u odnosu na proteklo vrijeme u Aura konsenzusu

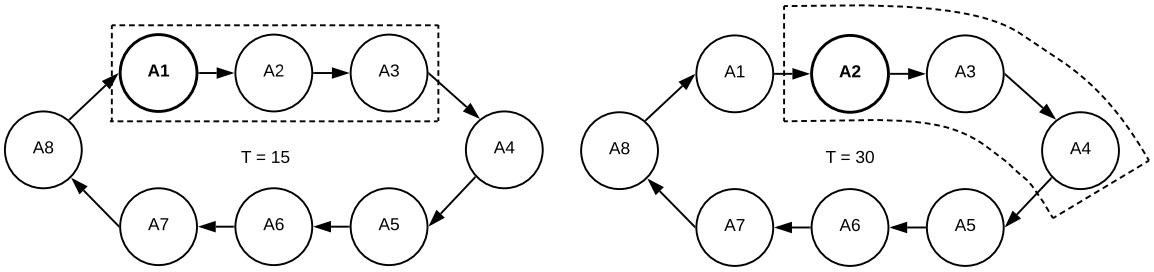
Clique

Clique [2] je Geth [7], Etherum klijent napisan u Golang-u, implementacija PoA konsenzus algoritma. Algoritam djeluje u epohama koje su indetificirane prethodnim slijedom spremljenih blokova. Kada započne nova epoha, unutar bloka se specificira skup autoriteta, odnosno njihove adrese, koji će se koristiti kao snimak trenutog blockchain-a od strane novh autoriteta koji se priključuju i sinkroniziraju s mrežom [18].

Za razliku od Aura konsenzusa, Clique izračunava trenutni korak i povezanog predvodnika pomoću formule koja kombinira broj bloka i broj autoriteta. Također, u Clique konsenzusu, osim predvodnika trenutnog koraka, drugim autoritetima je dopušteno predlaganje blokova u određenom koraku. Kako bi se spriječilo nepošteno ponašanje, takvo da jedan bizantski autoritet, nametanjem velikog broja blokova uguši mrežu, svakom autoritetu je predlaganje bloka dopušteno samo u svakih $\frac{n}{2} + 1$ koraka. Stoga u bilo kojem trenutku postoji najviše $N - (\frac{N}{2} + 1)$ autoriteta koji mogu predložiti blok. Slično kao i kod Aura konsenzusa, ako se autoritet ponaša zlonamjerno, može ga se izglasati iz popisa legitimnih autoriteta. Konkretno, glasovanje protiv autoriteta se može dati na svakom koraku, te se ako se postigne većina usuglašenih autoriteta, autoritet se briše s popisa legitimnih autoriteta [2, 7, 18].

Zbog toga što je više autoriteta u mogućnosti predložiti blok, mogu se pojaviti račvanja blockchain-a. Zato je vjerojatnost istodobnih predlaganja blokova ograničena činjenicom da je svaki autoritet, koji nije trenutni predvodnik koraka, odgađa propagaciju svog bloka

za nasumično vrijeme, čime daje prednost stvarnom predvodniku koraka. Ukoliko dođe do račvanja, prije navedeni GHOST protokol će razriješiti račvanje, na način da predvodnici koraka imaju prednost, čime se zapravo garantira da će se račvanje konačno riješiti [2, 7, 18]. Na slici 2.8 su prikazane dva uzastopna koraka unutar Clique konsenzusa.



Slika 2.8: Redosljed mogućnosti predlaganja blokova u odnosu na proteklo vrijeme u Clique konsenzusu [18]

2.5. DES - Simulacija diskretnih događaja

Simulacija diskretnih događaja (Discrete event simulation) je metoda koja se koristi za modeliranje sustava stvarnog svijeta koji se mogu dekomponirati na skup logički odvojenih procesa koji autonomno napreduju kroz vrijeme. Svaki se događaj izvršava u određenom procesu, te mu se dodjeljuje logičko vrijeme (vremenska oznaka). Rezultat ovog događaja može se prenositi na jedan ili više drugih procesa. Sadržaj ishoda može rezultirati generiranjem novih događaja koji će se obraditi u nekom određenom budućem logičkom vremenu. Simulacijski modeli analiziraju se numeričkim, a ne analitičkim metodama. Analitičke metode koriste deduktivno matematičko rezoniranje kako bi „riješili" model. U slučaju simulacijskih modela, koji upotrebljavaju numeričke metode, modeli se „izvode", a ne rješavaju, to jest, umjetna povijest sustava generirana je iz pretpostavki modela, a opažanja se prikupljaju kako bi se analizirala u svrhu procijene mjera performansi stvarnih sustava [11].

Svaki dinamički i stohastički sustav koji se mijenja na diskretan način sadrži neke od najbitnijih elemenata prikazanih u tablici 2.1. Aktivnost tipično predstavlja vrijeme usluge, vrijeme između dolaska entiteta, ili bilo koje drugo vrijeme obrade čije je trajanje karakterizirano i definirano. Trajanje aktivnosti može se definirati na više načina: deterministički (npr. svakih 5 minuta), statistički (npr. nasumičan odabir iz skupa s jednakim vjerojatnostima), funkcijski ovisno varijablama sustava i/ili atributa entiteta (npr. vrijeme ukrcavanja autobusa kao funkcija maksimalnog broja sjedala i brzina ulaska osoba u minuti). Za razliku od aktivnosti, modeliranje odgađanja nije predefinirano, već je određeno uvjetima sustava. Vrlo se često mjeri vrijeme trajanja odgode, te je ono jedno od željenih izlaz modela. Tipično, odgoda završava kada neki skup logičkih uvjeta postane istinit ili se dogodi jedan ili

Element	Značenje
Sustav	Skup entiteta koji tijekom vremena djeluju međusobno kako bi ostvarili jedan ili više ciljeva
Model	Apstraktna reprezentacija sustava koja obično sadržava strukturne, logičke ili matematičke odnose koji opisuju sustav u pogledu stanja, entiteta i njihovih atributa, procesa, događaja, aktivnosti i odgoda
Stanje sustava	Skup varijabli koji sadrži sve potrebne informacije kako bi se sustav opisao u bilo koje vrijeme
Entitet	Objekt ili komponenta sustava koja zahtjeva eksplicitnu reprezentaciju unutar modela (npr. klijent, poslužitelj)
Atributi	Svojstva entiteta (npr. prioritet klijenta koji čeka)
Lista	Skup logički poredanih trajnih ili privremenih entiteta
Događaj	Trenutačna pojava koja mijenja stanje sustava
Obavijest o događaju	Zapis o događaju koji će se dogoditi u trenutnom ili budućem vremenu, zajedno sa povezanim podacima potrebnim za izvršenje događaja.
Lista događaja	Lista obavijesti za buduće događaje, poredana prema vremenu pojavljivanja (FEL - Future Event List)
Aktivnost	Vremensko trajanje određene duljine za koje je poznat početak (npr. vrijeme usluge)
Odgoda	Vremensko trajanje neodređene duljine, koje nije poznato dok se ne završi
Sat	Varijabla koja predstavlja simulirano vrijeme

Tablica 2.1: Elementi simulatora diskretnih događaja [11]

više događaja. Ti sustavi se mijenjaju tijekom vremena. Stoga, stanje sustava, atributi entiteta i broj aktivnih entiteta, aktivnosti i odgode trenutno u tijeku su sve funkcije vremena koje se stalno mijenjaju kroz vrijeme. Varijabla *Sat* predstavlja vrijeme unutar simulacije. Simulacija diskretnih događaja proizvodi niz snimaka sustava koji reprezentiraju evoluciju sustava kroz vrijeme. Mehanizam za vremensko upravljanje simulacije i jamstva za kronološko izvršavanje događaja baziran je na listi budućih događaja. Ta lista sadrži sve obavijesti o događajima koji su zakazani za buduće vrijeme. U stvarnom svijetu, većina budućih događaja nije zakazana, već se samo događaju [11].

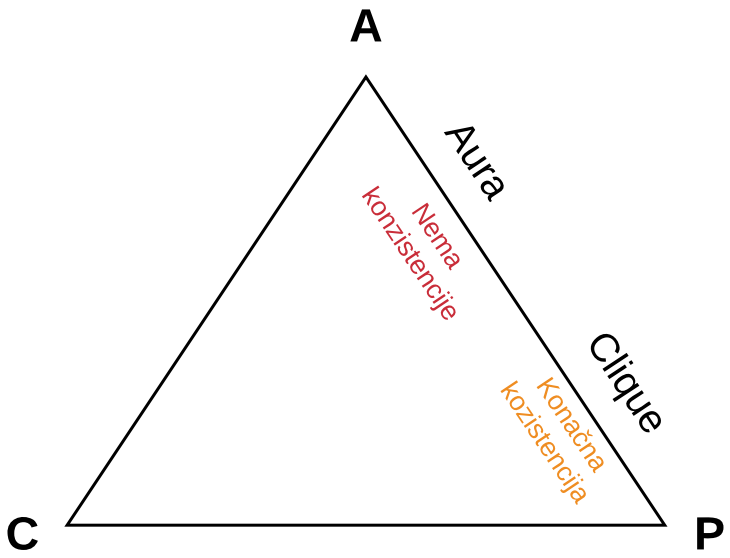
3. Primjena CAP teorema na Aura i Clique konsenzus algoritme

CAP teorem [14] navodi da se u distribuiranoj bazi podataka mogu postignuti samo dva od tri sljedećih svojstava: konzistentnost (C), dostupnost (A) i tolerancija particije (P). Svaka se distribuirana baza podataka može okarakterizirati na temelju najviše dva svojstva za koje može garantirati. Kombinacije tih svojstava mogu biti CA, CP ili AP, međutim budući da replicirani sustavi mogu biti podložni kvarovima koji se mogu smatrati particioniranjem, i budući da tolerancija particije mora biti garantirana, ostaju samo CP ili AP kombinacije svojstava [17, 18].

Blockchain postiže konzistentnost kada se izbjegavaju račvanja lanca (fork). Ovo se svojstvo naziva konačnost konsenzusa. Kada se konzistencija ne može postići, potrebno je razlikovati jesu li račvanja razriješena prije ili kasnije (konačna konzistencija) ili nisu uopće razriješena (nema konzistencije). Blockchain se smatra dostupnim ako se transakcije koje su dostavili klijenti obrađuju, i konačno dodaju u lanac. Kada dođe do mrežne particije, autoriteti su podijeljeni na nepovezane grupe na takav način da čvorovi u različitim skupinama ne mogu međusobno komunicirati. Iz ovih razloga permissioned blockchain mreže moraju tolerirati te nepovoljne situacije: razdoblja u kojima se mreža ponaša asinkrono; određen broj bizantskih autoriteta s ciljem ometanja dostupnosti i dosljednosti (maksimalan broj toleriranih bizantskih autoriteta za PoA je N/2) [17, 18].

Budući da se Aura temelji na UNIX usklađenom vremenu, satovi autoriteta devijacijom mogu postati ne sinkronizirani. Kada su autoriteti geografski distribuirani na širokom području, postupci resinkronizacije ne mogu biti djelotvorni zbog konačne sinkronizacije mreže. Stoga mogu postojati razdoblja u kojima se autoriteti ne slažu oko trenutnog koraka i samim time oko trenutnog predlagatelja bloka. Za odstupanja satova može se pretpostaviti da su niža od trajanja jednog koraka, te mogu biti u trajanju od nekoliko sekundi, stoga postoje kratki vremenski okviri gdje su dva različita autoriteta smatrani kao predlagatelj bloka u dvije nepovezane skupine autoriteta. Takva pojava može kritički utjecati na dosljednost cijelog sustava. Zato što Clique dopušta više od jednom autoritetu predlaganje bloka, s različitim nasumičnim odgodama, može se nositi s autoritetima koji nisu predložili nikakav blok

zbog asinkrone mreže ili bizantske greške. Iako zbog višestrukog predlaganja blokova dolazi do račvanja lanca, ta se račvanja svejedno rješavaju GHOST protokolom, te stoga postoji konačna konzistencija [17, 18]. Klasifikacija Clique i Aura algoritama prema CAP teoremu prikazana je na slici 3.1.



Slika 3.1: Klasifikacija Clique i Aura konsenzus algoritma prema CAP teoremu [18]

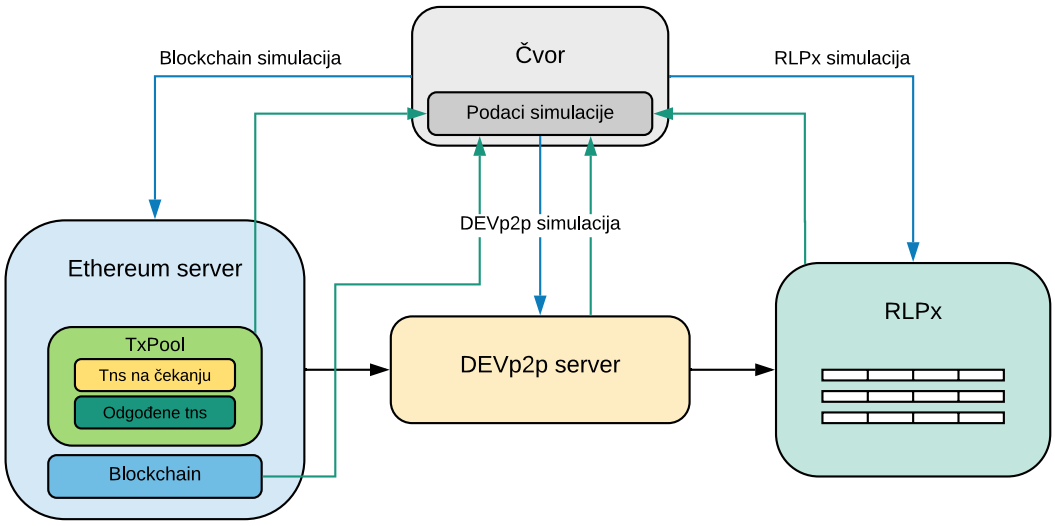
4. Simulator

4.1. Uvod u simulator

Simulator je implementiran u Golang [3] programskom jeziku, koristeći Godes [24] knjižnicu za simulaciju diskretnih događaja, te Ethereum implementaciju blockchain-a, Geth [7]. Simulator obuhvaća sva tri sloja Ethereum mreže:

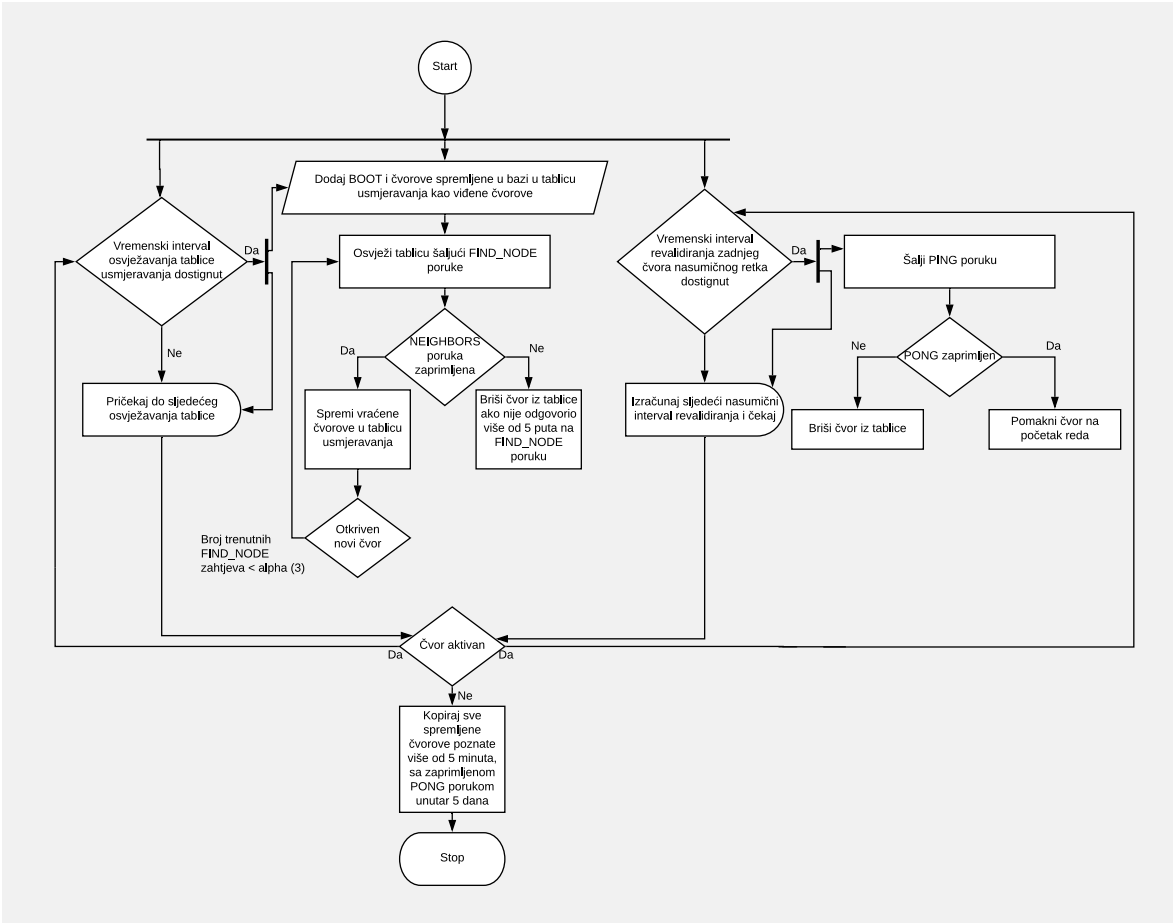
- Sloj pronalaska čvorova - RLPx
- Sloj povezivanja čvorova mreže - DEVp2p
- Sloj blockchain-a - Ethereum subprotokol

Simulator omogućava simuliranje svakog od slojeva zasebno. Svaki od slojeva aktivno prikuplja podatke tijekom simulacije. Također za simulaciju svakog od viših slojeva, simulira se i prethodni sloj. Na slici 4.1 prikazana je struktura čvora simulatora. Čvor sadrži strukturu u kojoj se spremaju svi podaci vezani za taj čvor tijekom simulacije. Ovisno o načinu pokretanja simulacije, čvor pokreće potrebne slojeve. RLPx sloj se u suštini uvijek simulira jer je on najniži simulacijski sloj. Slojevi sve svoje promjene spremaju u strukturu podataka vezanu za čvor. Podaci se spremaju po vremenskim grupama koje su određene parametrom simulacije.



Slika 4.1: Reprezentacija modela čvora simulatora

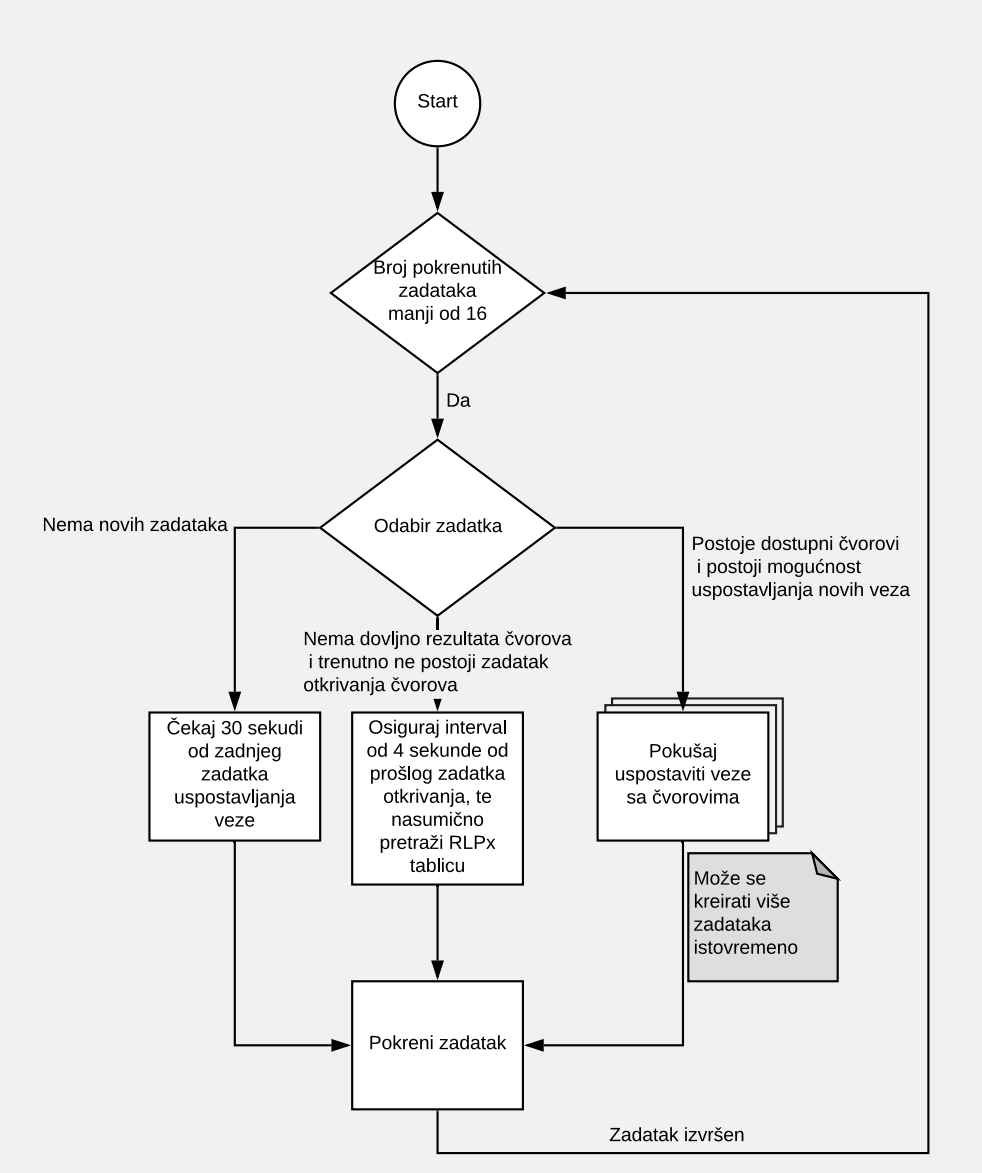
Sloj RLPx, prikazan na slici 4.2 implementiran je na sličan način kao i u Geth [7] implementaciji. Prilikom pokretanja, tablica usmjeravanja puni se s BOOT čvorovima, te čvorovima spremljenim u bazi. Tablica se tada osvježava šaljući *FIND_NODE* poruke čvorovima iz tablice usmjeravanja. Ukoliko čvor ne odgovori na više od *maxFindnodeFailures* (5), isti se briše iz tablice usmjeravanja. Čvorovi koje čvor dobije kao odgovor, spremaju se u tablicu usmjeravanja, ukoliko zapis o tom čvoru već ne postoji. Svakom novo saznalom čvoru šalje se *FIND_NODE* poruka s istim ciljem pretraživanja, sve dok je broj trenutnih *FIND_NODE* zahtjeva manji od *alpha*. Pri pokretanju, pokreću se dva paralelna zadataka. Prvi zadatak svakih *refreshInterval* minuta osvježava tablicu, na isti način kao što je prije opisano. Drugi zadatak u nasumičnim intervalima, s gornjom granicom od *revalidateInterval* sekundi, šalje *PING* poruku zadnjem čvoru nasumičnog *k*-retka. Ukoliko čvor ne odgovori s odgovarajućom *PONG* porukom briše se iz tablice usmjeravanja.



Slika 4.2: Princip rada RLPx sloja

Na slici 4.3 prikazan je rad DEVp2p sloja. Prilikom pokretanja, pokreću se zadaci povezivanja i otkrivanja čvorova. Broj paralelno pokrenutih zadataka ograničen je sa *maxActiveDialTasks*. Ukoliko nema dovoljno čvorova za uspostavljanje veza, pokreće zadatak otkrivanja novih čvorova nad RLPx tablicom usmjeravanja. Rezultati pronalaska spremaju se u

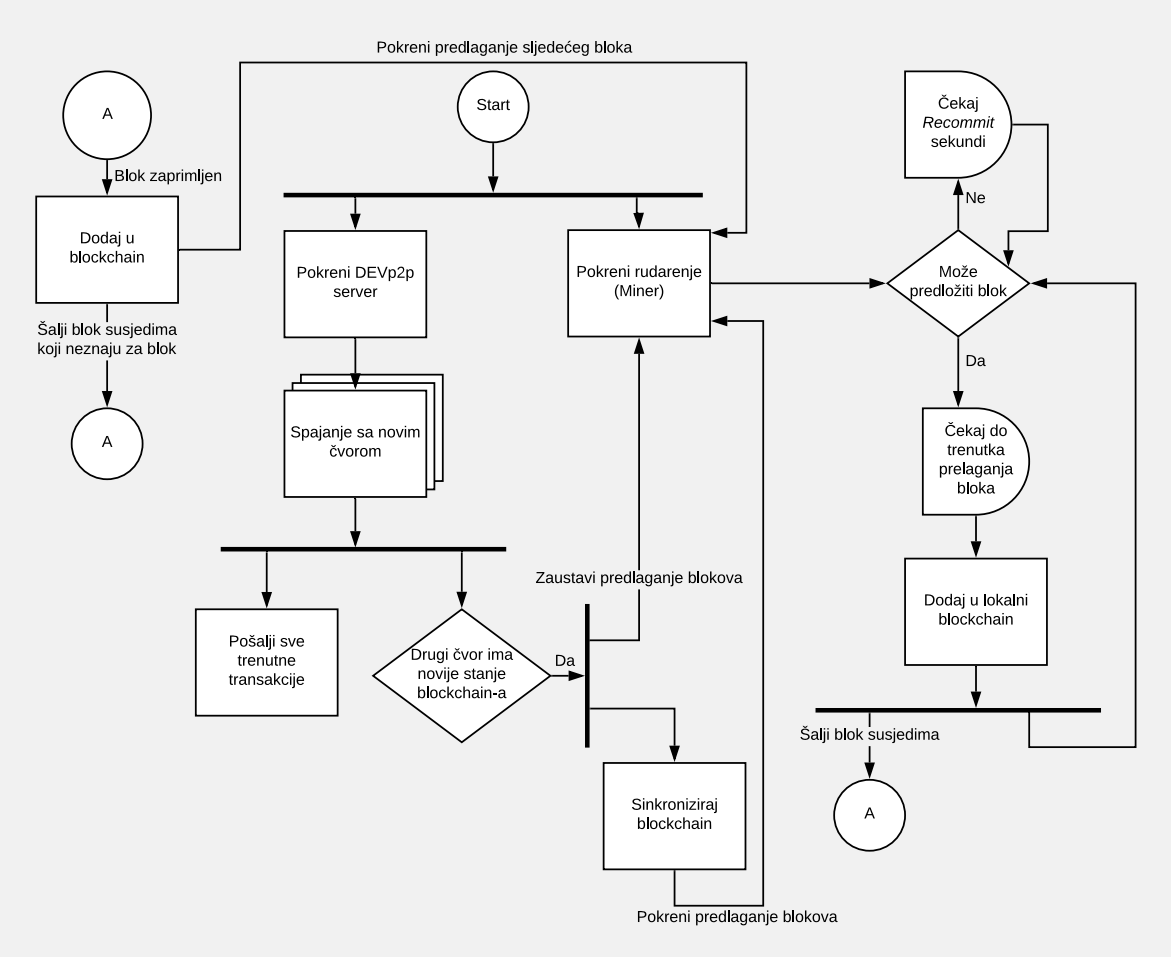
listu rezultata, koja se koristi pri uspostavljanju novih veza. Samo jedan zadatak otkrivanja može biti pokrenut u određenom vremenu. Ukoliko nema čvorova na kojih se može spojiti, a postoji mogućnost spajanja na nove čvorove, pokreće se zadatak čekanja, nakon kojeg se ponovno ispituje mogućnost uspostavljanja novih veza. Kao vrijeme čekanja uzima se 30 sekundi od zadnjeg zadatka uspostavljanja veze.



Slika 4.3: Princip rada DEVp2p sloja

Slika 4.4 prikazuje način rada blockchain sloja. Prilikom pokretanja paralelno se pokreću DEVp2p server i *Miner* (predlagatelj blokova). Prilikom povezivanja čvorova na DEVp2p razini, pokreće se blockchain protokol. Čvorovi tada razmjenjuju podatke o mreži na kojoj se nalaze, genesis bloku i hash-u zadnjeg bloka. Nakon uspješnog razmjenjivanja poruka, čvorovi međusobno razmjenjuju trenutno tekuće transakcije. Paralelno se provodi sinkronizacija blockchain-a, ukoliko čvor zaključi da drugi čvor ima novije stanje blockchain-a. Prilikom

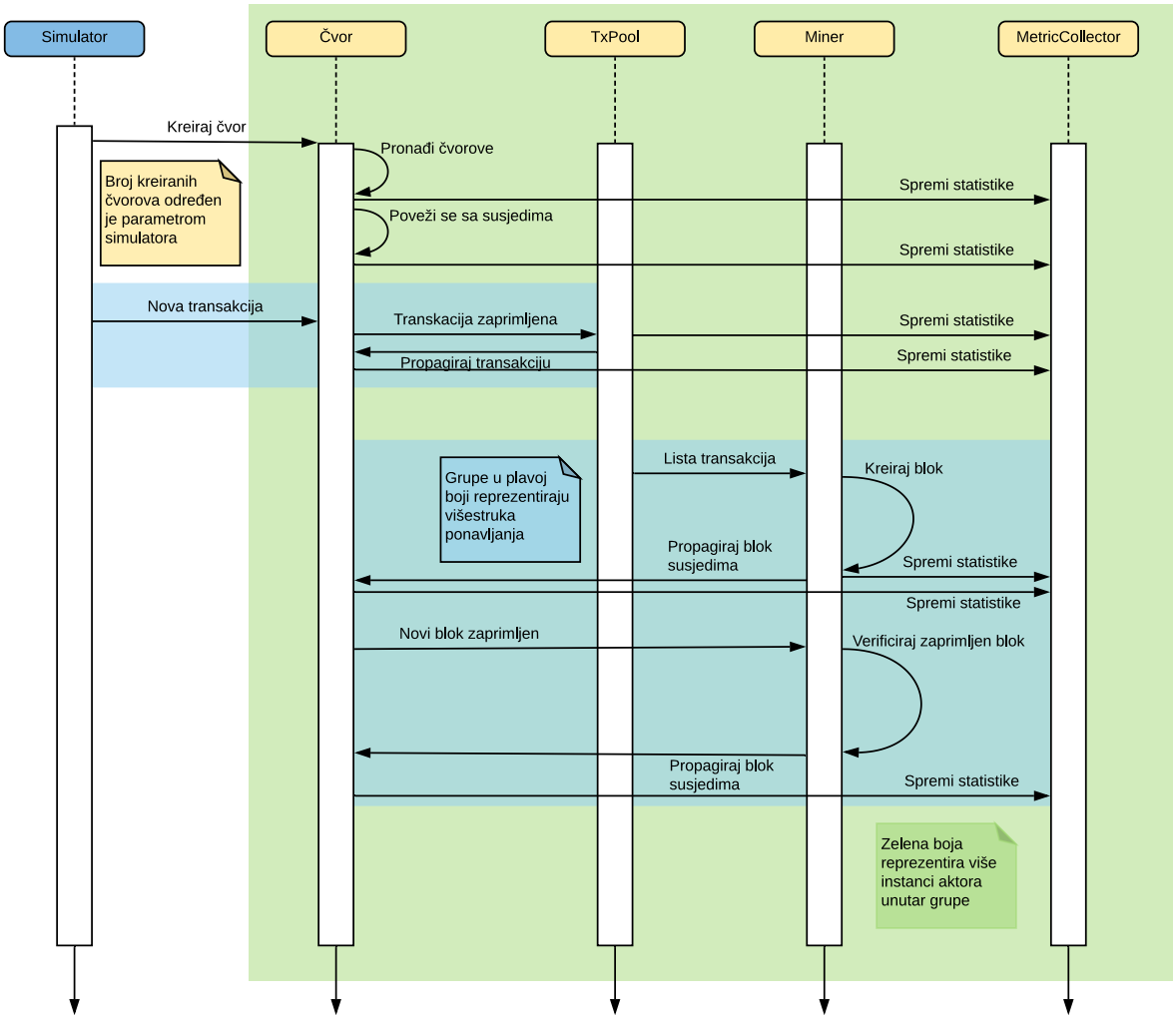
sinkronizacije, predlaganje blokova se zaustavlja, jer bi svi eventualno predloženi blokovi bili odbačeni od strane blockchain mreže. Nakon što je sinkronizacija završena, pokreće se modul predlaganja blokova. Prilikom uspješnog kreiranja bloka, čvor taj blok dodaje u lokalni blockchain, te ga zatim šalje svojim susjednim čvorovima. Svaki od susjednih čvorova prilikom zaprimanja bloka, dodaje isti u svoj blockchain, te ga propagira svojim susjedima koji ne znaju za blok.



Slika 4.4: Princip rada blockchain sloja

Kako bi se mogao simulirati rad blockchain-a, simulator ima mogućnost generiranja transakcija. Na slici 4.5 prikazan je tijek simulacije s generiranjem transakcija. Simulator kreira broj čvorova prema postavljenom parametru. Nakon što su se svi čvorovi pronašli kroz RLPx sloj i povezali kroz DEVp2p sloj, simulator započinje generiranje transakcija po postavljenoj distribuciji. Generirana transakcija dodaje se nasumičnom čvoru. Čvor tada provjerava valjanost transakcije, te ukoliko je transakcija valjana, dodaje istu u listu trenutnih transakcija, ili u listu budućih transakcija. Transakcija se tada propagira susjednim čvorovima, na sličan način kao i kod propagacije blokova. Prilikom kreiranja bloka, uzimaju se transakcije iz liste trenutnih transakcija, te se uključuju u blok. Svaka transakcija se obrađuje, pa se

tako za svaku obrađenu transakciju povećava *GasPool* za *IntrinsicGas* (21000), sve dok se ne dostigne *GasLimit* bloka.



Slika 4.5: Rad simulatora s generiranjem transakcija

Za podešavanje ponašanja simulatora koriste se globalni parametri prikazani u tablici 4.1. Svaki od slojeva može se ima dodatne parametre koji su konstantni ili promjenjivi.

RLPx sloj ne dopušta nikakvo parametriziranje, odnosno sve vrijednosti su konstante koje se ne mijenjanju. Nadalje, neki od važnijih parametara specifičnih za RLPx sloj prikazani su tablicom 4.2.

DEVp2p uz neke ne parametrizirajuće parametre dopušta i nekoliko parametrizirajućih parametara. Ne parametrizirani/konstantni parametri dani su tablicom 4.3, dok su parametrizirajući parametri dani tablicom 4.4.

Parametri blockchain sloja prikazani su tablicom 4.5 za konstantne parametre, te tablicom 4.6 za promjenjive parametre.

Parametar	Značenje
SimulationTime	Vrijeme trajanja simulacije
NodeCount	Broj čvorova mreže
NodeStabilisationTime	Vrijeme stabilizacije mreže
ChurnEnabled	Mogućnost čvorova za napuštanje i vraćanje mreži
NodeArrivalDistr	Distribucija pridruživanja čvorova mreži
NodeSessionTimeDistr	Distribucija trajanja sesija čvorova (ChurnEnabled: true)
NodeIntersessionTimeDistr	Distribucija trajanja nedostupnosti čvora (ChurnEnabled: true)
NodeLifetimeDistr	Distribucija trajanja cjelokupnog života čvora
NetworkLatency	Distribucija latencija mreže
SimMode	Simulacija RLPx-a, DEVp2p-a ili Blockchain-a
GroupFactor	Faktor vremenskog grupiranja podataka
Metrics	Imena metrika koje se želi prikupit
ExportType	Tip prikazivanja metrika (PNG, CSV)
CollectType	Funkcija (Sum, Avg) koja se primjenjuje na sve metrike
MetricCollectType	Funkcija (Sum, Avg) koja se primjenjuje na specifične metrike

Tablica 4.1: Globalni parametri simulatora

Parametar	Vrijednost	Značenje
alpha	3	Faktor konkuretnosti slanja *FIND_NODE* poruka
bucketSize	16	Veličina retka tablice usmjeravanja
maxReplacements	10	Maksimalan broj zamjenjivih čvorova *k*-retka prilikom otkrivanja novih čvorova
maxFindnodeFailures	5	Maksimalan broj neodgovorenih *FIND_NODE* poruka, nakon kojih se čvor briše iz tablice
refreshInterval	30 minuta	Interval osvježavanja tablice usmjeravanja (slanje *FIND_NODE* poruka)
revalidateInterval	10 minuta	Gornja granica nasumičnog vremenskog perioda u kojem se provjerava dostupnost zadnje viđenog čvora u nasumičnom k-retku tablice usmjeravanja

Tablica 4.2: Konstantni parametri RLPx sloja

Parametar	Vrijednost	Značenje
pingInterval	15 sekundi	Interval slanja *DEVp2p PING* poruke svakom od spojenih čvorova
maxActiveDialTasks	16	Maksimalan broj istovremenih zadataka za spajanje i pronalaženje DEVp2p čvorova

Tablica 4.3: Konstantni parametri DEVp2p sloja

Parametar	Značenje
MaxPeers	Maksimalan broj spojenih čvorova
DialRatio	Odnos ulaznih/izlaznih pokušaja ostvarivanja veze između čvorova mreže. Za vrijednost 0, uzima se zadana vrijednost: 3

Tablica 4.4: Promjenjivi parametri DEVp2p sloja

Parametar	Vrijednost	Značenje
evictionInterval	1 minuta	Vremenski interval izbacivanja budućih transakcija (imaju veći *nonce* nego zadnja obrađena + 1), od adresa koje nisu u proteklo *Lifetime* vrijeme poslali nikakvu transakciju
minBroadcastPeers	4	Minimalan broj čvorova kojim se propagira blok, ukoliko je korijen broja povezanih čvorova manji od *minBroadcastPeers*
minDesiredPeerCount	5	Minimalan broj povezanih čvorova potrebnih za sinkronizaciju nakon povezivanja s novim čvorom
forceSyncCycle:	10 sekundi	Interval sinkronizacije s ostalim čvorovima mreže, čak i ako je broj čvorova manji od *minDesiredPeerCount*

Tablica 4.5: Konstantni parametri blockchain sloja

Parametar	Značenje
NetworkId	Broj koji reprezentira mrežu na kojoj se čvor nalazi
Engine	Konsenzus algoritam koji se koristi (Clique, Aura)
GasFloor	Minimalni *Gas* bloka
GasCeil	Maksimalni *Gas* bloka
GasPrice	Cijena svakog potrošenog *Gas*-a
Recommit	Minimalno vrijeme između pokušaja kreiranja bloka
PriceLimit	Minimalna cijena *Gas*-a za zaprimanje transakcije
PriceBump	Minimalno postotno povećanje cijene *Gas*-a kako bi se zamijenila postojeća transakcija s istim *nonce*-om
AccountSlots	Maksimalan broj transakcija za račun
GlobalSlots	Maksimalan broj transakcija
AccountQueue	Maksimalan broj transakcija na čekanju za račun
GlobalQueue	Maksimalan broj transakcija na čekanju
Lifetime	Maksimalno vrijeme zadržavanja neobrađenih transakcija
Period	Vremenski period generiranja blokova

Tablica 4.6: Promjenjivi parametri blockchain sloja

4.2. Simulacije i rezultati

4.2.1. RLPx i DEVp2p

Kako bi se pokazale karakteristike RLPx i DEVp2p sloja provedena je simulacija s parametrima prikazanih tablicom 4.7. Nadalje, kako bi se spriječilo previsoke devijacije, prikupljeni podaci grupirani su po grupama od 30 sekundi.

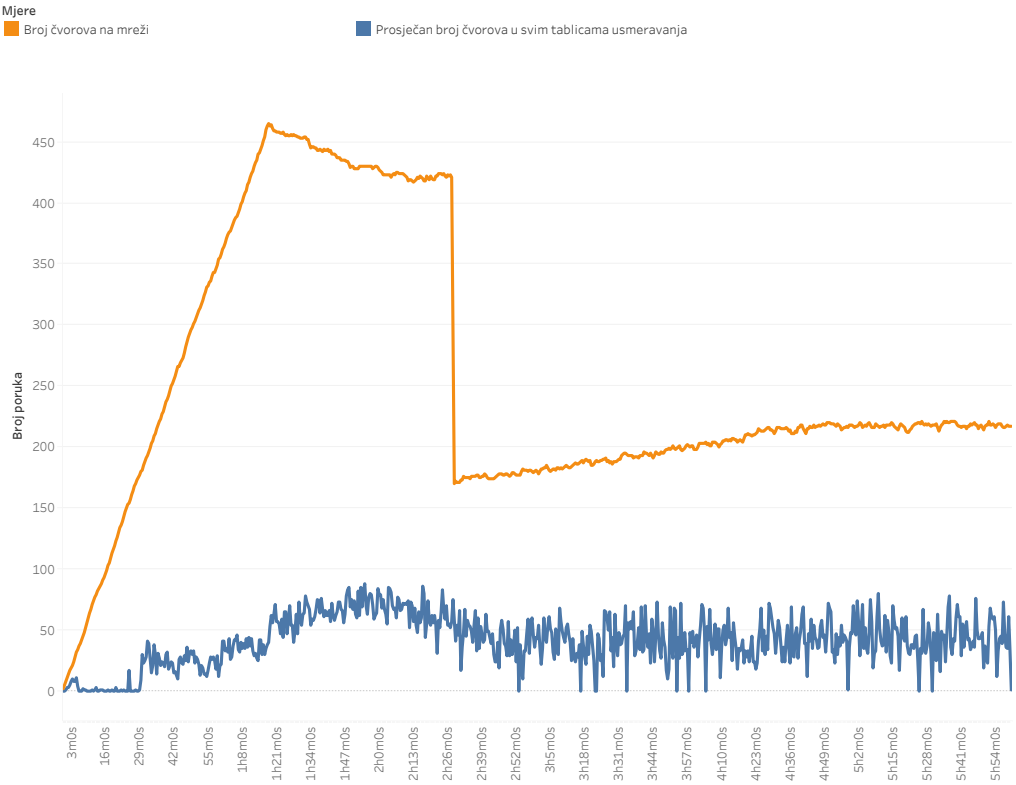
Cilj je bio dakle prikazati karakteristike tih dvaju slojeva, te je stoga kreiran scenarij gdje se polovica nasumično odabranih aktivnih čvorova izbaci iz mreže, sat vremena nakon stabilizacije iste. Na slici 4.6 prikazan je broj aktivnih čvorova mreže zajedno sa prosječnim brojem čvorova pohranjenih u tablicama usmjeravanja svih čvorova.

Bolju sliku opisuju poruke koje razmjenjuju čvorovi. Na slici 4.7 prikazana je suma svih poruka poslanih sa RLPx sloja. Na slici možemo vidjeti kako je broj *PONG* poruka u trenutku izbacivanja polovice čvorova iz mreže, znatno manji, što je i za očekivati. Nadalje, vrijeme koje je bilo potrebno da se broj *PONG* poruka stabilizira, odnosno da tablica usmjeravanja stare čvorove zamijeni s aktivnim je 18 minuta.

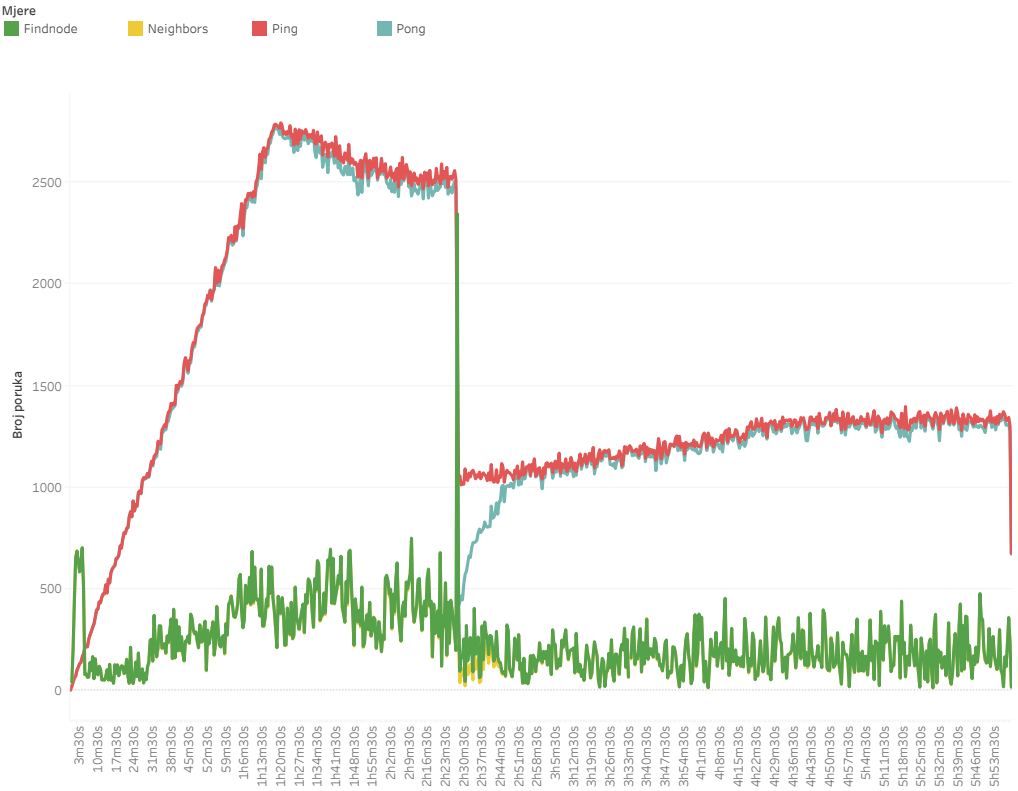
Na slici 4.8 odmah primjećujemo kako prosječan broj povezanih čvorova nije 25 kao što je specificirano u simulaciji. Razlog tome je *DialRatio*, odnos dolaznih/odlaznih zahtjeva uspostavljana veze koji nije specificiran u simulaciji te je zaprimio zadanu vrijednost 3. Na-

Parametar	Vrijednost
SimulationTime	6 sati
NodeCount	500
NodeStabilisationTime	10 minuta
ChurnEnabled	true
NodeArrivalDistr	Normal(9.6, 4.8)
NodeSessionTimeDistr	Expon(1 sat)
NodeIntersessionTimeDistr	Expon(1 minuta)
NetworkLatency	Lognormal(0.209, 0.157)
MaxPeers	25
SimMode	DEVp2p

Tablica 4.7: Parametri RLPx i DEVp2p simulacije



Slika 4.6: Prosječan broj čvorova RLPx tablice usmjeravanja u odnosu na broj čvorova mreže



Slika 4.7: Broj RLPx poruka svih čvorova mreže

Parametar	Vrijednost
ActorCount	Broj različitih adresa s kojih se šalju transakcije
TransactionIntervalDistr	Distribucija dolaska transakcija
TxPriceDistr	Distribucija cijena transakcija
Duration	Vrijeme generiranja transakcija

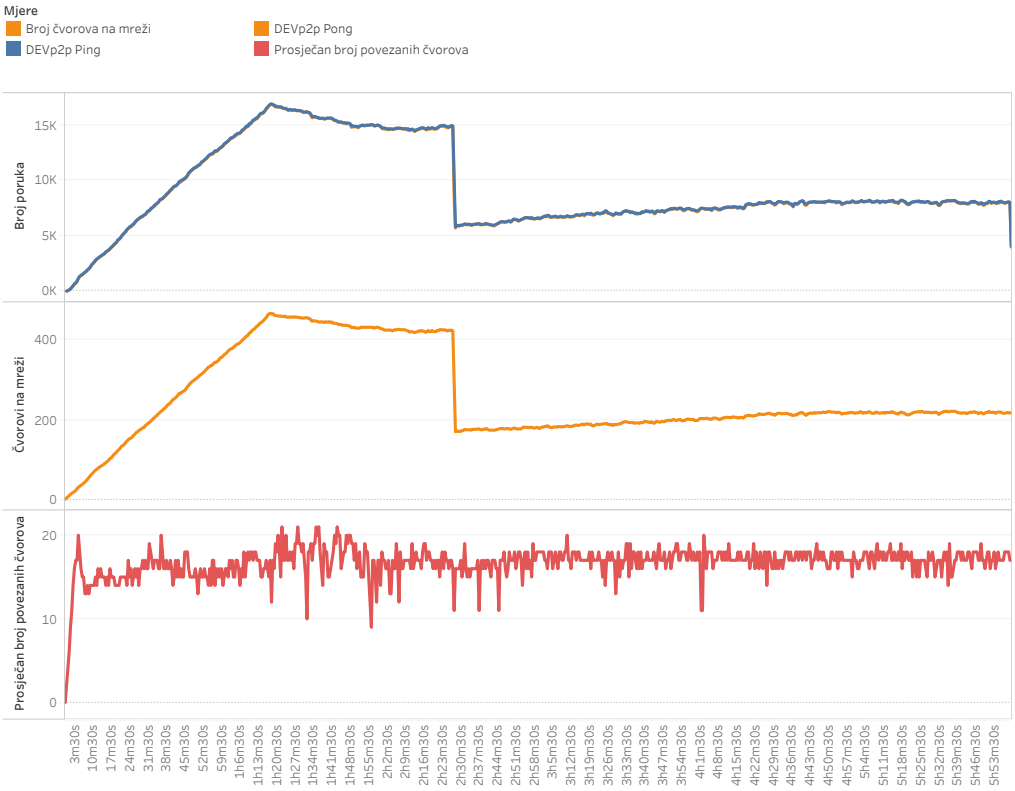
Tablica 4.8: Parametri modeliranja dolaska transakcija

dalje, prosječan broj povezanih čvorova nije pao ispod 10, što sugerira brzi oporavak (unutar 30 sekundi zbog grupacije podataka) povezanih čvorova.

4.2.2. Blockchain

Simulator ne simulira izvršavanje pametnih ugovora, jer su to u suštini transakcije koje se izvršavaju na čvoru. Za modeliranje dolaska transakcija, simulator pruža parametre prikazane tablicom 4.8. Za početnu usporedbu konsenzus algoritama, postavljeni parametri prikazani su tablicom 4.9.

Kako bi se bolje prikazale karakteristike konsenzus algoritama, prikupljeni podaci su grupirani svakih 15 sekundi, točnije, usklađeno je s vremenom između kreiranja dva bloka.

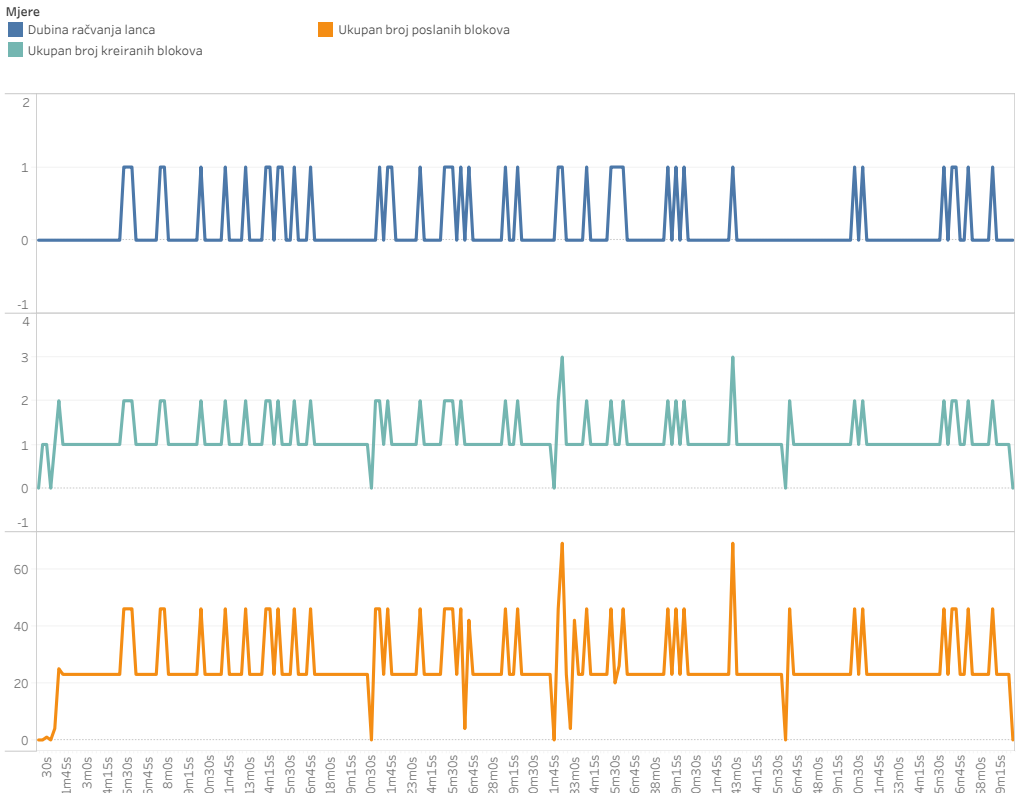


Slika 4.8: Broj DEVp2p poruka svih čvorova mreže s prosječnim brojem povezanih čvorova

Parametar	Vrijednost
SimulationTime	1 sat
NodeCount	6
ChurnEnabled	false
NetworkLatency	Lognormal(0.209, 0.157)
Engine	Clique i Aura
Period	15 sekundi

Tablica 4.9: Parametri simulacije konsenzus algoritama

Na slikama 4.9 i 4.10 prikazani su podaci o dubini račvanja lanca, ukupnom broju kreiranih blokova te broju blokova poslanih po mreži za Clique i Aura konsenzus algoritam. Na slici 4.9 vidi se kako Clique algoritam kreira višak blokova, čime povećava ukupan broj blokova poslanih mrežom kao i povećani broj račvanja lanca. Kod Aura algoritma situacija je čista zbog toga što svaki validator zna kada treba kreirati blok, te ne kreira novi blok u nikakvom drugom slučaju. Zbog toga se kod Aure ne pojavljuju račvanja lanca, niti se kreira višak blokova.

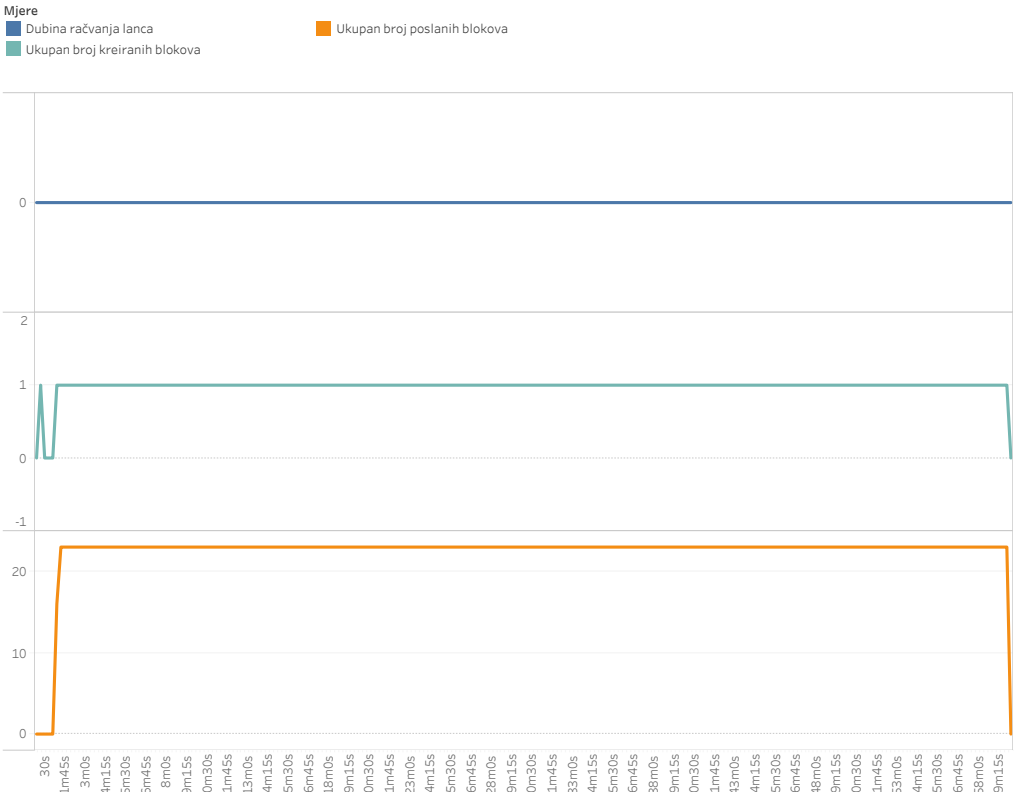


Slika 4.9: Kreiranje blokova s Clique konsenzus algoritmom

Ipak, u stvarnom svijetu čvorovi nisu cijelo vrijeme dostupni zbog višestrukih razloga. Čvor može izaći iz mreže, te se u istu vratiti u manjim ili većim vremenskim intervalima. Za prikaz karakteristika konsenzus algoritama u takvom okruženju provedena je simulacija s istim parametrima kao i kod prethodne simulacije uz promjene parametara prikazane tablicom 4.10.

Na slikama 4.11 i 4.12 prikazani su rezultati tih simulacija. Iz rezultata simulacije vidi se kako kod Clique algoritma nema značajnih promjena, manje promjene vidljive su tek u ukupnom broj propagiranih blokova mrežom. Nadalje, prema očekivanom, Aura algoritam nije kreirao nove blokove ukoliko predlagatelj bloka nije bio aktivan. U vremenu od sat vremena, Aura je kreirala 7 blokova manje nego Clique.

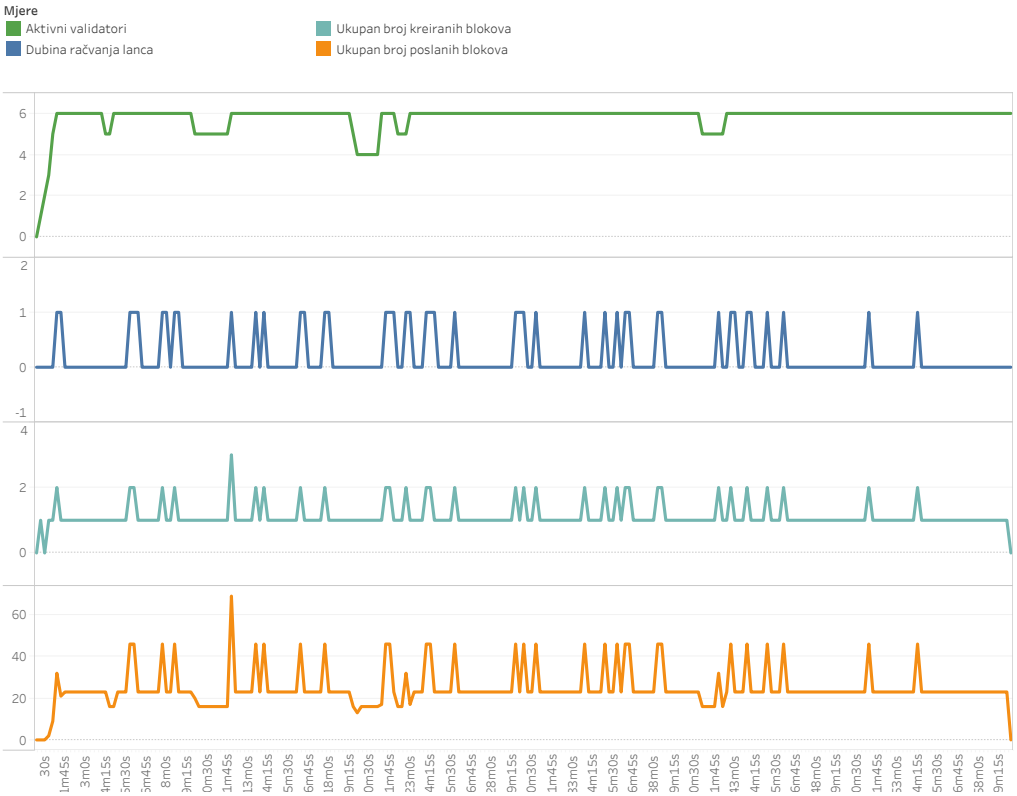
Nedostupnost čvorova zasigurno utječe na procesiranje transakcija. Utjecaj nedostup-



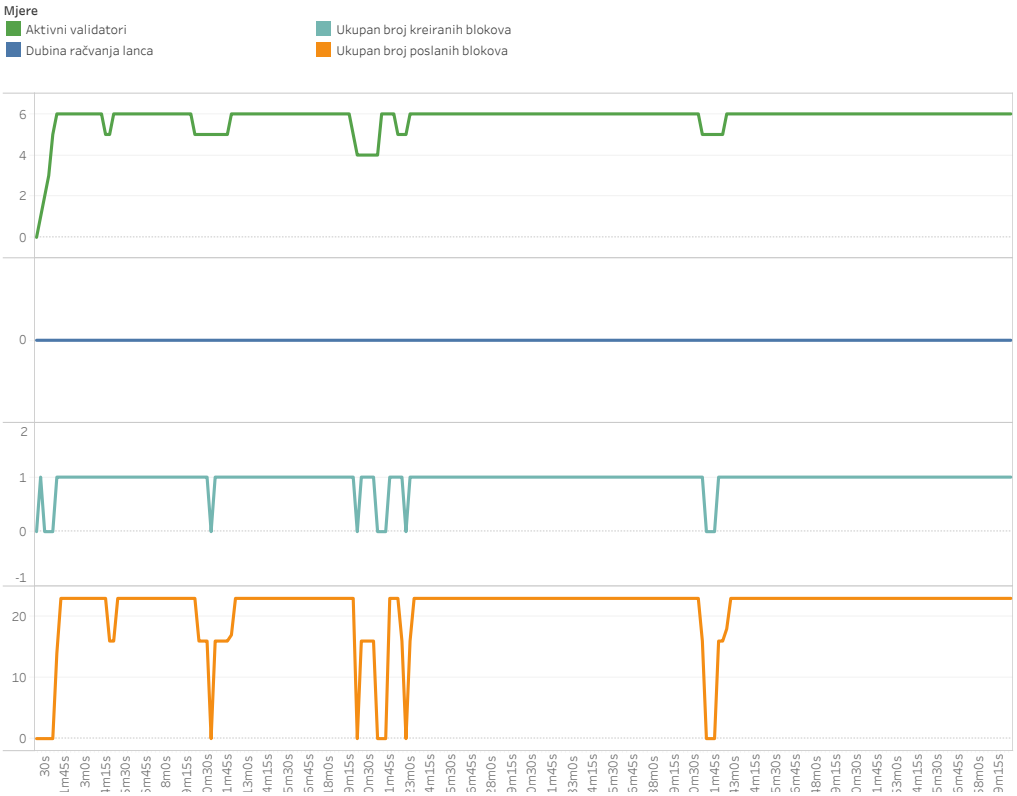
Slika 4.10: Kreiranje blokova s Aura konsenzus algoritmom

Parametar	Vrijednost
ChurnEnabled	true
NodeStabilisationTime	1 minuta
NodeSessionTimeDistr	Expon(1 sat)
NodeIntersessionTimeDistr	Expon(1 minuta)

Tablica 4.10: Parametri simulacije konsenzus algoritama u okruženju gdje čvor može napustiti mrežu, te se na istu vratiti u nakon određenog vremena



Slika 4.11: Ponašanje Clique algoritma u okruženju s nedostupnim čvorovima



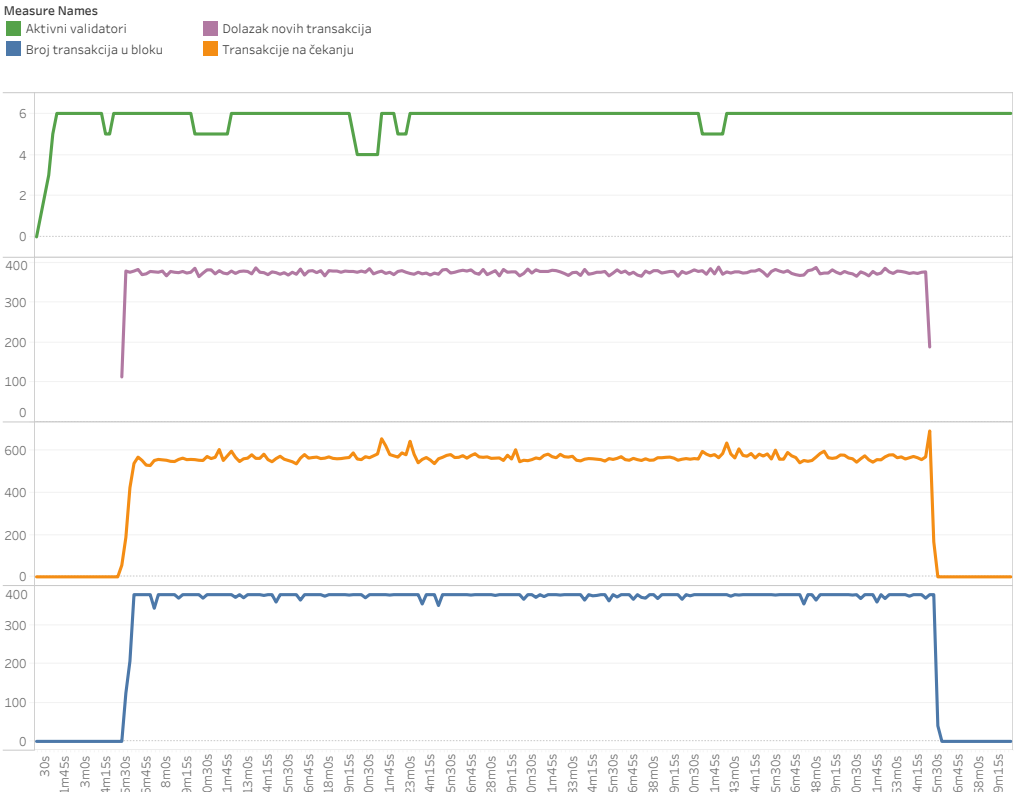
Slika 4.12: Ponašanje Aura algoritma u okruženju s nedostupnim čvorovima

Parametar	Vrijednost
ActorCount	10000
TransactionIntervalDistr	Normal(0.04, 0.01)
TxPriceDistr	Normal(3, 1)
Duration	55 minuta
PriceLimit	1
PriceBump	10
AccountSlots	16
GlobalSlots	4096
AccountQueue	64
GlobalQueue	1024
Lifetime	3 sata
GasFloor	8000000
GasCeil	8000000
GasPrice	1
Recommit	3 sekunde

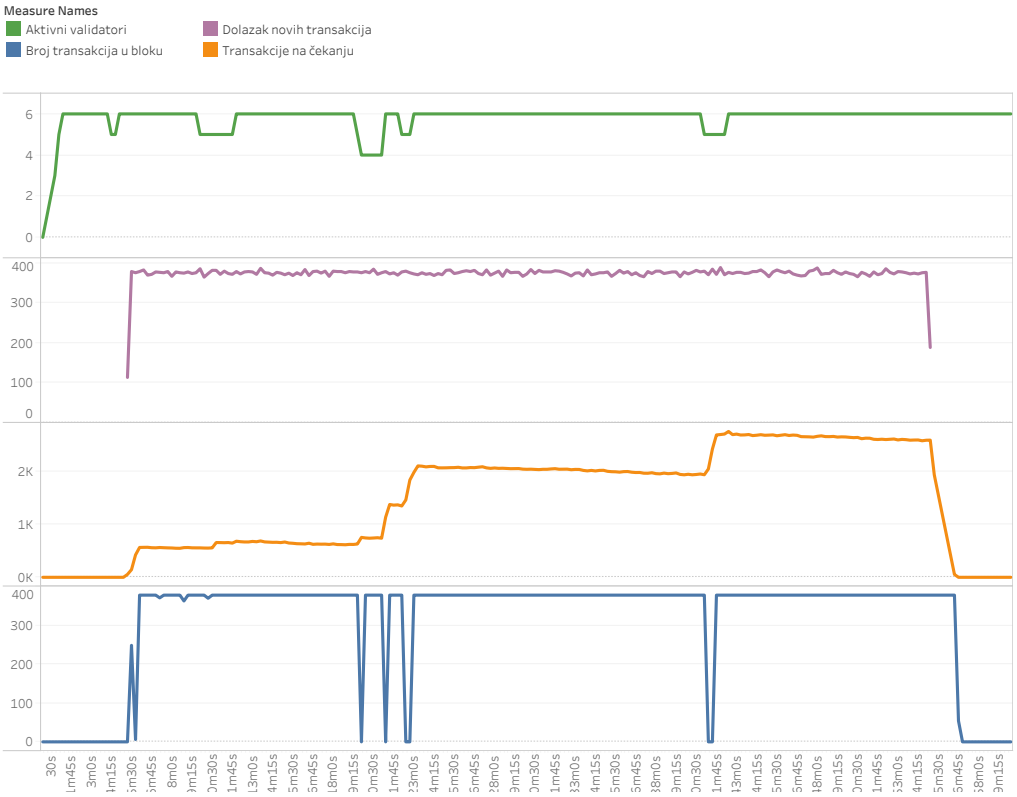
Tablica 4.11: Parametri generiranja transakcija

nosti čvorova na procesiranje transakcija prikazano je koristeći identične parametre kao i u prethodnoj simulaciji, uz dodatak parametara prikazanih tablicom 4.11.

Uzevši u obzir parametar *GasCeil* i cijene obrade jedne transakcije 21000, maksimalan broj transakcija po bloku iznosi 380. Distribucija generiranja transakcija modelirana je kako bi broj generiranih transakcija, za svakih 15 sekundi simulacije, u prosjeku iznosio 380. Na slikama 4.13 i 4.14 prikazani su rezultati simulacije. Clique algoritam se dobro nosi s opterećenjem, te je broj transakcija na čekanju stabilan, kao i broj transakcija uključenih u blok. Kod Aura algoritma, zbog nedostupnosti predlagatelja bloka, blokovi se ne kreiraju. Posljedica preskakanja kreiranja blokova je povećani broj transakcija na čekanju.



Slika 4.13: Konstantno opterećenje na Clique u okruženju s nedostupnim čvorovima



Slika 4.14: Konstantno opterećenje na Aura u okruženju s nedostupnim čvorovima

5. Prethodna istraživanja

5.1. Simulacije otkrivanja čvorova mreže (peer-to-peer discovery)

Jednostavnost, brzina, skalabilnost i troškovna učinkovitost simulacije kao evaluacijske tehnike rezultirala je velikim brojem simulatora. Isto vrijedi i za peer-to-peer (veza ravnopravnih računala) simulatore. Peer-to-peer simulatori kao što su ProtoPeer [22], PeerfactSim.KOM [35] i OverSim [12], simuliraju različite peer-to-peer mrežne protokole i ispituju karakteristike tih protokola [37].

ProtoPeer [22], je simulator diskretnih događaja, baziran na Javi, s već predefiniranim protokolima OverStat, SG-1 i T-Man, te implementiranim protokolima Pastry, Chord i BitTorrent.

PeerfactSim.KOM [35], je skalabilni simulator diskretnih događaja baziran na Javi. Podržava velik broj protokola: GIA, Gnutella, Napster, CAN, Chord, C-DHT, Kademlia, Pastry, Globase.

OverSim [12], je simulator diskretnih događaja baziran na OMNet++[1](napisan u C++), s podržanim peer-to-peer protokolima: Chord, Kademlia, Pastry, Koorde i Broose.

Spektar protokola koje ti i drugi simulatori implementiraju je velik. Unatoč tome, trenutno niti jedan simulator ne podržava RLPx peer-to-peer protokol.

5.2. Simulacije blockchain-a

U proteklih nekoliko godina pojavilo se nekoliko implementacija simulatora blockchain-a kako bi se pokušalo simulirati ponašanje pravih blockchain mreža.

BlockSim [9], je dinamički simulator diskretnih događaja koji apstrakcijama olakšava dizajnerima i analitičarima istraživanje kompromisa dizajna i konfiguracijskih pitanja blockchain-a. Autori kao cilj simulatora navode općenitost, jednostavnost i proširivost. Kako bi pokazali primjenjivost simulatora, autori su simulirali rad Bitcoin i Ethereum blockchain. Rezultati

[1]https://omnetpp.org

su pokazali kako simulator točno reprezentira realne sustave, uz manju grešku za propusnost Ethereum mreže za koju autori navode kako je najvjerojatniji uzrok mali uzorak transakcija.

VIBES [36], je konfiguracijski blockchain simulator koji može povoditi mrežne simulacije velikih razmjera u svrhu dobivanja empirijskih uvida u ponašanje određenog sustava. Autor navodi kako simulator postiže dva cilja: skalabilnost i brzinu.

eVIBES [19, 20], inspiriran VIBES simulatorom, je konfiguracijski simulator upravljan događajima za simuliranje velikih Ethereum mreža. Dizajniran s dva primarna cilja: skalabilnost i podesivost konfiguracije koju korisnik može podešavati i tijekom simulacije.

SimBlock [10], simulator upravljan događajima, koji kao događaje smatra generiranje bloka i prijenos/primanje poruka. Kako bi evaluirali simulator, autori su proveli simulaciju s istim uvjetima kao i u simulaciji [23], te potvrdili kako simulator dobro reprezentira realne blockchain sustave.

U [30] autori predlažu simulator double-spend napada kao simulacijski baziran pristup proučavanja arhitektonskih dizajnerskih odluka na određene atribute kvalitete blockchain-a.

BLOCKBENCH [21] je evaluacijski okvir za analizu privatnih blockchain-a. Omogućuje dublje razumijevanje različitih izbora sustava i platformi, koje se može integrirati u BLOCKBENCH putem jednostavnih API-a (Application Programming Interface) te usporediti s radnim opterećenjima različitih slojeva privatne blockchain mreže. Također mjeri performanse u smislu propusnosti, latencije, skalabilnosti i tolerancije na pogreške.

Iako sve implementacije navedenih simulatora daju približno realne vrijednosti mjerenih varijabli uz dobru skalabilnost, ograničeni su s tek nekoliko parametara koji su najčešće: stopa dolaska transakcija, broj čvorova, distribucija mrežne latencije, vrijeme generiranja bloka te broj susjednih čvorova.

6. Zaključak

U ovom radu prikazan je PoASim blockchain simulator. Simulator implementira sva tri sloja Ethereum blockchain-a kako bi dizajnerima mreža što bolje objasnio ponašanje blockchain mreže u raznim uvjetima s raznim konfiguracijskim parametrima. Simulator implementira konsenzus algoritme vrste Proof of Authority, Clique i Aura, koje nudi kao mogući izbor prilikom simulacije. Simulacijom opterećenja na blockchain mrežu s Clique konsenzusom, pokazalo se kako se ta mreža dobro nosi opterećenjem i u slučaju kada su predlagatelji trenutnog bloka nedostupni, dok se kod Aura konsenzusa blokovi ne kreiraju ako predlagatelj nije aktivan, pa se stoga vrijeme za obradu transakcije povećava prilikom velikog opterećenja. U stabilnom okruženju gdje čvorovi garantiraju visoku razinu neprekidnog rada, s usklađenim satovima, Aura konsenzus garantira konačnost konsenzusa, dok kod Clique konsenzusa može doći do račvanja lanca, koja se tada rješavaju GHOST protokolom.

U stvarnom svijetu satovi čvorova nisu sinkronizirani pa zbog toga može doći do neispravnog rada konsenzus algoritama. Stoga bi sljedeći poželjan korak u nadogradnji simulatora bio dodavanje sata na svaki čvor, odnosno odstupanje od stvarnog sata, koji će se koristiti kao trenutno vrijeme pri bilo kakvoj operaciji vezanoj za čvor. Simulator je dostupan na https://github.com/rosaj/poasim.

LITERATURA

[1] Aura. Dostupno na https://wiki.parity.io/Aura. Pristupljeno: 17.06.2019.

[2] Clique. Dostupno na https://github.com/ethereum/EIPs/issues/225. Pristupljeno: 17.06.2019.

[3] The Go Programming Language. Dostupno na https://golang.org/. Pristupljeno: 20.06.2019.

[4] Parity Ethereum. Dostupno na https://github.com/paritytech/parity-ethereum. Pristupljeno: 17.06.2019.

[5] Ethereum Wire Protocol (ETH), 2016. Dostupno na https://github.com/ethereum/devp2p/blob/master/caps/eth.md. Pristupljeno: 15.06.2019.

[6] What is ethereum, 2018. Dostupno na https://github.com/ethereum/wiki/wiki/What-is-Ethereum. Pristupljeno: 16.06.2019.

[7] Go Ethereum, 2019. Dostupno na https://github.com/ethereum/go-ethereum/. Pristupljeno: 15.06.2019.

[8] Parity Technologies, 2019. Dostupno na https://www.parity.io/. Pristupljeno: 17.06.2019.

[9] Alharby, Maher i van Moorsel, Aad. BlockSim. *ACM SIGMETRICS Performance Evaluation Review*, 46(3):135–138, 2019. ISSN 01635999.

[10] Aoki, Yusuke, Otsuki, Kai, Kaneko, Takeshi, Banno, Ryohei, i Shudo, Kazuyuki. SimBlock: A Blockchain Network Simulator. 2019.

[11] Banks, J., Carson, J. S., Nelson, B. L., i Nicol, D. *Discrete-Event System Simulation*. Prentice Hall, 5 izdanju, 2010. ISBN 0136062121.

[12] Baumgart, Ingmar, Heep, Bernhard, i Krause, Stephan. OverSim: A flexible overlay network simulation framework. *2007 IEEE Global Internet Symposium, GI*, (June 2007):79–84, 2007.

[13] Bertoni, Guido, Daemen, Joan, Peeters, Michael, i Van Assche, Gilles. The Keccak SHA-3 submission. stranice 1–14, 2011.

[14] Brewer, Eric. Cap twelve years later: How the "rules" have changed. *Computer*, 45: 23–29, 02 2012.

[15] Buchman, Ethan. Understanding the ethereum trie, 2014. Dostupno na `https://easythereentropy.wordpress.com/2014/06/04/understanding-the-ethereum-trie/`. Pristupljeno: 24.06.2019.

[16] Buterin, Vitalik. Merkling in Ethereum, 2015. Dostupno na `https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/`. Pristupljeno: 24.06.2019.

[17] De Angelis, Stefano. Assessing Security and Performances of Consensus algorithms for Permissioned Blockchains. 2018.

[18] De Angelis, Stefano, Aniello, Leonardo, Baldoni, Roberto, Lombardi, Federico, Margheri, Andrea, i Sassone, Vladimiro. PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain. *CEUR Workshop Proceedings*, 2058:1–11, 2018. ISSN 16130073.

[19] Deshpande, Aditya, Nasirifard, Pezhman, i Jacobsen, Hans-Arno. Configurable and Interactive Ethereum Blockchain Simulation Framework. *Proceedings of the 19th International Middleware Conference on - Middleware '18*, stranice 11–12, 2018.

[20] Deshpande, Aditya Shrikant. Design and Implementation of an Ethereum-like Blockchain Simulation Framework. 2018.

[21] Dinh, Tien Tuan Anh, Wang, Ji, Chen, Gang, Liu, Rui, Ooi, Beng Chin, i Tan, Kian-Lee. BLOCKBENCH: A Framework for Analyzing Private Blockchains. 2017.

[22] Galuba, W. ProtoPeer: Bridging the Gap Between Simulation and Live Deployment. *Lsirpeople.Epfl.Ch*, 2009.

[23] Gervais, Arthur, Karame, Ghassan O, Wüst, Karl, i Ritzdorf, Hubert. On the Security and Performance of Proof of Work Blockchains. *Bitcoin.org*, 2017.

[24] Goussiatiner, Alex. Godes, 2015. Dostupno na `https://github.com/agoussia/godes`. Pristupljeno: 20.06.2019.

[25] KIM, SEOUNG K. MEASURING ETHEREUM'S PEER-TO-PEER NETWORK. 2017.

[26] Kostal, Kristian, Krupa, Tomáš, Gembec, Martin, Veres, Igor, Ries, Michal, i Kotuliak, Ivan. On transition between pow and pos. 09 2018.

[27] Kyun Kim, Seoung, Ma, Zane, Murali, Siddharth, Mason, Joshua, Miller, Andrew, i Bailey, Michael. Measuring Ethereum Network Peers. stranica 14, 2018.

[28] Lamport, Leslie, Shostak, Robert, i Pease, Marshall. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, Srpanj 1982. ISSN 0164-0925.

[29] Leverington., Alex. The RLPx Transport Protocol, 2017. Dostupno na `https://github.com/ethereum/devp2p/blob/master/rlpx.md`. Pristupljeno: 15.06.2019.

[30] Marmsoler, Diego i Eichhorn, Leo. Simulation-Based Analysis of Blockchain Architectures. *Unpublished*, 2018.

[31] Maymounkov, Petar i Mazieres, David. Kademlia: A Peer-to-peer Information System Based on the XOR metric. 2002.

[32] Nakamoto, Satoshi. Bitcoin: A peer-to-peer electronic cash system, 2008. Dostupno na `http://www.bitcoin.org/bitcoin.pdf`.

[33] Saltini, Roberto. Correctness Analysis of IBFT. stranice 1–31, 2019.

[34] Steinmetz, Ralf i Wehrle, Klaus. *Peer-to-Peer Systems and Applications (Lecture Notes in Computer Science)*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 354029192X.

[35] Stingl, Dominik, Groß, Christian, Rückert, Julius, Nobach, Leonhard, Kovacevic, Aleksandra, Peerfactsim, Steinmetz, Simulation, K O M A, In, Peer-to-peer Systems, Stingl, Dominik, Gross, Christian, Julius, R, Nobach, Leonhard, Kovacevic, Aleksandra, i Steinmetz, Ralf. PeerfactSim . KOM : A Simulation Framework for Peer-to-Peer Systems. (July), 2011.

[36] Stoykov, Lyubomir, Zhang, Kaiwen, i Jacobsen, Hans-Arno. VIBES: fast blockchain simulations for large-scale peer-to-peer networks. (January 2018):19–20, 2017.

[37] Surati, Shivangi, Jinwala, Devesh C., i Garg, Sanjay. A survey of simulators for P2P overlay networks with a case study of the P2P tree overlay using an event-driven simulator. *Engineering Science and Technology, an International Journal*, 20(2):705–720, 2017. ISSN 22150986.

[38] Wood, Garvin. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER, 2017.

Usporedba radnih karakteristika konsenzus algoritama računalnom simulacijom

Sažetak

Blockchain je u proteklih nekoliko godina dobio široku pozornost zbog njegove primjene na kripto valutama i tehnologijama distribuiranih knjiga (Distributed Ledger), kao što su Bitcoin i Ethereum. Složena i decentralizirana priroda blockchain tehnologija otežava razumijevanje ponašanja pojedinih komponenata i njihovog učinka na blockchain sustav. Dolaskom novih konsenzus algoritama, kao što je Proof of Authority (PoA), razumijevanje ovog složenog sustava postaje izazovan zadatak. U ovom radu predlažemo PoASim, podesiv, diskretni simulacijski alat za simulaciju dvaju glavnih PoA algoritama, nazvanih Clique i Aura. PoASim može pomoći korisnicima da bolje razumiju temeljne slojeve Ethereum blockchain-a, kao i razumijevanje razlika konsenzusa i nedostataka između dva konsenzus algoritma. PoASim tako može poslužiti kao koristan alat za bolje razumijevanje ponašanja svakog konsenzusa pokretanjem simulacija s različitim parametrima i analizom ponašanja sustava.

Ključne riječi: Blockchain, Ethereum, Proof of Authority, simulacija, peer to peer

Comparison of performance characteristics of consensus algorithms by computer simulation

Abstract

Blockchain have received widespread attention over the past few years because of its application on cryptocurrencies and Distributed Ledger technologies, such as Bitcoin and Ethereum. Complex and decentralized nature of blockchain technologies makes it difficult to understand behaviors of individual components and their effect on the blockchain system. With the arrival of new consensus algorithms, such as Proof-of-Authority (PoA), understanding this complex system becomes a challenging task. In this thesis, we propose PoASim, configurable, discrete-event simulation tool for simulating two of the main PoA algorithms, named Clique and Aura. PoASim can help users better understand the underlying layers of the Ethereum blockchain, as well as understanding consensus differences and drawbacks between the two consensus algorithms. PoASim can thus serve as a useful tool for better understanding behavior of each consensus by running simulations with varying parameters and then analyzing the system behavior.

Keywords: Blockchain, Ethereum, Proof of Authority, simulation, peer to peer