Usluga grupne validacije Ethereum mreže korištenjem tehnologija za skaliranje mreža lanca blokova

Buratović, Jakov

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 3165

USLUGA GRUPNE VALIDACIJE ETHEREUM MREŽE KORIŠTENJEM TEHNOLOGIJA ZA SKALIRANJE MREŽA LANCA BLOKOVA

Jakov Buratović

Zagreb, veljača 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 3165

USLUGA GRUPNE VALIDACIJE ETHEREUM MREŽE KORIŠTENJEM TEHNOLOGIJA ZA SKALIRANJE MREŽA LANCA BLOKOVA

Jakov Buratović

Zagreb, veljača 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 2. listopada 2023.

DIPLOMSKI ZADATAK br. 3165

Pristupnik:	**Jakov Buratović (0036504283)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	izv. prof. dr. sc. Igor Čavrak

Zadatak: **Usluga grupne validacije Ethereum mreže korištenjem tehnologija za skaliranje mreža lanca blokova**

Opis zadatka:

Ethereum mreža, temeljena na distribuiranoj glavnoj knjizi, nije pogodna za obavljanje velikog broja transakcija u kratkom vremenu. Cijena transakcija raste zbog ograničene veličine bloka, što rezultira natjecanjem za mjesto u bloku te, posljedično, povećanjem cijene koju je korisnik voljan platiti za izvršavanje svoje transakcije. Za sudjelovanje u validaciji mreže Ethereum potrebno je imati tehničko znanje i 32 Ethera, što nije dostupno većini korisnika. Istodobno, validacijom mreže Ethereum validatori zarađuju stabilne prihode na svoj portfelj Ethera, što validaciju čini financijski privlačnom. Zaobilaženje navedenih ograničenja mehanizma validacije Ethereum mreže lanca blokova bilo bi moguće izradom sustava grupiranja uloga različitih korisnika, temeljenog na tehnologijama za skaliranje lanaca blokova koji grupiraju transakcije korisnika. Cilj ovog rada ostvarenje je mehanizma validacije Ethereum mreže dostupnijih širem broju korisnika, u cilju poboljšanja sigurnosti i decentraliziranosti mreže te omogućavanja njihova dodatnog izvora prihoda kao nagrade za pružanje usluge validacije mreže. Potrebno je osmisliti sustav, temeljen na tehnologijama skaliranja Ethereum mreže, za grupiranje uloga više korisnika u cilju smanjenja troškova i zahtjeva na resurse pojedinog korisnika mreže. Na osnovu idejnog rješenja potrebno je predložiti arhitekturu sustava te implementirati programsko rješenje u obliku mrežne usluge kojim se demonstriraju mogućnosti i ograničenja takvog sustava. Na kraju rada potrebno je provesti ispitivanje performanci i troškova razvijenog sustava, te ih usporediti s performancama i troškovima tradicionalne metode validacije Ethereum mreže.

Rok za predaju rada: 9. veljače 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 3165

USLUGA GRUPNE VALIDACIJE ETHEREUM MREŽE KORIŠTENJEM TEHNOLOGIJA ZA SKALIRANJE MREŽA LANCA BLOKOVA

Jakov Buratović

Zagreb, Veljača, 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 2. listopada 2023.

DIPLOMSKI ZADATAK br. 3165

Pristupnik:	**Jakov Buratović (0036504283)**
Studij:	Računarstvo
Profil:	Programsko inženjerstvo i informacijski sustavi
Mentor:	izv. prof. dr. sc. Igor Čavrak

Zadatak: **Usluga grupne validacije Ethereum mreže korištenjem tehnologija za skaliranje mreža lanca blokova**

Opis zadatka:

Ethereum mreža, temeljena na distribuiranoj glavnoj knjizi, nije pogodna za obavljanje velikog broja transakcija u kratkom vremenu. Cijena transakcija raste zbog ograničene veličine bloka, što rezultira natjecanjem za mjesto u bloku te, posljedično, povećanjem cijene koju je korisnik voljan platiti za izvršavanje svoje transakcije. Za sudjelovanje u validaciji mreže Ethereum potrebno je imati tehničko znanje i 32 Ethera, što nije dostupno većini korisnika. Istodobno, validacijom mreže Ethereum validatori zarađuju stabilne prihode na svoj portfelj Ethera, što validaciju čini financijski privlačnom. Zaobilaženje navedenih ograničenja mehanizma validacije Ethereum mreže lanca blokova bilo bi moguće izradom sustava grupiranja uloga različitih korisnika, temeljenog na tehnologijama za skaliranje lanaca blokova koji grupiraju transakcije korisnika. Cilj ovog rada ostvarenje je mehanizma validacije Ethereum mreže dostupnijih širem broju korisnika, u cilju poboljšanja sigurnosti i decentraliziranosti mreže te omogućavanja njihova dodatnog izvora prihoda kao nagrade za pružanje usluge validacije mreže. Potrebno je osmisliti sustav, temeljen na tehnologijama skaliranja Ethereum mreže, za grupiranje uloga više korisnika u cilju smanjenja troškova i zahtjeva na resurse pojedinog korisnika mreže. Na osnovu idejnog rješenja potrebno je predložiti arhitekturu sustava te implementirati programsko rješenje u obliku mrežne usluge kojim se demonstriraju mogućnosti i ograničenja takvog sustava. Na kraju rada potrebno je provesti ispitivanje performanci i troškova razvijenog sustava, te ih usporediti s performancama i troškovima tradicionalne metode validacije Ethereum mreže.

Rok za predaju rada: 9. veljače 2024.

Zahvaljujem se mentoru doc. dr. sc. Igoru Čavraku na podršci i pomoći pri izradi ovog rada. Također se zahvaljujem i svojoj obitelji koja me usmjeravala i pomagala tokom diplomskog studija.

Sadržaj

1. Uvod

Većina modernih implementacija mreža lanca blokova (engl. *blockchain*) ne koristi dokaz o radu (engl. *Proof of Work*) algoritam za postizanje konsenzusa. Tako je i *Ethereum* mreža prešla s *Proof of Work* na dokaz o zalogu (engl. *Proof of Stake*) algoritam za postizanje konsenzusa kako bi se omogućio veći stupanj decentralizacije i zaustavila "kartelizacija rudara" koji su se postajali sve veća prijetnja cjelokupnoj mreži. *Proof of Stake* ne rješava izravno problem kartelizacije, ali omogućuje izgradnju složenijih aplikacija koji mogu izravnati interese ulagača i korisnika mreže.

U ovom radu je prikazano nekoliko izazova za korisnike trenutnog izdanja mreže *Ethereum*. Jedan od izazova je skaliranje broja i složenosti transakcija. Izrađen program za ovaj rad omogućuje široj publici da sudjeluje u decentralizaciji i osiguravanju mreže *Ethereum*.

U prvom poglavlju, čitatelj će se podsjetiti na osnovne značajke *Blockchaina* i algoritama za postizanje konsenzusa. Nakon toga predstavljen je pojam *Blockhain trilema* koji prikazuje temeljne izazove za svaku *blockchain* mrežu. Glavni dio rada zalazi dublje u *Proof od Stake* i različite načine na koji korisnici mogu sudjelovati u decentralizaciji i sigurnosti mreža.

Uz algoritam konsenzusa, važno je objasniti i tehnologije za skaliranje koje su trenutno u uporabi, te konačno napraviti presjek između tehnologija za skaliranje i validacije mreže za što je izražen i prigodan projekt u sklopu ovog diplomskog rada. Program je detaljno opisan u posljednjem dijelu rada, te su objašnjena moguća proširenja i izmjene koje bi ovaj program napravile primjenjive na više različitih tehnologija za skaliranje.

2. Osnovno o *blockchainu* i *Ethereumu*

2.1. Blockchain

Blockchain tehnologija je digitalni koncept koji omogućava sigurnu i transparentnu razmjenu podataka unutar decentralizirane mreže. Karakterizira ga niz blokova podataka koji su kriptografski povezani i distribuirani među svim sudionicima mreže. Svaki blok sadrži jedinstveni skup transakcija, a jednom kada se podaci upišu u blok, postaju nepromjenjivi, što osigurava visoku razinu sigurnosti i povjerenja među korisnicima.

Dok su blockchain i distribuirana knjiga (*DLT*) često korišteni izmjenično, važno je razumjeti njihove razlike. *DLT* se odnosi na široku kategoriju tehnologija koje omogućuju distribuiranu obradu i pohranu podataka preko više lokacija ili entiteta. Blockchain je specifična vrsta *DLT*-a koja koristi lanac blokova za pohranu podataka u kontinuiranom nizu. Ključna razlika leži u strukturi i načinu pohrane podataka; dok blockchain strogo slijedi linearan, kronološki redoslijed povezivanja podataka, *DLT* može koristiti različite strukture za postizanje distribuirane konsenzusne mreže.

Dvije najpopularnije implementacije blockchain tehnologije su *Bitcoin* i *Ethereum*. *Bitcoin* je isključivo zamišljen kao digitalna, odnosno kriptografska valuta, dok *Ethereum* pruža mnogo više mogućnosti za razvoj i korištenje.

2.2. *Ethereum*

Cilj *Ethereuma* jest biti alternativan protokol za izgradnju decentraliziranih aplikacija, pružajući drugačiji pristup koji bi trebao biti vrlo koristan velikom skupu decentraliziranih aplikacija, s posebnim naglaskom na situacije gdje kratko vrijeme razvoja i visok stupanj sigurnosti za male i rijetko korištene aplikacije mora biti ispunjen kako bi one

međusobno komunicirale[1].

Često se opisuje kao svjetsko računalo (engl. *world computer*). Iz perspektive računalne znanosti, *Ethereum* je deterministički i praktično neograničeni automat stanja (engl. *state machine*) koji se sastoji od globalno dostupnog stanja i virtualnog stroja koji primjenjuje promjene na to stanje.

U praksi, *Ethereum* je globalna decentralizirana infrastruktura za izvođenje programa koji se zovu pametni ugovori (engl. *smart contracts*). Koristi *blockchain* za usklađivanje i pohranu promjena stanja, uz kripto valutu *ether*, za ograničavanje potrošnje resusursa[2].

2.3. Blockchain trilema

Termin *blockchain* trileme, odnosno trileme skalabilnosti je uveo tvorac *Ethereuma*, Vitalik Buterin u svom članku. Trilema skalabilnosti kaže da postoje tri svojstva koja *blockchain* treba posjedovati, držeći se jednostavnih principa, moguće je imati samo dva od ta tri svojstva. Svojstva su:

1. **Skalabilnost**: *blockchain* može procesuirati više transakcija nego običan čvor mreže (npr. prijenosno računalo) može provjeriti

2. **Decentralizacija**: *blockchain* može raditi bez oslanjanja na povjerenje male skupine velikih čvorova u mreži. Ovo je tipično interpretirano tako da ne smije postojati nikakvo povjerenje u set čvorova u koji se ne može bilo tko pridružiti sa svojim računalom

3. **Sigurnost**: *blockchain* se može oduprijeti velikom postotku čvorova koji ga pokušavaju napasti (idealno 50%)[3]

Tradicionalno, *blockchain* mreže suočavaju se s izazovom da maksimiziraju samo dva od ova tri aspekta istovremeno, često na štetu trećeg. Na primjer, *blockchain* koji prioritizira sigurnost i decentralizaciju može patiti od problema sa skalabilnošću, što se vidi u ranim verzijama Bitcoina i *Ethereuma*. S druge strane, povećanje skalabilnosti kroz centralizirane rješenja može ugroziti sigurnost ili decentralizaciju.

Rješavanje *blockchain* trileme je centralno za razvoj i usvajanje *blockchain* tehnolo-

gije. Različiti pristupi, uključujući slojevita rješenja (engl *layered solutions*), razlamanje (engl. *sharding*), i prelazak s *Proof of Work* (PoW) na *Proof of Stake* (PoS) mehanizme konsenzusa, pokušavaju adresirati ove izazove, svaki s vlastitim skupom prednosti i kompromisa. *Ethereumov* prijelaz na PoS, na primjer, predstavlja značajan korak prema rješavanju trileme, obećavajući veću skalabilnost (ne izravno) i energetsku efikasnost uz očuvanje sigurnosti i decentralizacije.

U zaključku, *blockchain* trilema ostaje ključni izazov koji oblikuje evoluciju *blockchain* tehnologije. Njeno rješavanje zahtijeva inovativne pristupe i kontinuirani razvoj, s ciljem stvaranja robustnih, sigurnih i široko skalabilnih *blockchain* mreža koje mogu podržati širok spektar aplikacija u digitalnom dobu.

Daleko najozbiljniji problem *Ethereuma* u ovom trenutku je skalabilnost- *Ethereum* podržava svega 15 transakcija u sekundi. Ovaj problem se pojavio prvi put u praksi porastom popularnosti inicijalnih ponuda tokena (engl. *Initial Coin Offering (ICO)*).

Mnogi postavljaju pitanje zašto jednostavno ne povećati bolokove *Ethereuma* ili se smanjiti period između dva bloka. Jedan od odgovora je rizika centralizacije. Vitalik smatra da svaka osoba kod kuće treba biti u mogućnosti pokretati čvor decentralizirane mreže. Pojednostavljeno, ukoliko se veličina blokova udvostruči, udvostručit će se i zahtjevi za kapacitetom za pohranu. Osim pohrane, povećat će se i zahtjevi na mrežnu infrastrukturu za propagaciju informacija između čvorova. Novi, veći blokovi bi se vrlo brzo ponovno popunili, a pokretanje čvorova bi sada bilo samo posao organizacija koje imaju pristup kvalitetnom i skupocjenom sklopovlju na modernoj mrežnoj infrastrukturi [4].

Predložena rješenja uključuju razlamanje (engl. *sharding*), bočne lance (engl. *side chains*) i kompresijske slojeve (engl. *rollups*)[5].

Danas se najviše koriste kompresijski slojevi. Njihova temeljna ideja je "prebaciti posao negdje drugdje". Za razliku od bočnih slojeva koji također prebacuju posao na novu mrežu ali imaju svoj mehanizam konsenzus, kompresijski slojevi se potpuno oslanjaju na sigurnost korijenskog lanca. Kompresijski slojevi zapravo "žive" u obliku pametnog ugovora na korijenskom lancu. Komputacija se događa negdje drugdje, jeftinije, ali postoje kriptografski i ekonomski mehanizmi koji osiguravaju da se prilikom spremanja stanja

na korjenski lanac ne upiše pogrešno stanje[6].

Kompresijski slojevi imaju svoj sustav naplaćivanja transakcija korisnicima koji je inspiriran *Ethereumovim* sustavom jedinica *gas*. Prisjetimo se da je *gas* osnovna mrežna jedinica za trošak. Isključivo se može platiti nativnim tokenom. U slučaju *Ethereuma*, to je *ether*. Cijena je postavljena transakcijama i validatori slobodno mogu ignorirati transakcije za čije je izvršavanje priloženo malo tokena[7]. Obzirom na konačnu veličinu bloka, validatorima je u interesu proizvesti blok s što više priloženog tokena kako bi njihova nagrada bila veća.

Kod kompresijskih slojeva, komputacija se ne obavlja nužno na jednak način. To znači da određene operacije koje su resursno "skupe" na korjenskom lancu, mogu biti prilično jeftine na kompresijskom sloju. I obrnuto isto može vrijediti. Dodatno, kompresijski sloj mora redovno postaviti svoje stanje na pametni ugovor u kojem živi na korjenskom lancu. Ta transkacija obično košta mnogo jer se puno podataka upisuje[8]. Trenutno se radi na poboljšanju *Ethereum* mreže koje optimizira prostor za pohranu specifično za kompresijske slojeve. Nadogradnja ima kodni naziv EIP-4844.

2.3.1. Monolitna i modularna arhitektura

U kontekstu *blockchain* tehnologije, razlikovanje između monolitnih i modularnih blockchaina odnosi se na pristup dizajnu i arhitekturi mreže. Svaki pristup ima svoje specifične karakteristike, prednosti i nedostatke, te je usmjeren na rješavanje određenih izazova, uključujući one povezane s *blockchain* trilemom.

Monolitni Blockchain

Monolitni *blockchain* predstavlja tradicionalni pristup dizajnu blockchaina gdje se sve ključne funkcije izvršavaju unutar jednog sloja. To uključuje obradu transakcija, konsenzusni mehanizam, i finalizaciju stanja. *Bitcoin* i *Ethereum* (u izvornom obliku) su primjeri monolitnih blockchaina.

Karakteristike:

- **Integracija funkcija**: Obrada transakcija, postizanje konsenzusa, i pohrana podataka odvijaju se unutar jedne, neodvojive platforme.

- **Jendostavnost**: Jednostavnija arhitektura olakšava razumijevanje i održavanje.
- **Sigurnost**: Visoka razina sigurnosti zbog centraliziranog modela obrade i validacije.

Izazovi:

- **Skalabilnost**: Kako mreža raste, monolitni blockchaini često se suočavaju s ograničenjima u pogledu broja transakcija koje mogu efikasno obraditi.
- **Fleksibilnost**: Nadogradnje i promjene zahtijevaju široko usvajanje unutar mreže, što može biti spor i izazovan proces.

Modularni Blockchain

Modularni blockchaini predstavljaju evoluciju u dizajnu blockchaina, gdje se različite funkcije (kao što su obrada transakcija, konsenzus, i dostupnost podataka) odvajaju u različite module ili slojeve. Ovaj pristup omogućava specijalizaciju svakog sloja, poboljšavajući performanse, sigurnost, i skalabilnost.

Karakteristike:

- **Specijalizacija**: Svaki sloj može biti optimiziran za specifične funkcije, omogućavajući veću efikasnost i fleksibilnost.
- **Inovacija**: Modularna struktura omogućava brže i sigurnije uvođenje novih tehnologija i ažuriranja u određene dijelove blockchaina bez utjecaja na cjelokupni sustav.
- **Skalabilnost**: Veća skalabilnost može se postići optimizacijom pojedinačnih slojeva, kao što su slojevi obrade transakcija ili slojevi konsenzusa.

Izazovi:

- **Kompleksnost**: Veća složenost u dizajnu i koordinaciji između slojeva može dovesti do izazova u implementaciji i održavanju.
- **Interoperabilnost**: Potreba za učinkovitom komunikacijom i interoperabilnošću između slojeva može predstavljati tehničke izazove.

Izbor između monolitnog i modularnog pristupa ovisi o specifičnim zahtjevima i ciljevima projekta blockchaina. Monolitni blockchaini nude robustan i testiran pristup, idealan za aplikacije koje zahtijevaju visoku razinu sigurnosti i jednostavnost. S druge strane, modularni blockchaini nude veću fleksibilnost, skalabilnost, i mogućnost za inovacije, što ih čini privlačnima za složenije aplikacije i one koje teže rješavanju izazova povezanih s *blockchain* trilemom. Razvojem tehnologije, pokazuje se da je modularni pristup bolje prihvaćen.

2.4. *Ethereum* migracija mehanizma konsenzusa

Ethereum je izvorno koristio *Proof of Work* kao mehanizam konsenzusa, no 2022. godine je izvršena migracija na *Proof of Stake* radi povećanja sigurnosti, smanjenja utroška energije i postavljanja temelja za nove mehanizme skaliranja.

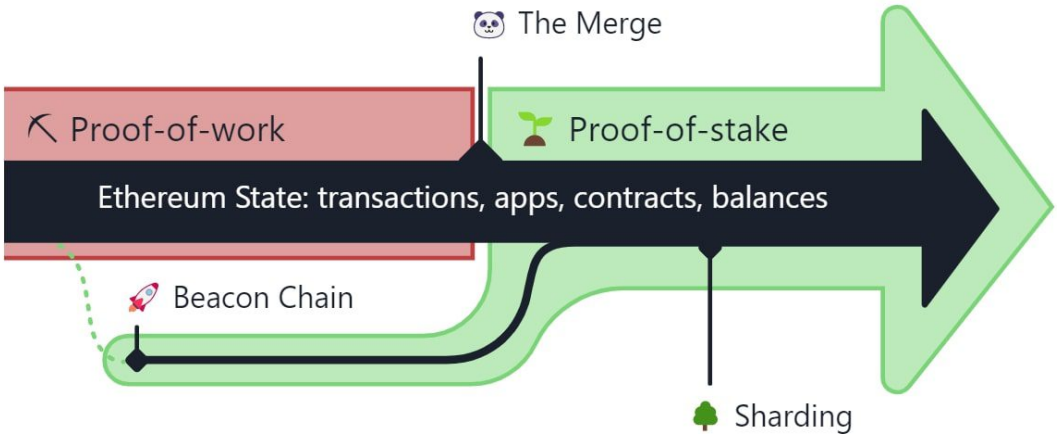
2.4.1. *Proof of Stake*

Proof of Stake je način provjere pri kojim validatori (tvorci blokova) postavljaju zalog koji ima određenu vrijednost u mrežu kako bi se taj zalog oduzeo u slučaju ponašanja koji nije u skladu s pravilima protokola. U *Ethereum Proof of Stake* implementaciji, validatori ulažu kapital u obliku *ethera* (ETH) u pametni ugovor koji živi na *Ethereumu*. Validator je onda odgovoran za provjeru da su novi blokovi koji propagiraju kroz mrežu ispravni. Povremeno i sami kreiraju novi blok koji onda ostali validatori provjeravaju[9]. Redoslijed odabira validatora za proizvodnju bloka je također definiran kroz algoritme u samom protokolu. Kako bi se osiguralo da mehanizam ne prioritizira isključivo validatore s najvećim zalogom i pritom centralizira sustav, postoje dodatni algoritmi koji se bave odabirom validatora zaduženog za proizvodnju idućeg bloka.

2.4.2. Proces prijelaza

Prijelaz *Ethereuma* s *Proof of Work* na *Proof of Stake* se nije dogodio odjednom. Proces zvan sljubljivanje (engl. *The Merge*) je krenuo lansiranjem lanca odašiljača (engl. *Beacon chain*) nakon čega je slijedio dulji period u kojem su se paralelno izvodila dva lanca, originalni *Ethereum* čije blokove proizvode rudari (engl. *miners*) algoritmom *Proof of Work* i novi lanac odašiljač s *Proof of Stake* algoritmom. U tom periodu, lanac odašiljač nije

postizao konsenzus oko stvarnih *Ethereum* transakcija. Umjesto toga, isključivo se bavio konsenzusom svog stanja potvrđujući aktivne validatore i njihovo stanje na računu. Nakon opsežnog i dugotrajnog testiranja, došlo je vrijeme da lanac odašiljač počne raditi i na konsezusu za transakcije sloja izvođenja (engl. *execution layer*). Taj događaj se dogodio 15. rujna 2022. godine. Sljubljivanje predstavlja službeni prijelaz *Ethereuma* na *Proof of Stake*, rudari nakon sljubljivanja nisu više zaslužni za proizvodnju blokova. Umjesto njih, validatori su preuzeli tu ulogu. Važno je napomenuti da se povijest transakcija i stanje nije izgubilo sljubljivanjem[10].

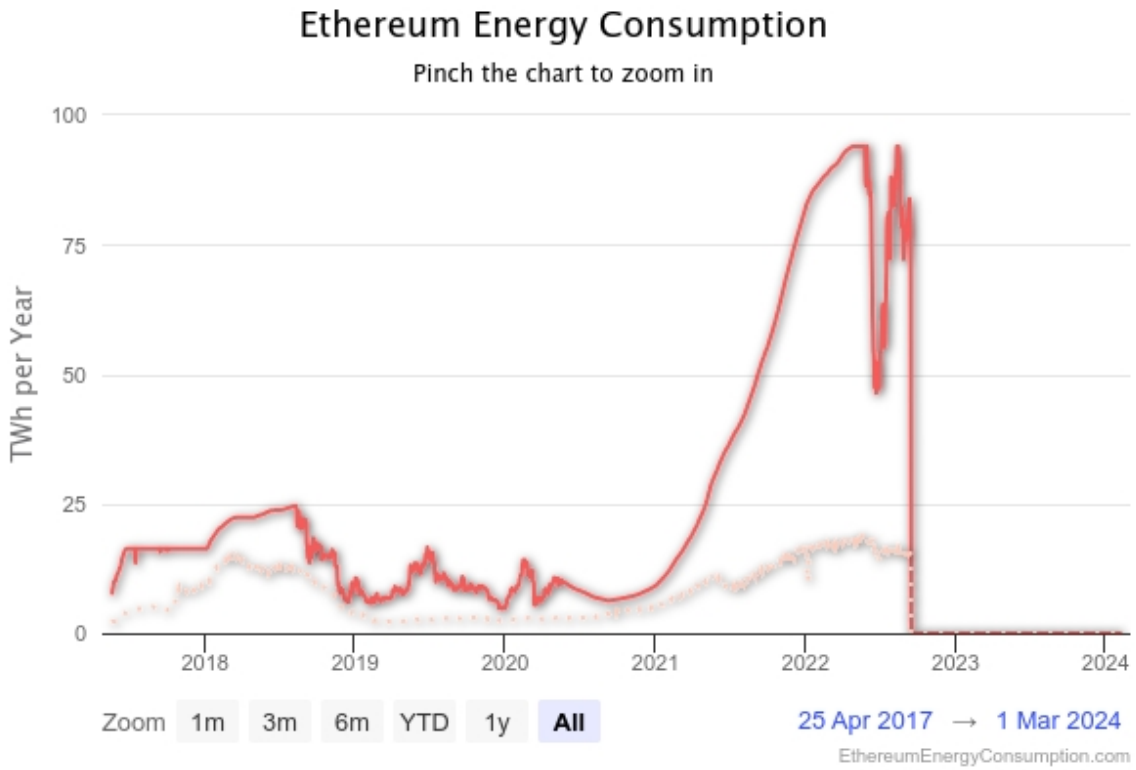


Slika 2.1. *Ethereum* sljubljivanje
[10]

Za korisnike mreže se nije ništa promijenilo. Sve decentraliziranje aplikacije su nastavile raditi bez prekida. Korisnička sučelja kao što su digitalni novčanici (engl. *wallet*) su također ostali u potpunosti kompatibilni s novim lancem.

Najveća prednost prijelaska dolazi u obliku smanjenja troškova za proizvodnju blokova. Kako se *Proof of Work* oslanja isključivo na pogađanje hasheva, potrebno je utrošiti energiju za izračun. Obzirom da su rudari, pojedinci ili udruge koji posjeduju računala za proizvodnju blokova financijski nagrađeni ako prvi dođu do rješenja, rudari su se natjecali između sebe. To natjecanje je uzrokovalo da manji broj udruga s velikom financijskom moći može priuštiti najsnažnije uređaje koje su držali u prostorima adekvatno opskrbljeni električnom energijom i opremljeni sustavima za hlađenje.

13. kolovoza 2022. godine, potrošnja električne energije *Ethereuma* je dostigla vrh, skaliranjem na godišnju potrošnu rezultirala je 93,975 TWh što je više nego potrošnja države Filipini[11]. Nakon prijelaska, potrošnja je pala za 99,95



Slika 2.2. Potrošnja energije prije i poslje sljubljivanja u TWh, preuzeto s `https://digiconomist.net/ethereum-energy-consumption`

2.4.3. Različite vrste pristupu validaciji za korisnike

Postoje tri glavna načina kojima korisnici *Ethereuma* mogu sudjelovati u validaciji mreže:

1. Samostalno ulaganje (engl. *Solo staking*)
2. Ulaganje kao servis (engl. *Staking as a service*)
3. Grupno ulaganje (engl. *Pooled staking*)

Svaki od pristupa ima određene prednosti i mane. Nije moguće odrediti koji je najbolji pristup jer to ovisi o samom korisniku i njegovim sposobnostima i željama.

Samostalno ulaganje

Samostalno ulaganje je izvorni i izravni način sudjelovanja u validaciji *Ethereuma*. Prednosti su:

- Maksimalne nagrade izravno od protokola (nema posrednika)
- Nagrade od uspješno proizvedenih blokova i provjera tuđih blokova

- Nagrade od transakcijske provizije

Izazovi su:

- Rizik gubljenja založenog *ethera* zbog odsutnog validatora
- Gubljenje založenog *ethera* zbog neispravnog ili zlonamjernog ponašanja koji nije u skladu s pravilima protokola

Zahtjevi su:

- Depozit od 32 ETH
- Održavanje sklopovlja koji izvršava izvršni i konsenzusni sloj (engl. *execution and consensus layer*)
  - Zahtjeva tehničko znanje
  - Zahtjeva fizičko sklopovlje

Ulaganje kao servis

Ovaj način je vrlo sličan samostalnom ulaganju. Dvije su osnovne razlike:

1. Sklopovlje i čvorove održava treća strana
2. Za taj servis se naplaćuje provizija što umanjuje nagrade

Dodatan rizik je oslanjanje na treću stranu koja ima kontrolu nad čvorovima. Privatni ključevi su najčešće i dalje pospremljeni kod korisnika.

Grupno ulaganje

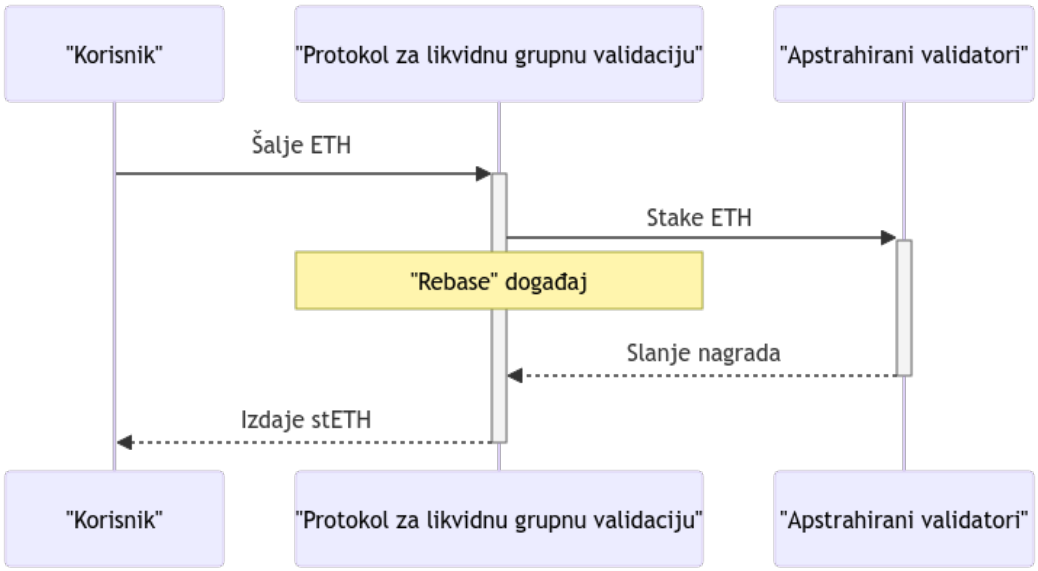
Postoji više vrsta grupnog ulaganja, no najveća prednost ovog pristupa jest da korisnik ne treba imati svih 32 ETH da bi sudjelovao već se može udružiti s drugim korisnicima u grupe kako bi ujedinili svoje zaloge. Ovisno o udjelu u ukupnom zalogu, korisnik dobiva isti udio od ukupne nagrade[12].

Likvidno grupno ulaganje Najpopularniji oblik grupnog ulaganja jest likvidno grupno ulaganje (engl. *liquid staking*). Svoju popularnost je stigao zbog dvije vrlo važne zna-

čajke:

1. Moguće je sudjelovati s proizvoljno malim ili velikim zalogom
2. Nakon ulaganja korisnik i dalje ima token (potvrdu o zalogu) koji može slobodno koristiti

Upravo takav protokol će se koristiti u sklopu ovo rada zbog niske barijere za ulaz. Lido je decentralizirani financijski servis koji omogućava korisnicima da sudjeluju u stakingu kriptovaluta, kao što je *Ethereum* (ETH), bez potrebe za zaključavanjem sredstava ili upravljanjem staking infrastrukturom. Ključna značajka Lida je tzv. "liquid staking", koji korisnicima omogućava da zadrže likvidnost svojih sredstava čak i dok su založena (staked). Korisnik u bilo kojem trenutku može zatražiti podizadnje svog zaloga, čime se inicira proces u dvije faze. Prva faza je slanje zahtjeva za podizanjem zaloga i slanje likvidnog tokena protokolu. Nakon što se zahtjev obradi (duljina trajanja obrade ovisi o aktivnosti mreže), korisnik može preuzeti svoje *ethere* iz protokola. Primjerice kod *solo staking* pristupa, korisnik bi trebao maknuti potpuni zalog s validatora da bi to napravio.



Slika 2.3. Dijagram korisničke interakcije s protokolom za likvidno grupno ulaganje

Koraci iz dijagrama:

1. Depozit: Korisnik šalje svoje ETH tokene u pametni ugovor za zaloge. Umjesto da izravno založi ETH na *Ethereum* i pokrene validator, korisnici založe svoje tokene kroz likvidni grupni protokol za validaciju.

2. Izdavanje stETH Tokena: U zamjenu za založene ETH tokene, korisnici odmah dobivaju stETH tokene, koji predstavljaju potvrdu za založeni iznos.

U slučaju *Lido* protokola, odnos između stETH i ETH tokena jest 1:1. stETH nastaje samo i samo ako je ispunjen jedan od dva uvjeta:

1. Korisnik založi ETH

2. Validator koji je dio Lido skupa validatora proizvede nagradu

U drugom slučaju, nagrada se ravnomjerno rasporedi svim vlasnicima stETH tokena na način da se njihovo stanje na računu poveća. To se zove ponovno baziranje (engl. *rebase*) događaj.

U Lido protokolu postoji i drugi token, omotani stETH (wstETH). Razlog postojanja tog tokena je kompatibilnost s ostalim protokolima decentraliziranih financija. wstETH za razliku od stETH tokena ne radi ponovno baziranje već mu vrijednost raste shodno. Korisnik neće vidjeti razliku u broju tokena na svom računu, ali ako odmota wstETH u stETH dobit će veći broj stETH tokena nego prije nagrada. To znači da omjer između ETH i wstETH tokena nije 1:1 kao što je kod ETH i stETH tokena. Zbog svoje jednostavnosti i potražnje, trenutno je samo wstETH dostupan na ostalim mrežama kao što su *Arbitrum*, *Optimism* i *Polygon PoS*.

Likvidnost: stETH tokeni su "likvidni", što znači da ih korisnici mogu trgovati, koristiti u protokolima decentraliziranih financija (DeFi) protokolima za posudbe, osiguranje, ili kao jamstvo za druge financijske proizvode unutar *Ethereum* ekosustava. Ovo omogućava korisnicima da zadrže pristup svojim sredstvima i koriste ih za druge investicijske prilike dok njihov *ether* pomaže validiranju i zarađuje nagrade za svoj zalog.

Decentralizacija i Sigurnost: Lido se oslanja na skup operatora čvorova koja distribuira ETH preko više validatora. Ovo pomaže u smanjenju rizika i povećanju sigurnosti sredstava, dok decentralizirani model upravljanja Lido protokola osigurava da su odluke o projektu vođene zajednicom, a ne jednim entitetom koji bi imao potpunu kontrolu[13].

Prednosti Lido protokola

Duboka likvidnost: Korisnici Lido protokola su izloženi najdubljoj likvidnosti. U

veljači 2024. godine je likvidnost različitih tokena uparenih s stETH tokenom kumulativno iznosila oko 525 milijuna američkih dolara.

Fleksibilnost: Korisnici mogu ulagati stETH tokene u druge DeFi aplikacije za dodatne prihode. Zbog vrlo likvidnog tržišta oko Lido stETH tokena, stETH token je prisutan kao osnovni token u većini drugih DeFi protokola.

Likvidnost: Lido protokol nudi jednostavan način za sudjelovanje u validaciji bez tehničkih složenosti ili potrebe za minimalnim iznosom zaloga.

Decentralizirano upravljanje: Lido je upravljan zajednicom putem LDO tokena, omogućavajući korisnicima da sudjeluju u donošenju odluka o budućem razvoju projekta glasanjem.

3. Grupna validacija *Ethereuma* s Polygon PoS bočne mreže

Kao što je već objašnjeno u prethodnom dijelu, validacija *Ethereum* mreže je dobra za ekosustav i sudionici su za to prigodno nagrađeni. Tehnologije za skaliranje omogućavaju jeftiniji pristup decentraliziranim aplikacijama kao što su primjerice decentralizirane mjenjačnice, tržišta za posuđivanje digitalnih tokena i slično. Navedeni primjeri decentraliziranih aplikacija se lagano mogu replicirati na bilo kojoj *blockchain* mreži jer ne ovise niti o jednoj drugoj, odnosno mogu postojati izolirano. Obzirom da se validacija *Ethereum* mreže odvija isključivo na *Ethereum* mreži, omogućavanje korisnicima s udaljenih lanaca kao što su kompresijski slojevi ili bočni lanci nije trivijalno. Potrebno je razdvojiti decentraliziranu aplikaciju na komponente koje se nalaze na udaljenom lancu (lanac dijete u nastavku) i na matičnom lancu (lanac korijen u nastavku). Kako bi te komponente komunicirale na siguran i decentraliziran način, komunikacija se smije provoditi isključivo putem komunikacijskih mostova zaslužnih za sinkronizaciju stanja lanca djeteta i korijenskoga lanca.

U svrhu ovog rada, odabran je Polygon PoS bočni lanac, no princip rada je sličan i za ostale bočne lance te kompresijske slojeve. Najveće razlike se nalaze u mehanizmu prijenosa podataka između lanaca putem mostova. Za različite mostove, vrijeme trajanja prijenosa podataka se može razlikovati. Ono je ograničeno mehanizmom provjere ispravnosti podataka koji se prenose prije upisa na odredišni lanac.

3.1. Polygon PoS

Polygon PoS, kao što stoji u samom nazivu je lanac u kojem se konsenzus postiže *Proof of Stake* mehanizmom. Obzirom da ima svoj zaseban mehanizam konsenzusa, te skup validatora koji je neovisan o *Ethereum* validatorima, *Polygon PoS* nije kompresijski sloj.

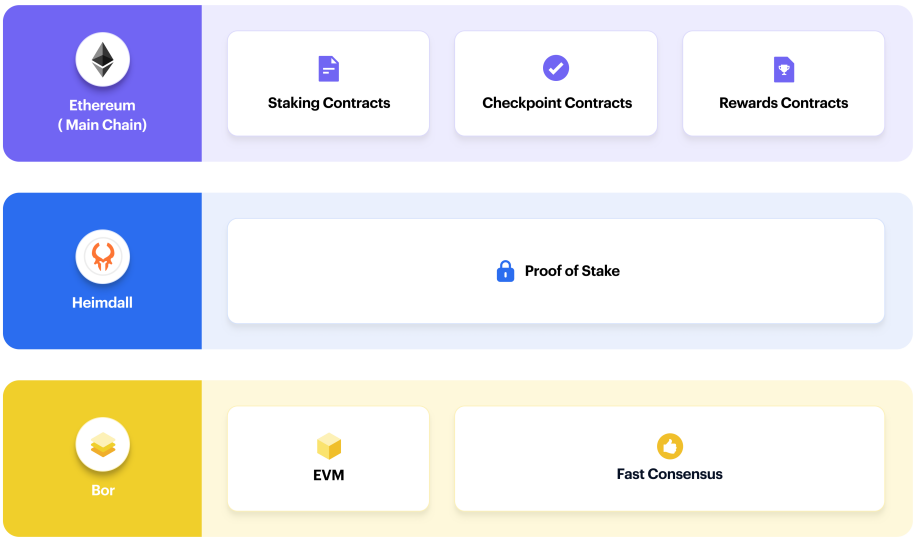
Ono što ga čini bočnim lancem je njegova varijacija *Tendermint* arhitekture koja sinkronizira stanje s *Polygon PoSa* na *Ethereum* mrežu. Iako je tema ovog rada bazirana oko *Ethereum* mreže i njenih validatora, važno je razumjeti arhitekturu Polygon PoS mreže jer o njoj ovisi mehanizam prijenosa podataka između korijenskog i lanca djeteta.

Postoje dva sloja:

1. sloj proizvodnje blokova (engl. *block producing layer*)
2. validacijski sloj (engl. *validator layer*)

Polygon PoS ima modularnu arhitekturu te je za svaki sloj zadužena zasebna implementacija čvora. Čvor sloja za proizvodnju blokova se zove *Bor*, a čvor sloja za validaciju se zove *Heimdall*.

Heimdall konstantno kontrolira pametne ugovore za validaciju Polygon PoS mreže koji žive na *Ethereumu*, te postavlja kontrolne točke na *Ethereum*. Također je zaslužan za odabir redoslijeda kojim će *Bor* validatori biti prozvani za proizvodnju blokova.



Slika 3.1. Arhitektura Polygon PoS lanca[14]

3.1.1. Validacija Polygon PoS mreže

Pametni ugovori na *Ethereum* mreži za validaciju Polygon PoS bočnog lanca omogućuju tri glavne značajke:

1. Mogućnost da bilo tko postavi svoje MATIC tokene u pametne ugovore za validaciju i pridruži se sustavu kao validator
2. Nagradu za validaciju promjene stanja Polygon PoS mreže
3. Spremanje kontrolnih točaka na *Ethereum* mrežu

Heimdall sloj agregira blokove koje proizvodi *Bor* u *Merkle* stablo, te periodički šalje korijen stabla na *Ethereum*. To se zove mehanizam kontrolnih točaka ranije spomenut.

Za svakih nekoliko blokova koje proizvede *Bor*, *Heimdall*:

1. Provjeri sve blokove od posljednje kontrolne točke
2. Proizvede *Merkle* stablo od hasheva tih blokova
3. Poštalje hash korijena na ugovor koji se nalazi na *Ethereumu*

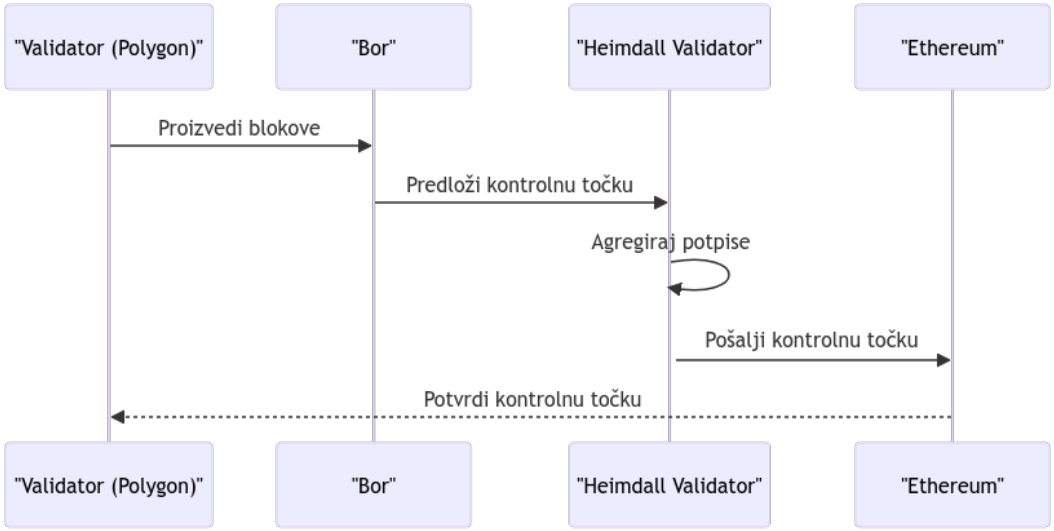
Kontrolne točke su važne iz dva razloga:

1. Služe kao oznaka konačnosti na lancu korijenu
2. Pružaju dokaz o uništavanju tokena prilikom prijenosa s lanca djeteta na korijenski lanac[14]

Pametni ugovori koji se nalaze na *Ethereum* mreži pružaju jezgrenu logiku Polygon PoS mreže. Sadrže razne mehanizme kao što su depoziti i izlazi s *Ethereum* mreže na bočni lanac i obrnuto. Također sadrže i izlazni prioritetni red, periodične kontrolne točke, mehanizme zaštite od prevare, mehanizam izlaska validatora iz seta i druge[15].

3.1.2. Most za prebacivanje tokena

Most implementira mehanizam zaključavanja i stvaranja (engl. *lock and mint*). To je mehanizam koji omogućava prijenos vrijednosti između dviju različitih mreža, često koristeći tzv. omotane (engl. *wrapped*) tokene kao sredstvo za predstavljanje originalnih



Slika 3.2. Dijagram toka sinkronizacije bočnog lanca i *Ethereuma*

sredstava na ciljnoj mreži. Ovaj pristup je ključan za interoperabilnost u *blockchain* ekosustavima, omogućavajući korisnicima da iskoriste specifične prednosti različitih mreža bez potrebe za trajnim prelaskom sredstava iz jednog lanca u drugi. Evo kako proces funkcionira:

Korak 1: Zaključavanje (engl. *Locking*)

1. **Početni Korak**: Korisnik koji želi prenijeti sredstva s originalne *blockchain* mreže (npr. *Ethereum*) na ciljnu mrežu (npr. Polygon) započinje proces tako što "zaključava" svoja sredstva u pametnom ugovoru na originalnoj mreži. Ovo "zaključavanje" učinkovito uklanja sredstva iz cirkulacije unutar originalne mreže.

2. **Sigurnost**: Pametni ugovor koji upravlja zaključavanjem sredstava dizajniran je tako da osigura da zaključana sredstva ne mogu biti pristupačna nikome osim kada korisnik vraća sredstva s ciljnog lanca.

Korak 2: Stvaranje (engl. *Minting*)

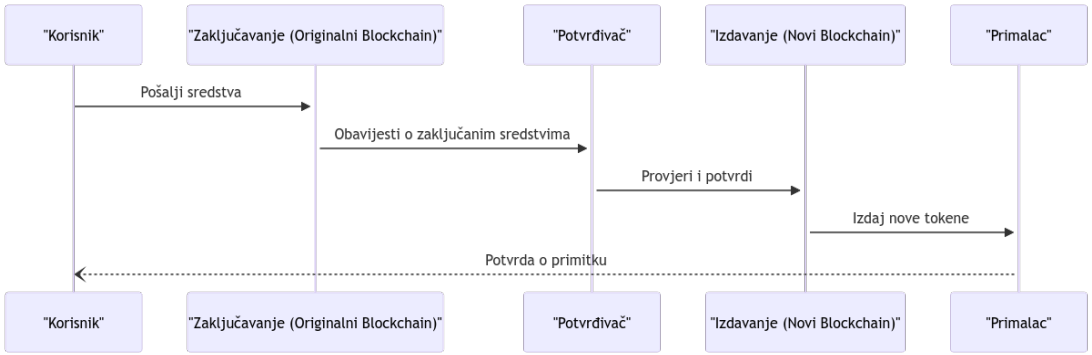
1. **Verifikacija**: Nakon što su sredstva zaključana, relevantne informacije o transakciji se verificiraju na ciljnoj mreži putem mehanizma konsenzusa ili putem određenih validatora zaduženih za nadzor mosta.

2. **Izdavanje omotanih tokena**: Kada se transakcija potvrdi, ekvivalentna količina

omotanih tokena se stvara na ciljnoj mreži korisniku. Omotani tokeni su specifični za ciljnu mrežu i dizajnirani su tako da imaju istu vrijednost kao i originalna sredstva koja su zaključana na početnoj mreži. U svakom trenutku u kojem postoji jedan omotani token na ciljnoj mreži, za njega postoji jedan zaključani token u pametnom ugovoru mosta na korijenskoj mreži.

Korak 3: Upotreba na Ciljnoj Mreži

Korisnik sada može koristiti omotane tokene na ciljnoj mreži za različite svrhe, uključujući trgovanje, ili sudjelovanje u decentraliziranim aplikacijama koje se izvršavaju na toj mreži.



Slika 3.3. Dijagram toka prebacivanja tokena

Korak 4: Otključavanje i Povrat Originalnih Sredstava (Opcionalno)

Ako korisnik želi povratiti svoja originalna sredstva, može "spaliti" (engl. *burn*) omotane tokene na ciljnoj mreži. Ovo spaljivanje signalizira pametnom ugovoru na početnoj mreži da "otključa" i vrati originalna sredstva korisniku, zatvarajući time ciklus prijenosa.[16]

Važno je napomenuti da ovo nije jedini dostupan most na *Polygon PoS* mreži. Ovo je nativni most, odnosno most koji je prepostavljen (engl. *default*) i dio samog protokola. Samim time, najviše povjerenja se daje takvom mostu.

Postoje mostovi postavljeni od trećih strana. Ti mostovi su obično mnogo brži, ali rade određene kompromise da bi to postigli. Kompromisi mogu biti komponente mosta koje nisu decentralizirane i izvode se na običnim poslužiteljima, zasebne mreže validatora za postizanje konsenzusa i slični. Problem koji se javlja s više mostova na jednom

lancu je taj da svaki most proizvede svoju inačicu omotanog tokena. U praksi to znači da ukoliko korisnik prebaci wstETH putem nativnog mosta na odredišnu mrežu i nekada kasnije koristi most treće strane (npr. Wormhole), imat će 2 potpuno različita skupa tokena iako su na korjenskom lancu bili isti. Problem postaje vidljiv prilikom interakcije s DeFi protokolima zbog fragmentacije likvidnosti. Obično je jedna inačica tokena dominantna, a sve ostale inačice se moraju zamijeniti za dominantu da bi bile iskoristive i interoperabilne.

Osim problema s korisničkim iskustvom, neki mostovi mogu donijeti velik sigurnosni rizik. To je primjerice problem kod mostova koji koriste dodatni skup validatora za postizanje konsenzusa oko prijenosa podataka, no taj skup validatora nije dovoljno decentraliziran. Ukoliko netko preuzme većinu, može potencijalno ukrasti tokene, zamrznuti prijenos ili cenzorirati transakcije[17].

3.2. Dizajn i arhitektura decentralizirane aplikacije

Kao što je objašnjeno u prethodnom dijelu, decentralizirane aplikacije koje ne mogu izolirano i samostalno odrašivati svoju funkciju na jednom lancu imaju složeniju arhitekturu jer se njihova implementacija nalazi na više lanaca istovremeno i stanje između tih komponenti mora biti sinkronizirano. Jedan takav primjer je objašnjen kroz implementaciju mosta između *Polygon PoS* mreže i *Ethereum* mreže.

Ideja aplikacije je omogućiti korisnicima bočnog lanca *Polygon PoS* jeftin i jednostavan način da sudjeluju u validaciji *Ethereuma*.

Ovaj projekt je prilagođen za *Polygon PoS* koji je bočni lanac *Ethereumu*, ali uz modifikacije dijela koda zaslužnog za prebacivanje podataka izmežu mreža, moguće je napraviti sustav koji radi primjerice s *Arbitrum* kompresijskim slojem. Korisnik *Polygon PoS* mreže koji posjeduje omotani *ether* (wETH) token može poslati svoje tokene u ovaj protokol kako bi postavio svoj *ether* u Lido protokol za validaciju. Nakon što protokol prikupi depozite od ostalih korisnika, on će procesuirati *shuttle* i svi korisnici će moći preuzeti svoje wstETH tokene na *Polygon PoS* mreži.

Motivacija

Ukoliko bi korisnik ručno želio napraviti tu radnju trebao bi proći kroz ove korake:

1. Započni prebacivanje wETH tokena na *Ethereum*
2. Preuzmi ETH na *Ethereumu*
3. Pošalji ETH u Lido i primi stETH
4. Dozvoli wstETH ugovoru da koristi stETH
5. Omotaj stETH u wstETH
6. Dozvoli Polygon bridge ugovoru da koristi wstETH
7. Pošalji wstETH na Polygon

Za korisnika to znači da bi trebao platiti 7 transakcija od kojih su 6 na *Ethereumu*. Ovisno o trenutnoj utilizaciji mreže i cijeni *ether*a, korisnik bi mogao platiti i do nekoliko stotina EUR za izvršavanje ovog slijeda što je neisplativo za većinu korisnika.

Koristeći grupnu validaciju i ovaj protokol to izgleda ovako:

1. Dozvoli `Child Pool` ugovoru da koristi wETH
2. Pošalji wETH u `Child Pool`
3. Preuzmi wstETH na Polygon PoS mreži iz `Child Poola`

Broj transakcija se smanjio sa sedam na tri. Ono što je još važnije je da su sve tri transakcije na *Polygonu* tako da korisnik može očekivati trošak od svega nekoliko desetaka Euro centi. Osim smanjenja troška, značajno je olakšan proces za korisnika i od njega se ne zahtjeva duboko razumijevanje nekoliko različitih protokola da bi mogao sudjelovati u poboljšanju decentralizacije i sigurnosti *Ethereuma*.

3.2.1. Glavni dijelovi aplikacije

Jezgru ove decentralizirane aplikacije čine 4 komponente:

1. `Child Pool`

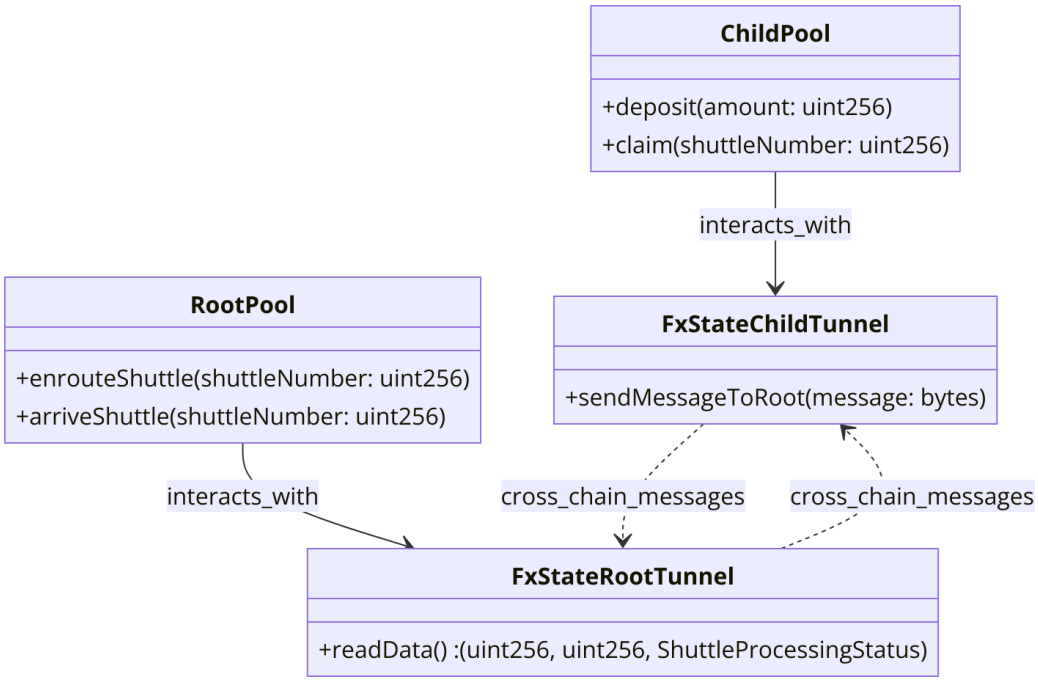
2. `Root Pool`

3. `Lido adapter`

4. `FxTunnel`

Pametni ugovor (Child Pool), odnosno bazen lanca djeteta upravlja korisničkim depozitima i zahtjevima na lancu djetetu. Ovaj ugovor omogućuje korisnicima da pošalju wETH tokene u trenutni (shuttle), te pruža mehanizme za obradu, slanje i finalizaciju (shuttle) transakcija preko `Child Tunnel` ugovora.

Pametni ugovor `Root Pool`, odnosno bazen korijenskoga lanca obrađuje (shuttle) transakcije primljene s lanca djeteta i šalje tokene natrag na isti. Ovaj ugovor omogućava depozit *ethera* preko Lido adaptera i slanje wstETH tokena natrag na `Child Pool`.



Slika 3.4. UML dijagram glavnih komponenata aplikacije

FxTunnel ugovori se nalaze na oba lanca. Oni služe kao pristupne točke za dvosmjernu komunikaciju između lanaca. Prijenos tokena s *Polygon PoSa* se izvršava kroz funkciju u samoj implementaciji omotanog tokena koja poziva interne metode mosta. *FxStateRootTunnel* i *FxStateChildTunnel* ugovori služe za prijenos informacija o trenutnom *shuttleu*. Ti podaci su:

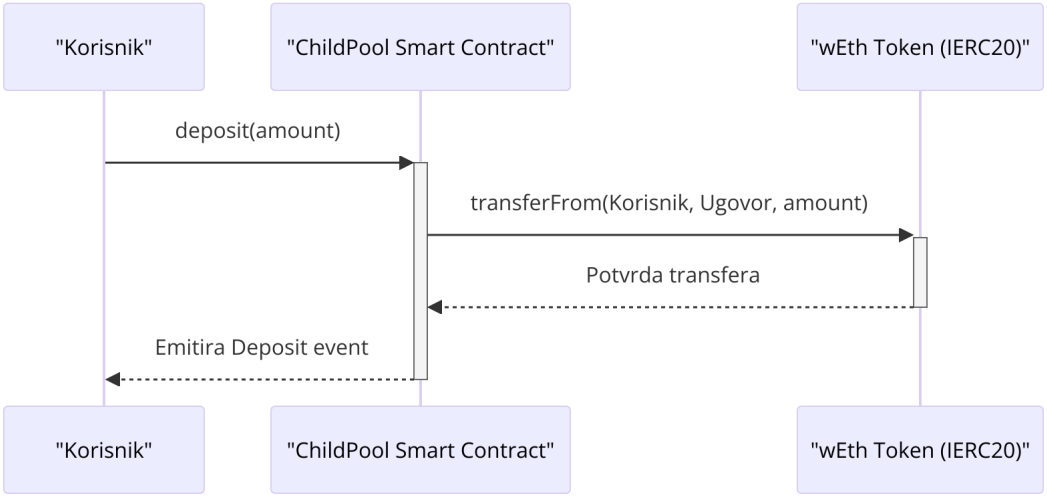
- Identifikator trenutnog *shuttlea*
- Ukupna količina wETH, odnosno wstETH tokena koji se prebacuje
- Status trenutnog *shuttlea*

Obzirom da se unutar iste transakcije odvija započinjanje prijenosa tokena i slanje poruke s dodatnim informacijama, te informacije u istom bloku postaju dostupne i na odredišnom lancu.

3.2.2. Child Pool

Korisnička interakcija započinje i završava s ovim ugovorom. Sve ostale komponente su apstrahirane od korisnika. Za njega su bitne samo funkcije `deposit` i `claim`. Prva funkcija prima depozit wETH tokena nakon što je korisnik prethodno dao dozvolu svojim

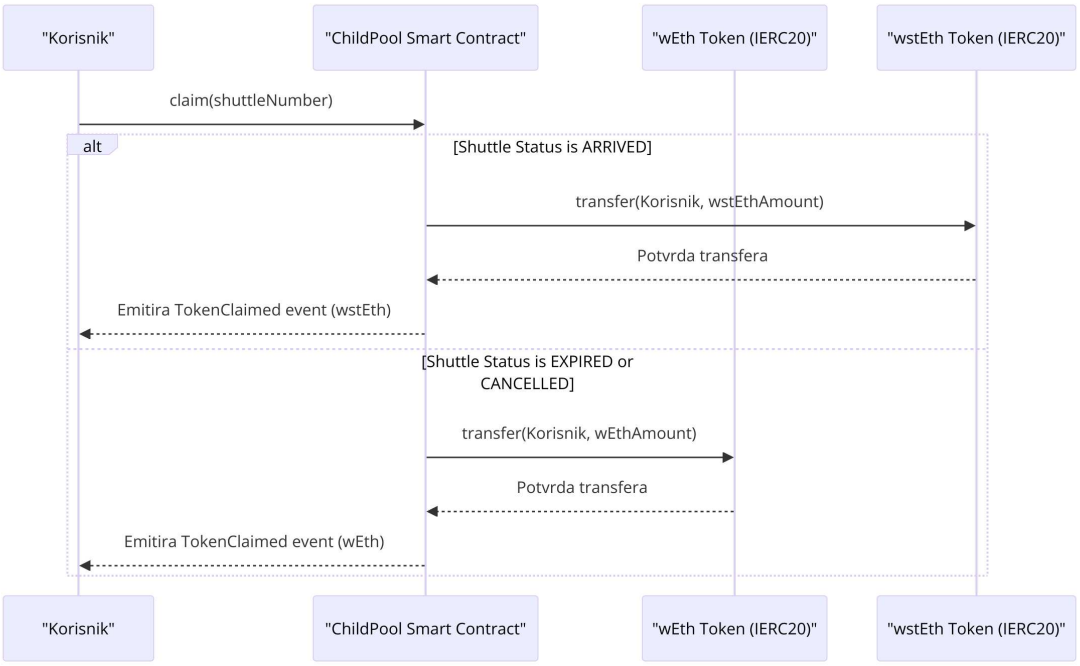
tokenima da ih konzumira `Child Pool` ugovor. To je sigurnosno ograničenje koje ima svaki token koji implementira ERC20 standard. Nativni tokeni kao MATIC na *Polygonu* ili *ether* na *Ethereumu* ne trebaju posebnu dozvolu, već je sam potpis privatnim ključem dozvola za njihovu potrošnju. Razlika je što je jedini način za slanje nativnog tokena tako da korisnik potpiše transakciju za slanje, dok ERC20 tokeni mogu uz dozvolu biti povučeni od strane ugovora ili korisnika kojem je dana dozvola. To je vrlo korisna funkcionalnost, ali potencijalno i rizična ukoliko se ne koristi ispravno.



Slika 3.5. Dijagram funkcije za depozit

Za `claim` funkciju `Child Pool` razlikuje dva slučaja:

1. Status *shuttlea* je `ARRIVED` -> korisnik prima wstETH token
2. Status *shuttlea* je `ARRIVED/EXPIRED` -> korisnik prima wETH token



Slika 3.6. Dijagram funkcije za preuzimanje wETH odnosno wstETH tokena

Nakon što su krajnji korisnici poslali "zadovoljavajuću" količinu tokena, operator je zadužen za procesiranje *shuttlea*. Zadovoljavajuća količina nije određena brojem, ovisi o poslovnom modelu koji ovdje nije promatran. Pretpostavka je da je cilj prikupiti što više korisnika koji su napravili depozit kako bi se trošak procesiranja podijelio na što više dijelova. Obzirom da je cilj rada pokazati da je ovakav mehanizam moguć na praktičan način, nisu postavljeni nikakvi dodatni troškovi niti provizije na krajne korisnike koji bi plaćali operatora za njegov rad.

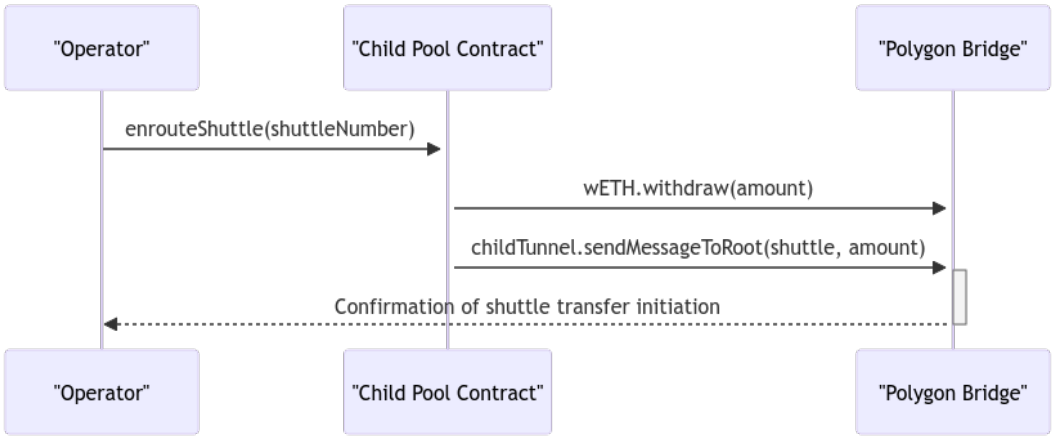
Operator je aktor koji ima ovlasti pozivati funkcije za procesiranje *shuttlea*. To su primarno funkcije `enrouteShuttle` i `arriveShuttle`.

enrouteShuttle Nakon poziva `enrouteShuttle`, prikupljeni wETH tokeni se šalju na `Root Pool` ugovor putem mosta s porukom koja sadrži dodatne informacije o prijenosu (identifikator i količinu tokena). Kao što je spomenuto u prethodnom dijelu (3.2.1.), iniciraju se dvije različite akcije za prijenos na *Ethereum*:

1. Token transfer

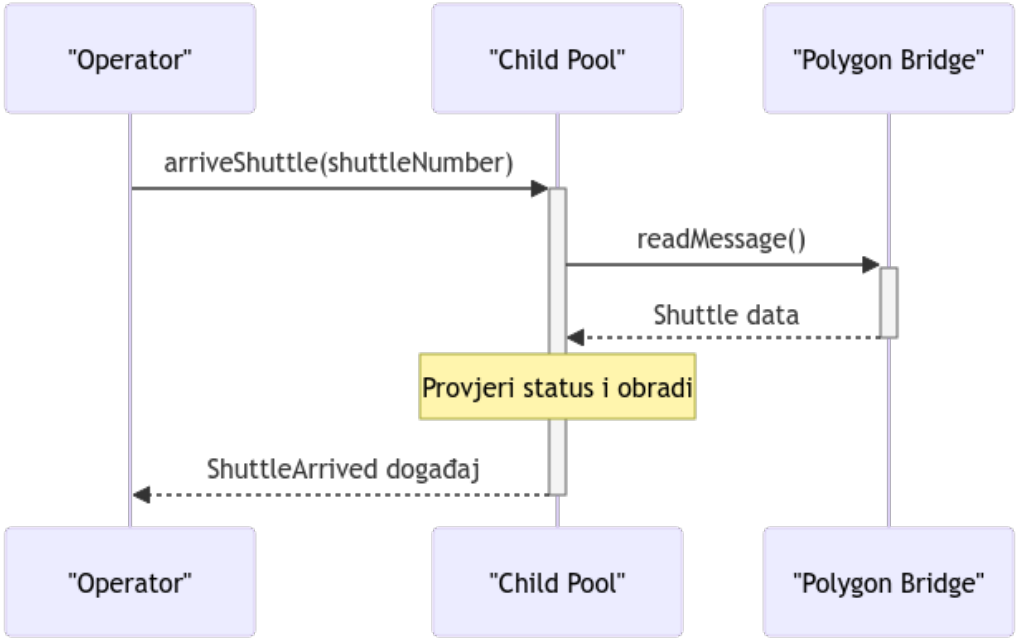
2. Transfer poruke

Svaki od tih događaja (engl. *event*) imaju svoj potpis (engl. *signature*) koji se koristi za generiranje dokaza o njihovom izvršenju na lancu djetetu. Bez tog dokaza nije moguće primiti poruke na lancu korijenu, odnosno *Ethereumu*. Za generiranje dokaza, potrebno je pričekati da *Heimdall* upiše sljedeću kontrolnu točku na ugovor koji upravlja kontrolnim točkama. To obično traje oko dvadeset minuta, ali ovisi o uvjetima na mreži. Operator je zadužen da provjeri kada je moguće generirati dokaze, generira ih i pošalje `Root Poolu`) na daljnje izvršavanje koje je opisano u nastavku (3.2.3.).



Slika 3.7. Dijagram `enrouteShuttle` funkcije

arriveShuttle Kad wstETH tokeni uspješno stignu s *Ethereuma*, operator poziva `arriveShuttle` funkciju s identifikatorom *shuttlea* te označava da je taj *shuttle* uspješno stigao. Korisnici nakon toga mogu zvati `claim` i preuzeti svoje tokene.



Slika 3.8. Dijagram `arriveShuttle` funkcije

3.2.3. `Root Pool`

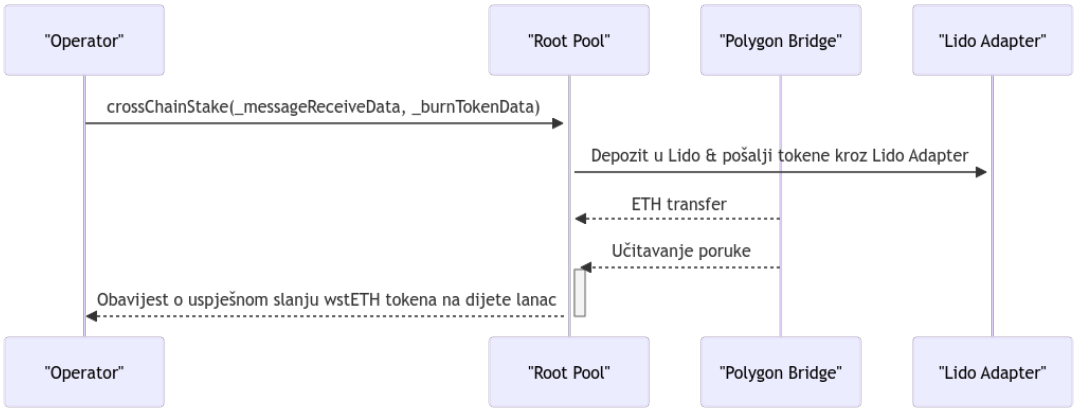
Ovaj ugovor živi na *Ethereumu*. Krajnji korisnici ne kreiraju transakcije s njim, već je za to zadužen operator. Nakon što je generirao dokaze o uništavanju tokena i prijenosu poruke s lanca djeteta, može pozvati `crossChainStake` s tim argumentima.

crossChainStake Nakon poziva `crossChainStake` funkcije odvija se slijed:

1. Poziv interne funkcije `_receieveShuttleFromChild` koja povlači tokene i čita poruku poslanu s lanca djeteta

2. Poziva se `depositForAndBridge` nad Lido adapterom koja apstrahira depozit u Lido i slanje tokena na dijete lanac

3. Poziva se `sendMessageToChild` sa informacijama o stanju *shuttlea* (status, identifikator i količina tokena)



Slika 3.9. Dijagram `crossChainStake` funkcije

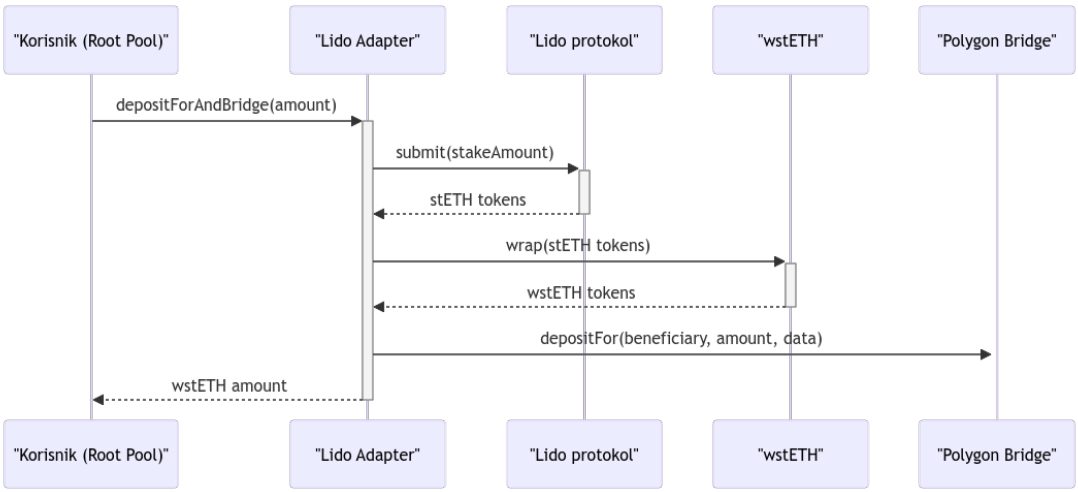
Nakon uspješnog izvršenja, tokeni postaju dostupni na *Polygonu* te operator može pozvati `arriveShuttle` funkciju (3.2.2.).

3.2.4. Lido adapter

Za validaciju *Ethereuma* u ovoj aplikaciji se koristi protokol *Lido*. Zbog modularne arhitekture aplikacije, moguće je izraditi adaptere i za druge protokole koji se koriste za validaciju, no *Lido* je odabran zbog svoje popularnosti, sigurnosti i kvalitetne dokumentacije koja olakšava proces integracije.

Lido adapter je odgovoran za izvršavanje depozita *ethera* u protokol, omatanje primljenog stETH tokena u wstETH token te iniciranje prijenosa wstETH tokena nazad na dijete lanac putem *Polygon* mosta.

Jedina javna funkcija adaptera jest `depositForAndBridge`, i ona će biti pozvana isključivo od strane `Root Pool` ugovora kad se na njemu pozove `crossChainStake` funk-



Slika 3.10. Dijagram toka funkcionalnosti Lido adaptera

cija.

4. Rezultati i rasprava

U nastavku su prikazane dvije tablice u kojima su navedeni koraci, odnosno transakcije izvorno prikazane u motivaciji za izradu ove aplikacije (3.2.). Obzirom da je za izvođenje ovih koraka potrebno izvršavati transakcije na *Polygonu* i *Ethereumu*, trošak u jedinicama *gas* je odvojeno zbrojen. Cijena za korisnika se računa formulom:

$$cijena = gasJedinice * gasCijena$$

Cijene su izražene u jedinicama gwei. Cijena jedinice gas ovisi o utilizaciji mreže. Kod *Polygona*, konačna cijena za korisnika je izražena u (MATIC) tokenu, dok je kod *Ethereuma* izražena u ETH tokenu. Krajnji korisnici obično bolje razumiju cijenu u nekoj od tradicionalnih fiat valuta, pa je za cijenu u EUR potrebno pogledati trenutni tečaj ETH i MATIC tokena.

Radi boljeg razumijevanja, niže su prikazane prosječne cijene jedinica *gas* 1.12.2023. i cijene tokena ETH i MATIC u istom danu.

Korištenjem cijena iz tablice (4.1.), cijena izvršavanja ovog slijeda za krajnjeg korisnika jest:

Ethereum gas u gwei	38,6
Polygon gas u gwei	173
ETH cijena u EUR	1935
MATIC cijena u EUR	0,73

Tablica 4.1. Cijena jedinice `gas` i tokena ETH/MATIC 1.12.2023.[1]

Tablica 4.2. Trošak jedinica *gas* za krajnjeg korisnika bez korištenja aplikacije za grupnu validaciju

Akcije:	Utrošak jedinica
Započni prebacivanje wETH tokena na *Ethereum*	39,620 (*Polygon*)
Preuzmi ETH na *Ethereumu*	277,823 (*Ethereum*)
Pošalji ETH u Lido i primi stETH	97,402 (*Ethereum*)
Dozvoli wstETH ugovoru da koristi stETH	65,310 (*Ethereum*)
Omotaj stETH u wstETH	113,751 (*Ethereum*)
Dozvoli *Polygon* most ugovoru da koristi wstETH	46,180 (*Ethereum*)
Pošalji wstETH na Polygon	107,123 (*Ethereum*)
Ukupni trošak *Ethereum*	707,589
Ukupni trošak *Polygon*	39,620

Ethereum

$$gasJedinice = 707589 \tag{4.1}$$

$$gasCijena = 38,6 \tag{4.2}$$

$$cijena(gwei) = 707590 * 38,6 \tag{4.3}$$

$$cijena(gwei) = 27312935,5 \tag{4.4}$$

$$cijena(ETH) = 0,0273129355 \tag{4.5}$$

$$cijena(EUR) = 52,85 \tag{4.6}$$

$$\tag{4.7}$$

Polygon

$$gasJedinice = 39620 \tag{4.8}$$

$$gasCijena = 173 \tag{4.9}$$

$$cijena(gwei) = 39620 * 173 \tag{4.10}$$

$$cijena(gwei) = 6854260 \tag{4.11}$$

$$cijena(MATIC) = 0,00685426 \tag{4.12}$$

$$cijena(EUR) = 0,00 \tag{4.13}$$

$$\tag{4.14}$$

Tablica 4.3. Trošak jedinica *gas* za krajnjeg korisnika uz korištenje aplikacije za grupnu validaciju

Akcije:	Utrošak jedinica
Dozvoli `Child Pool` ugovoru da koristi wETH	51,748 (Polygon)
Pošalji wETH u `Child Pool`	140,268 (Polygon)
Preuzmi wstETH na Polygon PoS mreži iz `Child Poola`	68,721 (Polygon)
Ukupni trošak *Ethereum*	0
Ukupni trošak Polygon	260,737

Ethereum

$$gasJedinice = 0 \tag{4.15}$$

$$gasCijena = 38,6 \tag{4.16}$$

$$cijena(gwei) = 0 * 38,6 \tag{4.17}$$

$$cijena(gwei) = 0 \tag{4.18}$$

$$cijena(ETH) = 0 \tag{4.19}$$

$$cijena(EUR) = 0 \tag{4.20}$$

$$\tag{4.21}$$

Polygon

$$gasJedinice = 260737 \tag{4.22}$$

$$gasCijena = 173 \tag{4.23}$$

$$cijena(gwei) = 260737 * 173 \tag{4.24}$$

$$cijena(gwei) = 45107501 \tag{4.25}$$

$$cijena(MATIC) = 0,045107501 \tag{4.26}$$

$$cijena(EUR) = 0,03 \tag{4.27}$$

$$\tag{4.28}$$

S ovim podacima krajnji korisnik troši samo $0,03/52,85 * 100 = 0,05\%$ od troška koji bi imao da samostalno izvodi cijeli proces.

Iako izračunata cijena u EUR ne odgovara stvarnoj trenutnoj cijeni, dobar je pokazatelj razlike za krajnjeg korisnika u odabranom trenutku. Kako je za ovaj rad odabran *Polygon PoS* i Lido kao protokol za validaciju, ukupna cijena ovisi uveliko i o kompleksnosti funkcija koje se pozivaju na tim pametnim ugovorima. Neka druga mreža, primjerice *Arbitrum* možda ima manje kompleksnu funkciju za depozit u most i samim time oba scenarija ispadaju povoljnija za krajnjeg korisnika.

Ova decentralizirana aplikacija bi mogla biti unaprijeđena na način da se odvoji funkcionalnost transfera tokena putem mosta od adaptera za depozit u Lido. Oba adaptera mogu implementirati sučelja kojima pristupa `Root Pool` ugovor. Na taj način arhitektura postaje potpuno modularna i neovisna o protokolu za validaciju i tehnologiji za skaliranje.

U realnom, produkcijskom okruženju, pametni ugovori aplikacije bi bili složeniji zbog dodatnih sigurnosnih značajki kao što su kontrola pristupa bazirana na zalozima i mogućnost nadograđivanja pametnih ugovora uz prigodne ovlasti. Dodatkom tih značajki, kompleksnost bi porasla što znači da bi se za izvođenje trošilo i više jedinica *gasa*.

5. Zaključak

U posljednje vrijeme *Ethereum* ekosistem postaje najvažniji skup platformi za razvoj decentraliziranih aplikacija. Do početka razvoja tehnologija za skaliranje kao što su bočni lanci i kompresijski slojevi, *Ethereum* je bio monolitna mreža u kojoj su se nalazile sve decentralizirane aplikacije. Prednost je bila visoka razina interoperabilnosti, no ubrzo broj transakcija koje *Ethereum* može procesuirati u bloku nije bio dovoljan za povećani broj transakcija i korisnika koji su došli s novim primjenama.

Trenutno najpopularniji tip decentraliziranih aplikacija se bavi financijama. To su primjerice aplikacije za razmjenu tokena, tržišta za kredite i slično. Pojava i popularnost aplikacija koje se bave financijama povlači i da vrlo velika količina kapitala cirkulira tim aplikacijama.

Da bi minimizirao rizik centralizacije, *Ethereum* protokol nagrađuje sudionike u validaciji transakcija i kreiranje blokova. Vrlo je važno da je ta aktivnost dostupna krajnjim korisnicima neovisno o njihovoj financijskoj moći. Ukoliko bi samo financijski snažni korisnici mogli sudjelovati, mreža bi efektivno pripadala validatorima manjeg broja imućnijih organizacija što je bio problem s originalnim algoritmom konsenzusa koji je koristio dokaz o radu. Povezivanjem validacije i decentraliziranih financija kroz protokole kao Lido, korisnici imaju dodatan razlog za sudjelovanje u validaciji jer više ne moraju odabrati jedno ili drugo, već imaju priliku sudjelovati u obje aktivnosti istovremeno.

Smatram da će vremenom krajnji korisnici u potpunosti prestati izvršavati transakcije na *Ethereumu* i da će se kompleksnosti i fragmentacije između različitih mreža u potpunosti apstrahirati sofisticiranijim alatima za komunikaciju s mrežama blok-lanac kao što su napredni digitalni novčanici koji agregiraju podatke s više mreža i pritom nude

iskustvo na kakvo je šira javnost navikla kroz klasične mobilne aplikacije.

Literatura

[1] V. Buterin, “Ethereum whitepaper”, 2013. [Mrežno]. Adresa: https://ethereum.org/whitepaper/

[2] A. M. Antonopoulos, *Mastering Ethereum: Building Smart Contracts and DApps*. O’Reilly Media, 2018.

[3] V. Buterin, “Why sharding is great: demystifying the technical properties”, 2021. [Mrežno]. Adresa: https://vitalik.eth.limo/general/2021/04/07/sharding.html

[4] ——, “https://vitalik.eth.limo/general/2021/05/23/scaling.html”, 2021. [Mrežno]. Adresa: https://vitalik.eth.limo/general/2021/05/23/scaling.html

[5] S. Šredl, “Ethereum - revolucija kroz decentralizaciju”, 2019. [Mrežno]. Adresa: http://darhiv.ffzg.unizg.hr/id/eprint/11152/1/Ethereum%20%E2%80%93%20revolucija%20kroz%20decentralizaciju.pdf

[6] D. Boneh, “Scaling the blockchain part ii: Rollups”, 2023. [Mrežno]. Adresa: https://cs251.stanford.edu/lectures/lecture17.pdf

[7] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger byzantium version”, 2019. [Mrežno]. Adresa: https://ethereum.github.io/yellowpaper/paper.pdf

[8] e. a. Lee Bousfeld, “Arbitrum nitro: A second-generation optimistic rollup”, 2022. [Mrežno]. Adresa: https://github.com/OffchainLabs/nitro/blob/master/docs/Nitro-whitepaper.pdf

[9] E. sudionici u Github repozitorij, “Proof of stake faq”, 2024. [Mrežno]. Adresa: https://ethereum.org/developers/docs/consensus-mechanisms/pos

[10] ——, “The merge”, 2023. [Mrežno]. Adresa: https://ethereum.org/roadmap/merge

[11] B. M. Elie Kapengut1, “An event study of the ethereum transition to proof-of-stake”, 2023. [Mrežno]. Adresa: https://arxiv.org/pdf/2210.13655.pdf

[12] E. sudionici u Github repozitorij, “How to stake your eth: Comparison of staking options”, 2023. [Mrežno]. Adresa: https://ethereum.org/staking#comparison-of-options

[13] L. D. sudionici, “Lido docs”, 2023. [Mrežno]. Adresa: https://docs.lido.fi

[14] P. Fundacija, “Polygon pos dokumentacija”, 2023. [Mrežno]. Adresa: https://docs.polygon.technology/pos/

[15] A. A. Jaynti Kanani, Sandeep Nailwal, “Matic whitepaper”, 2019. [Mrežno]. Adresa: https://github.com/maticnetwork/whitepaper

[16] Chainsecurity, “Code assessment of the pos portal smart contracts”, 2023. [Mrežno]. Adresa: https://chainsecurity.com/wp-content/uploads/2023/04/Polygon_PoS_Portal_-Smart-Contract-Audit_ChainSecurity.pdf

[17] B. Kiepuszewskij, “Rollups are the most secure bridges”, 2022. [Mrežno]. Adresa: https://archive.devcon.org/resources/6/rollups-are-the-most-secure-bridges.pdf

Sažetak

Usluga grupne validacije Ethereum mreže korištenjem tehnologija za skaliranje mreža lanca blokova

Jakov Buratović

U ovom radu su istraženi mehanizmi koji osiguravaju i decentraliziraju blok-lanac mreže čiji se konsenzus postiže mehanizmom dokaza o zalogu (engl. *Proof of Stake*). Dan je pregled različitih načina na koji korisnici mogu sudjelovati validaciji mreže. Predstavljene su prednosti za korisnike koji sudjeluju, no i mane koje se pojavljuju prisutnošću većeg broja transakcija na mreži u jedinici vremena. Objašnjen je predložen pristup koji povezuje tehnologije skaliranja blok-lanac mreža i validacije uz analizu rezultata i prijedloge poboljšanja te decentralizirane aplikacije.

Ključne riječi: ethereum; skaliranje; validacija; blok-lanac; most; pametni ugovori; čvorovi; transakcije

Abstract

A Group Validation Service of the Ethereum Network Using Blockchain Scaling Technologies

Jakov Buratović

This paper explores mechanisms which that secure and decentralise blockchain networks powered by Proof of Stake consensus algorithm. The paper gives an overview of various options users have to participate in validating the network. Advantages for validating participants are displayed along with bottlenecks that are becoming present with an increasing number of transactions in a unit of time. Finally, a solution that unifies blockchain scaling technologies and validating is proposed with an analysis of results and suggestions for future improvements of the decentralized application.

Keywords: ethereum; scaling; validation; blockchain; bridge; smart contracts; nodes; transactions

Privitak A: Postavljanje pametnih ugovora na mrežu

Za postavljanje (engl. *deployment*) pametnih ugovora treba prethodno instalirati nodejs module, postaviti varijable okružena i prevesti Solidity kod.

Pretkoraci (izvršavanje iz korjenskog direktorija projekta)

1. `npm run install`
2. `cp .env.example .env`
3. U `.env` datoteci postaviti tražene varijable
4. `npx hardhat compile` za prevođenje koda
5. `npm run test` za pokretanje testova prije deploymenta

Deployment Napomena: Nakon svakog *deploymenta*, potrebno je pospremiti adresu novonastalog ugovora. Najjednostavnije sučelje za interakciju s funkcijama ugovora jest putem pretraživača blokova kao što su Etherscan i Polygonscan. Kako bi pretraživač mogao prikazati funkcije, potrebno je verificirati izvorni kod naredbom:

```
npx hardhat verify [adresa] --mreža (goerli/mumbai)
```

U ovom primjeru se koriste testne mreže:

1. Goerli za *Ethereum* (`https://goerli.etherscan.io`)
2. Mumbai za Polygon PoS (`https://mumbai.polygonscan.com`)

1. Deploy Tunela:

```
$ npm run deploy-tunnel-goerli
```

```
$ npm run deploy-tunnel-mumbai
```

2. Postavi FxRoot na Child Tunnel i FxChild na Root Tunnel.

3. Deploy Lido adaptera koristeći

4. ```
   $ npm run deploy-adapter-goerli
   ```

5. Deploy bazena (pool), pazite da koristite ključ s istim nonceom. Oba bazena trebaju imati istu adresu.

6. ```
   $ npm run deploy-pool-goerli
   $ npm run deploy-pool-mumbai
   ```

7. Postavi adresu RootPool-a na rootTunnel.

8. Postavi adresu（Child Pool)-a na childTunnel.

9. Inicijaliziraj RootPool i（Child Pool).

Ovi koraci pružaju detaljne upute za deployment pametnih ugovora, uključujući postavljanje tunela, deploy Lido adaptera, i inicijalizaciju bazena na Goerli i Mumbai mrežama.

Privitak B: Adrese postavljenih ugovora

Za demonstraciju rada protokola, moguće je koristiti već postavljene ugovore na mreži. Ograničenje je da samo adresa `0x1F0d32e5A1E4748E061AcB5CBeE87b2243778e1E` ima ulogu operatora. To znači da korisnici moraju čekati da operator procesuira *shuttle*.

Oznake 5 i 8001 su identifikatori mreža Goerli i Mumbai.

```
    5: {
    checkPointManager: "0x2890bA17EfE978480615e330ecB65333b880928e",
    fxRoot: "0x3d1d3E34f7fB6D26245E6640E1c50710eFFf15bA",
    rootTunnel: "0xf9CE62874D1e31f6a9a0884eaDfd486d7E0A7075",
    rootManagerProxy: "0xBbD7cBFA79faee899Eaf900F13C9065bF03B1A74",
    etherPredicateProxy: "0xe2B01f3978c03D6DdA5aE36b2f3Ac0d66C54a6D5",
    depositManagerProxy: "0xBbD7cBFA79faee899Eaf900F13C9065bF03B1A74",
    lidoAdapter: "0x3a9B88aBCA123447AdD846B34388eFcbF90D37f2",
    rootPoolOwner: "0x1F0d32e5A1E4748E061AcB5CBeE87b2243778e1E",
    rootPool: "0xDc911Cd802B8381F0Ad88282036D245f66F85A10",
},
    80001: {
    fxChild: "0xCf73231F28B7331BBe3124B907840A94851f9f11",
    childTunnel: "0xeB0F6d99782C2b91b4bE0999F4F181CBA60dE3f7",
    wETHTokenOnChild: "0xA6FA4fB5f76172d178d61B04b0ecd319C5d1C0aa",
    wstEthToken: "0xF9a4BBAa7fA1DD2352F1A47d6d3fcfF259A6D05F",
    shuttleExpiry: 10,
    childPoolOwner: "0x1F0d32e5A1E4748E061AcB5CBeE87b2243778e1E",
    childPool: "0xDc911Cd802B8381F0Ad88282036D245f66F85A10",
},
```

Svi ugovori su verificirani i povijest transakcija je vidljiva na pretraživačima blokova.

Privitak C: Korisničke upute

Upute za Korištenje za Krajnjeg Korisnika

Ove upute omogućavaju krajnjim korisnicima da šalju wETH tokene na lanac dijete i kasnije pozivaju `claim` na istom lancu. Svi procesi u pozadini, uključujući depozit u Lido, transfere između lanaca i upravljanje *shuttleima*, apstrahirani su od korisnika.

Depozit wETH tokena

Za slanje wETH tokena u aplikaciju:

1. **Depozit wETH Tokena:** Za depozit, koristi se funkcija `deposit` u `ChildPool` ugovoru. Potrebno je specificirati količinu wETH tokena za slanje. Ovaj korak zahtijeva interakciju s pametnim ugovorom putem odgovarajućeg sučelja na blockchainu (npr., kroz web aplikaciju ili direktno preko *blockchain* walleta).

```
function deposit(uint256 amount) external;
```

Primjer kako to učiniti putem web aplikacije ili walleta koji podržava interakciju sa smart ugovorima:

1. Otvoriti pretraživač blokova `https://mumbai.polygonscan.com`.
2. Unijeti adresu `ChildPool` ugovora.
3. Povezati novčanik i pozovite funkciju `deposit` s količinom wETH za slanje.

2. **Čekanje na Obradu:** Nakon što je depozit izvršen, tokeni će biti uključeni u *shuttle* koji će ih prenijeti na *Ethereum* i staviti u Lido protokol putem Lido adaptera. Ovaj proces može potrajati neko vrijeme, ovisno o dinamici mreže i *shuttle* rasporedu.

Pozivanje Claim na Child Chainu

Nakon što je depozit obrađen i tokeni su stavljeni postavljeni u Lido, moguće je pozvati `claim` na `ChildPool` ugovoru za preuzimanje wstETH tokena.

1. **Poziv funkcije Claim:** Za preuzimanje tokena, koristiti funkciju ‘`claim` u `ChildPool` ugovoru. Potrebno je specificirati broj *shuttlea* iz kojeg treba preuzeti tokene.

```
function claim(uint256 _shuttleNumber) external;
```

Kako pozvati ‘claim‘:

1. Otvoriti pretraživač blokova `https://mumbai.polygonscan.com`.
2. Unijeti adresu `ChildPool` ugovora.
3. Povezati novčanik i pozovite funkciju `claim` s identifikatorom *shuttlea*. Nakon uspješnog poziva funkcije, wstETH tokeni će biti preneseni na adresu pozivatelja.

Privitak D: Upute za operatora

Upute za Operatora za Procesuiranje *Shuttlea*

Operatorova uloga ključna je u procesu slanja akumuliranih wETH tokena s lanca djeteta na korijenski lanac i daljnje obrade.

Slanje *Shuttlea* s djeteta na korijen

1. **Provjera Spremnosti Shuttlea:** Prije slanja *shuttlea*, potrebno je provjeriti je li zadovoljavajuća količina wETH tokena akumulirana nu `ChildPool` ugovoru. To se može učiniti pregledom stanja (Child Pool) ugovora.

2. **Slanje Shuttlea:** Kada se utvrdi da je *shuttle* spreman, poziva se funkcija `enrouteShuttle` u `Child Pool` ugovoru za slanje *shuttlea* na korijenski lanac. Ova funkcija prenosi wETH tokene u odgovarajući *shuttle* i označava ga kao `ENROUTE`.

```
function enrouteShuttle(uint256 _shuttleNumber) external;
```

- Ova funkcija zahtijeva da je adresa koja interaktira autentificirana kao operator i da je *shuttle* u statusu koji omogućuje slanje.

- *Shuttle* se označava kao `ENROUTE` što znači da je u procesu slanja na korijen.

Obrada Shuttlea na lancu korijenu

Nakon što je *shuttle* poslan na lanac korijen, slijedeći koraci se odnose na njegovu obradu.

1. **Prijem *Shuttlea*:** Operator, pratit dolazak *shuttlea* na lanac korijen. To uključuje praćenje poruka koje dolaze preko `RootTunnel` ugovora.

2. **Zalog i slanje na udaljeni lanac:** Jednom kada *shuttle* stigne na korijen, potrebno je izvršiti depozit prikupljenih wETH tokena preko Lido adaptera i slanje nastalih wstETH tokena natrag na lanac dijete. Ovo se postiže pozivanjem funkcije `crossChainStake` u `RootPool` ugovoru.

```
function crossChainStake(bytes memory _messageReceiveData, bytes memory _burnTo
```

- `_messageReceiveData` i `_burnTokenData` su dokazi potrebni za verifikaciju transakcije i obradu. Za generiranje dokaza, postoji skripta za čije su korištenje napisane upute u nastavku ovog dokumenta.
- Ova funkcija obrađuje *shuttle*, stavlja wETH tokene u Lido protokol preko Lido adaptera, i šalje nastale wstETH tokene natrag na lanac dijete.

Završetak Procesa

Nakon uspješnog depozita u Lido i slanja na dijete lanac:

- *Shuttle* se označava kao obrađen, a odgovarajući wstETH tokeni se šalju natrag na lanac dijete za korisnike koji su sudjelovali u *shuttleu*.
- Kao operator, važno je osigurati da se svi koraci pravilno dokumentiraju i prate kako bi se osigurao transparentan i siguran proces.

Napomene

- Operateri moraju imati odgovarajuće ovlasti (ROLE) za izvršavanje funkcija ugovora.
- Svi koraci zahtijevaju pažljivo praćenje i upravljanje kako bi se osigurala sigurnost i efikasnost procesa.
- Komunikacija između lanaca koristi složene mehanizme poput tunela, što zahtijeva razumijevanje kako ti mehanizmi funkcioniraju.

Generiranje dokaza za *Token burn* i *Message Transfer*

Kao operator, važno je generirati dokaze za uništavanje tokena i transfer poruke s lanca djeteta kako bi se mogla pozvati funkcija `crossChainStake` na korijenskom lancu.

Dokazi se mogu generirati korištenjem skripte `proof_generation.js`.

Koraci za Generiranje Dokaza

1. **Priprema:** Prvo, osigurati da su instalirani potrebni paketi za pokretanje skripte. Skripta koristi `@maticnetwork/maticjs`, ‘`@maticnetwork/maticjs-web3`, i druge povezane pakete.

2. **Konfiguracija Skripte:** Urediti skriptu `proof_generation.js`‘ kako biste postavili relevantne konfiguracije za mreže. Potrebno je specificirati pristupne točke (provider URLs) za *Ethereum* (korijen) i *Ethereum* (dijete) mreže.

3. **Specifikacija Događaja:** Unijeti potpis događaja transfera ili poruke (TRANSFER_EVENT_SIG) koji je potreban za generiranje dokaza. Ovo je heksadecimalni string koji predstavlja potpis *Ethereum* događaja.

4. **Unos Polygon TX Hasha:** Unijeti hash transakcije na *Polygon PoSu* (`BURN_TX_HASH`) koja predstavlja *withdrawal* akciju za koju treba generirati dokaz.

5. **Pokretanje Skripte:** Pokrenuti skriptu `proof_generation.js`‘ kako biste generirali dokaze. Skripta će ispisati potrebne informacije koje su potrebne za poziv funkcije na korijenskom lancu.

```
node proof_generation.js
```

Privitak E: Razvojna dokumentacija za `Child Pool`

Pregled

Pametni ugovor `ChildPool` upravlja korisničkim depozitima i zahtjevima na lancu djetetu. Ovaj ugovor omogućuje korisnicima da pošalju wETH tokene u trenutni *shuttle*, te pruža mehanizme za obradu, slanje i finalizaciju procesa putem tunela djeteta.

Ključne Komponente

- **IFxStateChildTunnel**: Sučelje za komunikaciju sa korjenskim lancem.
- **IWEthToken & IERC20**: Sučelja za interakciju s wETH i wstETH tokenima.
- **Shuttle**: Struktura koja predstavlja pojedinačni *shuttle*, sadrži informacije kao što su ukupna količina, status, primljeni tokeni i rok trajanja.

Inicijalizacija

- **Initialize(IFxStateChildTunnel _childTunnel, IWEthToken _wEthToken, IERC20 _wstEthToken, uint256 _shuttleExpiry, address _owner)**: Postavlja početne vrijednosti ugovora, uključujući adrese tunela i tokena, vrijeme isteka *shuttlea*, i vlasnika.

Glavne Funkcije

- **deposit(uint256 amount)**: Omogućuje korisnicima da pošalju wETH tokene u trenutni *shuttle*.
- **enrouteShuttle(uint256 _shuttleNumber)**: Operater može poslati trenutni *shuttle* prema *Ethereumu*, pokrećući prenos sredstava.

- **arriveShuttle(uint256 _shuttleNumber)**: Finalizira *shuttle* nakon primanja sredstava i poruka s korjenskog lanca.
- **claim(uint256 _shuttleNumber)**: Korisnici mogu zahtijevati svoje tokene nakon što je *shuttle* finaliziran.
- **expireShuttle(uint256 _shuttleNumber)**: Omogućuje istek *shuttlea* ako nije poslan prije definiranog roka.
- **cancelShuttle(uint256 _shuttleNumber)**: Operater može otkazati *shuttle* koji je u dostupnom statusu, omogućujući korisnicima da zahtijevaju povrat deponiranih wETH tokena.
- **setShuttleExpiry(uint256 _shuttleExpiry)**: Postavlja vrijeme isteka *shuttlea* u blokovima.

Dogadjaji (engl. *Events*)

- **ShuttleCreated**: Emitira se kada je stvoren novi *shuttle*.
- **Deposit**: Emitira se kada korisnik pošalje tokene u *shuttle*.
- **ShuttleEnrouted**: Emitira se kada je *shuttle* poslan.
- **ShuttleArrived**: Emitira se kada *shuttle* stigne i sredstva su primljena.
- **TokenClaimed**: Emitira se kada korisnik preuzme svoje tokene.
- **ShuttleExpired**: Emitira se kada istekne vrijeme *shuttlea*.
- **ShuttleCancelled**: Emitira se kada je *shuttle* otkazan.
- **ShuttleExpiryChanged**: Emitira se kada je postavljeno novo vrijeme isteka *shuttlea*.

Sigurnosni Moduli

- **ReentrancyGuard**: Sprječava *reentrancy* napade.
- **Pausable**: Omogućuje pauziranje ugovora u hitnim situacijama.

- **AccessControl**: Upravlja pristupom različitim funkcijama ugovora.

Kako Koristiti

1. **Inicijalizacija**: Prije korištenja, ugovor mora biti inicijaliziran s potrebnim parametrima i adresama.

2. **Depozit**: Korisnici mogu poslati wETH tokene pozivom funkcije ‘deposit‘.

3. ***Shuttle* Upravljanje**: Operater može upravljati *shuttleima*, uključujući njihovo slanje, finalizaciju, otkazivanje ili označavanje kao istekle.

4. **Zahtijevanje Tokena**: Nakon finalizacije *shuttlea*, korisnici mogu preuzeti svoje wstETH ili wETH tokene, ovisno o statusu *shuttlea*.

Privitak F: Razvojna dokumentacija za `Root Pool`)

Pregled

Pametni ugovor `Root Pool` obrađuje *shuttle* transakcije primljene s lanca djeteta i šalje tokene natrag na isti lanac. Ovaj ugovor omogućava depozit ETH u Lido putem adaptera i slanje wstETH tokena natrag na `Child Pool`.

Ključne Komponente

- **IFxStateRootTunnel**: Sučelje za komunikaciju s lancem djetetom.
- **IRootManagerProxy**: Sučelje za upravljanje `Root Manager` ugovorom.
- **ILidoAdapter**: Sučelje za Lido adapter ugovor.

Inicijalizacija

- **initialize(IFxStateRootTunnel _rootTunnel, IRootManagerProxy _rootManagerProxy, ILidoAdapter _lidoAdapter, address _owner)**: Postavlja početne vrijednosti ugovora, uključujući adrese tunela, adresu `Root Manager Proxy` ugovora, Lido adapter, i vlasnika.

Glavne Funkcije

- **crossChainStake(bytes memory _messageReceiveData, bytes memory _burnTokenData)**: Izvršava depozit u protokol za validaciju. Prima poruku poslanu s `Child Tunnel` ugovora, potvrđuje uništene tokene na Polygon lancu, stavlja tokene u Lido protokol i šalje wstETH na `Child Pool`.

Interna Metodika

- **_receieveShuttleFromChild(bytes memory _messageReceiveData, bytes memory _burnTokenData)**: Interna metoda za primanje *shuttlea* koji je pokrenut s `Child Pool` ugovora. Ova metoda dekodira i obrađuje primljene podatke, potvrđuje tokene poslane s *Polygona* na *Ethereum*, i provjerava iznos za depozit.

Setters

- **setLidoAdapter(ILidoAdapter _lidoAdapter)**: Postavlja adresu Lido adapter ugovora.

Događaji (engl. (*Events*))

- **ShuttleProcessed**: Emitira se nakon što je *shuttle* procesiran.
- **ShuttleProcessingInitiated**: Emitira se kada započne obrada *shuttlea*.

Kako Koristiti

1. **Inicijalizacija**: Inicijalizirajte ugovor s potrebnim parametrima i adresama. Samo jednom potrebno izvršiti.
2. **Cross-Chain Stake**: Da biste izvršili depozit i transfer na *Polygon*, operater mora pozvati `crossChainStake` s potrebnim dokazima i podacima.
3. **Postavljanje Lido Adaptera**: Governance može postaviti adresu Lido adaptera pozivom `setLidoAdapter`.

Napomene

- Zahtijeva da operatori i governance imaju odgovarajuće role za izvršenje funkcija
- Korištenje ovog ugovora zahtijeva poznavanje procesa transfera između mreža koristeći most i depozit u Lido putem Lido adaptera.