SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2546

Ulančani blokovi i raspodijeljene glavne knjige s ograničenim pravom pristupa i pametnim ugovorima

Martin Pavić

Zagreb, lipanj 2021.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

SADRŽAJ

POPIS SLIKA

1. Uvod

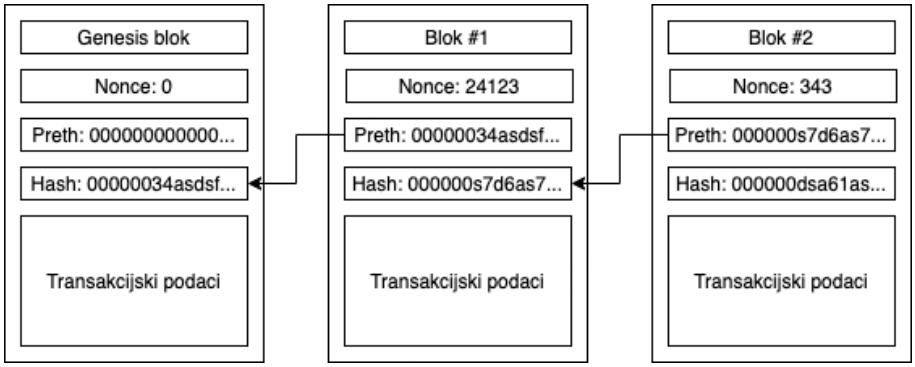
Baze podataka su posvuda. Koristimo ih svaki dan. Na primjer, popis kontakata u mobitelu je jednostavna baza podataka - elektronička verzija papirnatog adresara. Detaljnije baze podataka uključuju popise kupaca, zaposlenika, pacijenata ili glasača i njihove svojstva i odnose. Složenije baze podataka mogu sadržavati čak i programe koji mogu međusobno komunicirati. Zapravo, o bazi podataka možemo razmišljati kao o bilo kojem organiziranom skupu informacija koje možete pro- naći i eventualno ažurirati stavke. Otkako je računalna revolucija započela pedesetih godina prošlog stoljeća, baze podataka igrale su važnu ulogu u poslu i društvu.

Mnoge su baze podataka danas dijeljene. Do sada je svijet toliko povezan da različiti ljudi često trebaju pristupiti istim podacima. Da bi se udovoljilo ovoj potrebi, pojavile su se distribuirane baze podataka, u kojima određenim dijelovima podataka može pristupiti više osoba odjednom. Primjerice, radi pojednostavljenja postavljanja ili ponovnog zakazivanja sastanaka, svi članovi radne skupine mogu dijeliti svoje sastanke na internetskom kalendaru. To se ne može učiniti s papirnatim kalendarom. Naravno, u poslu se koriste složenije zajedničke baze podataka. Neka od pitanja koja možemo postaviti pri radu s bazama podataka.

- Vjerujete li onome s kime dijelite podatke?
- Kako možete znati da je netko onaj za koga kaže da je?
- Što smiju raditi s bazom podataka?
- Tko rješava sukobe ili sporove?

Jasno je da postoji mnogo praktičnih problema s dijeljenjem baze podataka. Tijekom vremena ljudi su isprobali mnogo različitih rješenja. Jedan uzbudljiv novi način dijeljenja baza podataka koji može pomoći u rješavanju tih problema je tehnologija ulančanih blokova (eng. Blockchain).

Blockchain je tehnologija koja stoji iza Bitcoina. Mediji su prepuni priča o Bitcoinu i drugim kriptovalutama. Većina tvrtki ne brinu previše o kriptovalutama. Većina tvrtki je sretna što kupuje i prodaje s dolarima, eurima, funtama, jenima ili bilo ko-



Slika 1.1: *Blockchain*

jom drugom prihvaćenom valutom - možda čak i kriptovalutom - sve dok to djeluje za njih. Za poduzeća je daleko značajnija tehnologija koja stoji iza kriptovaluta, tzv *blockchain*.

Blockchain je novi oblik zajedničke baze podataka. Blockchain je distribuirana baza podataka bez središnjeg tijela i bez točke povjerenja. Kada želite dijeliti bazu podataka, ali nemate puno povjerenja u druge ljude koji bi mogli koristiti tu bazu podataka, blockchain vam može biti od velike pomoći. U tom kontekstu, "povjerenje" može značiti mnogo stvari. Povjerenje bi moglo značiti vjerovanje drugima da će s bazom podataka postupati ispravno. Povjerenje bi moglo značiti da jedna strana drugoj ne pokušava doći do privatnih informacija. Ili povjerenje može značiti ne ponižavanje tuđeg učinka radi stjecanja konkurentske prednosti.

Rasprava o povjerenju otvara dvije glavne vrste blockchaina. Većina kriptovaluta koristi blockchain bez dozvola gdje se svatko može pridružiti i imati puno prava na njegovo korištenje. Na primjer, svatko može kupiti Bitcoin ili Ether jer oni koriste širom otvorene, blockchaine bez ograničenja pristupa. S druge strane, poslovni blockchaini imaju tendenciju da im se ne može pristupiti bez dozvole. To znači da osoba treba ispuniti određene zahtjeve za izvođenje određenih radnji na blockchainu. Neki blockchaini ograničavaju pristup čak i prethodno provjerenim korisnicima koji su već dokazali da su oni za koje kažu da jesu, što se može dogoditi kada postoje podatci na blockchainu kojima mogu pristupiti samo određeni sudionici. Drugi dopuštaju bilo kome da se pridruži, ali dopuštaju samo provjernim identitetima da verificiraju transakcije na blockchainu.

U bazi podataka koju dijele dvije različite stranke postavlja se pitanje, što se događa ako obe stranke žele pristupiti podatcima u isto vrijeme? Ako bi se tom bazom podataka upravljalo s blockchainom, ovaj bi se problem mogao riješiti kroz proces koji se naziva konsenzus. Blockchaini koriste konsenzusne sustave kako bi bili sigurni da

su podaci u bazi podataka uvijek ispravni. Primjerice, konsenzusni sustav koristio bi unaprijed utvrđena pravila za određivanje koja stranka prva dobiva pristup podatcima. Konsenzusni sustavi imaju mnogo različitih oblika s različitim imenima. Na primjer, Bitcoin koristi dokaz izvršenog rada (eng. proof of work, skr. PoW) konsenzus, gdje računala sudionika rješavaju teške matematičke zadatke. Najbitnije svojstvo koje takvi sustavi moraju zadovoljavati naziva se *Byzantine Fault Tolerance* ili skraćeno BFT.

Zašto je blockchain važan poslovnim ljudima? Pomoću blockchaina mogu se usmjeriti mnogi postojeći poslovni procesi u mnogim industrijama kako bi uštedjeli vrijeme, uštedjeli novac i smanjili rizik. Mnogi potpuno novi procesi - možda čak i potpuno nove industrije mogu biti izumljeni. Prva generacija interneta bila je izvrsna za dijeljenje informacija: stvari poput e-pošte, dokumenata, fotografija, web stranica, pjesama i videozapisa, ali postojao je problem. Bilo je teško nekome dokazati da su oni koji su rekli da jesu. Svaka transakcija koja je uključčivala bilo koju vrijednost zahtijevala je posrednika, poput banke, da potvrdi kupca i prodavača i potvrdi transakciju. To je stvorilo trenje, kašnjenje i trošak - i središnja točka neuspjeha koju bi napadači mogli napasti. Blockchain otvara vrata drugoj generaciji interneta za koju je mnogo prikladnija razmjena vrijednosti, uključčujući vrijedne informacije. Pomoću blockchaina ljudi mogu utvrditi tko su i zatim mijenjati stavke poput novca, dionica i obveznica, intelektualno vlasništvo, djela, glasove, bodove lojalnosti i sve ostalo što ima vrijednost. Čak i ako se stranke ne znaju ili ne vjeruju jedni drugima, mogu vjerovati tehnologiji za bilježenje transakcija. A tehnologija uklanja potrebu za bilo kojim posrednikom, što štedi vrijeme i smanjuje troškove.

2. Blockchain s ograničenim pravom pristupa

Postoje razne arhitekture blockchaina. S ograničenjem pristupa i bez ograničanja pristupa, javni i privatni. Nama su, u ovom radu, najzanimljivija arhitektura blockchaina koja ima ograničen pristup. Takvi blockchaini zahtijevaju posebna dopuštenja za čitanje, pristup i upisivanje podataka na njih. Suštinska konfiguracija takvih blockchaina kontrolira transakcije sudionika i definira njihove uloge u kojima svaki sudionik može pristupiti i doprinijeti blockchainu. To također uključuje i održavanje identiteta svakog sudionika blockchaina na mreži. Takvi se blockchaini nazivaju blockchaini s ograničenim pravom pristupa (eng. permissioned blockchains, skr. PBs). PB-i također se razlikuju od privatnih blockchaina, koji omogućuju sudjelovanje samo poznatim čvorovima. Na primjer, banka možda pokreće privatni blockchain koji posluje kroz određeni broj čvorova unutar banke. Suprotno tome, PB-i mogu dopustiti bilo kome da se pridruži mreži nakon što se definiraju njihov identitet i uloga.

Prednosti. Brojne su prednosti blockchaina s ograničenim pravom pristupa, koje ga čine povoljnijim za upotrebu od javnih blockchaina. PB-e karakterizira učinkovitija izvedba. Kada bi uspoređivali PB-e i javne blockchaine, oni nude bolju izvedbu. Glavni i najočigledniji razlog tome je ograničen broj ččvorova koji sudjeluje u blockchainu, za razliku od javnih blockchaina gdje, u teoriji, može biti neograničeni broj ččvorova. Iz toga slijedi uklanjanje nepotrebnih proračuna potrebnih za postizanje konsenzusa, što svakako poboljšava izvedbu i performanse, a još i k tome imaju unaprijed određene čvorove koji validiraju transakcije. PB-i imaju pravilno organiziranu strukturu upravljanja, što rezultira tome da administratori trebaju manje vremena za ažuriranje pravila mreže, što je značajno brže u odnosu na javne blockchaine. Jedan od problema javnih blockchaina je taj što ne rade svi čvorovi zajedno kako bi implementirali ažuriranja mreže. Čvorovi mogu staviti svoje vlastite interese iznad interesa cijelog sustava, što rezultira usporavanjem ažuriranja mreže. Za usporedbu, PB-i nemaju taj problem jer čvorovi unutar takvog sustava rade zajedno kako bi brže ažurirali

mrežu, i zato što si međusobno vjeruju i surađuju, u interesu im je da je mreža što bolja i funkcionalnija. Jedna od vrlo važnih karakteristika koja pridonosi korisnosti PB-a je decentralizirana pohrana. PB-i potpuno iskorištavaju tehnologiju, tako što osim samih transakcija koje se događaju na mreži, također spremaju i druge važne podatke koje svi (ili neki) ččvorovi u mreži koriste. Ti su podatci spremljeni između čvorova organizacija koje su sudionici PB-a, a održavaju se jednakima za sve konsezusnim algoritmima. Sve to dovodi do toga da PB-i zahtijevaju manje troškova od drugih modela poslovanja te možemo zaključiti da su isplativi, pogotovo u usporedbi s javnim blockchainima, gledajući iz perspektive poslovnih rješenja.

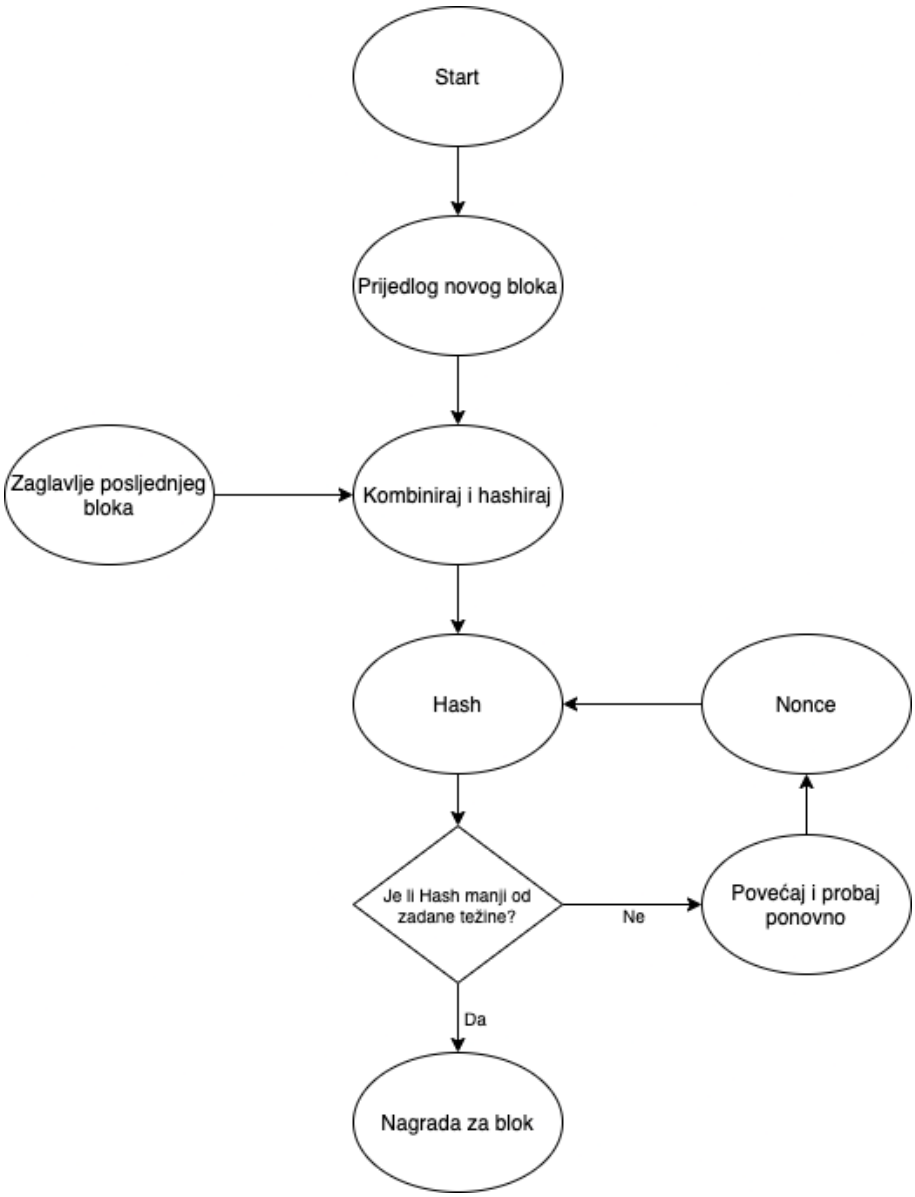
Nedostaci. Naravno, PB-i nisu oslobođeni nedostataka. Sigurnost PB-a dobra je koliko i integritet člana. Maliciozni dio PB-a može promjeniti i ugroziti podatake pohranjene u mreži. Na taj se način može ugroziti integritet mreže. Da bi ga riješio, sustav treba imati odgovarajuća dopuštenja, tako da se loši čvorovi ne mogu udružiti i kompromitirati mrežu. Kontrola, cenzura i regulacija - u idealnom svijetu PB-i bi trebali funkcionirati kao javni blockchain, ali s propisima. Međutim, propisi uvode cenzuru u mrežu, gdje tijelo može ograničiti transakciju ili kontrolirati da se ona dogodi. To je prijetnja bilo kojoj tvrtki ili organizaciji koja koristi PB. Ovaj pristup također onemogućuje PB da maksimalno iskoristi cijeli blockchain ekosustav.

2.1. Konsenzus

Svi čvorovi koji sudjeluju u mreži moraju se složiti oko svake poruke koja se prenosi između čvorova. Ako je grupa čvorova oštećena ili je poruka koju oni prenose oštećena, ona ipak ne bi trebala utjecati na mrežu kao cjelinu i trebala bi se oduprijeti ovom "napadu". Ukratko, mreža se u cijelosti mora složiti oko svake poruke prenesene u mreži. Ovaj se sporazum naziva konsenzusom. Konsenzus je temeljni problem distribuiranih sustava otpornih na kvarove. Konsenzus pokušava riješiti problem Bizantskih generala (eng. *Byzantine Generals Problem*). Analogija korištena za problem Bizantskih generala u osnovi ide ovako: Nekoliko divizija bizantske vojske smješteno je neposredno ispred neprijateljskog grada i priprema se za bitku. Razni generali mogu međusobno komunicirati samo putem glasnika. Moraju se dogovoriti oko zajednikog djelovanja. Međutim, moramo pretpostaviti da su neki generali izdajice koji žele spriječiti lojalne generale da se dogovore oko zajednikog postupka. Potreban je algoritam kako bi se osiguralo da mala skupina izdajica ne može poremetiti komunikaciju. Da bi riješili problem bizantskih generala, lojalni generali trebaju siguran način da se dogovore o planu - konsenzus - i izvrše svoj odabrani plan.

2.1.1. Dokaz izvršenog rada i dokaz uloga

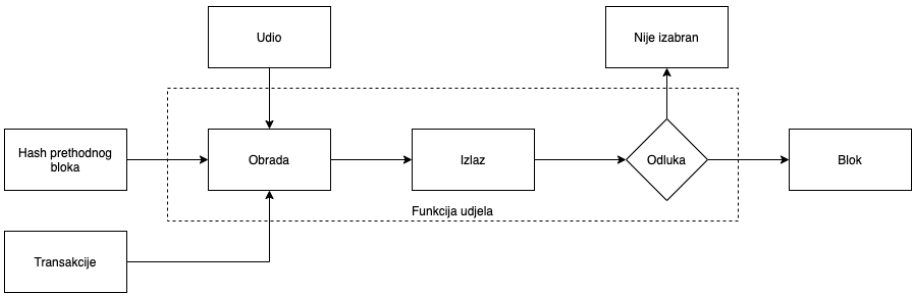
Dokaz izvršenog rada i dokaz uloga su dva najpoznatija konsenzusna algoritma koja se koriste unutar javnih blockchain sustava. Iako se u ovom radu nećemo baviti njima, važno ih je spomenuti jer su oni, posebno PoW, glavni čimbenici koji su pridonijeli razvoju današnjih blockchain sustava.



Slika 2.1: Dokaz izvršenog rada

Dokaz izvršenog rada radi tako da rudari pokušavaju generirati hash koji je manji od zadanog praga koji se zove težina rudarenja. Do toga hasha dolaze tako da mjenjaju ulaz u kriptografsku hash funkciju dodavajući jedan cijeli broj koji se zove *nonce*. Dokaz izvršenog rada funkcionira na način da je vjerojatnost rudarenja bloka propor-

cionalna količini računalnih resursa rudara. Isplate za blokove vremenom postaju sve manje, kako bi se smanjila mogućnost većinskog napada na mrežu. PoW sustavi vremenom postaju centralizirani, jer se rudari udružuju u takozvane "bazene rudara".



Slika 2.2: Dokaz uloga

Dokaz uloga funkcionira tako da rudar može rudariti blok ovisno o količini resursa (kriptovalute) koju posjeduje, dakle ne riješava nikakvu kriptografsku slagalicu što uklanja potrebu za velikim računalnih resursima. Takvi sustavi čine većinski napad jako skupim, stoga se napadačima i ne isplati pokušavati. Takvi su sustavi također i više decentralizirani, ali potrebno je puno truda uložiti kako bi se izgradile zajednice oko takvih sustava.

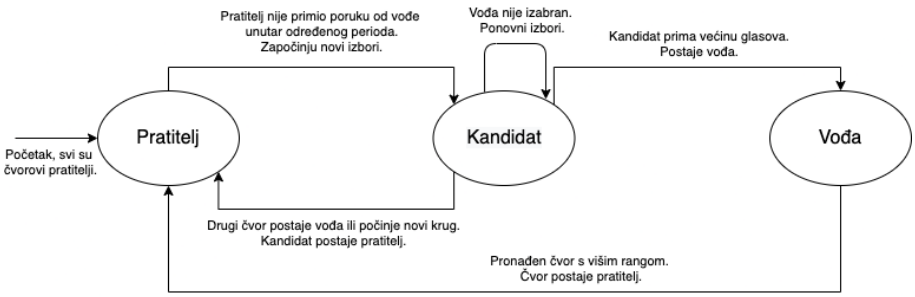
2.1.2. Paxos

Paxos je obitelj konsenzusnih protokola koji osiguravaju dosljednost replika u distribuiranom sustavu nepouzdanih procesora, odnosno sustavu u kojem poslužitelji mogu zakazati. Protokol Paxos uveo je 1989. Leslie Lamport, nazvan po izmišljenom zakonodavnom sustavu konsenzusa koji se koristio na otoku Paxos u Grčkoj. Cilj Paxos algoritma je održavanje istog redoslijeda naredbi među više replika, tako da sve replike na kraju konvergiraju u istu vrijednost. To je slično slučaju kada više automobila slijedeći iste upute stiže na isto konačno odredište.

2.1.3. Raft

Raft je konsenzusni algoritam koji je dizajniran tako da ga je lako razumjeti. Jednak je Paxosu u toleranciji kvarova i performansama. Razlika je u tome što je razložen na relativno neovisne podprobleme i bavi se svim glavnim dijelovima potrebnim za praktične sustave na uredniji način. Kako bi se poboljšala razumljivost, Raft razdvaja ključne elemente konsenzusa, kao što su izbor vođe, replikacija zapisnika i sigurnost te provodi snažniji stupanj konherentnosti kako bi se smanjio broj stanja koja se moraju

uzeti u obzir. Raft također uključuje novi mehanizam za promjenu članstva klastera, koji koristi preklapajuće većine kako bi se zajamčila sigurnost. Ongaro i Ousterhout (2014)



Slika 2.3: Raft konsenzus

Osnove

- Za ispravan rad mreže potreban je kvorum (većina) čvorova.
- čvorovi mogu biti u tri stanja - vođa, kandidat za vođu ili sljedbenik.
- Vođa upravlja ažuriranjem stanja mreže. Vođa se određuje glasovanjem.
- Kada prođe vremensko ograničenje izbora (sljedbenik ne primi nikakvu komunikaciju određeno vrijeme) započinje novi izbor vođe.
- Vođa periodički šalje heartbeat poruke, kako bi zadržao autoritet nad sljedbenicima.

Razlike u odnosu na Paxos

- Snažan vođa - Raft koristi jači oblik vodstva od ostalih konsenzusnih algoritama. Na primjer, zapisnik se šalje samo od voditelja do drugih čvorova. Ovo pojednostavljuje upravljanje repliciranim zapisnikom i čini Raft lakšim za razumijevanje.
- Izbor vođe - Raft koristi nasumične tajmere za biranje vođe. To dodaje samo malu količinu dodatnog mehanizma već otprije potrebnim mehanizmima za bilo koji algoritam konsenzusa, dok sukobe rješava jednostavno i brzo.
- Promjene članstva - Raftov mehanizam za promjene skupa čvorova u klasteru koristi novi zajedniki konsenzusni pristup gdje se većina dvije različite konfiguracije preklapa tijekom prijelaza. To omogućuje klasteru da nastavi raditi normalno tijekom promjena konfiguracije.

2.2. Pametni ugovori

Pojam pametni ugovor korišten je tijekom godina za opisivanje širokog spektra različitih stvari. Devedesetih je kriptograf Nick Szabo skovao taj pojam i definirao ga kao "skup obećanja, specificiranih u digitalnom obliku, uključujući protokole unutar kojih stranke izvršavaju ostala obećanja". Od tada se koncept pametnih ugovora razvijao, posebno nakon uvođenja decentraliziranih blockchain platformi s izumom Bitcoina 2009. Taj pojam najčešće je povezan s Ethereum blockchainom, no kontekstu Ethereuma, taj je izraz zapravo pomalo pogrešan naziv, s obzirom na to da pametni ugovori Ethereum nisu ni jedno ni drugo. Nisu pametni niti pravni ugovori, ali termin je zapeo. Što su onda zapravo pametni ugovori? Ethereum definira pametne ugovore u sljedećih nekoliko stavki:

- **Računalni program** - Pametni ugovori su jednostavno računalni programi. Riječ "ugovor" u ovom kontekstu nema pravno značenje.
- **Nepromjenjiv** - Jednom postavljen, kôd pametnog ugovora ne može se mijenjati. Za razliku od tradicionalnog softvera, jedini način izmjene pametnog ugovora je postavljanje nove instance.
- **Deterministiki** - Ishod izvršenja pametnog ugovora jednak je za sve koji ga izvode, s obzirom na kontekst transakcije koja je pokrenula njegovo izvršenje i stanje blockchaina u trenutku izvršenja.
- **Kontekst Ethereumovog virtualnog stroja** - Pametni ugovori rade s vrlo ograničenim kontekstom izvršenja. Mogu pristupiti vlastitom stanju, kontekstu transakcije koja ih je pozvala i nekim informacijama o najnovijim blokovima.
- **Decentralizirano svjetsko računalo** - EVM radi kao lokalna instanca na svakom Ethereum čvoru, ali budući da sve instance EVM-a rade u istom početnom stanju i proizvode isto konačno stanje, sustav u cjelini djeluje kao jedinstveno "svjetsko računalo".

Pametni ugovori obično su napisani u programskom jeziku visoke razine, kao što je Solidity. Ali da bi se izvodili, moraju se prevesti u bajtni kod niske razine koji se izvodi u Ethereum virtualnom stroju. Jednom sastavljeni, oni se raspoređuju na Ethereum platformi pomoću posebne transakcije stvaranja ugovora, koja se identificira kao takva slanjem na adresu za stvaranje posebnog ugovora 0x0. Svaki ugovor identificiran je Ethereum adresom koja je izvedena iz transakcije stvaranja ugovora kao funkcija izvornog računa i nonce-a. Ethereum adresa ugovora može se koristiti u transakciji kao primatelj, šaljući sredstva na ugovor ili pozivajući jednu od funkcija ugovora. Za raz-

liku od vanjskih računa, ne postoje ključevi povezani s računom stvorenim za novi pametni ugovor. Stvaratelj ugovora, ne dobiva nikakve posebne povlastice na razini protokola (iako ih se može izričito kodirati u pametni ugovor). Važno je da se ugovori izvršavaju samo ako ih transakcija pozove. Svi pametni ugovori u Ethereumu izvršavaju se u konačnici zbog transakcije pokrenute od vanjskog računa. Ugovor može pozvati drugi ugovor koji može pozvati treći ugovor, i tako dalje, ali prvi ugovor u takvom lancu izvršenja uvijek će biti pozvan transakcijom od vanjskog računa. Ugovori se nikad ne izvršavaju sami ili u pozadini. Ugovori učinkovito miruju sve dok transakcija ne pokrene izvršenje, bilo izravno ili neizravno kao dio lanca ugovora. Transakcije su atomske, ili se uspješno izvršavaju ili poništavaju.

1. ako se transakcija pošalje iz vanjskog računa na drugi vanjski račun, tada se bilježe sve promjene u globalnom stanju (npr. stanja računa) izvršene transakcijom

2. ako se transakcija pošalje iz vanjskog računa na ugovor koji se ne poziva na bilo koji drugi ugovor, tada se bilježe sve promjene globalnog stanja (npr. stanja na raćunima, varijable stanja ugovora)

3. ako se pošalje transakcija iz vanjskog računa do ugovora koji samo poziva druge ugovore na način koji propagira pogreške, tada se bilježe sve promjene globalnog stanja (npr. stanja na računima, varijable stanja ugovora)

4. ako se transakcija pošalje iz vanjskog računa na ugovor koji poziva druge ugovore na način koji ne propagira pogreške, tada mogu biti zabilježene samo neke promjene u globalnom stanju (npr. stanja na računu, varijable stanja ugovora bez pogrešaka), dok se druge promjene globalnog stanja ne bilježe (npr. varijable stanja ugovora s pogreškama). Inače, ako se transakcija poništi, svi se njezini učinci (promjene stanja) vraćaju unatrag, kao da se transakcija nikada nije pokrenula. Neuspjela transakcija i dalje se evidentira kao pokušaj, a Ether potrošen na gas za izvršenje oduzima se s izvornog računa, ali inače nema drugih učinaka na ugovor ili stanje računa.

Kao što je prethodno spomenuto, važno je zapamtiti da se kod ugovora ne može mijenjati. Međutim, ugovor se može "izbrisati", uklanjajući kod i njegovo unutarnje stanje (pohranu) s adrese, ostavljajući prazan račun. Sve transakcije poslane na tu adresu računa nakon brisanja ugovora ne rezultiraju izvršenjem koda, jer tamo više nema koda za izvršavanje. Brisanje ugovora ne uklanja povijest transakcija ugovora, jer je sam blockchain nepromjenjiv. Također je važno napomenuti da će mogućnost

biti dostupna samo ako je autor ugovora programirao pametni ugovor da ima tu funkcionalnost. Pametni ugovori s ograničenim pristupom postavljeni na blockchainu s ograničenim pristupom, postaju sve popularniji u poslovnom svijetu. U usporedbi s neučinkovitim i skupim procesima provjere valjanosti javnih blockchaina, blockchaini s ograničenim pristupom prikladniji su za poticanje poslovne suradnje. Javni pametni ugovori nameću neizbježne opasnosti za privatnost korisnika. Stoga se u poslovnom svijetu više koriste pametni ugovori s ograničenim pristupom, a to uključuje banke, lanac opskrbe, glasanje, IoT, osiguranje, itd.

2.2.1. Tokeni

U Blockchain ekosustavu bilo koja imovina koja je digitalno prenosiva između dvije stranke naziva se "token". Tokeni na blockchainu (kriptografski tokeni) predstavljaju skup pravila, kodiranih u pametnom ugovoru - ugovoru tokena. Svaki token pripada blockchain adresi. Ti su tokeni dostupni preko virtualnog novčanika, onog koji komunicira s blockchainom i upravlja parom javno-privatnih ključeva povezanim s adresom blockchaina. Samo osoba koja ima privatni ključ za tu adresu može pristupiti odgovarajućim tokenima. Token može predstavljati na primjer imovinu, pravo pristupa ili pak glas na izborima, a vlasnik ga može prenijeti (odobriti) tako da to potpiše svojim privatnim ključem, stvarajući digitalni potpis. Valjanost i sigurnost kriptografskih tokena kontrolira pametni ugovor koji ga je stvorio zajedno s osnovnom distribuiranom glavnom knjigom i većinskim konsenzusom. Ethereum je razvio standardizirani pametni ugovor, standard ERC-20, koji definira zajedniki popis pravila za Ethereum tokene, uključujući način na koji se tokeni prenose s jedne Ethereum adrese na drugu i kako se pristupa podacima unutar svakog tokena. Ovi relativno jednostavni pametni ugovori upravljaju logikom i održavaju popis svih izdanih tokena i mogu predstavljati bilo koju imovinu koja ima značajke zamjenjive robe. Besplatni i otvoreni ERC-721 standard opisuje kako napraviti takozvane nezamjenjive tokene na Ethereum blockchainu. Ovo je uvelo doba složenijih obilježja tokena. Ovaj je standard olakšao stvaranje tokena koji predstavlja bilo koju vrstu kolekcionarskih predmeta, umjetnina, vlasništva ili personaliziranih prava pristupa. Ovi nezamjenjivi tokeni imaju posebna svojstva koja token čine jedinstvenim i povezani su s identitetom određene osobe.

Tokene trenutno nije moguće prebacivati između različitih mreža, jer se izdaju pametnim ugovorima specifinim za blockchain koji njima upravljaju. Ti različiti blockchaini imaju različite standarde i nisu interoperabilni. Rješavanje tih problema moglo bi dovesti do masovnog usvajanja tokena širom tehnološkog svijeta.

ERC-20

Standard ERC-20 (Ethereum zahtjev za komentare 20, eng. *Ethereum Request for Comments* 20), predložen od Fabiana Vogelstellera u studenom 2015. godine, je standard tokena koji predstavlja sučelje po kojem se programiraju pametni ugovori. Pruža funkcionalnosti poput prijenosa tokena s jednog računa na drugi, dohvaćanja trenutnog stanja tokena na računu, kao i ukupne zalihe tokena dostupnih na mreži. Osim njih, ima i neke druge funkcije poput odobrenja da iznos tokena s jednog računa može potrošiti račun treće strane. Ako pametni ugovor implementira sljedeće metode i događaje, može se nazvati ERC-20 token ugovorom, a nakon primjene bit će odgovoran za praćenje stvorenih tokena na Ethereumu:

```
function name() public view returns (string)
function symbol() public view returns (string)
function decimals() public view returns (uint8)
function totalSupply() public view returns (uint256)
function balanceOf(address _owner) public view returns (uint256
    balance)
function transfer(address _to, uint256 _value) public returns (bool
    success)
function transferFrom(address _from, address _to, uint256 _value)
    public returns (bool success)
function approve(address _spender, uint256 _value) public returns (
    bool success)
function allowance(address _owner, address _spender) public view
    returns (uint256 remaining)
```

```
event Transfer(address indexed _from, address indexed _to, uint256
    _value)
event Approval(address indexed _owner, address indexed _spender,
    uint256 _value)
```

ERC-721

Standard ERC-721 (Ethereum zahtjev za komentare 721, eng. *Ethereum Request for Comments* 721), koji su u siječnju 2018. predložili William Entriken, Dieter Shirley, Jacob Evans, Nastassia Sachs, standard je nezamjenjivog tokena koji predstavlja sučelje po kojem se programiraju pametni ugovori. Pruža funkcionalnosti poput prijenosa tokena s jednog računa na drugi, dobivanjatrenutnog stanja tokena na računu, dobivanja vlasnika određenog tokena, kao i ukupne zalihe tokena dostupnih na mreži. Osim njih, on ima i neke druge funkcije poput odobrenja da iznos tokena s računa može

premjestiti račun treće strane. Ako pametni ugovor implementira sljedeće metode i događaje, može se nazvati ERC-721 ugovorom o nezamjenjivom tokenu i nakon što se jednom primijeni, bit će odgovoran za praćenje stvorenih tokena na Ethereumu.

```
function balanceOf(address _owner) external view returns (uint256);
function ownerOf(uint256 _tokenId) external view returns (address);
function safeTransferFrom(address _from, address _to, uint256
    _tokenId, bytes data) external payable;
function safeTransferFrom(address _from, address _to, uint256
    _tokenId) external payable;
function transferFrom(address _from, address _to, uint256 _tokenId)
    external payable;
function approve(address _approved, uint256 _tokenId) external
    payable;
function setApprovalForAll(address _operator, bool _approved)
    external;
function getApproved(uint256 _tokenId) external view returns (
    address);
function isApprovedForAll(address _owner, address _operator)
    external view returns (bool);
```

```
event Transfer(address indexed _from, address indexed _to,  uint256
    indexed _tokenId);
event Approval(address indexed _owner, address indexed _approved,
    uint256 indexed _tokenId);
event ApprovalForAll(address indexed _owner, address indexed
    _operator, bool _approved);
```

2.3. Corda, Quorum i Hyperledger Fabric

2.3.1. Corda

Corda je platforma za distribuirane glavne knjige koje se koriste za evidentiranje i obradu financijskih sporazuma. Platforma Corda podržava pametne ugovore koji odgovaraju definiciji **?**. Corda pametni ugovor je ugovor čije je izvršenje automatizirano pomoću računalnog koda koji radi s ljudskim unosom i kontrolom, i čija su prava i obveze pravno izvršni. Pametni ugovor povezuje poslovnu logiku i poslovne podatke s povezanim pravnim osobama kako bi se osiguralo da financijski sporazumi na platformi budu ukorijenjeni vrsto u zakonu i da se mogu provoditi te da postoji jasan put u slučaju dvosmislenosti, neizvjesnosti ili spora. Corda je specijalizirana za uporabu u reguliranim financijskim institucijama. Inspiriran je blockchain sustavima, ali bez

dizajna koji čini tradicionalne blockchaine neprikladnim za mnoge financijske scenarije. Corda pruža radni okvir za izvođenje pametnih ugovora sa sljedećim ključnim aktivnostima i značajke:

- Zapisivanje i upravljanje evolucijom financijskih sporazuma i drugih dijeljnih podataka između dvije stranke na način koji je utemeljen na postojećim pravnim konstrukcijama i kompatibilan sa postojećim i novim regulacijama.
- Koreografiranje tijeka rada između tvrtki bez središnjeg tijela.
- Konsenzus između tvrtki na razini pojedinačnih dogovora, a ne na globalnoj razini.
- Podrška uključivanju regulatornih i nadzornih čvorova promatraa.
- Validiranje transakcija isključivo između strana u transakciji.
- Podržavanje različitih mehanizama konsenzusa.
- Zapisivanje eksplicitnih veza između pravnih dokumenata i koda pametnog ugovora.
- Ograničavanje pristupa podacima u okviru sporazuma samo onima koji su izričito s pravom ili logički privilegirani na to.

Te značajke doprinose dizajnu platforme prikladne za upotrebu u složenim financijskim uslugama. Ovaj dizajn ne koristi izvorne kriptovalute i ne nameće globalno ograničenje brzine transakcije.

Glavna značajka Corda koncepta je **objekt stanja**, koji je zapravo digitalni dokument koji zapisuje postojanje, sadržaj i trenutno stanje sporazuma između dvije ili više stranaka. U Cordi se ažuriranja primjenjuju pomoću transakcija koje troše postojeće objekte stanja i proizvode nove objekte stanja.

Postoje dva aspekta konsenzusa:

1. Valjanost transakcije: stranke mogu biti sigurne da je predložena transakcija valjana tako da izvrši kod ugovora uspješno i da ima sve valjane potpise. Ovo također mora vrijediti i za sve transkacije na koje se ova transakcija referira.

2. Jedinstvenost transakcije: stranke mogu biti sigurne da je transakcija u pitanju jedinstveni potrošač svih njenih ulaznih stanja. Odnosno, ne postoji nijedna druga transakcija za koju je prethodno postignut konsenzus (valjanost i jedinstvenost), koja troši bilo koje od istih stanja. Stranke se mogu dogovoriti o valjanosti transakcija samostalnim pokretanjem istog koda ugovora i logike validacije. Me-

đutim, konsenzus oko jedinstvenosti zahtijeva unaprijed određenog promatrača, koji će u mnogim sluajevima morati biti neovisan.

2.3.2. Quorum

Quorum je razvila tvrtka J.P. Morgan kao inaicu Ethereum blockchaina s ograničenim pristupom. Ethereum je javni blockchain bez ograničenog pristupa koji se koristi za izradu decentraliziranih aplikacija unutar različitih domena. Ima podršku za Turing-potpune pametne ugovore i stoga se može koristit za izradu aplikacija opće namjene. Iako je javan i bez ograničenja pristupa, sigurnost Ethereuma proizlazi iz Dokaz izvršenog rada konsenzusnog algoritma i interne kriptovalute Ether. Quorum uključuje sljedeće ključne promjene u odnosu na Ethereum:

- Ograničeno sudjelovanje - stavlja restrikciju na sudjelovanje u blockchain mreži samo onim ččvorovima kojima je to dopušteno. Samo je njima dopušteno povezivanje na Quorum blockchain, provjeravanje transakcija, pokretanje pametnih ugovora i održavanje glavne knjige.
- Konsenzusni algoritmi - Quorum prema zadanim postavkama koristi Raft i Istanbulski BFT konsenzus. Ova dva algoritma pružaju zamjenu za Ethereumov PoW. Dok PoW štiti javni blockchain postepeno uvodeći težinu u kriptografsku slagalicu, potpuno je nepotrebno i rastrošno (velika potrošnja energije) za sustav ograničenog pristupa gdje si stranke međusobno vjeruju. Raft i IBFT algoritmi dovode do bržeg konsenzusa i pružaju trenutnu konačnost transakcije, čineći ih prikladan izbor za PB. Quorum također podržava lako promjenjivu arhitekturu u koju se po potrebi može uključiti drugačija implementacija konsenzusa.
- Privatnost: jedan od ciljeva dizajna Quoruma. Dozvoljava manjem dijelu konzorcija da vrše transakcije između sebe bez da ih javno iznose ostatku konzorcija. Privatnost u Quorum je omogućena podjelom veće javne glavne knjige na javnu i privatnu. Javna glavna knjiga je vidljiva svim čvorovima u mreži, a privatna samo onima koji vrše transkacije međusobno. Samo hash privatne transakcije je vidljiv na javnoj glavnoj knjizi, ali samo stranke između kojih je izvršena transakcija imaju ključ s kojim mogu pregledati transakciju. Također se i pametni ugovori mogu dijeliti na privatan način, između stranaka koje međusobno vrše transakcije.
- Ukidanje cijena transakcija: Quorum je eliminirao dodavanje troškova transak-

cijama pomoću *gas*-a. Stoga, Quorum nema troškova povezanih s izvođenjem transakcija na Quorum mreži. S obzirom da je Quorum kod "forkan" od Ethereuma, korištenje gas-a postoji ali je namješteno na 0. Gas je korišten u Ethereum mreži za plaćanje izvođenja transakcija (koristi se ether kriptovaluta) kako bi se incentivizirali rudari i kako bi se Ethereum mreža zaštitila od spam-a i DDoS napada.

2.3.3. Hyperledger Fabric

Mnogi PB-i trpe zbog mnogih ogranienja, koji esto proizlaze iz njihovih predaka - javnih blockchaina, ili iz upotrebe arhitekture "poredaj-izvrši" (eng. order-execute):

- konsenzus je vrsto kodiran unutar platforme, što je u suprotnosti s ustaljenim shvaćanjem da ne postoji jednoznačni (BFT) konsenzus protokol;
- model povjerenja validacije transakcija je određen konsenzusnim protokolom i ne može se prilagoditi zahtjevima pametnog ugovora;
- pametni ugovori moraju biti pisani na fiksan, nestandardan ili specifičan za domenu jezik, što ometa široko usvajanje i može dovesti do pogrešaka prilikom programiranja;
- slijedna izvedba svih transakcija od svih čvorova ograničava performanse, te su potrebne komplicirane mjere kako bi se spriječio DDoS napad;
- transakcije moraju biti deterministike, što je teško ostvariti programski;
- svaki pametni ugovor se izvršava na svim čvorovima, što je u suprotnosti povjerljivosti i zabranjuje širenje koda ugovora i stanje podskupu čvorova.

U ovom radu opisujemo Hyperledger Fabric ili jednostavno Fabric, blockchain platformu otvorenog koda koja prevladava ta ograničenja. Fabric je jedan od projekata Hyperledgera u okviru pokroviteljstva zaklade Linux. Fabric se koristi u više od 400 prototipova, dokaza o konceptu, produkcijskim raspodijeljenim glavnim knjigama i u različitim industrijama. Ovi slučajevi korištenja uključuju, ali nisu ograničeni na područja kao što su rješavanje sporova, trgovinska logistika, devizna mreža, sigurnost hrane, upravljanje ugovorima, dijamantna provenijencija, upravljanje bodovima, trgovanje niskom likvidnosti, upravljanje identitetima i namirba putem digitalne valute.

Fabric uvodi novu blockchain arhitekturu s ciljem elastičnosti, fleksibilnosti, skalabilnosti i povjerljivosti. Dizajniran kao modularni i proširivi PB, Fabric je prvi blockchain sustav koji podržava izvršavanje distribuiranih aplikacija napisanih u standardnim programskim jezicima, na način koji omogućuje njihovo dosljedno izvršavanje

preko mnogih čvorova, ostavljajući dojam izvršenja na jednom globalno distribuiranom blockchain računalu. To Fabric čini prvim distribuiranim operativnim sustavom za PB. Hyperledger Fabric uvodi i još jedan imbenik modularnosti i fleksibilnosti, a to su takozvani kanali. Kanali omogućavaju sudionicima stvarnje virtualnih grupa i održavanje neovisnih glavnih knjiga koje su nevidljive drugim kanalima u mreži. Kanali pružaju fleksibilnost poslovnom konzorciju da sigurno dijeli informacije samo s relevantim strankama. Arhitektura Fabric slijedi novu "izvrši-poredaj-validiraj" paradigmu za distribuirano izvršavanje nepovjerljivog koda u nepouzdanom okruženju. Tok transakcije razdvaja na tri koraka koji se mogu izvoditi na različitim entitetima u sustavu:

1. izvršavanje transakcije i provjera njezine ispravnosti i valjanosti;
2. redoslijed (eng. ordering) putem konsenzusnog protokola, bez obzira na semantiku transakcije;
3. validacija transakcije po pretpostavkama specifičnim aplikaciji, što također sprječava tzv. *race conditions*.

Ovaj dizajn radikalno odstupa od paradigme *order-execute* tako da Fabric obično izvršava transakcije prije postizanja konačnog dogovora o njihovom redoslijedu. Kombinira dva pristupa repliciranju, pasivni i aktivni. Prvo, Fabric koristi pasivnu ili primarno-sigurnosnu replikaciju kao što se čestonalazi u distribuiranim bazama podataka, ali s posredničkim softverom koji procesuira ažuriranja asimetrično i u nepouzdanoj okolini s Bizantskim pogreškama. U Fabricu svaka transakcija se izvršava (odobrava) samo na podskupu čvorova (eng. peer), što dopušta paralelno izvršavanje i rješava potencijalni nedeterminizam, oslanjajući se na BFT replikaciju "izvrši-provjeri" (eng. execute-verify). Fleksibilna politika odobravanja transakcija određuje koji peer-ovi ili koliko njih treba jamčiti za ispravno izvršenje zadanog pametnog ugovora. Drugo, Fabric uključuje aktivnu replikaciju u smislu da učinci transakcije na stanje glavne knjige pišu se tek nakon postizanja konsenzusa o ukupnom redoslijedu među njima, u koraku determinističke validacije koji izvršava svaki peer pojedinačno. Ovo omogućava Fabricu da se prilagodi pretpostavkama povjerenja specifičnih za aplikaciju. Štoviše, ordering ažuriranja stanja delegirana je modularnoj komponenti radi konsenzusa, koja je bez stanja i logički odvojena od peerova koji izvršavaju transakcije i održavaju glavnu knjigu. Zbog konsenzusa koji je modularan, njegova se primjena može prilagoditi pretpostavkama o povjerenju određenog razvoja. Iako je moguće

koristiti blockchain peer-ove i za provođenje konsenzusa, razdvajanje dviju uloga dodaje fleksibilnost i omogućuje osloniti se na dobro uspostavljene alate za otpornost na kvarove (eng. Crash Fault Tolerance, skr. CFT). Sve u svemu, ovaj hibridni replikacijski dizajn koji miješa pasivnu i aktivnu replikaciju u bizantskom modelu i paradigma *execute-order-validate* predstavljaju glavnu inovaciju u arhitekturi Fabrica. Oni rješavaju prethodno spomenute probleme i omugu ćuju da Fabric bude skalabilni sustav za PB koji podržava fleksibilne pretpostavke povjerenja. Da bi implementirao ovu arhitekturu, Fabric sadrži modularne građevne blokove za svaku od sljedećih komponenata:

- usluga poravnanja transakcija atomski emitira ažuriranja stanja peer-ovima i uspostavlja konsenzus na redoslijed transakcija;
- usluga članstva (eng. Membership service provider, skr. MSP) je odgovorna za pridruživanje kriptografskih identiteta peer-ovima. Zapravo održava ograničenja pristupa unutar Fabrica;
- opcionalna *peer-to-peer gossip* usluga širi izlaze blokova svim peer-ovima;
- pametni ugovori u Fabricu izvršavaju se unutar kontejnera, izolirani. Mogu biti napisani u standardnim programskim jezicima, ali nemaju direktan pristup stanju glavne knjige;
- svaki peer lokalno održava glavnu knjigu u formi blockchaina na koji je moguće samo dodavati blokove i kao snapshot posljednjeg stanja u "ključ-vrijednost" pohrani.

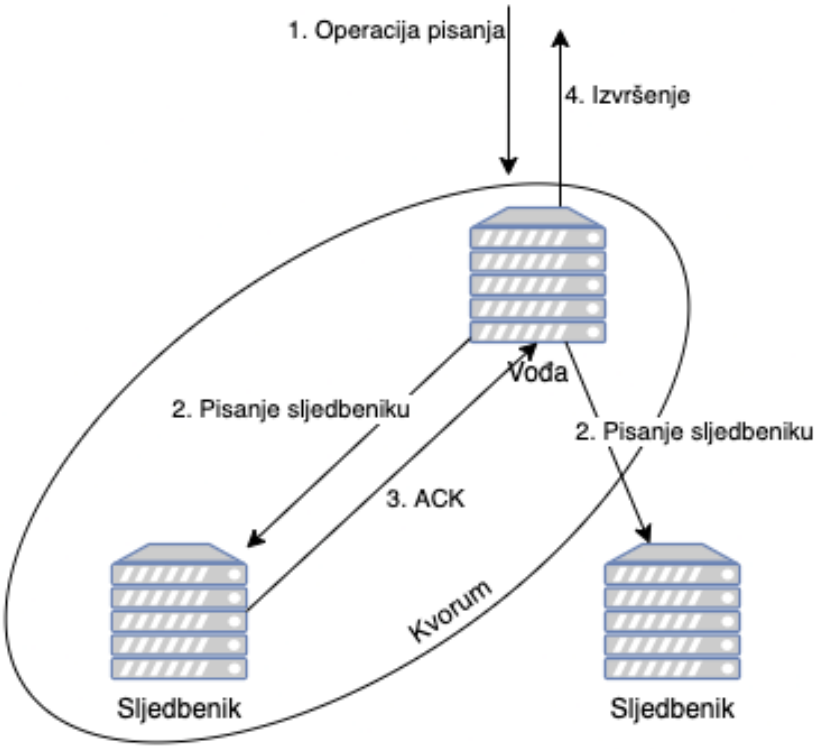
2.3.4. Usporedba

U ovoj sekciji usporedit ću tri prethodno navedene izvedbe blockchaina s ograničenim pravom pristupa. Provest ću analizu svake izvedbe prema tome kako implementiraju konsenzus, pametne ugovore, autentifikaciju i autorizaciju i prema propusnosti transakcija.

Konsenzus

U blockchain mreži, tijekom obrade bloka, čvorovi obavljaju određene aktivnosti, kao što su sudjelovanje u provjeri autentičnosti i provjeri transakcija, komunikacija preko mreže i izgradnja povjerenja unutar blockchain sustava bez uplitanja središnjeg autoriteta. Uvijek postoji rizik da se pojedini čvorovi mogu ponašati zlonamjerno ili djelovati protiv osnovnog cilja i pokušati srušiti komunikaciju u mreži. Da bi se osigurala

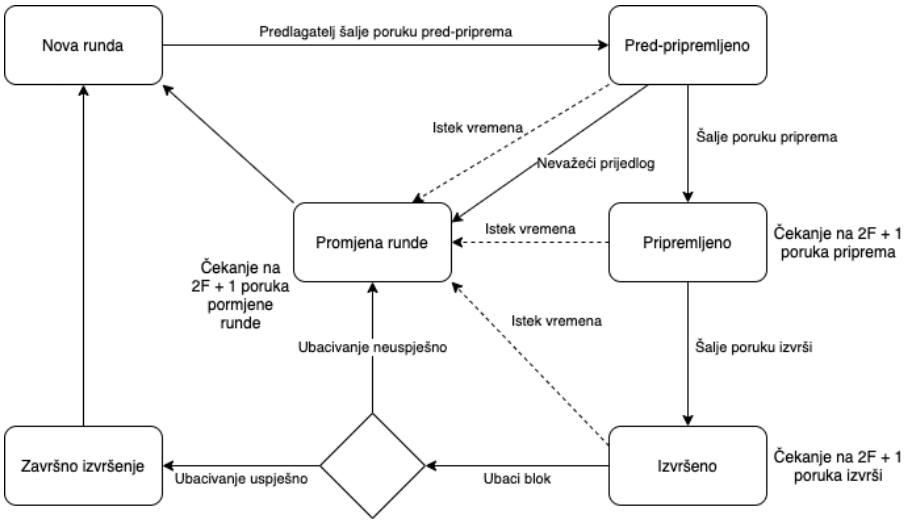
kontinuirana usluga koja pruža dostupnost, povjerljivost, integritet, i pristupačnost potreban je siguran mehanizam da bi svi čvorovi sudionici postigli globalni dogovor koje informacije trebaju biti dodane blockchainu. Ovaj postupak je poznat kao konsenzus. Hyperledger Fabric u prvoj verziji koristi konsenzus protokol Apache Kafka. Kafka nije tradicionalni konsenzusni protokol, nego je "*publish-subscribe*" rješenje. Kafka koristi isti "vođa i sljedbenik" koncept kao i Raft kao što je vidljivo na slici 2.4, u kojem su transakcije (Kafka koristi pojam "poruka") replicirane od čvora vođe do čvorova sljebenika. Od verzije 2.0 Fabric koristi Raft konsenzus prethodno opisan u poglavlju **??**.



Slika 2.4: Kafka konsenzus

Quorum po zadanim postavkama koristi Raft za toleranciju pada sustava i Istanbulski BFT za toleranciju Bizantske pogreške. Istanbulski BFT konsenzus je konsenzus u 3 faze: pred-priprema, priprema i izvršenje. Ovaj konsenzus sustav može tolerirati da jedna trećina svih čvorova u mreži bude neispravan, a opet osigurava konačnost transakcije. Čvorovi koji su izabrani kao "validatori" bloka odabiru "predlagača" za predlaganje novog bloka u konsenzusnom krugu. Predlagatelj će tada predložiti novi blok i emitirati ga zajedno s porukom "pred-priprema". Po primitku "pred-priprema"

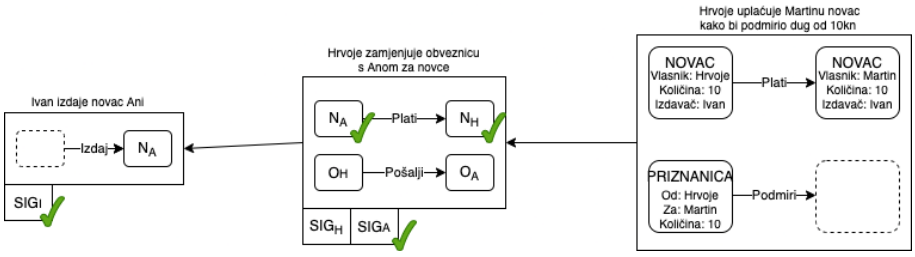
poruke predlagatelja, validatori ulaze u stanje "pred-pripremljeno", a zatim emitiraju "priprema" poruku. Ovaj korak želi osigurati da svi validatori rade u istom slijedu i istom krugu. Nakon primanja poruke "priprema" od dvije trećine mrežnih čvorova, validator ulazi u stanje "pripremljeno", a zatim emitira "izvrši" poruku. Ovaj korak služi tome da se obavijeste peer-ovi da validator prihvaća predloženi blok i da će ga dodati u lanac. Na kraju, validatori čekaju da dvije trećine poruka "izvrši" uđu u stanje "izvršeno" i zatim umetnu blok u lanac.



Slika 2.5: IBFT

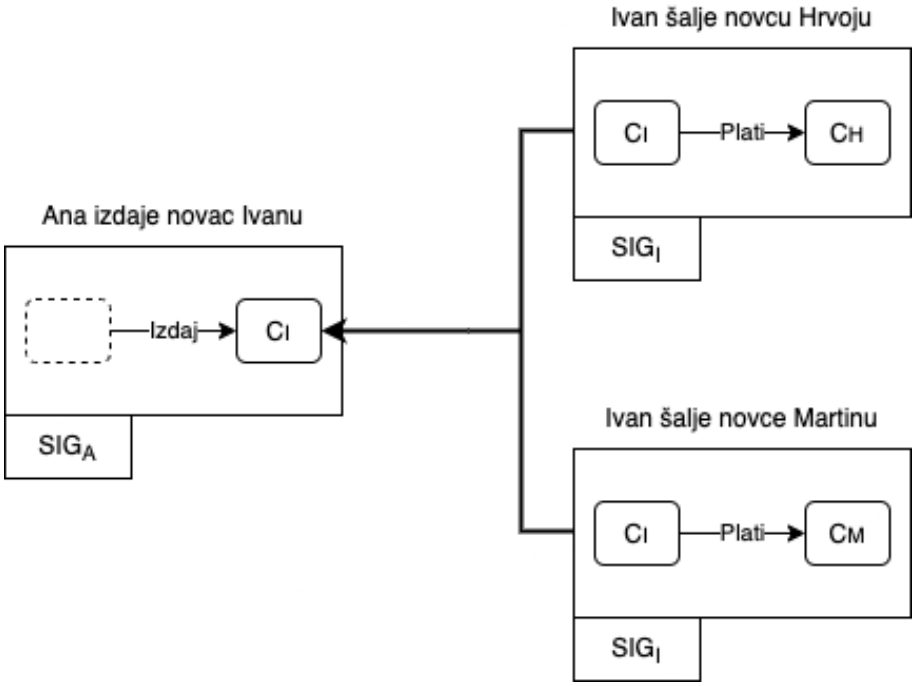
Corda ima zamjenjivu uslugu jedinstvenosti. To služi tome da se poboljša privatnost, skalabilnost, kompatibilnost pravnog sustava i algoritamska agilnost. Jedna usluga može biti sastavljena od mnogih međusobno nepovjerljivih čvorova koji se koordiniraju preko BFT algoritma ili može biti vrlo jednostavna, poput jednog stroja. U nekim slučajevima, kada ažuriranje stanja zahtijeva potpise svih relevantnih strana, možda uopće ne treba usluga jedinstvenosti. Konsenzus oko valjanosti transakcije provodi se samo između stranaka između kojih obavljena ta transakcija. Stoga se podaci dijele samo s tim stranama. Ostale platforme uglavnom postižu konsenzus na razini glavne knjige. Dakle, bilo koji akter u sustavu Corda vidi samo podskup ukupnih podataka kojima upravlja sustav u cjelini. Dio podataka zapisan je u glavnu knjigu ako su barem dva aktera u sustavu konsenzusni o njegovom postojanju i pojedinostima i dopuštaju se proizvoljne kombinacije aktera da sudjeluju u postupku konsenzusa za bilo koji podatak.

Prilikom provjere predložene transakcije, određena stranka ne mora imati svaku transakciju u lancu transakcija koju trebaju provjeriti. U tom slučaju, može zatražiti transakcije kojih nema od predlagatelja transakcija. Predlagatelji transakcije uvijek će



Slika 2.6: Valjanost

imati potpuni transakcijski lanac, budući da bi to zatražili prilikom provjere transakcije koja je stvorila predložena ulazna stanja.



Slika 2.7: Jedinstvenost

Valjani prijedlog transakcije također mora postići konsenzus jedinstvenosti. Konsenzus jedinstvenosti je uvjet da ni jedno od ulaznih stanja predložene transakciju nije već konzumirano u drugoj transakciji. Ako je jedan ili više ulaza već konzumirano u drugojvtransakciji, to je poznato kao dvostruka potrošnja, a prijedlog transakcije smatra se nevažećim. Konsenzus jedinstvenosti omogućuju bilježnici (eng. *Notaries*) koji vrše provjeru dvostruke potrošnje.

Pametni ugovori

Pametni ugovor u Corda sustavu je sporazum iju je izvedbu moguće automatizirati računalnim kodom i čija prava i obveze, navedene u takozvanoj legalnoj prozi (eng. legal prose), su legalno provedive. Pametan ugovor u Cordi ima tri ključna elementa, a to je izvršni kod, objekte stanja i naredbe. Izvršni kod uglavnom provjerava promjene objekata stanja u transakcijama. Objekti stanja su podaci koji bilježe postojanje, sadržaj i trenutno stanje sporazuma između dvije ili više strana i rade kao ulaz ili izlaz transakcije. Naredbe su dodatni podaci koji su uključeni u transakcije. Oni uglavnom opisuju što se događa i govore izvršnom kodu na koji način verificirati transakcije.

Quorum je blockchain s ograničenim pravom pristupa implementiran na Ethereum protokolu. Osnovna ideja koja stoji iza Quoruma je korištenje kriptografije kako bi se spriječili svi osim onih koji sudjeluju u transakcijama od gledanja osjetljivih podataka. Rješenje uključuje jedan zajedniki blockchain i kombinaciju softverske arhitekture pametnih ugovora i modifikacija Ethereuma. Arhitektura pametnih omogućuje segmentaciju privatnih podataka. Izmjene Ethereum izvornog koda uključuju modifikacije u procesu prijedlaganja bloka i procesu validacije. Proces validacije je modificiran tako da svi čvorovi potvrđuju javne transakcije i bilo koje privatne transakcije u kojima sudjeluju izvršenjem koda pametnog ugovora povezanog s transakcijama. Za ostale privatne transakcije, čvor će jednostavno preskoiti postupak izvršavanja koda ugovora.

Hyperledger Fabric često izmjenjuje izraze pametni ugovor i lančani kod (eng. *Chaincode*). Općenito, pametni ugovor definira transakcijsku logiku koja kontrolira životni ciklus poslovnog objekta koji se nalazi u globalnom stanju. Zatim se pakira u lančani kod koji se zatim raspoređuje u blockchain mrežu. Pametni ugovor može opisati gotovo beskonačan niz slučajeva poslovne upotrebe koji se odnose na nepromjenjivost podataka u višeorganizacijskom donošenju odluka. Posao programera pametnih ugovora je preuzeti postojeći poslovni proces koji može regulirati financijske cijene ili uvjete isporuke i izraziti ga kao pametni ugovor u programskom jeziku kao što su JavaScript, Go ili Java.

Autentifikacija i autorizacija

Ethereum razlikuje dvije vrste računa koji se koriste za transakcije, a to su računi ugovora i vanjski računi. Svaki račun ima svoje stanje, a globalno stanje mreže je skup svih stanja računa. računi u Quorumu, koji je izveden iz Ethereuma, imaju potpuno istu strukturu. Quorum je blockchain s ograničenim pravom pristupa, a osigurava da u razmjeni poruka mogu sudjelovati samo ovjereni čvorovi. Autentifikacija čvora poši-

ljatelja oslanja se na autentifikaciju na temelju ključa koristeći ECDSA (eng. Elliptic Curve Digital Signature Algorithm) potpis - svakom je korisniku dodijeljen par asimetričnih ključeva generiranih pomoću eliptine krivulje secp256k1 i sadrže popis drugih javnih ključeva čvorova u mreži. **?** ECDSA omogućava primatelju izvlačenje javnog ključa pošiljatelja iz potpisa poruke koji uspoređuje s listom javnih ključeva drugih čvorova u mreži. Ako se ključ podudara, čvor je autentificiran, u protivnom se odbija veza. Opcionalno se može omogućiti TLS koji zahtijeva uzajamnu provjeru autentičnosti klijenta ili poslužitelja. Dostupno je nekoliko načina rada: TOFU, CA i bijela lista (eng. whitelist) (TOFU i CA kombinirano). Ključevi za identitet i TLS mogu sem ponovno iskoristiti.

- TOFU (eng. Trust-on-first-use) - samo prvi čvor koji se poveže s identifikacijom određenog host-a moći će se povezati samo kao isti host u budućnosti. Oslanja se na provjeru autentičnosti na temelju ključa;
- CA (eng. Certificate Authority - samo čvorovi s ispravnim certifikatom i lancom povjerenja se mogu povezati. Oslanja se na provjeru autentičnosti na temelju certifikata;
- WHITELIST - samo čvorovi koji su se prethodno povezali s tim čvorom i dodani su u datoteku "poznati klijenti" (eng. *knownclient*) moći će se povezati. Oslanja se na provjeru autentičnosti na temelju ključa.

Autentifikacija incijatora transakcije, slično Ethereumu, oslanja se na autentifikaciju na temelju ključa pomoću ECDSA potpisa koji se koriste za potvrdu da je pošiljatelj vlasnik računa s kojeg transakcija proizlazi. Quorum uvodi koncept privatnosti transakcija i razlikuje javne transakcije od privatnih transakcija. Javne transakcije su identične Ethereum transakcijama i dostupna svima u mreži, dok je privatna transakcija dostupna ograničenom broju čvorova koje definira pošiljatelj. Uz svaku transakciju povezana je lista kontrole pristupa koja sadrži javne ključeve autoriziranih čvorova. U Hyperledger Fabric-u postoje dvije vrste transakcija opisane u nastavku:

- transakcija postavljanja (eng. Deploy transaction) - stvara novi chaincode i prima program kao parametar. Kada se ta transakcija uspješno izvrši, chaincode je instaliran na blockchain;
- transakcija pobude (eng. Invoke transaction) - izvodi operaciju u kontekstu prethodno postavljnog chaincode-a. Ta transakcija odnosi se na chaincode i na jednu od funkcija koju chaincode nudi.

Servis članstva (eng. Membership service provider, skr. MSP) nudi apstrakciju

arhitekture lanstva. Upravlja provjerom autentičnosti i autorizacijom u mreži, a podijeljen je na različite razine:

- MSP mreže - definira sudionike mreže putem popisa MSP-a organizacija i pruža njihovu autorizaciju, naprimjer ovlaštenje za stvaranje kanala;
- MSP kanala - definira administrativna i participativna pravan na razini kanala. Svaka organizacija koja sudjeluje u kanalu mora imati definirani MSP. Peer-ovi i orderer-i u kanalu dijele isti popis MSP-ova, stoga mogu ispravno autentificirati sudionike kanala;
- lokalni MSP (orderer-i, peer-ovi, klijenti) - omogućava korisniku da se autentificira u svojim transakcijama kao član kanala, ili kao vlasnik određene uloge (autorizacija) u sistemu (npr. administrator organizacije). Lokalni MSP su definirani za klijente i za čvorove (peer-ovi i orderer-i).

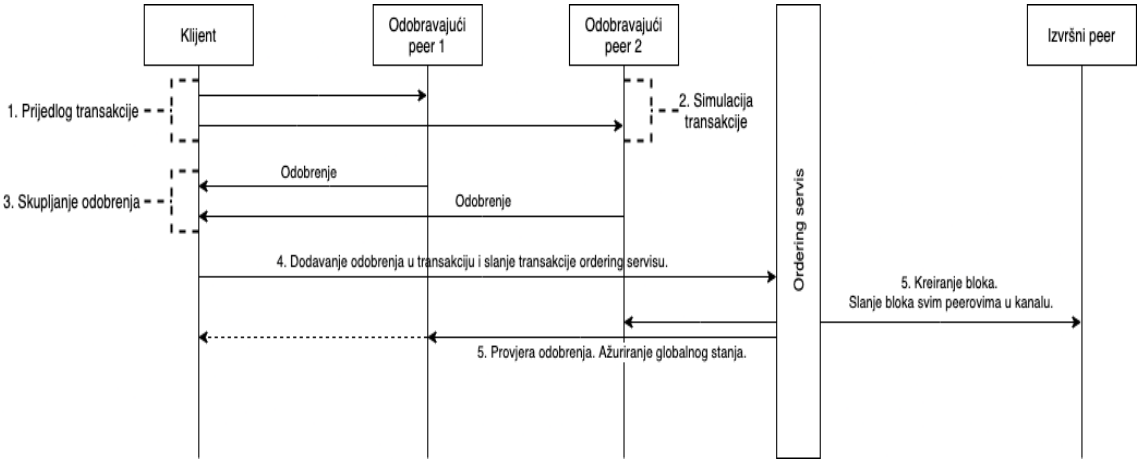
Hyperledger Fabric pruža opciju korištenja predefiniranog tijela za izdavanje certifikata koje se može koristiti za generiranje certifikata i ključeva. Međutim može se zamijeniti za bilo koji CA koji može generirati ECDSA certifikate. Protokol za provjeru izvornosti poruke koristi certifikat X.509 na temelju ECC-a **?** i opcionalne TLS certifikate. Za provjeru autentičnosti pošiljatelja poruke od strane kanala, mora predstaviti certifikat s provjerljivim putem do točno jednog od korijena povjerljivog certifikata. Da bi se pridružio kanalu, novi čvor mora poslati zahtjev za potpisivanje certifikata na jedan od kanalovih korijenskih ili posredničkih CA-ova. Nadalje, mora konfigurirati svoj lokalni MSP sa sljedećim informacijama:

- korijen povjerljivog certifikata;
- (opcionalno) posrednički CA certifikati;
- certifikat MSP administratora;
- popis opoziva certifikata koji odgovaraju prethodno navedenim CA-ovima (korijenskim ili posredničkim);
- TLS korijen povjerljivog certifikata;
- (opcionalno) posrednički TLS CA;
- (opcionalno) lista organizacijskih jedinica koje član tog MSP-a mora sadržavati u svom certifikatu.

Kanal u Fabric-u po želji definira određene uloge unutar kanala pomoću kontrole pristupa zasnovane na atributima (eng. *Attribute-Based Access Control*, skr. ABAC) koja se temelji na atributima identiteta sadržanih u certifikatima. Transakcije postavljanja ne zahtijevaju posebno odobrenje. Unutar kanala svi imaju pristup transakcijama

i ne postoje sheme autorizacije za pristup određenoj transakciji, transakcije se tretiraju kao uobičajene poruke i stoga se provjera autentičnosti pošiljatelja transakcija obrađuje slično autentifikaciji pošiljatelja poruke. Autorizacijska shema odobrenja transakcija oslanja se na liste kotrole pristupa: da bi transakcija bila odobrena, mora se odobriti prema politici odobrenja pozvanog chaincode-a. Tok operacije odobrenja prikazan na slici 2.8 je sljedeći:

1. klijent kreira transakciju i šalje je odobravajućim peer-ovima koje odabere;
2. odobravajući peer simulira transakciju i proizvodi odobravajući potpis;
3. klijent prikuplja odobrenje transakcije i emitira ga putem servisa poravnavanja;
4. servis poravnavanja dostavlja transakciju peer-ovima.



Slika 2.8: Hyperledger Fabric tijek transakcija

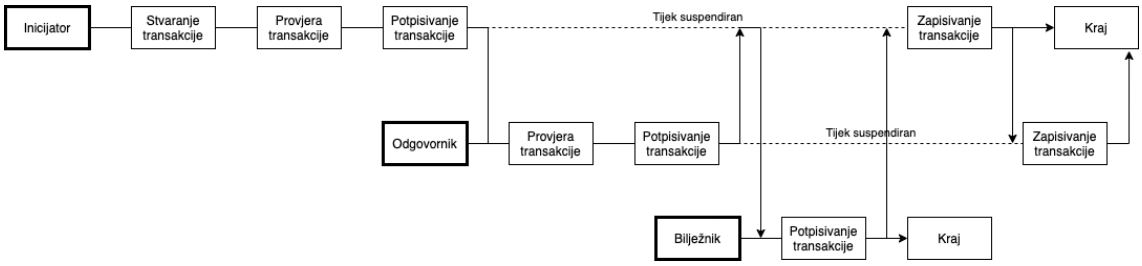
U Cordi, zona kompatibilnosti označava mrežu koja ima ograničen pristup. Corda mreža ima četiri vrste CA:

- korijenski CA;
- vratar CA - ponaša se kao posrednički CA;
- CA čvora - svaki čvor služi kao svoj CA u izdavanju podređenih certifikata koje koristi za potpisivanje svojih ključeva identiteta i TLS certifikata;
- CA legalnih identiteta - "dobro poznati" legalni identiteti čvorova, osim potpisivanja transakcija, mogu izdavati certifikate za povjerljive legalne identitete.

Cordin dizajn integrira provjeru autentičnosti temeljenu na certifikatima koristeći digitalne potipse kako bi se provjerila autentičnost pošiljatelja poruke. Čvorovi posjeduju TLS certifikate i certifikate identiteta. Kada čvor primi zahtjev za povezivanjem,

provjerava lanac povjerenja TLS certifikata i certifikata identiteta. Potpisi oba certifikata u lancu moraju biti valjani skroz do korijenskog CA. Da bi se čvor priključio mreži potrebno je proći kroz sljedeće korake:

1. novi čvor mora posjedovati korijenski CA u svojoj pohrani povjerenja, adresu vratara i kartu mreže;
2. čvor se mora prijaviti sa svojim certifikatom na vratara podnošenjem zahtjeva za potpisivanje certifikata da bi dobio CA certifikat čvora;
3. iz CA certifikata čvora, čvor stvara i potpisuje još dva certifikata, TLS certifikat i certifikata za čvorov "dobro poznati" identitet;
4. konačno, čvor gradi zapis podataka o čvoru koji sadrži njegovu adresu i dobro poznati identitet te ga registrira s uslugom karte mreže.



Slika 2.9: Tijek transakcija u Cordi

Corda specificira akcije koje je čvoru dopušteno poduzeti koristeći ABAC - tipovi identiteta su definirani u certifikatu. Tipovi mogu biti sljedeći:

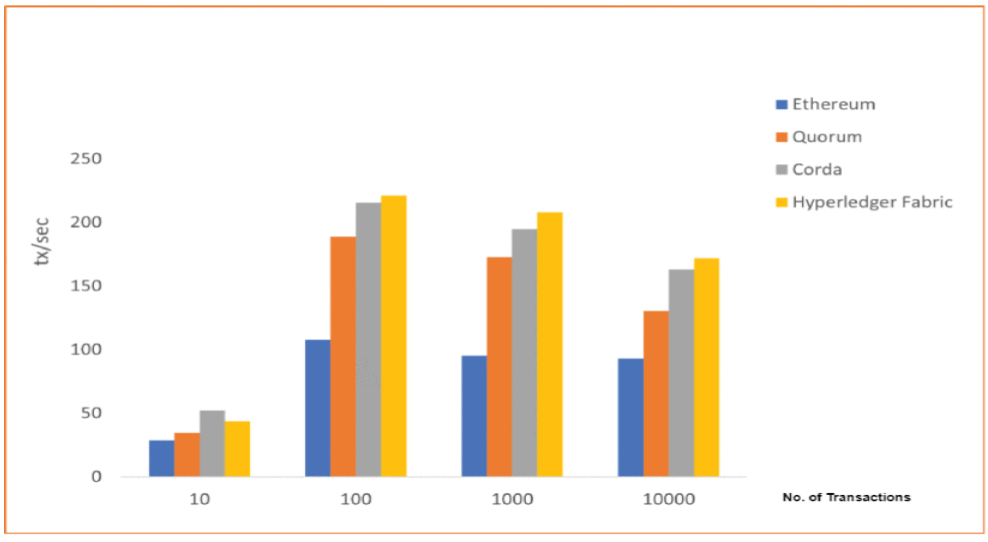
- vratar (eng. *doorman*);
- karta mreže (eng. *network map*);
- identitet servisa (eng. *service identity*);
- autoritet certifikata čvora (eng. *node certificate authority*);
- sigurnost transportnog sloja (eng. *transport layer security*);
- "dobro poznati" legalni identitet (eng. *well-known legal identity*);
- povjerljivi legalni identitet (eng. *confidential legal identity*).

U suprotnosti s mnogim blockchain platformama s ograničenim pravom pristupa, u Cordi sve transakcije su privatne što znači da su djeljene samo između dionika, i autorizaciju za pozivanje transakcije daje bilježnik. Da bi transakcija bila valjana, sva

ulazna stanja moraju se prvo prebaciti na istog bilježnika. To se radi pomoću posebne transkacije za promjenu bilježnika koja proizvodi izlazno stanje slično ulaznom stanju, ali prebačeno na novog bilježnika. Udaljena autentifikacija korisnika podržana je autentifikacijom putem lozinke kojom se upravlja na razini čvora. Opcionalno se može postaviti provjera autentičnosti klijenta temeljem certifikata pomoću TLS-a. Programski sučelje čvora potupuno je izloženo udaljenim pozivima. Ovlaštenje za upotrebu Corda udaljenog programskog sučelja koristi popise sposobnosti. Udaljeni korisnik posjeduje popis sposobnosti po udaljenom čvoru kojem pristupa. Dakle, metode koje udaljeni korisnik smije pozivati, ovise o autorizaciji koja mu je odobrena.

Propusnost transakcija

Propusnost transakcija je brzina kojom su valjane transakcije postavljene na blockchain u definiranom vremenskom periodu. Kao što se vidi na slici 2.10 koja prikazuje mjerenja od (Monrat et al., 2020), Hyperledger Fabric je najbolji izbor za izgradnju blockchaina koji treba podnijeti veliki broj transakcija. Doduše, ove analize napravljene su pomoću alata za testiranje koje su razvile tvrtke koje stoje iza samih platformi te bi se to moglo smatrati nedostakom ove analize. Analiza bi bila efikasnije i više "fer" da postoji univerzalni alat kojim je moguće provesti analizu nad svim navedenim blockchain platformama.



Slika 2.10: Usporedba propusnosti transakcija

Fabric, razvijen od strane Linux fundacije u sklopu multi-projekta otvorenog koda je radni okvir koji sam odlučio koristiti za implementacijski dio ovoga rada. Analiza propusnosti transakcija odrađena je od strane (Monrat et al., 2020) između Ethereum,

Quorum, Corda i Hyperledger Fabric platforme. Možemo primjetiti da Fabric nadmašuje Quorum i Ethereum s velikom razlikom, ali samo nešto bolje od Corde. Fabric ima nisku latenciju obrade transakcija u usporedbi s ostalim blockchainima s ograničenim pravom pristupa. Stoga je ova platforma sposobna također pružiti i bolju propusnost transakcija. Njihova eksperimentalna opažanja otkrivaju da Hyperledger Fabric ima bolje rezultate od ostalih platformi zbog svog jednostavnog i uinkovitog modularnog pristupa konsenzusu.

3. Implementacija

Implementacijski dio ovog rada odradio sam koristeći Hyperledger Fabric radni okvir. Ovaj radni okvir omogućava da sustav koji želimo napraviti prilagodimo potrebi aplikacije vrlo detaljno. Od određivanja uloga korisnika sustava do same poslovne logike. U Fabricu svaki čvor je pokrenut kao Docker kontejner (eng. *container*). Docker je set "platforma kao servis" (eng. *Platform as a Service,* skr. PaaS) proizvoda koji koriste virtualizaciju na razini operativnog sustava kako bi dostavili softver u pakiranjima koji se zovu kontejneri (eng. container). Kontejneri su odvojeni jedni od drugih, ali skupa rade na istoj mreži. Datoteka koja služi za definiranje konfiguracije Docker-a je *docker-compose.yaml*. U ovoj datoteci definirani su kontejneri i mreža kojoj su pridruženi i njihove varijable okruženja. Pokreće se naredbom *docker-compose* kojoj se pridodaje putanja do *docker-compose.yaml*datoteke. Implementirao sam primjer dodjeljivanja hipotetskih bodova za menzu studentima i dodjeljivanje rektorovih nagrada u obliku virtualnih tokena. Bodovi za menzu predstavljeni su tokenom koji slijedi ERC-20 standard, što znači da je token "zamjenjiv" (eng. *fungible*), a tokenom koji slijedi ERC-721 standard predstavljene su rektorove nagrade, a takav token je "nezamjenjiv" (eng. *non-fungible*).

Direktorij u kojem se nalaze izvršne datoteke potrebne za pokretanje Fabric mreže je "bin" direktorij unutar "network" direktorija. U "bin" direktoriju nalaze se sljedeće datoteke:

- *peer* - ova datoteka služi kao naredba preko komandne linije i dijeli se na više podnaredbi:
  - *peer node* - naredba omogućuje administratoru da pokrene peer čvor, resetira sve kanale kojima je peer član do genesis (nultog) bloka ili vrati kanal na zadani broj bloka.
  - *peer channel* - naredba omogućuje administratorima izvođenje operacija povezanih s kanalom na peer-u, poput pridruživanja peer-a kanalu ili popisa kanala kojima je peer pridružen.
  - *peer chaincode* - naredba omogućuje administratorima izvođenje ope-

racija povezanih s lančanim kodom, poput instaliranja, instanciranja, pozivanja, pakiranja, postavljanja upita i nadogradnje lančanog koda.

- *peer lifecycle chaincode* - podnaredba omogućuje administratorima da koriste životni ciklus lančanog koda za pakiranje lančanog koda, njegovo instaliranje na peer-ove, odobravanje definicije lančanog koda za organizaciju i potom definiranje na kanalu. Lančani kôd spreman je za upotrebu nakon što je uspješno definiran na kanalu.

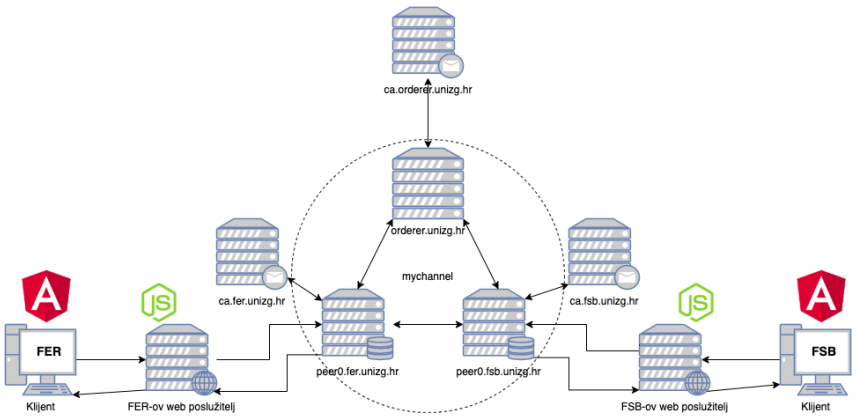
– *configtxgen* - naredba omogućuje korisnicima stvaranje i pregled artefakata povezanih s konfiguracijom kanala. Sadržaj generiranih artefakata diktira sadržaj datoteke configtx.yaml. Ta datoteka nalazi se u korijenskom "network" direktoriju i služi za definiranje konfiguracije kanala i politike u kanalu.

– *configtxlator* - naredba omogućuje korisnicima prevođenje između protobuf i JSON inačica struktura podataka i stvaranje ažuriranja konfiguracije. Naredba može pokrenuti REST poslužitelj da izloži svoje funkcije preko HTTP-a ili se može koristiti izravno kao alat naredbenog retka.

– *fabric-ca-client* - naredba omogućuje upravljanje identitetima (uključujući upravljanje atributima) i certifikatima (uključujući obnavljanje i opoziv).

– *fabric-ca-server* - naredba omogućuje inicijalizaciju i pokretanje procesa poslužitelja koji može poslužiti jedno ili više tijela za izdavanje certifikata. Konfiguracija se nalazi u fabric-ca-server-config.yaml datoteci.

– *orderer* - naredba omogućuje pokretanje orderer čvora. Konfiguracija za taj čvor nalazi se u orderer.yaml datoteci.



Slika 3.1: Arhitektura rješenja

Mreža se sastoji od sveukupno tri organizacije. FER, FSB i ordering organizacija. FER i FSB imaju svaka po jedan glavni čvor koji je zadužen za održavanje raspodi-

jeljene glavne knjige, blockchaina i pametnih ugovora. Taj čvor se naziva peer. Ti peer-ovi, skupa s orderer čvorom, povezani su u jednom kanalu. Uz peer-ove ključni su i poslužitelji autoriteta za izdavanje certifikata.

3.1. Servis članstva

Autoriteti za izdavanje certifikata izdaju identitete generiranjem javnog i privatnog ključa koji oblikuje par ključeva koji se može koristiti za dokazivanje identiteta. Budući da se privatni ključ nikada ne može javno dijeliti, potreban je mehanizam kako bi se omogućilo dokazivanje identiteta, i tu dolazi "servis članstva" (eng. Membership Service Provider, skr. MSP). Na primjer, peer koristi svoj privatni ključ za digitalno potpisivanje, ili odobravanje, transakcije. MSP na ordering usluzi sadrži javni ključ peer-a koji se tada koristi za provjeru da je potpis transakcije validan. Privatni ključ se koristi za proizvodnju potpisa na transakciji koju samo odgovarajući javni ključ, koji je dio MSP-a, može otključati. Dakle, MSP je mehanizam koji omogućuje da se identitetu može vjerovati i prepoznati od strane ostatka mreže bez otkrivanja privatnog ključa člana.

Implementacija MSP-a je skup direktorija koje se dodaju konfiguraciji mreže i koristi se za definiranje organizacije iznutra (organizacije odlučuju tko su njeni administratori) i izvana (dopuštajući drugim organizacijama da potvrde da subjekti imaju ovlasti za činjenje onoga što pokušavaju učiniti). Autoriteti za izdavanje certifikata generiraju potvrde koje predstavljaju identitete, MSP sadrži popis odobrenih identiteta.

MSP se pojavljuje u dvije domene:

- Lokalna - definirani su za klijente i čvorove (peer i orderer). Svaki čvor mora imati lokalni MSP.
- Kanalska - definiraju administrativna prava i prava sudjelovanja na razini kanala. Identificira tko ima vlasti na razini kanala. Svaka organizacija koja sudjeluje u kanalu mora imati MSP definiran za to. MSP sustava uključuje MSP svih organizacija koje sudjeluju u ordering servise. Lokalni MSP-ovi su definirani samo na datotečnom sustavu čvora ili korisnika. Kanalni MSP je također instanciran na datotečnom sustavu svakog čvora u kanalu i čuva se sinkroniziranim konsenzusnim protokolom.

Organizacija se također može podijeliti u više organizacijskih jedinica (eng. Organizational Unit, skr. OU), od kojih svaka ima određeni skup odgovornosti, koje se nazivaju i afilijacije. Navođenje OU-a nije obavezno. Ako se OU ne koriste, svi

identiteti koji su dio MSP-a - kako su identificirani u korijenskim i posredničkim CA direktorijima - smatrat će se članovima organizacije.

Uz to, postoji posebna vrsta OU, koja se ponekad naziva OU čvora, a koja se može koristiti za dodjeljivanje uloge identitetu. Ove uloge OU čvora definirane su u datoteci "/msp/config.yaml" i sadrže popis organizacijskih jedinica čiji se članovi smatraju dijelom organizacije koju predstavlja ovaj MSP. To je osobito korisno kada želite ograničiti članove organizacije na one koji imaju identitet (potpisan od jednog od MSP-a određenih CA-a) s određenom ulogom OU čvora.

Da bi se koristile uloge OU čvora, mora biti omogućena značajka "klasifikacija identiteta" za mrežu. Kada se koristi struktura MSP-a koja se temelji na mapi, to se postiže omogućavanjem "Node OU" u datoteci config.yaml koja se nalazi u korijenu mape MSP:

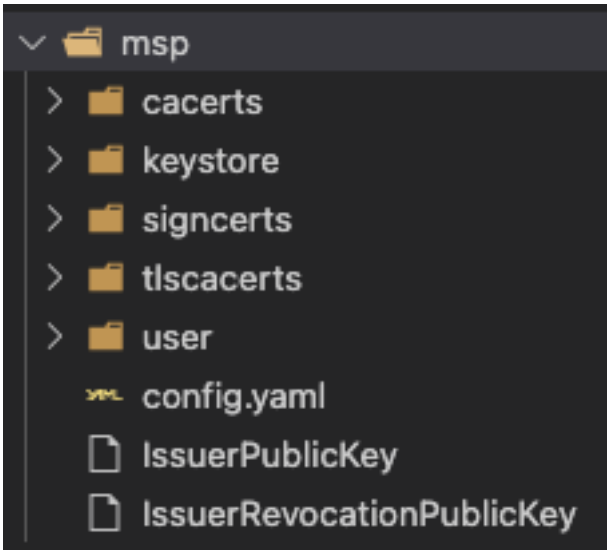
```
NodeOUs:
  Enable: true
  ClientOUIdentifier:
    Certificate: cacerts/localhost-7054-ca-fer.pem
    OrganizationalUnitIdentifier: client
  PeerOUIdentifier:
    Certificate: cacerts/localhost-7054-ca-fer.pem
    OrganizationalUnitIdentifier: peer
  AdminOUIdentifier:
    Certificate: cacerts/localhost-7054-ca-fer.pem
    OrganizationalUnitIdentifier: admin
  OrdererOUIdentifier:
    Certificate: cacerts/localhost-7054-ca-fer.pem
    OrganizationalUnitIdentifier: orderer
```

U ovom primjeru nalaze se 4 moguće uloge:

- klijent (eng. client)
- peer
- administrator (eng. admin)
- orderer

Struktura MSP direktorija prikazana na slici 3.2:

- config.yaml - Koristi se za konfiguriranje značajke klasifikacije identiteta u Fabricu omogućavanjem "Node OUs" i definiranjem prihvaćenih uloga.



Slika 3.2: Struktura msp direktorija

- cacerts - Ovaj direktorij sadrži popis samopotpisanih X.509 certifikata korijenskih CA-a kojima vjeruje organizacija koju predstavlja ovaj MSP.
- signcerts - Za peer ili orderer (ili u lokalnom MSP-u klijenta) ovaj direktorij sadrži certifikat čvora koji je izdao CA.
- keystore - Ovaj direktorij sadrži privatni ključ.
- tlscacerts - Ovaj direktorij sadrži popis samopotpisanih X.509 certifikata korijenskia CA-a kojima ova organizacija vjeruje za sigurnu komunikaciju između čvorova pomoću TLS-a.
- itd.

3.2. Peer

Blockchain mreža se sastoji uglavnom od skupa peer-ova. Peer-ovi su temeljni element mreže jer su poslužitelji glavnih knjiga i pametnih ugovora. Podsjetimo da glavna knjiga nepromjenjivo bilježi sve transakcije generirane pametnim ugovorima (koji su u Fabric-u sadržani u chaincode-u). Pametni ugovori i glavne knjige koriste se za enkapsuliranje zajedničkih procesa i zajedničkih informacija u mreži. Peer poslužuje instance glavnih knjiga i instance chaincode-ova. Budući da je peer poslužitelj za glavne knjige i chaincode-ove, aplikacije i administratori moraju komunicirati s peer-om ako žele pristupiti tim resursima. Zato se peer-ovi smatraju temljnim građevnim blokovima Fabric mreže.

U ovoj implementaciji koristim 2 peer-a. Jedan FER-ov peer0.fer.unizg.hr i FSB-ov peer0.fsb.unizg.hr. Ova nula poslije peer znači broj peer-a, a u ovom primjeru imamo samo jednog peer-a po organizaciji.

Transakcija ažuriranja prilično se razlikuje od transakcije upita jer pojedinačni peer ne može samostalno ažurirati glavnu knjigu - za ažuriranje je potreban pristanak ostalih peer-ova u mreži. Peer zahtijeva od ostalih peer-ova u mreži da odobre ažuriranje knjige prije nego što se može primijeniti na lokalnu glavnu knjigu peer-a. Taj se postupak naziva konsenzusom, koji traje puno dulje od jednostavnog upita. Ali kada svi peer-ovi odobre transakciju, peer-ovi će obavijestiti svoje povezane aplikacije da je glavna knjiga ažurirana. Točnije, aplikacije koje žele ažurirati glavnu knjigu uključene su u 3-fazni postupak, koji osigurava da svi peer-ovi u blockchain mreži održavaju svoje glavne knjige međusobno dosljednima.

- U prvoj fazi aplikacije rade s podskupom odobravajućih peer-ova, od kojih svaki šalje odobrenje predloženog ažuriranja glavne knjige aplikaciji, ali ne primjenjuje predloženo ažuriranje na svoju kopiju glavne knjige.
- U drugoj fazi, ova odobrenja prikupljaju se zajedno kao transakcije i pakiraju u blokove.

Prva faza - prijedlog. Faza 1 tijeka transakcije uključuje interakciju između aplikacije i skupa peer-ova - ne uključuje orderer-a. Faza 1 odnosi se samo na aplikaciju koja traži da odobravajući peer-ovi organizacija pristanu na rezultate predloženog pozivanja chaincode-a. Da bi započeli fazu 1, aplikacije generiraju prijedlog transakcije koji šalju svakom od potrebnih skupova peer-ova na odobrenje. Zatim, svaki od ovih odobravajućih peer-ova samostalno izvršava chaincode koristeći prijedlog transakcije za generiranje odgovora na prijedlog transakcije. Ne primjenjuje ovo ažuriranje na glavnu knjigu, već ga jednostavno potpisuje i vraća aplikaciji. Nakon što aplikacija dobije dovoljan broj potpisanih odgovora na prijedlog, prva faza tijeka transakcije je završena.

Faza 2 - redanje i pakiranje transakcija u blokove. Orderer je ključan za ovaj dio procesa, prima transakcije koje sadrže odobrene odgovore na prijedloge transakcija od aplikacija i reda transakcije u blokove.

Faza 3 - validacija i postavljanje. Posljednja faza tijeka transakcije uključuje distribuciju i provjeru valjanosti blokova od orderer-a do peer-ova, gdje mogu ažurirati glavnu knjigu. Točnije, kod svakog peer-a, svaka transakcija unutar bloka provjerava se kako bi se osiguralo da su je sve relevantne organizacije dosljedno odobrile prije nego što se ažurira glavna knjiga. Neuspjele transakcije zadržavaju se za reviziju, ali

ne mijenjaju glavne knjige. Faza 3 započinje orderer-om koji distribuira blokove svim peer-ovima povezanim s njim. Svim peer-ovima povezanim s orderer-om bit će poslana kopija novog bloka. Svaki će peer obrađivati ovaj blok neovisno, ali na potpuno isti način kao i svaki drugi peer na kanalu. Na taj ćemo način vidjeti da se glavna knjiga može održavati dosljednom. Također je vrijedno napomenuti da ne mora svaki peer biti povezan s naručiteljem - vršnjaci mogu kaskadno blokirati druge peer-ove koristeći *gossip* protokol **?**, koji ih također mogu samostalno obraditi.

3.3. Glavna knjiga

U Fabricu, glavna knjiga se sastoji od dva različita, iako povezana, dijela - globalno stanje i blockchain. Svaki od njih predstavlja skup činjenica o skupu poslovnih objekata.

Prvo, postoji globalno stanje - baza podataka koja ima trenutne vrijednosti skupa stanja glavne knjige. Globalno stanje olakšava programu za izravno pristup trenutnoj vrijednosti stanja, bez da ga mora izračunavati prelaskom kroz dnevnik stanja. Stanja glavne knjigu su, prema zadanim postavkama izražene kao parovi ključ-vrijednost. Globalno stanje se može često mijenjati, jer se stanja mogu stvoriti, ažurirati i izbrisati.

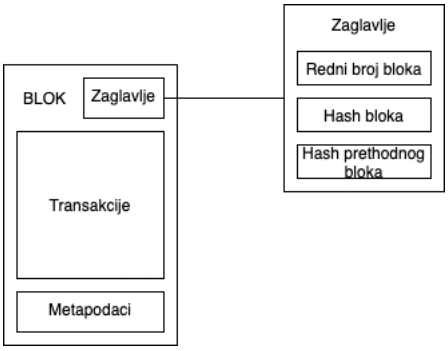
Drugo, tu je blockchain - dnevnik transakcija koji bilježi sve promjene koje su rezultirale trenutnim globalnim stanjem. Transakcije se prikupljaju unutar blokova koji se dodaju na blockchain - omogućujući pregled povijest promjena koje su rezultirale trenutnim globalnim stanjem. Blockchain struktura podataka je vrlo različita od globalnog stanja, jerjednom napisana, ne može se mijenjati.

Globalno stanje fizički je implementirano kao baza podataka, kako bi se omogućilo jednostavno i učinkovito pohranjivanje i pronalaženje stanja glavne knjige. Stanja glavne knjige mogu imati jednostavne ili složene vrijednosti, a kako bi se to prilagodilo, implementacija baze podataka može se razlikovati, što omogućuje učinkovitu implementaciju tih vrijednosti. Opcije za bazu podataka globalnog stanja trenutno uključuju LevelDB i CouchDB.

U mom primjeru koristio sam CouchDB. CouchDB je posebno prikladan izbor kada su stanja glavne knjige strukturirana kao JSON dokumenti jer CouchDB podržava bogate upite i ažuriranje bogatijih tipova podataka koji se često nalaze u poslovnim transakcijama. Implementacijski, CouchDB radi u odvojenom procesu operativnog sustava, ali postoji odnos 1:1 između peer-a i instance CouchDB baze podataka. Sve je to nevidljivo za pametni ugovor.

Da ponovimo, Blockchain u Fabricu strukturiran je kao sekvencijalni dnevnik me-

đusobno povezanih blokova, gdje svaki blok sadrži slijed transakcija, a svaka transakcija predstavlja upit ili ažuriranje globalnog stanja. Blockchain je implementiran kao datoteka.



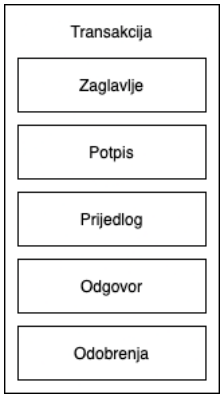
Slika 3.3: Blok

Struktura bloka prikazana na slici 3.3:

- Zaglavlje - sastoji se od tri polja:
  - Redni broj bloka - počinje od 0 (genesis blok) i povećava se za 1 sa svakim dodanim blokom u lanac.
  - Hash trenutnog bloka - hash svih transakcija koje trenutni blok sadrži.
  - Hash prethodnog bloka
- Podaci o transakcijama - ovaj dio sadrži popis transakcija poredanih redom. Zapisuje se kada je blok kreiran od orderer servisa.
- Metapodaci o bloku - ovaj dio sadrži certifikat i potpis tvorca bloka koji se koristi za provjeru bloka na mrežnim čvorovima. Postavljač bloka dodaje pokazatelj vrijedi/ne vrijedi za svaku transakciju u "bitmap"-u koja se također nalazi u metapodacima bloka, kao i hash kumulativnih ažuriranja stanja do i uključujući taj blok, kako bi se otkrilo račvanje (eng. fork) stanja. Ovaj dio nije ulaz u funkciju za izračunavanje hash-a bloka.

Struktura transakcije prikazana na slici 3.4:

- Zaglavlje - bitni metapodaci o transakciji, na primjer, naziv odgovarajućeg chaincode-a i njegova verzija.
- Potpis - kriptografski potpis, stvoren od strane klijentske aplikacije. Ovo se polje koristi za provjeru da detalji transakcije nisu kompromitirani, jer za njegovo generiranje potreban je privatni ključ aplikacije.
- Prijedlog - kodira ulazne parametre koje aplikacija šalje pametnom ugovoru koji kreira predloženo ažuriranje glavne knjige. Kada se izvrši pametni ugovor,



Slika 3.4: Transakcija

ovaj prijedlog pruža skup ulaznih parametara koji, u kombinaciji s trenutnim globalnim stanjem, određuju novo globalno stanje.

– Odgovor - bilježi globalno stanje prije i poslije, kao skup za čitanje i upisivanje (eng. Read-Write set, skr. RW-set). To je rezultat pametnog ugovora i ako se transakcija uspješno potvrdi, primijenit će se na glavnu knjigu i ažurirati globalno stanje.

– Odobrenja - ovo je popis potpisanih odgovora na transakcije svake potrebne organizacije dovoljan da zadovolji politiku odobrenja.

3.4. Poravnavanje transakcija

Mnogi distribuirani blockchaini, poput Ethereuma i Bitcoina, nemaju ograničen pristup, što znači da bilo koji čvor može sudjelovati u procesu konsenzusa, pri čemu se transakcije poredavaju i grupiraju u blokove. Zbog ove činjenice, ti se sustavi oslanjaju na algoritme vjerojatnog konsenzusa koji na kraju jamče dosljednost glavne knjige do visokog stupnja vjerojatnosti, ali koji su i dalje ranjivi na divergentne glavne knjige (poznati i kao "fork" glavne knjige), gdje različiti sudionici mreže imaju drugačiji pogled na prihvaćeni redoslijed transakcija.

U Hyperledger Fabric-u ima čvor pod nazivom *orderer* koji radi poredavanje transakcija, koji zajedno s drugim čvorovima orderer-a čini uslugu poredavanja. Budući da se Fabric dizajn oslanja na deterministične konsenzusne algoritme, bilo koji blok potvrđen od strane peer-a zasigurno će biti konačan i točan. Glavne knjige ne mogu divergirati na način na koji mogu u mnogim drugim distribuiranim blockchain mrežama bez ograničenog pristupa.

Osim promicanja konačnosti, odvajanje odobrenja izvođenja chaincodea (što se

događa na peer-ovima) od poredavanja transakcija daje prednosti Fabric-u u performansama i skalabilnosti i uklanjanje uskih grla koje se mogu pojaviti kada izvršenje i poredavanje provode isti čvorovi.

U svojoj implementaciji odlučio sam orderer čvor staviti u zasebnu orderer organizaciju . Koristim samo jedan orderer čvor orderer.unzig.hr radi jednostavnije demonstracije Fabric radnog okvira. Za bolje djelovanje Fabric mreže poželjno je imati 3 ili 5 ordering čvora.

3.5. Chaincode

Chaincode je program napisan u programskom jeziku Go, Node.js ili Java koji implementira propisano sučelje. Chaincode radi u zaštićenom Dockerovom kontejneru izoliranom od procesa odobravajućeg peer-a. Chaincode inicijalizira stanje glavne knjige i upravlja njime putem transakcija koje prima od aplikacija.

Chaincode obično obrađuje poslovnu logiku s kojom su se složili članovi mreže, pa se može smatrati "pametnim ugovorom". Ažuriranja glavne knjige stvorena chaincode-om opsega su isključivo na taj chaincode i ne može im se izravno pristupiti drugim chaincode-om. Međutim, unutar iste mreže, s obzirom na odgovarajuće dopuštenje, chaincode može pozvati drugi chaincode za pristup svom stanju.

Životni ciklus chaincode-a u Fabric mreži zahtjeva da se ogranizacije slože u parametrima koji definiraju chaincode, kao što su ime, verzija i politiku odobravanja. Članovi kanala dolaze do sporazuma kroz sljedeća četiri koraka, ali ne moraju sve organizacije napraviti svaki korak:

1. Pakiranje chaincode-a - ovaj korak može napraviti jedna, više ili sve organizacije u kanalu. Naredba koja se koristi za ovaj korak je: *peer lifecycle chaincode package*

2. Instaliranje chaincode-a na peer - svaka organizacija koja će koristiti chaincode kako bi odobrila transakciju ili zatražila podatke s glavne knjige mora izvršiti ovaj korak. Naredba: *peer lifecycle chaincode install*

3. Odobrenje definicije chaincode-a za organizaciju - svaka organizacija koja će koristiti chaincode mora izvršiti ovaj korak. Definiciju chaincode-a mora odobriti dovoljan broj organizacija kako bi se zadovoljile politike životnog vijeka kanala (većina, prema zadanim postavkama) prije nego što se chaincode može pokrenuti na kanalu. Naredba: *peer lifecycle chaincode approveformyorg*

4. Postavljanje definicije chaincode-a na kanal - transakciju postavljanja mora predati jedna organizacija nakon što je potreban broj organizacija na kanalu odobrilo chaincode. Podnositelj zahtjeva prvo prikuplja odobrenja od dovoljnog broja organizacija, a zatim predaje transakciju kako bi postavio definiciju chaincode-a na kanal. Naredba: *peer lifecycle chaincode commit*

U ovoj implementaciji koristio sam dva chaincode-a. Chaincode za ERC-20 token koji predstavlja bodove za menzu i chaincode za ERC-721 token koji predstavlja rektorovu nagradu. Pisani su u JavaScript programskom jeziku.

3.6. Kanal

Kanal je primarni komunikacijski mehanizam kojim članovi konzorcija mogu međusobno komunicirati. U mreži mogu biti više kanala. Kanali su korisni jer pružaju mehanizam za privatne komunikacije i privatne podatke između članova konzorcija. Kanali pružaju privatnost od drugih kanala, ali i od mreže. Fabric je moćan u tom pogledu, jer organizacijama omogućuje dijeljenje infrastrukture i čuvaju je privatnom u isto vrijeme. Ovdje nema kontradikcije - različite konzorcije unutar mreže imat će potrebu za različitim informacijama i procesima da se na odgovarajući način dijele, a kanali pružaju učinkovit mehanizam za to. Kanali pružaju učinkovito dijeljenje infrastrukture uz zadržavanje privatnosti podataka i komunikacije.

3.7. Klijent

Klijentska aplikacija se sastoji od dva dijela,"prednji" (eng. *frontend*) - sučelje i "stražnji" (eng. *backend*) - poslužitelj. Ovakva struktura je standardna u aplikacijama koje koriste Hyperledger Fabric mrežu. Omogućava odvajanje direktne komunikacije između aplikacije i mreže te aplikacije i klijenta. Prednost tome je jednostavnije korištenje i programiranje aplikacije.

3.7.1. Sučelje

"Prednji" dio aplikacije ili popularno nazvano *frontend*, programirano je koristeći radni okvir Angular. Angular je radni okvir otvorenog koda i napisan je u programskom jeziku TypeScript od strane Angular tima u Google-u. Ovaj radni okvir, koji se koristi za izradu web aplikacija, uz korištenje Angular Material biblioteke čini programiranje frontenda relativno jednostavnim.

3.7.2. Backend

"Stražnji" dio aplikacije ili *Backend*, programirao sam koristeći TypeScript programski jezik i Node.js *runtime* okruženje. Ovaj dio aplikacije izveden je uz REST načela **?** programiranja web poslužitelja. Uz Node.js koristio sam i radni okvir Express.js koji olakšava i poboljšava programiranje aplikacija na poslužiteljskoj strani.

U "običnim" web aplikacijama, ovaj dio bi komunicirao s "običnom" bazom podataka, na primjer MySQL, PostgreSQL i slično. U ovom primjeru komunicira s Fabric blockchain mrežom, koja zapravo služi kao baza podataka, raspodijeljena baza podataka do koje pristup imaju, i svi podaci su jednaki, sve organizacije koje su članovi te mreže ili kanala unutar te mreže.

Tijek aplikacije na primjeru stvaranja bodova za menzu:

1. Klijent otvori aplikaciju u svom internetskom poslužitelju te napravi zahtjev za stvaranje novih bodova za menzu.

2. Taj zahtjev preko HTTP protokola stiže na Node.js poslužitelj.

3. Poslužitelj obradi zahtjev i ako je zahtjev u redu, šalje zahtjev dalje prema blockchain mreži, točnije poziva metodu "Mint" erc20 chaincode-a.

4. Chaincode obrađuje zahtjev i provjerava identitet pošiljatelja. Ako je sve u redu, stvara nove bodove i dodaje ih u ukupan zbroj bodova u mreži i dodaje ih na račun stvaratelja.

5. Nakon što server primi poruku od mreže da su bodovi uspješno stvoreni, vraća odgovor klijentu koji na svom ekranu vidi nove bodove.

4. Zaključak

Blockchain je tehnologija koja sasvim sigurno ostavlja veliki utjecaj u četvrtoj industrijskoj revoluciji. Predstavlja novi način na koji razmišljamo o djeljenju podataka. Donosi rješenja za probleme koje se dugo vremena nije moglo riješiti. To je naročito zainteresiralo poslovni svijet. Stoga su nastale razne implementacije te tehnologije koje se natječu u rješavanju poslovnih problema, a u ovom radu smo spomenuli Quorum, Corda i Hyperledger Fabric.

Hyperledger Fabric je radni okvir za implementaciju blockchaina s ograničenim pravom pristupa koji je korišten za implementacijski dio rada. Fabric je najmodularniji i pokazuje najbolje performanse od svih prethodno spomenutih izvedbi blockchaina s ograničenim pravom pristupa.

Na jednostavnom primjeru bodova u menzi i rekotorovih nagrada između FER-a i FSB-a pokazalo se i analiziralo kako izgleda konkretan primjer korištenja Hyperledger Fabric platforme. Sama blockchain mreža se koristi kao raspodijeljena baza podataka koja se dijeli između FER-a i FSB-a, a klijentska aplikacija je inicijator transakcija kojima se dolazi do podataka na mreži ili ih se ažurira.

LITERATURA

Christopher D. Clack, Vikram A. Bakshi, i Lee Braine. Smart contract templates: foundations, design landscape and research directions. *CoRR*, abs/1608.00771, 2016. URL http://arxiv.org/abs/1608.00771.

Michael Downes. *Short Math Guide for LaTeX*. American Mathematical Society, 2002. URL ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf.

Ahmed Afif Monrat, Olov Schelén, i Karl Andersson. Performance evaluation of permissioned blockchain platforms. U *2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, stranice 1–8, 2020. doi: 10.1109/CSDE50874.2020.9411380.

Diego Ongaro i John Ousterhout. In search of an understandable consensus algorithm. U *2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14)*, stranice 305–319, 2014.

Weiqin Zou, David Lo, Pavneet Singh Kochhar, Xuan-Bach D. Le, Xin Xia, Yang Feng, Zhenyu Chen, i Baowen Xu. Smart contract development: Challenges and opportunities. *IEEE Transactions on Software Engineering*, stranice 1–1, 2019. doi: 10.1109/TSE.2019.2942301.

Ulančani blokovi i raspodijeljene glavne knjige s ograničenim pravom pristupa i pametnim ugovorima

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: ključne riječi, odvojene zarezima.

Permissioned blockchain and distributed ledger technologies with smart contracts

Abstract

Abstract.

Keywords: Keywords.