VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

BlockChain sistemų mastelio keitimo savybių tyrimas taikant agentais grįstą modeliavimą

Investigation of the BlockChain Systems‘ Scalability Features Using the Agent Based Modelling

Magistro baigiamasis darbas

Atliko:	Aleksas Šulnius	(parašas)
Darbo vadovas:	dr. Karolis Petrauskas	(parašas)
Recenzentas:	lekt. Donatas Čiukšys	(parašas)

Vilnius – 2017

Padėka

Darbo autorius dėkoja darbo vadovui dr. Karoliui Petrauskui už patarimus, pastabas, naujų idėjų palaikymą bei didelę kantrybę viso magistro darbo rengimo metu.

Santrauka

BlockChain technologijos išradimas dažnai lyginamas su interneto išradimu. Kartu yra pripažįstama, kad BlockChain šiuo metu išgyvena fazę prilyginamą ankstyvajam interneto veikimo laikotarpiui. Dar tik bus išrastos priemonės, metodologijos, patvirtintos tvarkos ir teisinis reglamentavimas leisiantis BlockChain technologijai būti pritaikytai finansinių, reguliacinių, socialinių klausimų sprendimui. Neilga BlockChain istorija parodė, kad priimami neteisingi fundamentalūs BlockChain plėtros sprendimai lemia didžiulius finansinius ir reputacinius tam tikrų platformų nuostolius. Kartais per vėlai pastebėtos techninės klaidos lemia net atskirų BlockChain platformų ir bendruomenių išnykimą.

Šiame darbe nagrinėjamos galimybės iš anksto įvertinti tam tikrų BlockChain architektūrinių spendimų įtaką platformų veikimo parametrams. Šiame darbe tiriamos galimybės pasinaudoti agentais grįsto modeliavimo principais tokių sistemų modeliavimui. Konkrečiai tiriant BlockChain platformų mastelio keitimo savybes. Darbe yra siūloma BlockChain platformų modeliavimo agentais metodika. Pagrindiniai metodikos principai yra išbandomi su populiariausia BlockChain realizacija BitCoin. Modelis yra validuojamas pasiremiant su BitCoin platforma atliktais mastelio keitimo eksperimentais.

Raktiniai žodžiai: modeliavimas agentais, BlockChain, BitCoin, mastelio keitimas, simuliacija.

Summary

BlockChain currently is in the spotlight of all the FinTech industry. This technology is being called revolutionary, ground breaking, disruptive and even the WEB 3.0. On the other hand it is widely agreed that the BlockChain is in its early stages of development. In its current state BlockChain is in similar position that the Internet was in the early nineties. In order for this technology to gain mainstream adoption in the financial, regulatory and social sectors, new frameworks, methodologies and legal systems need to be developed. During its short history BlockChain has already received several major blows to its reputation when ingenious decisions were made concerning several BlockChain platforms' fundamental architectural principles.

This paper investigates how the impact of different decisions' could be modeled in advance, to be able to foresee possible risks and their outcomes. In this work the agent based modelling is proposed for the modeling of the BlockChain systems. Model, simulating the BitCoin scalability features is developed. Also this model is validated with the help of already performed real life BitCoin scalability experiments.

Key words: agent modelling, BlockChain, BitCoin, scalability, simulation.

TURINYS

Įvadas

Temos naujumas ir aktualumas

Šiuo metu pasaulyje vykstančiose Fintech ir Regtech konferencijose, diskusijose, programavimo varžybose (angl. – *Hackatons*) viena tema susilaukia ypač daug susidomėjimo ir įvairių nuomonių. Ši tema yra Blokų grandinės (angl. ir toliau tekste – *BlockChain*). BlockChain kaip technologija, kurios pagrindu veikia BitCoin kripto valiutos sistema, po truputį išeina už kripto valiutų ribų ir pradedama pritaikyti kitų uždavinių sprendimui. Daugybė pasaulio bankų, logistikos, IT įmonių, valstybinių institucijų tiria įvairias BlockChain pritaikymo galimybes (angl. – *use cases*). Šiuo metu labiausiai tikėtinos BlockChain pritaikymo sritys yra identifikacijos, autentifikavimo, duomenų nepaneigiamumo sprendimai. Nors daugybė skirtingų įmonių bei institucijų atlieka technologinius tyrimus, kol kas nėra plačiai žinomų realiai veikiančių sprendimų BlockChain pagrindu. Tokios situacijos priežastimis dažniausiai įvertinami dabartinių BlockChain principus įgyvendinančių platformų trūkumai. Daug šaltinių vieną pagrindinių šių technologijų trūkumą įvardija prastas mastelio keitimo (angl. – *Scalability*) savybes. Šie trūkumai išryškėja ne tik bandant pritaikyti BlockChain idėjas naujose srityse, bet taip pat ir stebint šiuo metų populiariausių kripto valiutų sistemų veikimą. Populiariausios BlockChain platformos BitCoin transakcijų apdorojimo greitis tūkstančiais kartų atsilieka nuo VISA/MasterCard sistemų pajėgumų. Šiame darbe yra tiriamos galimybės modeliuoti BlockChain principais veikiančias sistemas agentais grįstu modeliavimu. Taip pat yra pateikiami esamų modelių papildymai, leidžiantys modeliuoti BlockChain sistemų mastelio keitimo savybes.

Dėl esminių BlockChain savybių (decentralizacija, savireguliacija, tinklo dalyvių pasiekiamas konsensusas) yra pasirinktas modeliavimo agentais būdas. Agentais grįstas modeliavimas remiasi atskirų modelyje veikiančių esybių tarpusavio simuliacijos kūrimu ir iš jų tarpusavio sąveikos kylančių rezultatų gavimu. BlockChain principus realizuojančiose platformose nėra centrinės, ekosistemos veiklą valdančios esybės. Platformos veikimo rezultatai gaunami veikiant tinklo dalyviams. Strateginiai platformos veikimo sprendimai yra priimami dalyvaujant visiems tinklo nariams. Dėl šios priežasties yra tiriama galimybė modeliuoti BlockChain tinklo dalyvius agentais. Yra tiriama ar pritaikius jau šiuo metu egzistuojančius agentais grįsto modeliavimo taikymus IT sistemų modeliavime, įmanoma orientuoti juos į BlockChain technologijomis veikiančių sistemų modeliavimą. Iš pradžių nagrinėjama kokie BlockChain technologijų aspektai gali būti modeliuojami remiantis agentais, kokie turėtų būti

atlikti kitų agentų modelių pakeitimai ar papildymai, kad jie būtų pritaikyti BlockChain technologijų modeliavimo sprendimams.

Darbo tikslas, uždaviniai ir laukiami rezultatai

Šio magistrinio darbo tema – BlockChain principu veikiančių sistemų mastelio keitimo savybių tyrimas taikant agentais grįstą modeliavimą.

Darbo tikslas:

- Ištirti galimybes ar agentais grįstas modeliavimas gali būti taikomas *BlockChain* sistemų simuliacijai;
- Jeigu BlockChain sistemų modeliavimas agentais yra įmanomas, sukurti agentais grįstą modelį. Ištirti BlockChain sistemų mastelio keitimo savybes bei apibendrinti išvadas.

Pagrindiniai uždaviniai:

1. Apibrėžti BlockChain esybes, kurios galėtų būti modeliuojamos agentais;
2. Apibrėžti agento sąvoką BlockChain ekosistemoje;
3. Dabar egzistuojančių agentais grįstų modeliavimo sprendimų pagrindu, parengti modelių papildymus, leidžiančius modeliuoti BlockChain sistemas;
4. Identifikuoti BlockChain mastelio keitimo savybes, kurios bus modeliuojamos agentais;
5. Parengti modelio veikimo pavyzdžius;
6. Validuoti modeliavimo rezultatus, lyginant juos su realybėje įvykdytų BlockChain sistemų bandymų rezultatais;
7. Pagal modelio veikimo rezultatus įvertinti galimų mastelio keitimo ateities sprendimų įtaką BlockChain platformų veikimui;
8. Parengti pasiūlymus galimiems ateities BlockChain mastelio keitimo savybių pagerinimo tyrimams;
9. Pateikti išvadas galimiems ateities modelio plėtimo darbams.

Laukiamas rezultatas –agentais grįstas BlockChain modelis, modeliuojantis bent vieną, šiuo metu aktualią BlockChain mastelio keitimo savybę. O taip pat pasiūlymai dėl tokio modelio plėtimo galimybių ateityje.

1. Agentais grįstas modeliavimas

Į klausimą kodėl modeliavimas agentais tampa vis populiaresnis ir juo vis labiau domimasi yra pateikiamas atsakymas: „Nes gyvename vis labiau sudėtingėjančiame pasaulyje“ [MNo09]. Turima omeny tai, kad sistemos, kurias kyla poreikis analizuoti, ištirti, modeliuoti tampa vis labiau paremtos atskirų vienas nuo kito nepriklausomų komponentų sąveika. Modeliavimas agentais tai realių biologinių, socialinių, ekonominių, techninių ir bet kokių kitų sistemų modeliavimo būdas, kurio pagrindas tam tikroje kontroliuojamoje aplinkoje veikiančios ir tarpusavyje sąveikaujančios esybės (agentai) [AT14] [MNo09]. Tokie iš agentų sudaryti modeliai simuliuoja sudėtingas sistemas. Tinkamai pritaikyti agentų tarpusavio veikimą bei jų veikimo aplinką apibrėžiantys algoritmai leidžia modeliuoti daugelį realaus pasaulio sistemų, kurių pagrindą sudaro jose veikiančių dalyvių veiksmai. Tokių sistemų spektras kinta nuo vertybinių popierių prekybos ir eismo reguliavimo iki cheminio ginklo teritorijų užterštumo ar senovės civilizacijų tyrimo sistemų. Modeliavimo agentais esmė yra tai, kad tinkamai sumodeliuoti agentai veikdami sistemoje, t.y. sąveikaudami tarpusavyje, sukuria naujus, iš anksto nenumatytus elgsenos modelius taip suteikiant daug informacijos apie pačią sistemą, kurios nebūtų galima gauti kitais analitiniais modeliavimo metodais.

Agentais paremtu modeliavimu sudarytas modelis dažniausiai pasižymi šiais požymiais [MNo09]:

- Aibė agentų, kurių kiekvienas turi savo tam tikrus atributus ir modeliuotojo sukurtas bei apibrėžtas elgsenas;
- Aibė agentų ryšių ir tarpusavio bendravimo (sąveikos) metodų, kurie nurodo kaip ir su kuo (konkrečiais kitais agentais, visais agentais, veikimo erdve) agentas bendrauja;
- Veikimo aplinka, kurioje veikia agentai. Kiekvienas agentas bendrauja tiek su kitais agentais tiek ir su aplinka. Aplinka taip pat daro įtaką agentų veikimui.

Konkretūs pramonėje ir versle veikiančių modelių sukurtų Agentais grįsto modeliavimo principais pavyzdžiai [MN11]:

- Virtuali rinkos mokymosi laboratorija (angl. – *The Virtual Market Learning Lab*) agentais grįstas modelis, skirtas prekybos rinkos modeliavimui. Pagrindinis modelio tikslas yra namų ūkių (vartotojų) ir gamybos įmonių veikimo laisvoje ekonominėje erdvėje simuliacija. Šis modelis, yra sėkmingai pritaikytas Procter & Gamble korporacijos.
- Vandenilio ekonominis modelis (angl. – *The Hydrogen Economy model*) vienoje iš modeliavimo erdvių Repast Symfony sukurtas modelis vykdantis Los Andželo miesto ir jo priemiesčių eismo dalyvių ir vandenilio kuro kolonėlių elgsenos simuliacija ir pagal

daugelį skirtingų parametrų modeliuojantis vandenilio kuro plitimo tendencijas Kalifornijos valstijoje. Yra panaudota Geografinė informacinė sistema, kuri yra viena iš modeliavimo agentais aplinkos Repast savybių. Ši galimybė leidžia modelyje panaudoti informaciją apie 13000 kvadratinių kilometrų greitkelių ir vietinės reikšmės kelių tinklą.

- InSTREAM modelis, simuliuojantis lašišų elgseną, skirtas tirti žuvų elgsenos pokyčius priklausančius nuo upės ekologinių ir gamtinių sąlygų.
- Elektros rinkos sudėtinga prisitaikanti sistema (angl. – *Electricity Market Complex Adaptive Systems Model*) modelis skirtas simuliuoti elektros energijos rinkos dalyvių elgseną.

1.1 Agentas

Agentais grįsto modeliavimo esmė – atskirų agentų tarpusavio sąveika. Agentas yra esminė šio modeliavimo būdo esybė, tačiau iš skirtingų straipsnių matosi, kad kol kas nėra sutarta tikslaus agento apibrėžimo [MNo09]. Galima apibrėžti kelis universalius požymius, kuriais turėtų pasižymėti agentas [MNo09] [AT14]:

- Agentas veikia savarankiškai. Savarankiškai bendrauja su kitais agentais ir aplinka, kurioje veikia. Agento veiksmų aprašymo galimybių spektras yra labai platus. Priklausomai nuo modelio sudėtingumo agento veikimas gali būti aprašomas paprastais sąlygos sakiniais, sudėtingais dirbtinio intelekto algoritmais ar atskiromis sistemomis veikiančiomis agento viduje;
- Agentas gali būti ir yra identifikuojamas kaip atskira esybė sistemoje. Taip pat jis turi griežtai apibrėžtas ribas, kurios nustato kas modelyje yra agento dalis, o kas ne;
- Agentas turi įdiegtus algoritmus, kurie apibrėžia kaip agentas bendrauja su kitais agentais ir aplinka, kurioje jis veikia. Šio funkcionalumo dėka agentas ne tik sugeba susikalbėti su kitais agentais sistemoje, bet taip pat jis atpažįsta kitus agentus, fiksuoja jų aktyvumą ir atitinkamai į jį reaguoja;
- Agentas turi konkrečią būseną konkrečiu metu ir jis žino kokioje būsenoje jis šiuo metu yra;
- Agentas simuliacijoje atitinkamai keičia savo elgseną ir prisitaiko prie besikeičiančių sąlygų;
- Agentas turi savo resursus, kurių kiekis veikiant modeliui kinta. Resursais gali būti pinigai, energija, atmintis, sąlyginiai sveikatos vienetai ir panašiai;

- Agentas gali turėti užbrėžtus tikslus, pagal kuriuos jis derins savo veikimą. Tikslai nebūtinai turi būti kaip galutinis rezultatas, bet jie gali būti naudojami kaip kokybiniai rodikliai pagal kuriuos agentas gali koreguoti savo veikimą.

Agentais grįstą modeliavimą nagrinėjant labiau iš informacinių sistemų modeliavimo pusės, agento elgsenų aprašymas susideda iš dviejų svarbių esybių: agento veiksmų ir tam tikrų mechanizmų, parenkančių atitinkamą agento veiksmą atitinkamoje situacijoje [BMV09]. Terminas „agento veiksmas“ yra esminė agento savybė, kuri gali pakeisti aplinką ar kitus modelyje veikiančius agentus. Agentas gali [BMV09]:

- Kartu su kitais agentais įtakoti visą sistemos būseną.
- Atlikti lokalius aplinkos pakeitimus.
- Atsakyti į atitinkamus veiksnius, įtakojančius agento veikimo erdvę.
- Vykdyti duomenų apdorojimo procesų agento viduje naudojant įvesties parametrus gautus iš aplinkos taip atitinkamai pakeičiant savo būseną.
- Pakeisti savo fizinę vietą aplinkoje.

Yra du agentų tipai [BMV09]. Agentas gali būti arba reaktyvus arba pažintinis (angl. – *Cognitive*). Reaktyvus agentas tai agentas be būsenos ir tuo pačiu „be atminties“. Jis veikia fiksuotoje aplinkos pozicijoje. Toks agentas atlieka veiksmų seką, įtakotą agento fiksuojamų veiksnių, kylančių iš aplinkos arba iš kitų agentų. Dažniausiai tokių agentų veiksmai yra aprašomi sąlygos sakiniais ir pasak šios straipsnio tokie agentai negali būti proaktyvūs. Kaip alternatyva reaktyviems agentams straipsnyje yra pateikiami pažintiniai agentai. Šie agentai turi savo būseną, saugo žinias apie aplinką, dažniausiai turi ir atmintį apie ankstesnes veikimo patirtis. Veikiant išorės veiksniams tokie agentai bando parinkti atitinkamą veiksmų seką, kad būtų pasiekti toje situacijoje būtini tikslai. Kaip tokių agentų panaudojimo pavyzdys yra pateiktas BDI (angl. – *Belief, Desire, Intention*) modelis. Pagal BDI, agento būsena yra sudaryta iš trijų duomenų struktūrų, kurios apibrėžia agento tikėjimą (informacija apie aplinką), troškimą (informacija apie tikslus) ir intencijas (konkrečiu metu, pagal konkrečią veikimo logiką pasirinkti troškimai – tikslai).

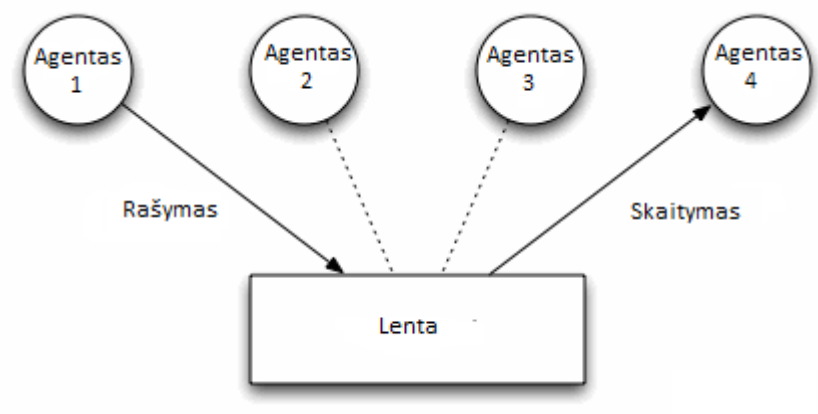
Nagrinėjant agentais grįstą modeliavimą labai svarbu apibrėžti agentų bendravimą su kitais agentais ir su aplinka. Išskiriami du bendravimo būdai [BMV09]: tiesioginis ir netiesioginis. Agentas kaip esybė turi bendrauti su aplinka ir su kitomis esybėmis kaip jis. Agentais sumodeliuotų sistemų esmė yra bendra sistemų dinamika, kurią sukuria agentų bendravimas tarpusavyje ir su aplinka [BMV09]. Agentų bendravimo mechanizmai daro didelę įtaką modelio

veikimui. Dažnu atveju šie mechanizmai nulemia ir patį sistemų modeliavimą, projektavimą ir kūrimą. Modeliavimo agentais būdas priklauso nuo to koks yra realizuojamas agentų bendravimo mechanizmas [BMV09].

Modeliuojant agentų tiesioginį bendravimą, dažniausiai yra nekreipiama dėmesio į kanalo, kuriuo keliautų pranešimai, modeliavimą ir net nėra matoma tokios būtinybės. Ir nors įmanoma modeliuoti pranešimų perdavimo kanalus pagal kompiuterinių tinklų protokolų pavyzdžius, yra siūloma naudotis Agentų komunikavimo kalba (ACL – Agent Comunication Language) [BMV09]. Nors ACL ir formalizuoja bendravimą tarp agentų, tačiau neišsprendžia agentų vienas kito radimo problemos [BMV09]. Tam, kad agentas *A* galėtų bendrauti su agentu *B*, agentas *A* turi žinoti agentą *B* vienareikšmiškai identifikuojantį vardą. Taip pat agentui *A* turi būti pranešta apie agentą *B* bei turi būti realizuotas agentų atpažinimas sistemoje. Šio uždavinio sprendimui yra siūloma kurti kitus agentus (tarpininkus) kurie būtų atsakingi už informacijos radimą, taip pat ir modelius atsakingus už ryšių tarp agentų modeliavimą (modeliai modeliuose) [BMV09]. Agentais grįstame modeliavime agentai nebūtinai turi elgtis kaip informacinių sistemų komponentai, pavyzdžiui kai sisteminiai komponentai bendraudami servisų pagalba vykdo veiksmus reaguodami į atitinkamas iš kitų komponentų ateinančias užklausas (SOA architektūra). Agentais grįstame modeliavime agentas nebūtinai turi sureaguoti į kitos esybės pasiųstą pranešimą. Agentai yra laikomi proaktyviomis esybėmis, t.y. agentai pagal vidinius mechanizmus gali patys nuspręsti kada ir kokią veikimo logiką jie turėtų taikyti [BMV09]. Teiginys, kad agentų proaktyvumas yra svarbus tik modeliavime, o ne informacinių sistemų projektavime [BMV09] nėra teisingas. Yra tokių pavyzdžių kaip išskaidytų komponentų programavimas specialiame karkase pagal SOA architektūros principus, panaudojant aktyvius objektus kur kiekvienas objektas yra izoliuota esybė valdoma vienos, jai priskirtos gijos [BHR13]. Tarpusavyje tokie objektai bendrauja asinchroniškai tam tikrų metodų iškvietimais. Objektas sulaukęs metodo iškvietimo stato jį apdorojimui į iškvietimų eilę ir veikdamas ant atskiros nuo kitų procesų izoliuotos gijos pagal savarankišką logiką sprendžia apie iškvietimų apdorojimą ar jų ignoravimą [BHR13]. Tai galėtų būti įdomi tema ateities agentų kūrimo uždaviniams spręsti.

Kalbant apie netiesioginį agentų bendravimą, bet koks bendravimas, netgi tas kas dokumentacijose yra vadinamas tiesioginiu, yra tam tikra prasme netiesioginis, nes bet kuriuo atveju visada turi būti realizuota vienokia ar kitokia infrastruktūra perduoti pranešimą iš vieno taško į kitą [BMV09]. Daugeliu atvejų, agentai bendrauja tarpusavyje, tuo pačiu dalindamiesi tam tikrus resursus. Standartinis automobilių eismo pavyzdys, kai automobiliai turi pasidalinti gatvėmis, o šviesoforai padeda jiems tą padaryti. Šviesoforas šiuo atveju yra artefaktas, o ne

agentas. Daugelis modeliavimo realizacijų modeliuoja tam tikrus bendrus resursus iš kurių ir į kuriuos skaitymas bei rašymas yra kontroliuojamas pagal iš anksto apibrėžtas taisykles [BMV09]. Tai iliustruoja mokyklinės rašymo lentos pavyzdys 1 pav. [BMV09]



1 pav.: Komunikavimo tarp agentų pavyzdys

Pavyzdyje pateiktas rašymo lentos – bendros duomenų saugyklos pavyzdys. Šiame modelyje veikiantys sisteminiai modeliai bendrauja tarpusavyje anonimiškai. Šis pavyzdys yra pateiktas pagal 1985 metų metodologiją, kurioje yra pristatomi komunikacijos sprendimai tarp skirtingų išskirstytų sistemų esybių [Gel85]. Rašymo lentos komunikavimo metodas yra vienas iš siūlomų sprendimų taip vadinamame Linda komunikacijos modelyje. Šis komunikacijos modelis, skirtas programų sistemų vidinių modulių komunikacijai, gali būti pritaikytas agentais grįstame modeliavime, rašymo lentos esybę apibrėžiant kaip artefaktą, panašiai kaip šviesoforus eismo simuliacijos modelyje, tik žinoma šviesoforai eisme jokios informacijos apie automobilius nesaugo [BMV09].

1.2. Modeliavimo agentais taikymai

Kaip jau buvo minėta Agentais grįstas modeliavimas yra neseniai atsiradusi modeliavimo koncepcija. Nepaisant to yra pateikta eilė pasiūlymų dėl šio modeliavimo būdo taikymų. Šiame skyriuje bus remiamasi šaltiniais, teikiančiais informacinių sistemų modeliavimo metodikas bei pasiūlymus grįstus modeliavimo agentais principais.

Yra siūloma modeliavimo agentais taikymo metodika skirta programų sistemų architektūrų tyrimui [Mik14]. Naudojamas programų sistemų architektūros apibrėžimas kaip rinkinys komponentų, jungčių ir konfigūracijų. Šiuo apibrėžimu yra siūloma paremti ir modeliavimą, t.y. modeliuoti atskirus komponentus, kurie susijungę ir sudaro programų sistemą. Komponentus yra siūloma modeliuoti kiekvieną atskirai: vienas programinis komponentas – vienas agentas [Mik14]. Kalbant apie pačio komponento savybes modelyje, yra pateikiama rekomendacijų

modelyje pasirinkti kaip įmanoma mažesnį komponento dydį, tai turėtų padėti lengviau prognozuoti komponento parametrus [Mik14]. Metodikoje išskiriamos pagrindinės komponento savybės [Mik14]:

- Komponentas turi savo parametrus, nusakančius įvairias komponento savybes;
- Komponentas vienu metu gali apdoroti baigtinį užklausų kiekį. Ši savybė yra modeliuojama komponento agentui pridedant maksimalaus užklausų kiekio parametrą;
- Komponentai bendrauja su kitais komponentais per taip vadinamus sąryšius.
- Komponento įeinantys ir išeinantys sąryšiai aprašomi komponento sąsajomis (angl. – *Interface*).

Yra aibė parametrų, kurie nusako koks komponentas yra agentas (Už kokį funkcionalumą programų sistemoje yra atsakingas) ir kokios to komponento savybės (Kaip yra įgyvendinimas konkretus funkcionalumas) [Mik14]. Vienas parametro pavyzdys jau yra minėtas maksimalaus vykdomų užklausų kiekio parametras. Kad būtų perteiktas kaip įmanoma labiau realistiškas programų sistemų veikimo vaizdas, yra siūloma naudoti parametrus, kurie leistų nurodyti tikimybę, kad komponentas yra neveikiantis ar, kad komponento veikimas pasibaigė klaida [Mik14].

Norint modeliuoti programų sistemų architektūrą būtina sukurti ne tik pačios iš komponentų susidedančios sistemos modelį, bet taip pat ir visą sistemos infrastruktūrą bei sistemos naudotojus. Programų sistemos veikimo simuliacijai yra būtina sukurti ryšius tarp programinių komponentų aplinkos, serverių, naudotojų ir kitų sistemų [Mik14]. Taip kaip kiekvienam programiniam komponentui yra sukuriamas atskiras agentas su savo programinio komponento parametrų reikšmėmis, taip pat yra sukuriami atskiri agentai kiekvienam programų sistemoje dalyvaujančiam serveriui, naudotojui ar kitai sistemai.

Taigi modelyje [Mik14] yra keturių rūšių agentai: *Serveris*, *Naudotojas*, *Programinis komponentas*, *Kita sistema*. Serverio ir kitos sistemos agentai lyginant su programinio komponento agentais turi sąlyginai nedaug parametrų (*Serveris* – 5, *Kita sistema* – 3). Serverio ir kitos sistemos agentai turi tokius parametrus kaip procesoriaus/atminties/tinklo resursų kiekiai, atsako laikas, klaidų tikimybė ir kiti. Taip pat modelyje yra ir sistemos naudotojų agentai, kurie yra aprašomi su daugiau parametrų, tokių kaip užklausų generavimo dažnumas, užklausų generavimo tikimybė, užklausų generavimo kiekis. Klasifikuojant skirtingus agentų tipus, serverio agentai yra priskiriami resursų modeliavimui, o naudotojo ir kitos sistemos agentai yra priskiriami aplinkos modeliavimui. Serverio agento penki parametrai aprašo tik resursų kiekį, kuris priklausomai nuo komponentų kreipimosi į serverį kinta pagal programinių komponentų

veikimo logiką. Todėl darytina prielaida, kad serverio agentas neturi savarankiškos veikimo logikos ir jo sukūrimo metu inicijuoti parametrai kinta tik priklausomai nuo programinių komponentų veikimo. Dėl šios priežasties ir serverio agento modeliavimą galima priskirti prie aplinkos modeliavimo.

Pateikta modeliavimo metodika [Mik14] yra skirta programų sistemų architektūrų tyrimui, konkrečiai yra pateikti SOA ir EDA modeliavimo pavyzdžiai. Kiekviename iš pavyzdžių visa architektūros veikimo logika yra perteikiama komponentų lygyje. Kitaip sakant modelyje yra sukuriami programiniai komponentai kurių išdėstymas, pavadinimai bei parametrai yra parengti pagal konkrečios architektūros specifikaciją. Modelis realizuotas Repast programiniame pakete, o veikimo logika realizuota Java programavimo kalba. Pateiktoje Java realizacijoje yra aprašyta klasės *„Užklausa“* ir *„Užduotis“*. Skirtingos klasės *„Užduotis“* yra klasės *„Užklausa“* vaikiniai objektai sugeneruojami užklausų vykdymo metu. Metodikoje nėra tiksliai aprašoma koks šių klasių vaidmuo modelyje. Sprendžiant iš klasių kintamųjų jų parametrai neatitinka nei vieno standartinio objekto *„Komponentas“, „Naudotojas“, „KviečiamaSistema“, „Serveris“*.

Ši siūloma metodika [Mik14] paaiškina kokius parametrus turi turėti kiekvienas agentas atliekantis savo vaidmenį modelyje. Šioje siūlomoje metodikoje nėra kalbama apie konkrečią situacijos realizaciją (programiniu kodu), todėl kaip ir ankščiau minėtame pavyzdyje yra sunku atskirti kur tam tikra esybė modelyje yra agentas, o kur kita modelio esybė. Ši apžvelgta metodologija taip pat neaprašo galimybės agentams turėti tam tikrą būseną. Minėtas serverio agentas, atstovaujantis modelyje vienai konkrečiai programų sistemos esybei, gali turėti tam tikrą būseną kaip atskirą požymį galinti įgauti reikšmę. Pavyzdžiui, serverio būsena *„įjungtas“*. Jeigu prie Serverio tipo agento būtų norima pridėti dar vieną būsenos požymį, būtų galima tiesiog agento aprašą papildyti tokio pačio pavadinimo komponentu *„įjungta“* su galimomis reikšmėmis 0 arba 1. Tačiau norint priskirti tam tikrą būseną agentui sukurtam programinio komponento pagrindu užduotis tampa sudėtingesnė. Visų pirma nežinant kokio būtent agento, kokį atributą ar atributus bus tikslinga naudoti kaip vienus iš agento būseną apibūdinančių parametrų, bus neįmanoma prieš inicializuojant modelį priskirti konkrečius parametrus programinių komponentų agentams. Pavyzdžiui siūlomame SOA architektūros modelyje Verslo procesų sluoksnio agentai Procesas ir Subprocesas turės kitokius būsenų parametrus (Pavyzdžiui parametrą: *ProcesoApdorotuUzklausuKiekis*) nei paslaugų sluoksnio paslaugų agentai (Pavyzdžiui parametrą: *PaslaugosKvietimųSkaicius*). Todėl arba modelyje prieš inicijavimą yra būtina aprašyti visus išvestinius agentus iš agento tipo *„Programinis komponentas“* arba sukurti būdą kaip parametrui priskirti struktūrų masyvą: *Būsenos[]({BusenosVardas; Busenos reikšmė})*.

Tokiu būdu tampa įmanoma formalizuojant modelį aprašyti galimas agento būsenas inicializuojant konkretų agentą programinio komponento pagrindu.

Tikslesni agentų su būsena panaudojimo pavyzdžiai yra aprašomi Java [CN14] programavimo kalba. Nagrinėjant pateikiamus agentų klasių pavyzdžius yra lengviau suprasti agentų, turinčių būseną modeliavimo principus. Pavyzdyje yra modeliuojama biologinio viruso verčiančių žmones zombiais plitimo simuliacija [CN14]. Agentus *„Žmogus“* ir *„Zombis“* aprašo dvi atskiros Java klasės, kuriose yra nurodomi atitinkami parametrai aprašantys šių dviejų agentų tipų atitinkamas būsenas. Agento *„Zombis“* klasė Java kalba yra apibrėžiama taip kaip pavaizduota 2 pav. [CN14]:

```
public class Zombis
{
     private ContinuousSpace < Object > erdve;
     private Grid < Object > tinklelis;
     public Zombis ( ContinuousSpace < Object >
erdve , Grid < Object > tinklelis )
   {
     this . erdve = erdve ;
     this . erdve = erdve ;
   }
}
```

2 pav.: Agento klasės Java programavimo kalba, pavyzdys

Šios klasės būseną nusakantys kintamieji yra *„erdve“ ir „tinklelis“*. Šie du kintamieji nusako agento *„Zombis“* koordinates erdvėje (kintamasis *„erdve“*) ir jo santykį su kitais simuliacijoje veikiančiais agentais (kintamasis *„tinklelis“*). Šie du kintamieji yra ir agento *„Žmogus“* klasėje, kartu su papildomais kintamaisiais *„energija“* ir *„pradineEnergija“*, nusakančiais modelyje veikiančio *„Žmogaus“* sveikatos būklę *t* laiko momentu ir momentu kai yra inicializuojama simuliacija. Straipsnyje vykdomoje simuliacijoje *„Zombis“* tipo agentai simuliacijos pabaigoje sunaikina visus *„Žmogus“* tipo agentus. Jeigu simuliaciją papildžius papildomais agentais, kurie naikintų agentus *„Zombis“*, jie turės tuos pačius būseną erdvėje nusakančius parametrus *„erdve“* ir *„tinklelis“* bei papildomus parametrus. Pavyzdžiui, sukūrus agentą *„Medikas“*, jo klasė galėtų turėti parametrą *„medikamentuKiekis“* arba agento *„Snaiperis“* klasei būtų aktualus, būklės parametras *„saudmenuKiekis“*. Tokiu atveju įmanoma apibrėžti abstraktų agentą *„biologineKlase“*, kuris turės būsenos parametrus nusakančius jo padėtį erdvėje, o visi išvestiniai jį realizuojantys agentai turės atitinkamai papildomus kitus parametrus. Grįžtant prie ankščiau apžvelgtos siūlomos modeliavimo agentais metodologijos

[Mik14], toks abstraktus agentas šiuo atveju yra *„PrograminisKomponentas“* jam galima priskirti parametrus nusakančius tam tikrą būseną. Programų sistemų architektūrų ir mastelio keitimo savybių tyrimo simuliacijose kuriant agentus *„PrograminioKomponento“* pagrindu yra tikslinga rasti būdą būsenos parametrus kurti priklausomai nuo konkretaus išvestinio agento veikimo logikos. Šiuo atveju yra reikalinga pateikti sprendimą aprašyti būsenų naudojimą abstrakčioje klasėje ir atitinkamų būsenų realizavimą išvestinėse klasėse. Nagrinėjant mastelio keitimo klausimų sprendimą modeliavimo agentais metodikomis, bus remiamasi siūloma metodologija [Mik14]. Šiame darbe ši metodika nebus pildyta agentų su būsena palaikymo galimybe, veikiančių agentų statusas bus nustatomas kitais sprendimais.

1.3. Modeliavimo aplinka

Kuriant sistemos simuliaciją agentais paremto modeliavimo principais yra ypač svarbu tiksliai apibrėžti agento ir aplinkos sąvokas bei jų atributus ir metodus. Dažnu atveju iš karto nepavyksta nuspręsti, kas iš visų sistemos atributų ir požymių turėtų būti prisikirta agentams, o kas aplinkai [MNo09]. Pavyzdžiui, modeliuojant teritorijos nuo branduolinio užterštumo valymo sistemą reikia apsispręsti ar radioaktyvaus spinduliavimo šaltinį laikyti agentu ar aplinkos veiksniu, kuris daro įtaką kitiems agentams. Šio darbo atveju, tokia dilema gali iškilti, pavyzdžiui, modeliuojant informacines sistemas ir kylančius tinklo sutrikimus juose: ar tinklo trukdžiai sklinda iš konkretaus *„Trukdžių agento“* ar jie yra aplinkos dalis.

Java ir C++ programavimo kalbos yra ne vieninteliai agentais grįsto modeliavimo realizavimo sprendimai. Kitas agentais grįsto modeliavimo aprašymo pavyzdys remiasi NetLogo modeliavimo aplinka [Jan12]. Šiuo atveju yra apibrėžiama bet kokios modeliavimo aplinkos esmė, tai yra algoritmai sukuriantys atitinkamas komandas, kurios apskaičiuoja agentų atributų pasikeitimus. Taip pat yra formalizuotas modelio veikimas NetLogo aplinkoje. Šis apibrėžimas, žinant modeliavimo aplinkų veikimo principus gali būti pritaikytas visiems modeliams leidžiamiems agentais grįsto modeliavimo aplinkose: $F(x,p)_t = F(x,p)_{t-1}$. Čia F – modelis, x – kintamųjų modelyje reikšmių aibė, p – modelio parametrai, t – laiko momentas. $F(x,p)_0$ yra inicijuojamas modelis.

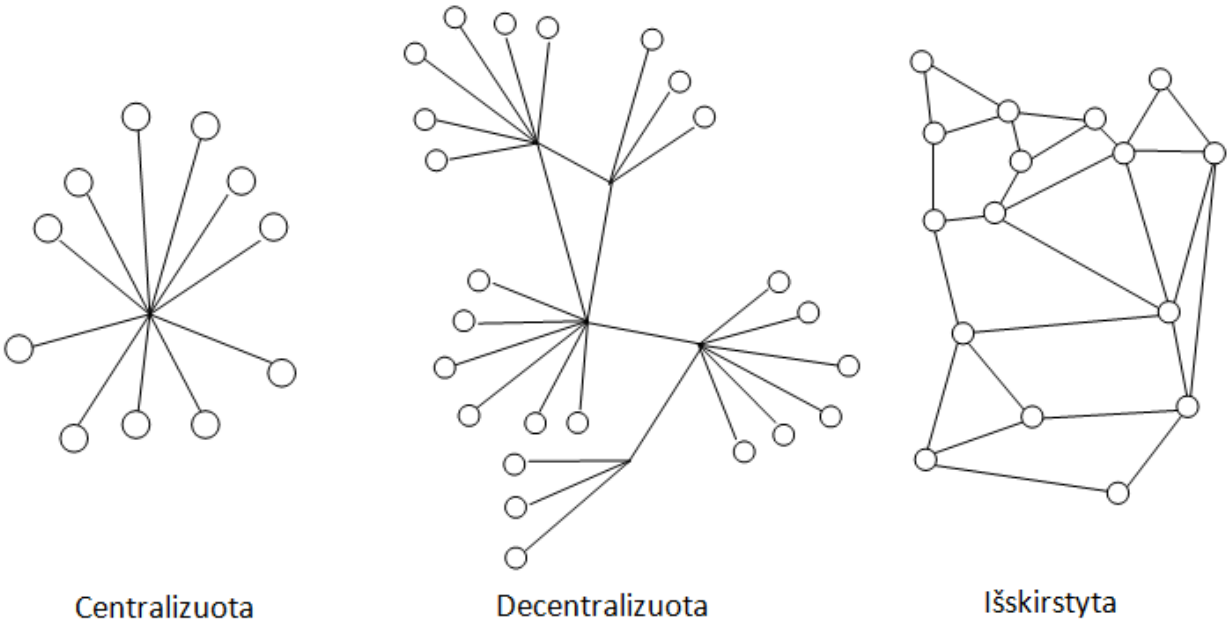
Nagrinėjant skirtingas modeliavimo aplinkas (platformas) ir jas apžvelgiančius šaltinius matosi, kad pati pirmoji Agentais grįsto modeliavimo terpė buvo Swarm kurios pagrindu buvo kuriamos kitos modeliavimo aplinkos [RLJ06] [Ber08]. Swarm vadinamas visų modeliavimo terpių pagrindu ir Swarm sukūrimas laikomas noru apibrėžti Agentais grįsto modeliavimo pagrindus ir principus [RLJ06]. Kitos aplinkos yra aprašomos, taip kaip jos realizuoja Swarm modelį. Kai kada Swarm yra laikoma tik kaip pirmoji agentų programavimo kalba ir su kitomis

aplinkomis nelyginama, nes jos realizuoja Swarm principus [Ber08]. Lyginant aplinkas NetLogo yra minima paprastos, edukacinės sistemos šviesoje, o Repast yra laikoma sudėtinga ir galinga sistema, lyginant priemones, kuriomis yra aprašoma agentų veikimo logika Repast suteikia daugiausiai galimybių [RLJ06] [Ber08]. Nors ir yra kalbama, jog pagrindine programavimo kalba yra Java [RLJ06], bet Repast oficiali dokumentacija mini ir daugiau įvairių programavimo kalbų panaudojimo galimybių, o dauguma pavyzdžių yra aprašyti C++ kalba [Col13]. Šiuo požiūriu Repast lenkia kitas aplinkas savo plačiu programavimo kalbų palaikymu, ypač ankščiau minėtą NetLogo, kuri palaiko tik mokymo tikslais naudojamą Logo kalbą.

2. BlockChain technologija

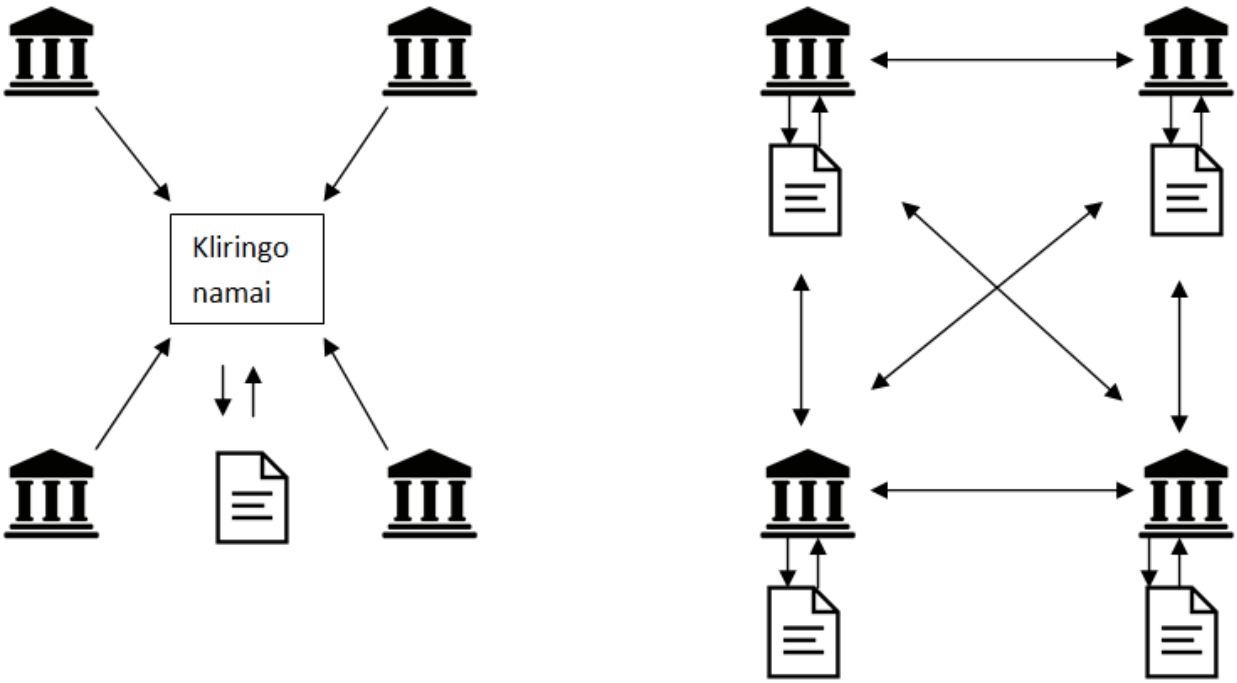
BlockChain technologija per paskutinius metus milžiniško populiarumo sulaukusi sritis [Shi16]. Labiausiai BlockChain technologijomis šiuo metu domisi finansų sektorius, bet nemažo populiarumo ši technologija sulaukia taip pat ir socialinėse, teisinėse ir kitose srityse [Gre16]. BlockChain technologija yra laikoma kaip viena iš *Fintech* varomųjų jėgų [Shi16]. Paskutiniu metu didėjantis susidomėjimas šia technologija neslopsta. Tiek pradedantieji startuoliai, tiek ir didžiulės korporacijos (IBM, Deloitte, Microsoft ir kitos) investuoja resursus ir laiką į šios technologijos nagrinėjimą ir panaudojimo galimybių paiešką.

BlockChain tai išskirstytas tinklas (*peer-to-peer* tipo) realizuojantis išskirstytos buhalterinės knygos (angl. – *Distributed ledger*) funkcionalumą. Išskirstytos sistemos tipas palyginus su kitais sistemų tipais pavaizduota 3 pav. [Nor16]:



3 pav.: Skirtingos sistemų topologijos

Distributed Ledger gali būti realizuojamas įvairiomis technologijomis, tačiau populiariausia šiuo metu yra BlockChain. Šios technologijos pagrindu veikia populiariausios kripto valiutų platformos tokios kaip BitCoin [Sat09], *Ethereum* [But15] ir kitos. Kiekviena iš kripto valiutų savaip įgyvendina BlockChain principus. Dažniausiai BlockChain skirtumas nuo „tradicinių“ centralizuotų sistemų yra pavaizduojamas kliringo pavyzdžiu. Viena iš BlockChain panaudojimo galimybių yra tarpininko (kliringo namų) mokėjimų tvirtinimo procese eliminavimas, kaip tai yra pavaizduota 4 pav. [Nor16]:



4 pav.: Kliringo schema: centralizuotas sprendimas (kairėje) ir BlockChain sprendimas (dešinėje)

Šis pavyzdys iliustruoja BlockChain privalumus už kripto valiutų ribų. Mokėjimų ar kitų transakcijomis paremtų sistemų dalyviams nebūtina savo pusėje valdyti atskirų duomenų bazių, kuriose reiktų užtikrinti, kad būtų užpildyta naujausiais kiekvienos transakcijos įrašų duomenimis. BlockChain leidžia visiems tinklo dalyviams pasiekti vieningą transakcijų duomenų bazę. Turėti ją savo pusėje ir būti tikriems, kad duomenys su kuriais jie dirba yra naujausi ir visi dalyviai juos mato vienodai [Hyp16], [Sat09].

2.1. Kaip veikia BlockChain

Nėra vieningos galutinės specifikacijos kas yra BlockChain technologija. Tačiau bendru sutarimu [Ant14][Nor16][Nak09][Hyp16][But14] minimas apibrėžimas, kad BlockChain yra išskirstyta „*Peer-to-peer*“ principu veikianti duomenų bazių sistema, pasižyminti saugumo, didelio greičio ir informacijos nepaneigiamumo savybėmis [Nor16][Hyp16]. BlockChain apibūdinama kaip duomenų struktūra leidžianti sukurti bendrą paskirstytą skaitmeninę buhalterinę knygą (angl. – *Distributed digital ledger* – toliau Knyga) ir ją bendrai naudoti įvairių įrenginių tinkle (angl. – *Internet of things*). Šios technologijos pagrindas yra kriptografija [Nak09], kuri leidžia kiekvienam tinklo dalyviui saugiai redaguoti Knygą be būtinybės turėti tam tikrą centrinę reguliuojančią instituciją. Pirmajame, labiausiai paplitusiame BlockChain realizuojančiame sprendime Bitcoin, buvo aprašytas esminis centrinės (trečiosios) pusės

eliminavimas [Nak09]. Tam, kad tai būtų įgyvendinta buvo būtina išspręsti buhalterijoje žinomą „dvigubo išleidimo“ (angl. – *Double spending*) problemą.

Kai tik tam tikra informacija yra įrašoma į BlockChain ją tampa labai sunku ar beveik net neįmanoma [Ant14] pakeisti ar pašalinti. Kai tinklo dalyvis nori papildyti tam tikru įrašu Knygą, visi tinklo dalyviai turi patvirtinti ar nauja transakcija gali būti įrašyta į Knygą. Jeigu dauguma dalyvių patvirtina, kad siūlomas naujas įrašas pagal praėjusių transakcijų istoriją yra galimas, tada atitinkamu įrašu yra papildoma Knygos blokų grandinė. Kiekvienas iš dalyvių gali įvertinti naujo įrašo atitikimą istorijai todėl, kad kiekvienas dalyvis savo pusėje saugo BlockChain kopiją [Nak09]. Patvirtinus naują įrašą kiekvienas iš dalyvių atnaujina savo saugomą kopiją. Užpildytas blokas visam laikui saugo į jį įrašytas transakcijas. Užbaigtame bloke surašytos transakcijos negali būti pakeistos. Toks bendras sutarimo priėmimas BlockChain tinkle yra vadinamas Konsensusu.

Yra daug įvairių BlockChain platformų, bet absoliučią jų daugumą vienija bendras veikimo principas [But14][Nak09][Hyp16]:

- Kuriamos naujos transakcijos ir yra paskleidžiamos BlockChain tinkle.
- Specialūs tinklo dalyviai Išgavėjai (angl. – *Miners*) pradeda surinkinėti transakcijas į blokus.
- Išgavėjai tvirtina transakcijas ir deda jas į blokus.
- Kai išgavėjas sugeneruoja bloką, blokas yra pridedamas prie bendros blokų grandinės. Apie jį sužino visi tinklo dalyviai ir atsisiunčia jį į savo turimas BlockChain kopijas.
- Tinklo dalyviai priima ir patvirtina bloką tik tada, jei visos transakcijos jame yra teisingos ir ankščiau nepriimtos (taip išvengiant „dvigubo išleidimo problemos“).
- Tinklo dalyviai patvirtina bloką pradėdami dirbti prie naujo bloko visi bendrai naudodami ką tik priimto į grandinę bloko *hash* reikšmę kaip nuorodą grandinėje į prieš tai buvusį bloką.
- Blokas yra tarsi buhalterinės knygos lapas į kurį yra surašomos naujausios transakcijos, kurios nebuvo iki tol papuolusios į jokį kitą bloką. Kai blokas yra užpildomas jis yra „užverčiamas“ ir yra „atverčiamas“ naujas blokas pildymui.

Tinklo dalyviai teisinga grandine laiko ilgiausią grandinę. Kai kuriems tinklo dalyviams gali tekti persijungti prie kitos grandinės, paaiškėjus, kad grandinė prie kurios jie dirba jau nebėra ilgiausia.

Čia yra aprašytas bazinis BlockChain veikimas, kuris yra sutinkamas daugumoje šią technologijų realizuojančių platformų. Skirtingos platformos skirtingai įgyvendina tam tikrus BlockChain aspektus.

Nors skirtingas BlockChain platformas ir sieja bendras blokų grandinių veikimo principas, tačiau visos jos skiriasi tam tikrais parametrais, kurie klasifikuojami taip [Ant14]:

Koks yra konsensuso algoritmas?

Tam, kad būtų išvengta minėtos dvigubo išleidimo problemos, visas tinklas turi prieiti prie bendro sutarimo dėl kiekvienos transakcijos tikrumo ir teisingumo. Dabartinėse BlockChain principus realizuojančiose platformose konsensusą galima pasiekti dviem skirtingais būdais arba jų sąveika:

- Įrodymas darbu (angl. – *Proof-of-work*) – turima aparatūrine įranga atliekant tam tikras aritmetines operacijas;
- Įrodymas turtu (angl. – *Proof-of-stake*) – tam tikrų protokolų pagalba įrodant nuosavybės teisę į tam tikrus platformos kripto valiutos kiekius.

Koks yra kripto valiutų generavimo mechanizmas?

BlockChain platformoje fiksuotas valiutos kiekis gali būti išplatintas platformos kūrimo metu, naudojant valiutą kaip priemonę pritraukti investicijas. Taip pat ji gali būti platinama kaip paskata prisidėti prie bendro platformos saugumo (BitCoin, Ethereum pavyzdys). Egzistuoja ir tokių platformų, kurios neturi jokios kripto valiutos (HyperLedger pavyzdys);

Ar tai privati ar vieša platforma?

Populiariausia BlockChain realizacija – BitCoin sistema yra vieša platforma prie kurios gali prisijungti visi norintys. Nėra jokio teisių valdymo mechanizmo, visi sprendimai dėl platformos vystymo yra priimami bendruomenės [Nak09]. Egzistuoja platformų, kurios gali būti diegiamos organizacijų (organizacijų grupių, valstybių) viduje, prisijungimas prie kurių būtų leidžiamas tik iš anksto patvirtintiems dalyviams.

2.2. Populiariausios platformos

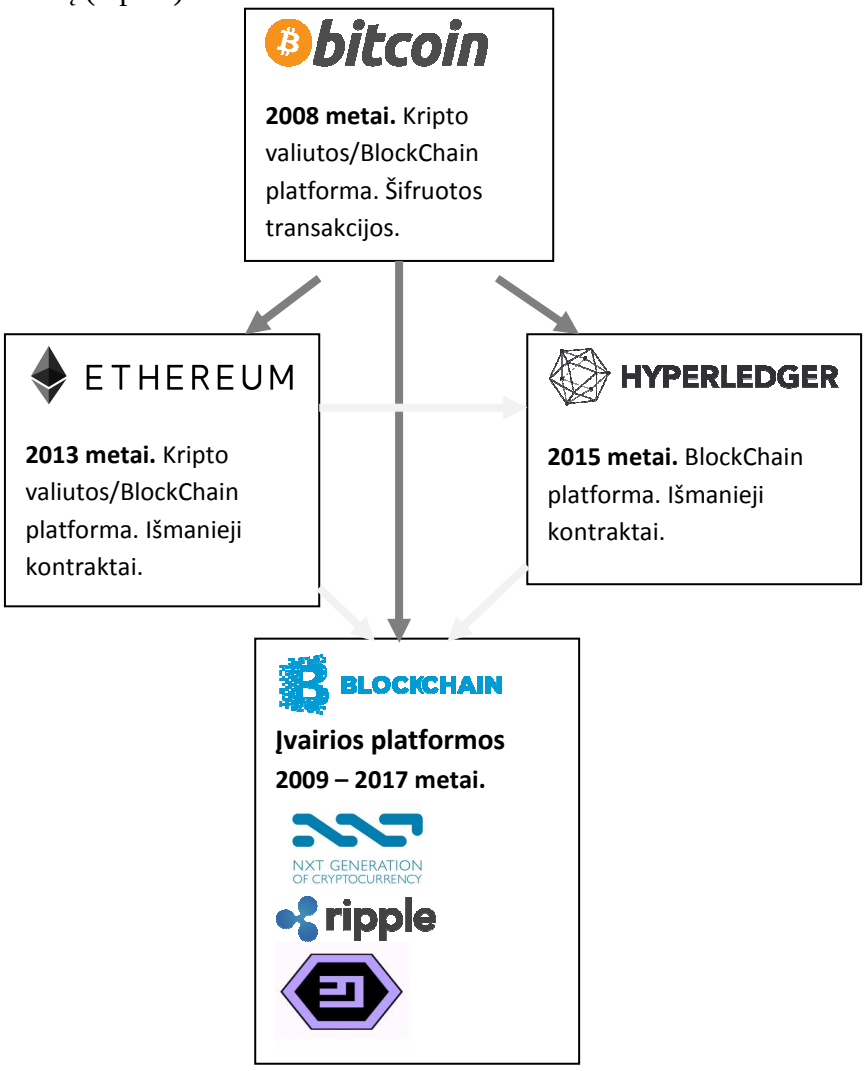
- BitCoin platforma [Nak09] – populiariausia kripto valiutos platforma. Atsiskaitymus BitCoin valiuta (toliau tekste – BTC) leidžia daugiau nei šimtas tūkstančių vietų, tarp jų: Expedia, Microsoft, Dell ir kitos. Taip pat naudojama duomenų nepaneigiamumui, autentiškumui užtikrinti. Rinkos kapitalizacija: 10,2 Mlrd. $, Naujas blokas sukuriamas per: 600 sekundžių, transakcijos kaina 0,08 $, Transakcijų per dieną: 200 000, tinklo dalyvių: 7100, konsensuso metodas: „*Proof of work* “.

- *Ethereum* platforma [But15] – platforma palaikanti išmaniuosius kontraktus. Kripto valiuta *Ether* antra pagal populiarumą kripto valiuta po BitCoin. Naudojama autonominių organizacijų, identifikavimo sistemų kūrimui. Rinkos kapitalizacija: 1 Mlrd. $. Blokas

sukuriamas kas: 12 sekundžių. Transakcijos kaina: 0,007 $. Transakcijų per dieną 40 000. Konsensuso metodas: „*Proof of work* “.

- *HyperLedger* platforma [Hyp16] – Kuriamas *Linux foundation* suburto konsorciumo, į kurį įeina tokios korporacijos kaip IBM, Cisco, Fujitsu, Hitachi ir kitos. Pagrindinis skirtumas nuo prieš tai apžvelgtų BitCoin ir *Ethereum* platformų yra tas, kad tai platforma orientuota į privačius tinklus, kontroliuojamus tam tikro organizacinio vieneto. Taip pat palaiko išmaniuosius kontraktus, bet neturi savo kripto valiutos vienetų. Konsensuso metodas: „*Proof of stake*“.

BitCoin esant pirmąją kripto valiutų platforma, absoliuti dauguma BlockChain platformų yra susikūrusios būtent iš BitCoin kiekvienai pridedant tam tikrą savo funkcionalumą. Remiantis šių trijų platformų dokumentacija galima pavaizduoti jų vystymosi ir esminių papildomų galimybių lyginant su BitCoin kelią (5 pav.):



5 pav.: BlockChain platformos

5 pav. pavaizduotos ne visos šiuo metu egzistuojančios BlockChain platformos. Egzistuoja įvairių BlockChain sprendimų, skirtų įvairioms užduotims (*Riple*, *NXT*, *EmerCoin* ir kitos). Įvairios platformos greitai atsiranda, taip pat greitai ir išnyksta. BitCoin, *Ethereum* platformos sėkmingai veikia nuo jų įsikūrimo, *HyperLedger* nors ir buvo inicijuota palyginus neseniai (2015 metų pabaigoje), tačiau minėtų didžiųjų korporacijų dalyvavimas šio projekto kūrime leidžia tikėtis, kad ši platforma turi perspektyvų.

Atsižvelgiant į tai, kad BitCoin šiuo metu yra pati populiariausia, daugiausia tinklo dalyvių turinti BlockChain sistema (pagal 2017 metų I ketvirčio http://www.blockchain.info duomenis), o taip pat, kad tai yra bazinė BlockChain platforma, kurios veikimo principus įgyvendina absoliuti dauguma kitų BlockChain realizacijų, tolimesnis darbas remsis būtent šia platforma.

2.3. BitCoin veikimo principas.

2009 metais BitCoin įkūrėjo ar slapyvardžiu Satochi Nakamoto (kol kas nėra žinoma kam priklauso šis slapyvardis) pasivadinusios įkūrėjų grupės išleistas straipsnis [Nak09] nusako pagrindinius BitCoin veikimo ir pačios pirmosios BlockChain pagrindu veikiančios sistemos pagrindus.

2.3.1. Transakcijos

BitCoin valiuta tai transakcijos [Nak09][DSt16]. Įvairiuose šaltiniuose bei žiniasklaidoje minimos BitCoin (toliau – BTC) monetos iš tikrųjų neegzistuoja, nei pačiame tinkle, nei tinklo dalyvių pusėje. Kiek ir kokiem tinklo dalyviam priklauso BTC yra nustatoma tik iš transakcijų įrašų BlockChain tinkle. Siunčiant tam tikrą kiekį „monetų“ (angl. – *token*) siuntėjas nurodo gavėjo adresą tinkle ir pasirašo transakciją parašu sugeneruotą pagal jo turimą privatų raktą. Tai nereiškia, kad siuntėjas perdavė BitCoin valiutos vienetus iš savo adreso į gavėjo adresą, tiesiog BlockChain atsirado naujas įrašas apie transakciją pagal kurią galima vienareikšmiškai identifikuoti naują konkrečių BTC vartotoją, kuris nuo šiol galės leisti jam priskirtus BTC. Taip pat įvertinti kiek BTC vienetų sumažėjo siuntėjo balansas. Naujasis savininkas tai tinklo vartotojas, kuris nuo šiol gali leisti naujai jam prisikirtas lėšas, pasirašydamas savo privačiu raktu, į kurio, pagal viešą raktą sugeneruotą adresą buvo prisikirta valiuta. Kuriant transakciją reikalinga nurodyti ne tik siunčiamą sumą ir gavėjo adresą BitCoin tinkle, bet taip pat ir nuorodą į prieš tai buvusias transakcijas, kuriose yra ankščiau perduota atitinkama BTC suma, kurios užtektų kuriamai transakcijai įvykdyti.

Transakcija tai tokia duomenų struktūra, kurios svarbiausi nariai yra „įvestis“ (angl. – *input*) ir „išvestis“ (angl. – *output*). Nepanaudota išvesties dalis yra vadinama *UTXO* (angl. – *unspent transaction output*). Iliustruoti transakcijų veikimo pavyzdį galima tokiu pavyzdžiu. Pavyzdžiui, klientas pardavėjui nori pervesti 0.6 BTC. Kliento *UTXO* bendrai sudaro 0.7 BTC, kuriuos jis gavo trimis skirtingomis transakcijomis: 0.2 BTC, 0.2 BTC ir 0.3 BTC. Kliento programinė įranga turi sukurti atitinkamą transakciją, kuri surinktų visus galimus *UTXO*, kurių bendra suma būtų lygi arba viršytų 0.6 BTC. Programinė įranga šiuo atveju surinktų visus *UTXCO* ir turėtų užtikrinti, kad pardavėjo adresui bus priskirta 0.6 BTC, o likę 0.1 BTC kaip grąža turėtų būti pervesti atgal į pirkėjo adresą. Jeigu būtų pamiršta nurodyti (ar specialiai nenurodyta) grąžos pervedimo adreso, 0.1 BTC būtų laikomas kaip transakcijos mokestis (angl. – *transaktion fee*) ir atitektų transakciją patvirtinusiam (bloką sugeneravusiam) tinklo dalyviui. Taigi tinkle nėra balansų, valiutos likučių. Bet kurioje BitCoin piniginėje matomi „balansai“ tai *UTXO* suma, kuria galima išleisti pasinaudojus privačiu raktu. Pametus privatų raktą nėra jokių kitų galimybių atstatyti teisę į BitCoin valiutą [Ant14].

Kiekvienas dalyvis BitCoin tinkle turi savo privatų ir viešąjį raktus, kurie yra naudojami autentifikavimui bei transakcijų pasirašymui. BitCoin tinklo dalyvis pagal transakcijų įrašus į savo adresą gavęs atitinkamą kiekį BitCoin valiutos gali siųsti juos kitiems tinklo dalyviams. Tokią teisė disponuoti tam tikrame adrese esančiais BitCoin valiutos vienetais yra suteikiama pagal turimą adreso viešąjį raktą, kuris atitinka tinklo vartotojo turimą privatų raktą. BitCoin adreso ir privataus rakto ryšį iliustruoja C# programavimo kalbos kodo fragmentas (6 pav.) [DSt16]:

```
Key privateKey = new Key();
PubKey publicKey = privateKey.PubKey;
var publicKeyHash = publicKey.Hash;
var mainNetAddress = publicKeyHash.GetAddress(Network.Main);
```

6 pav.: Ketvirtoje eilutėje perduodamas argumentas *Network.Main* gali būti pakeistas į *Network.TestNet* taip persijungiant į testinį BitCoin tinklą

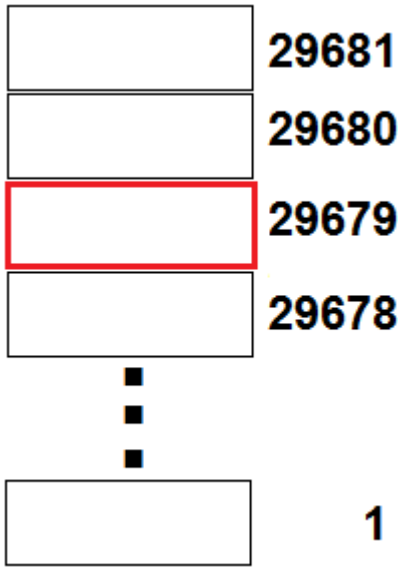
6 pav. matosi, kad tam, jog vartotojas norėtų gauti BTC jam tereikia sugeneruoti privatų ir viešąjį raktą, o pagal viešąjį raktą sugeneruoti adresą, kurį saugiai galima pateikti vartotojams, kurie turėtų siųsti BTC.

Visos BitCoin transakcijos yra viešos ir gali būti peržiūrėtos per specialius servisus (Pavyzdžiui http://www.blockchain.info). Tai yra visiškai anonimiška, nes BitCoin adresas yra sugeneruojamas pagal vartotojo privatų raktą. Pagal adresą neįmanoma identifikuoti privataus

rakto. Tam yra naudojamos *Hash* funkcijos. Tai deterministinės funkcijos (Perduodant vienodus argumentus gaunami vienodi rezultatai). Naudojamas *SHA-256 hash* algoritmas.

2.3.2. Blokai

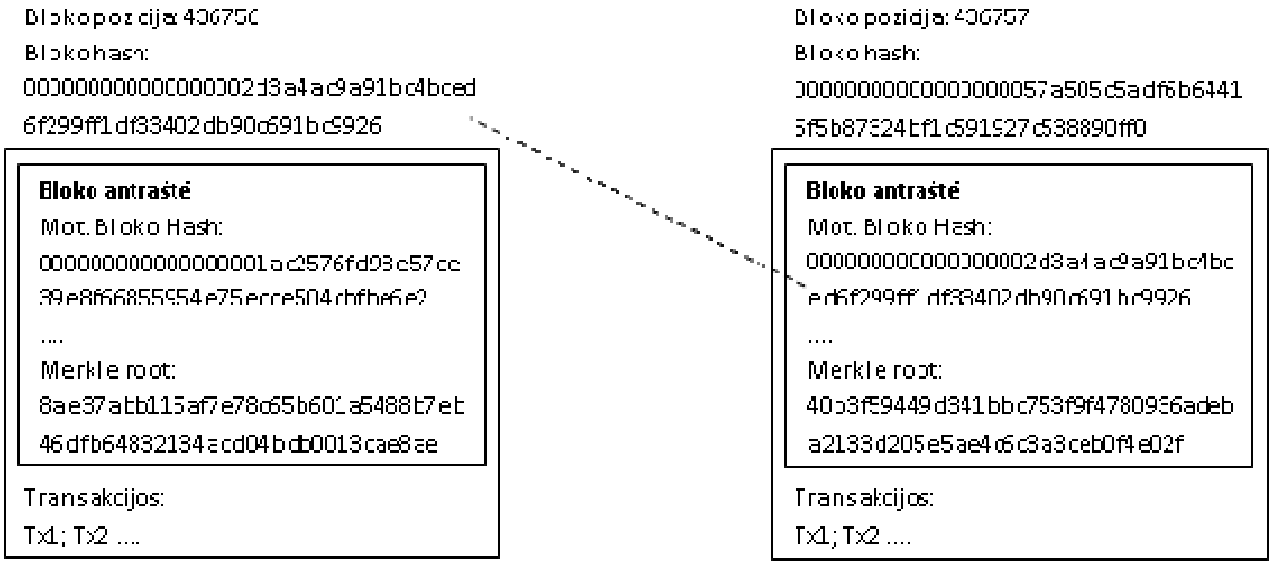
BlockChain yra paremtas blokais į kuriuos tarsi į buhalterinės knygos lapus yra surašomi transakcijų įrašai. Pavadinimas „Blokų grandinė“ puikiai nusako veikimo schemą. Blokai gali būti įsivaizduojami kaip sudėti vienas ant kito. Kiekvienas blokas turi nuorodą į prieš tai buvusį (motininį) bloką [Ant14][Nak09]. Terminas „aukštis“ nusako kokiame aukštyje yra konkretus blokas pradedant nuo pirmojo bloko grandinėje. Terminas „gylis“ nusako kokiam gylyje yra blogas pradedant nuo paskutinio bloko grandinėje. Bloko aukščio ir gylio pavyzdys yra pavaizduotas apačioje 7 pav. [Ant14]:



7 pav.: Pažymėto bloko aukštis: 29679, gylis: 3

Tai yra vienas iš galimų bloko identifikavimo būdų, bet jis nėra visiškai patikimas. Nors du blokai ir negali užimti tos pačios vietos grandinėje, tačiau gali pasitaikyti atvejų kai du ar daugiau blokų yra sugeneruojami ir bandomi pridėti į grandinę vienu metu [Ant14]. Patikimesnis blokų identifikavimo būdas yra pagal specialią SHA-256 algoritmu sugeneruotą *hash* reikšmę, kuri yra sugeneruojama pagal bloko antraštės informaciją. Toje pačioje antraštėje taip pat yra prieš tai einančio bloko *hash* reikšmė, kaip nuoroda į motininį bloką. Taip visi blokai yra sujungiami į blokų grandinę kurios pačioje pradžioje yra pats pirmasis 2009 metais sukurtas BitCoin blokas (angl. – *Genesis block*). Kiekvieno bloko identifikacinė *hash* reikšmė yra sugeneruojama dvigubos SHA-256 funkcijos (SHA-256(SHA-256()) pagalba. Kadangi šios funkcijos argumentas yra bloko antraštė, taigi taip pat kartu yra perduodama ir prieš tai buvusio

bloko *hash* reikšmė. Būtent toks mechanizmas užtikrina BlockChain duomenų nepaneigiamumo savybes. Pakeitus bet kokio bloko duomenis, automatiškai pasikeistų jo *hash* reikšmė ir dėl to prasidėtų grandininė reakcija visuose kituose po jo einančiuose blokuose, dėl ko kiltų jų perskaičiavimo būtinybė. Ir nors toks perskaičiavimas yra įmanomas naujiems ką tik prijungtiems ar 2–3 blokų gylyje esantiems blokams, 100 blokų gylyje esančios informacijos dėl BitCoin algoritmų ir skaičiavimo apribojimo pakeisti neįmanoma [Ant14][Nak09]. Panašiai yra užtikrinamas ir pačių transakcijų bloko viduj nepaneigiamumas. Kiekvieno bloko antraštėje yra laukas „*Merkle root*“, kuriame yra įrašoma speciali *hash* reikšmė. Bloke transakcijos yra sudėliojamos į specialaus tipo dvejetainį *Merkle* medį. Toks apverstas medis viršuje turintis šakninį elementą, o apačioje „lapus“ į kuriuos įeiną visų bloko transakcijų *hash* reikšmės sudaro vieningą *hash* reikšmę, vienareikšmiškai identifikuojančią transakcijų rinkinį. Ši vieninga *hash* reikšmė yra įrašoma į medžio viršuje esančią šaknį. Nesvarbu ar bloke būtų dvi ar tūkstantis transakcijų, algoritmas visada sudaro fiksuoto ilgio, 32 bitų *hash* reikšmę, pagal kurią įmanoma patikrinti bet kokios bloke esančios transakcijos egzistavimą. Šį blokų ir jų viduje esančių transakcijų sąryšį iliustruoja žemiau esanti schema (8 pav.). Pavyzdiniai blokų identifikatoriai paimti iš resurso http://www.blockchain.info:



8 pav.: du, vienas po kito einantys blokai

2.3.3. Transakcijų tvirtinimas

Pagrindinė įvardijama elektroninės valiutos problema tai jau minėta dvigubo išleidimo problema [Ant14][Nak09]. Kai tas pats mokėtojas gali sukurti kelias transakcijas, kuriose skirtingiems gavėjams būtų siunčiami tie patys valiutos vienetai. Tam, kad to išvengtų BitCoin (ar bet kuri kita virtuali, elektroninė ar fizinė valiutų sistema) tikra ir teisinga transakcija laiko tą, kuri atėjo ankščiausiai (t.y. pati pirma). Centralizuotoje transakcijų sistemoje šis uždavinys yra

sprendžiamas atitinkamos centrinės esybės. Tokioje sistemoje kaip *BitCoin* turi būtų pasiektas susitarimas tarp tinklo narių dėl to kokia transakcija bus laikoma pirma. Taip pat visi tinklo dalyviai turi būti tikri, kad jų pusėje turima transakcijų istorija (Knyga) yra naujausia ir tokia pati kaip daugumos kitų tinklo dalyvių. Kitaip sakant visi BitCoin tinklo dalyviai turi pasiekti bendrą sutarimą, konsensusą. Transakcijos BitCoin tinkle (Kaip ir *Ethereum* [But14]) yra tvirtinamos taip vadinamos „gavybos“ (angl. – *mining*) [Nak09] pagalba, naudojant „*proof of work*“ principą. Tam tikri dalyviai skiria savo skaičiavimo resursus tam, kad spręsdami sudėtingas aritmetines užduotis generuotų naujas identifikacines „*hash*“ reikšmes naujiems blokams. Taip naujus blokus su naujomis transakcijomis įrašydami į bendrą blokų grandinę. Vartotojui kuris „*išgauna*“ naują bloką skiriama premija (2017 metais tai 12.5 BTC), taip pat jo adresui priskiriami visi transakcijų mokesčiai prikabinti prie į bloką sudėtų transakcijų. Tokių būdu naujajame bloke yra sukuriama nauja transakcija, kuri neturi jokių įvesčių, bet turi išvestis su fiksuota BTC suma į bloką išgavusio vartotojo adresą.

Pats darbo įrodymo algoritmas taip pat remiasi BitCoin tinkle paplitusia SHA-256 funkcija. Šio algoritmo esmė yra ta, kad naujojo bloko antraštė yra pakartotinai leidžiama per SHA-256 funkcija keičiant tik viena parametrą, kol yra gaunama atitinkama reikalinga reikšmė. Funkcijos rezultato numatyti negalima, todėl vienintelis būdas gauti reikalingą reikšmę yra bandyti neapibrėžtą kiekį kartų. Blokų išgavimu užsiimantis vartotojas turi stengtis rasti tokį argumentą (skaitinę reikšmę), kurį perdavus kartu su naujojo bloko antrašte, būtų gauta naujo bloko identifikacinė *hash* reikšmė mažesnė už nustatytą slenksti (angl. – *threshold*). Kadangi SHA-256 funkcija su bet kokiais argumentais visada grąžina 256 bitų ilgio reikšmę, slenkstis yra reikalaujamas nulių kiekis funkcijos rezultato pradžioje (tai paaiškina nulius blokų identifikacinių *hash* reikšmių pradžioje 8 pav.) Nepavykus surasti atitinkamos reikšmės vartotojas padidina argumentą ir bando vėl. BitCoin tinklas automatiškai mažina slenkstį visiems dalyviams vienodai taip sunkindamas uždavinį, kad didėjant skaičiavimo pajėgumams vieno bloko sukūrimo laikas išliktų apie 10 minučių. Pagal (2017 metų sudėtingumo lygį blokus išgaunantys vartotojai turi bandyti (spėlioti) kvadralijonus kartų (Pagal http://www.blockchain.info informaciją).

2.3.4. BitCoin platformos panaudojimas

BitCoin platforma gali būti naudojama ne tik atsiskaitymams BTC valiuta. Jos saugumo, išskirstytų skaičiavimų ir duomenų nepaneigiamumo savybės gali būti išnaudojamos kitų informacinių sistemų. Pasinaudoti BitCoin sistemos funkcionalumu kitose informacinėse

sistemose galima Spalvotų monetų (angl. – *Colored coins*) ir „*MasterCoin*“ protokolo pagalba [Ant14][DSt16].

Spalvotos monetos tai meta protokolas galintis sujungti nedidelius BTC kiekius ir taip sukurti naują valiutą. Sistemos gali išnaudoti tokius naujus valiutos vienetus savo uždavinių sprendimui ir, reikalui esant, konvertuoti į/iš BTC [DSt16].

„*MasterCoin*“ protokolas tai platforma leidžianti įvairioms sistemos praplėsti BitCoin funkcionalumą. Šis protokolas naudoja MST valiutą „*MasterCoin*“ transakcijų vykdymui. Palyginimui yra pateikiamas [Ant14] kompiuterinių tinklų pavyzdys, kuriame *MasterCoin* yra tarsi HTTP protokolas veikiantis ant TCP infrastruktūros. *MasterCoin* veikia per specialų BitCoin adresą (1EXoDusjGwvnjZUyKkxZ4UHEf77z6A5S4P), vadinamą „*Exodus*“. Šis adresas yra lyginimas su HTTP protokolo 80 TCP prieiga (angl. – *Port*).

2.3.5. BlockChain bandymai, modeliavimai ir veikimo parametrai

Šiuo metu yra eilė darbų nagrinėjančių įvairias BlockChain sistemų savybes per BitCoin platformą. Visuose čia nagrinėjamuose darbuose vienaip ar kitaip yra nagrinėjama duomenų struktūrų sklaidos Blokų grandinių tinkluose aspektai. Darbuose [ESi14] [Mwa16] nagrinėjamos su BitCoin blokų išgavimo operacijomis susijusių atakų galimybės. Darbe [DWa14] nagrinėjama, kiek laiko užtrunka BitCoin tinkle išplisti tam tikro dydžio duomenų struktūrai.

Darbai [ESi14] [Mwa16] nagrinėja ir modeliuoja Savanaudiško išgavimo (angl. – *Selfish mining*) problemą. Savanaudiško išgavimo esmė – nesąžiningas išgavimo fondų (angl. – *Mining pools*) elgesys [Esi14]. Išgavimo fondai tai į komandą susibūrę BitCoin išgavėjai, sutelkę bendrus savo techninės įrangos resursus BitCoin valiutos išgavimui [Mwa16]. 2017 metais BitCoin valiutos išgavimo sudėtingumas yra pasiekęs tokį lygį, kad tam, jog išgavimas būtų pelningas, neužtenka įprastų asmeninių kompiuterių pajėgumų [Bei17]. Dėl šios priežasties internete kuriasi išgavimo fondai, kurie apjungę fondų dalyvių kompiuterinius resursus išgauna BitCoin valiutos vienetus ir atitinkamai paskirsto išgautą kiekį fondo dalyviams pagal jų turimos techninės įrangos indėlį į bendro išgavimo procesą. Tokiu būdu fondas prisideda prie BitCoin ekosistemos vystymo formuodamas blokus ir pagal savo techninius resursus gaudamas atlygį. Fondo dalyviams tai būdas gauti BTC valiutos [Bei17]. Darbe [Esi14] yra aprašoma situacija kai toks fondas tampa nesąžiningas ir sugeba pasisavinti daugiau BTC valiutos nei jam priklausytų pagal bendrą fondo dalyvių techninių pajėgumų indėlį į BitCoin išgavimo procesą. Nesąžiningas fondas priverčia sąžiningus BitCoin išgavėjus eikvoti savo skaičiavimo resursus blokų pridėjimui į Blokų grandinę kuri jau nėra ilgiausia. Tą nesąžiningas fondas pasiekia nepranešdamas BitCoin tinklui apie savo išgautus naujausius blokus. Fondas visus naujus savo

narių išgautus blokus jungia prie savo grandinės, o visi kiti BitCoin sistemos dalyviai jungia savo išgautus blokus prie viešos BitCoin grandinės, nors jie dar nežino, kad nesąžiningas fondas kontroliuoja grandinę, kuri šiuo metu yra ilgiausia. Tam tikru metu nesąžiningas fondas informuoja visus tinklo dalyvius apie savo pusėje išgautus blokus ir apie ilgiausią, jo pusėje esančią grandinę. Visi sąžiningi BitCoin dalyviai tada privalo persijungti prie ilgiausios grandinės. Šis faktas paverčia visus sąžiningų dalyvių išgautus blokus ir tuo pačiu investuotą techninės įrangos darbą beverčiais. Nesąžiningas fondas tuo metu gauna didesnį atlygį už du ar daugiau blokų, nei, kad būtų gavęs sąžiningos konkurencijos sąlygomis [Mwa16]. Darbe [Esi14] yra kuriamas modelis, kurio pagalba norima rasti kiek „slaptų“ blokų turėtų išgauti atakuojantis fondas, kad gautų maksimalų pelną. Kokie bebūtų galingi privataus fondo narių techniniai pajėgumai, ilguoju laikotarpiu jie negali konkuruoti su bendru BitCoin tinklo skaičiavimo pajėgumu. Ankščiau ar vėliau (Dažniausiai ankščiau [Esi14]) BitCoin vieša grandinė taps ilgesnė už privačią fondo grandinę ir tada jau fondo resursų investicijos į išgavimą taps bevertės. [Esi14] darbe yra nagrinėjama kiek blokų išgavęs ir kada apie juos informavęs tinklą atakuojantis fondas teoriškai gautų didžiausią pelną. Pats modelis yra realizuojamas sukuriant testinį BitCoin tinklą, susidedantį iš 1000 dalyvių iš kurių dalis išgavėjų turi specialų funkcionalumą vykdantį įvairius savanaudiško išgavimo algoritmus. Šis bandymas yra išplečiamas [Mwa16] darbe, sukuriant ne testinį privatų tinklą, o atskirą modelį parašytą C++ kalba. Modelyje taip pat yra modeliuojami tinklo trukdžiai platinant blokus. Tai yra vienas pagrindinių skirtumų nuo [Esi14] darbo, kuriame tinklo trukdžių nebuvo ir blokas pasiekdavo visus tinklo dalyvius iš karto. [Mwa16] darbe daroma prielaida, kad blokas yra išplatinamas tinkle per 10 sekundžių. Taip pat [Mwa16] modeliuoja situacijas kai tinkle veikia daugiau nei vienas savanaudiškas fondas ir jie tarpusavyje konkuruoja. Šie du darbai simuliuoja ir modeliuoja tam tikrą BitCoin platformos dalį, per kurią kyla rizika identifikuoti atakos vektorių. Šiame darbe bus pasinaudota tam tikromis idėjomis modeliuoti mastelio keitimo savybes BitCoin principais veikiančiame Blokų grandinės tinkle.

Darbas [DWa14] detaliau tiria duomenų esybių platinimą BitCoin tinkle. Bloko pasklidimo greitis (*angl.* - *propagation*) yra labai aktualus klausimas BitCoin bendruomenėje [DWa14]. Kiekvienas išgavėjas nori, kad jo sukurtas blokas būtų kaip įmanoma greičiau išplatintas tinkle (pasiektų 50% ir daugiau tinklo dalyvių) bei aplenktų kitus panašiu laiku sukurtus ir pradėtus platinti blokus. Kad tai išbandyti 2014 metais buvo atliktas eksperimentas [DWa14]. Jis buvo vykdytas sukuriant privatų 4000 dalyvių BitCoin tinklą ir vieną specialų tinklo dalyvį, kuris turi funkcionalumą, leidžiantį stebėti laikus tarp blokų ir transakcijų sukūrimo ir išsiuntimo kitiems tinklo dalyviams. Eksperimentas buvo vykdomas darant prielaidą, kad pranešimo su bloko

informacija dydis gali pasiekti 500 KB, nors pagal http://www.blockchain.info pateiktą statistiką vidutinis bloko dydis skirtingais 2014 metų mėnesiais neviršydavo 250 KB.

Šaltinyje yra nurodyti tokie transakcijų ir blokų išplatinimo laikai (1 lentelė):

Transakcijos išplatinimas:

Dalis	laikas (s)
50%	1,3
75%	2,3
90%	8,3
95%	120,4
99%	125,8

Bloko išplatinimas:

Dalis	laikas (s)
50%	4,5
75%	9,4
90%	32,5
95%	122,8
99%	134,4

1 lentelė.: BitCoin tinklo blokų ir transakcijų išplatinimo trukmė [DWa14]

Stulpelis „Dalis“ reiškia tinklo dalyvių dalį. Antras stulpelis „Laikas (s)“ reiškia, kiek laiko užtrunka pirmame stulpelyje nurodytai tinklo daliai gauti informaciją apie transakcijas ar blokus.

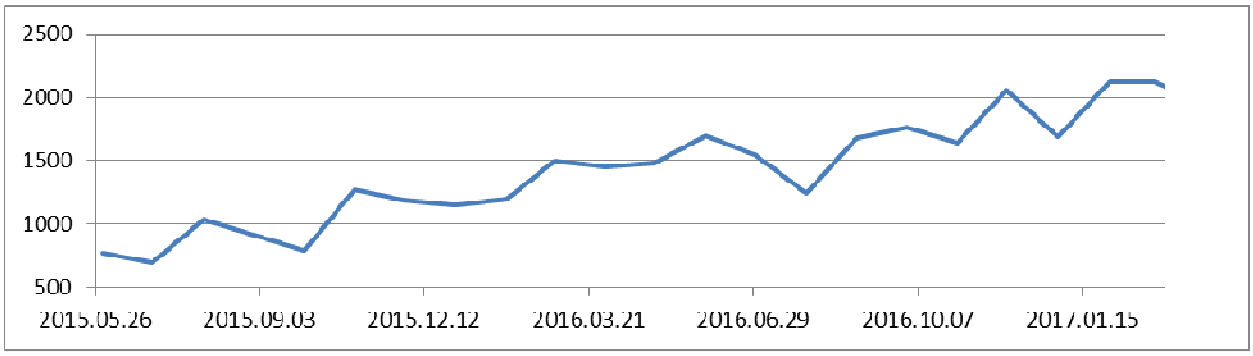
Šiame darbe pagal [DWa14] darbe atliktų eksperimentų rezultatus bus kuriamas agentais grįstas modelis, kurio tikslas bus priartinti tam tikrų BlockChain savybių modeliavimą kaip įmanoma arčiau realaus pasaulio rezultatų. Kuriamo modelio veikimo rezultatai bus lyginami su [DWa14] 1 lentelėje gautais rezultatais.

2.3.6 BitCoin mastelio keitimo problemos tyrimas

Mastelio keitimo (Scalability) sąvoka, yra aprašoma kaip sistemos savybė sklandžiai veikti didėjant sistemos apkrovai arba resursams [Gun14]. Dažniausiai mastelio keitimo savybės yra vertinamos pagal sistemos veikimo kokybinių parametrų (greičio, atsako laiko ir pan.) mažėjimo ar didėjimo greitį didėjant vartotojų, užklausų skaičiui ar pridedant techninius ir loginius resursus [SJ05]. Šiuo atveju kalbant apie BitCoin tinklą, BitCoin bendruomenėje aktualiausia problema yra sutalpinti kaip įmanoma daugiau transakcijų į vieną BitCoin bloką. Yra siūlomi įvairūs sprendimai iš kurių populiariausias yra maksimalaus bloko dydžio padindinimas [Her17]. Šiuo atveju bloko dydžio padidėjimas būtų tarsi loginis resursas (telpa daugiau transakcijų), bet kartu ir didesnė apkrova BitCoin tinklui.

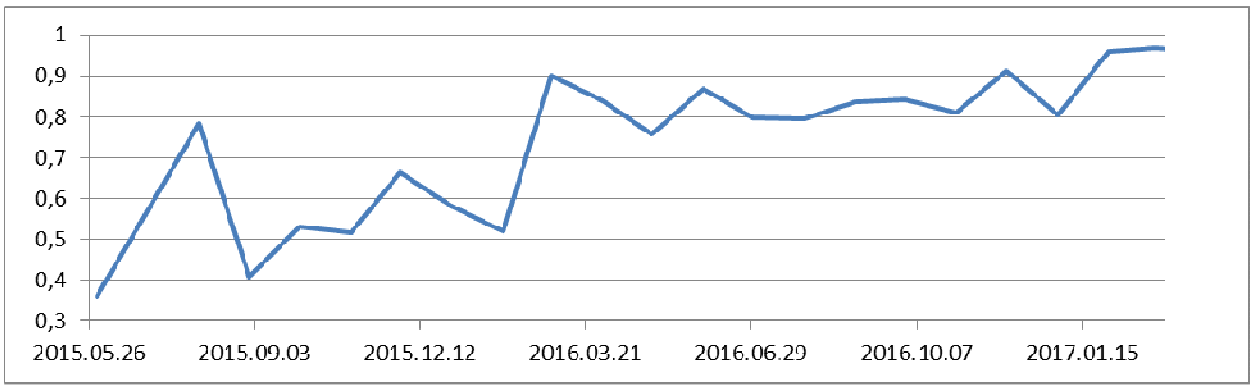
BitCoin mastelio plėtimo, keičiant maksimalų bloko dydį modeliavimas yra prasmingas ir reikalingas dėl to, kad būtų iš anksto numatyti šio veiksmo privalumai arba trūkumai. Šiuo metu BitCoin bendruomenėje verda karštos diskusijos dėl Bloko maksimalaus dydžio (1MB) padidinimo. Yra netgi siūloma taip vadinama Beribio BitCoin (*angl.* – BitCoin *Unlimited*, sutrumpintai BTU) vizija [Riz15]. Šio sprendimo pagrindinis skirtumas nuo paprasto BitCoin yra tas, kad maksimalus bloko dydis nėra fiksuotas BitCoin programinėj įrangoje. Maksimalus bloko dydis yra nustatomo parametro klausimas. Pagrindinė BTU vizija tai, kad pati bendruomenė nusprendžia koks turėtų būti maksimalus bloko dydis. Konsensuso dėka bendruomenė nuspręstų ar tai visiems priimtinas dydis ar ne. Kol kas BTU paplitimas yra nedidelis. 2017 kovo duomenimis [Her17] BTU programinį kodą naudoja iki 2,5% BTC dalyvių.

Augant BitCoin bloko dydžiui taip pat ir daugėja transakcijų, patenkančių į blokus, skaičius. Žemiau pavaizduotas transakcijų skaičiaus bloke grafikas (pagal http://www.blockchain.info informaciją):



9 pav.: transakcijų bloke didėjimas nuo 2015 metų gegužės iki 2017 metų kovo. Y ašyje – vidutinis transakcijų kiekis bloke

Pagrindinis BitCoin bloko didinimo argumentas yra jame telpančio transakcijų kiekio padidinimas. Manoma, kad padidinus transakcijų kiekį bloke automatiškai padidės ir per laiko vienetą patvirtinamų transakcijų kiekis [Caf15]. Dabar esantis transakcijų apdorojimo greitis lyginant, pavyzdžiui, su VISA/MasterCard kortelių sistemų transakcijų apdorojimo greičiu skiriasi kelis tūkstančių kartų [Ant14]. Pagal laisvai prieinamą BlockChain informaciją didėjant transakcijų kiekiams, BitCoin gyvavimo metu atitinkamai keitėsi ir blokų dydžiai, kaip pavaizduota grafike apačioje (pagal http://www.blockchain.info informaciją):



10 pav.: vidutinio bloko dydžio didėjimas 2015 metų gegužės iki 2017 metų kovo. Y ašyje – bloko dydis MB

10 pav. grafike matosi, kad vidutinis bloko dydis jau yra priartėjęs prie 1MB ribos. Remiantis šia bloko didėjimo informacija ir aktyviais bloko padidinimo paieškos tyrimais, galima daryti išvada, kad modeliai, tiriantis BitCoin sistemų veikimo savybes bloko dydžiui peržengus 1MB ribą artimiausiu metu tikėtina taps labai aktualūs.

3. BlockChain modeliavimas agentais

Darbe yra aprašomas BlockChain platformos paremtos BitCoin principais modelis. Šiame modelyje agentas yra BitCoin tinklo dalyvis. Skirtingai nuo realiai veikiančio BitCoin tinklo, pagrindinė agento būsena yra ne jo turimų BTC suma, o jo pusėje saugoma BlockChain duomenų bazė. Taip yra pasirinkta todėl, kad skirtingai nuo realios BitCoin sistemos, šiame darbe yra tiriamos ne ekonominės, bet techninės tinklo dalyvių sąveikos.

Šiame darbe modeliavimas agentais yra paremtas darbais [Mik14][Sil16]. Modeliavimą agentais nagrinėjančiame darbe [Mik14] nurodyti pagrindiniai modeliavimo šablonai: laiko peštynės (angl. – *scheduler scramble*), konteksto ir projekcijos hierarchija, strategija, mokymasis.

[Mik14][Sil16] darbai modeliavimą agentais tiria kaip įvairių architektūrų sistemose veikiančių atskirų esybių (vartotojų, serverių, modulių) sąveiką. Šiame darbe, yra pasinaudota tam tikrais minėtuose darbuose minimomis modeliavimo idėjomis, tačiau šiuo atveju agentu apibrėžiamas ne tam tikras sistemos modulis, vartotojas ar serveris, o BlockChain tinklo dalyvis. BlockChain *Peer-to-peer* tinkle nėra jokios valdančios centrinės esybės [Nak09]. Tinklo dalyviai, bendraudami tarpusavyje, realizuoja visą BlockChain sistemų veikimo logiką.

Remiantis [Mik14] aprašyta komponento sąvoka, komponentas yra identifikuojamas kaip:

- Nepriklausomas kūrimo vienetas;
- Trečiųjų šalių komponentai palaiko integraciją;
- Komponentai neturi iš išorės pastebimos būsenos;

- Komponentas turi kontraktu apibrėžtą interfeisą ir aiškiai apibrėžtas priklausomybes;
- Komponentas yra skirtas tam tikrai komponentų platformai.

Pagrindinis skirtumas nuo darbuose [Mik14][Sil16] aprašytų, skirtingas sistemų architektūras modeliuojančių modelių yra tas, kad šiuo atveju BlockChain „komponentai“ yra vienodos paskirties, su vienodomis bendravimo taisyklėmis. Šiame BlockChain agentų modelyje punktas „Trečiųjų šalių komponentai palaiko integraciją.“ nėra aktualus, nes sistema yra sudaryta iš vienodų komponentų („tinklo dalyvių“). Ši prielaida yra teisinga todėl, kad modelis yra kuriamas BitCoin sistemos pagrindu. Ateityje šis modelis galės būti praplėstas galimybėmis modeliuoti platformas, palaikančias išmaniųjų kontraktų funkcionalumą (pavyzdžiui, *Ethereum*). Tokiu atveju tinkle atsirastų esybės, turinčios specifinį, konkrečiom užduotims spręsti reikalingą programinį kodą.

Darbe [Mik14] yra minima, kad modeliavimą agentais verta taikyti tada, kai tenkinamas bent vienas iš šių kriterijų:

- kai problema turi natūralų atvaizdavimą į agentus;
- kai sprendimai ir elgesys gali būti tiksliai aprašyti;
- kai svarbu, kad agentai turėtų individualų elgesį;
- kai svarbu, kad agentai prisitaikytų ir keistųsi;
- kai svarbu, kad agentai mokytųsi;
- kai svarbu, kad agentai turėtų dinaminį ryšį su kitais agentais;
- kai svarbu, kad agentas turėtų erdvinius duomenis;
- kai praeitis nepadeda nuspėti ateities, nes augimas ir keitimasis yra dinaminiai;
- kai svarbi galimybė lengvai keisti agentų kiekį.

BlockChain *Peer-to-peer* tinklas neturintis nei centralizuotos esybės nei tinklo dalyvius aptarnaujančių serverių išskirtinai remiasi tik jame veikiančių tinklo dalyvių tarpusavio sąveika. Transakcijų siuntimas, tikrinimas, blokų generavimas, valiutos vienetų savininkų pasikeitimas sudaro BitCoin esmę. BlockChain natūralus atvaizdavimas į agentus, galimybė tiksliai aprašyti jų bendrą veikimą ir individualų elgesį, taip pat poreikis tirti agentų tarpusavio ryšius, tinklo veikimą besikeičiant agentų skaičiui, daro BlockChain puikiu kandidatų agentais paremto modelio tyrimams. Tokia išvada yra daroma, nes BlockChain tinklo dalyvių tarpusavio sąveika daro įtaką tam tikriems BlockChain tinklo parametrams, nuo kurių tiesiogiai priklauso tinklo greitaveika, teikiama pridėtinė vertė, saugumas. Pagrindiniai BlockChain parametrai, pagal kuriuos yra sprendžiama apie einamąją BlockChain tinklo būseną yra šie:

- Bendras BlockChain dydis. Nuo to tiesiogiai priklauso tinklo dalyvių atminties vietos poreikis;
- Transakcijų dydis. Nuo to priklauso blokų dydžiai, kas savo ruožtu dar įtaką viso BlockChain tinklo greitaveikai;
- Skirtingų dalyvių turima informacija apie BlockChain būklę. Nuo to priklauso bendro sutarimo, Konsensuso mechanizmas, o tai daro įtaką BlockChain tinklo patikimumui ir pritaikymo galimybėms.

Tokie patys principai galioja ir sudėtingesniems, papildomą funkcionalumą kaip išmanieji kontraktai pridedantiems tinklams (pavyzdžiui *Ethereum*). Išmanieji kontraktai taip pat veikia kaip savarankiškos esybės konkrečių transakcijų vykdymo metu ir tuo papildo tinklo dalyvių realizuojamą veikimo logiką [But14]. Nors išmaniųjų kontraktų funkcionalumo modeliavimas ir nėra šio darbo dalis, tačiau verta pabrėžti, kad tai taip pat yra tinkamas kandidatas modeliavimui agentais.

Šiame darbe aprašytame BlockChain modelyje visi agentai pagal nutylėjimą gali atlikti vienodas funkcijas, bet ne visi agentai jas visas vykdys. Kad būtų modeliuojamas kaip įmanoma labiau realistiškas BitCoin principu veikiantis BlockChain tinklas, dalyvių agentai turėtų būti dviejų tipų:

- Pilnas tinklo dalyvis. Savo pusėje turintis arba galintis atsinaujinti pilną BlockChain tinklą. Šiame darbe jis yra žymimas N (*Node*);
- Išgavėjas. Pilnas tinklo dalyvis aktyviai dalyvaujantis BTC išgavimo (angl. – *mining*) procese. Šiame darbe jis yra žymimas M (*Miner*);

Šiuos du agentų tipus skiria ne tik loginio lygio atributai, tokie kaip veikimo tikslai ir procesai, bet ir fiziniai, tokie kaip aparatūrinė įranga ir geografija. Šiame modelyje nėra atskirai išskirti supaprastinti mokėjimų tikrinimo dalyviai (angl. – *Simplified Payement Validation (SPV)*). SPV dalyviai savo pusėje neturi pilnos BlockChain duomenų bazės ir saugo informaciją tik apie konkrečioms privačių/viešų raktų poroms priskirtas transakcijas. Pilnus tinklo dalyvius modeliuojantys agentai padengia SPV (nes juos visada galima laikyti SPV su savo pusėje saugoma BlockChain baze), todėl modelyje jie nėra papildomai išskiriami.

3.1. Agentų funkcijos

Šiame darbe modeliuojamo BlockChain tinklo funkcionalumas yra realizuotas kaip tinkle veikiančių agentų sąveikos rezultatas. Modelio iniciavimo metu yra sugeneruojamas tam tikras

kiekis N ir M agentų. N agentai generuos ir siųs transakcijas. Agento išsiųsta transakcija yra persiunčiama kitiems agentams, kurie savo ruožtu įsitraukia transakciją į savo transakcijų erdvę ir siunčia ją toliau savo žinomiems kaimynams. Prieš persiųsdami transakciją toliau, agentai turėtų patikrinti ar ji teisinga ir ar jau nebuvo įtraukta į transakcijų erdvę ankščiau. Agentai M (jų modeliuojama mažiau nei agentų N) turėtų surinkinėti transakcijas į blokus. M agentai suformavę bloką informuoja apie jį visą tinklą tokiu pačiu būdu kaip N agentai informuoja apie naują transakciją. Blokas, priklausomai nuo dydžio keliauja tinklu tam tikrą laiką. Kiekvienas agentas papildo nauju bloku savo turimą BlockChain kopiją. Tam tikriems N ir M agentams konkrečiu metu nebuvus prisijungus prie tinklo, esant tinklo ar kitiems sutrikimams agentams turėtų nepavykti tinkamai atsinaujinti savo BlockChain bazės.

Skirtingai nuo [Mik14][Sil16] darbų, šiame modelyje nėra serverių. Kiekvienas agentas tuo pačiu turi ir vartotojo, ir serverio savybių:

- Agentas kaip vartotojas: siunčia transakcijas, susijungia su kitais agentais;
- Agentas kaip serveris: pagal savo turimus duomenis tikrina transakcijas, persiunčia jas kitiems dalyviams.

Taip pat atskirai nėra modeliuojama ir aplinka. Kaip ir realiai veikiančiuose BlockChain tinkluose, tam tikru momentu platformos būsena yra įvertinama ne pagal kažkokius globalius aplinkos parametrus, o įvertinant agentų pusėje saugomų BlockChain kopijų duomenis. 51 % viso tinklo agentų turima bendra, sutarta informacija yra laikoma visos BlockChain bendra būsena.

Šiame darbo etape iškeliami tokie reikalavimai agentų veikimui:

- Visi agentai yra sujungti į *peer-to-peer* tipo tinklą. Tinklas nepasižyminti jokia konkrečia struktūra ar topologija. Kiekvienas agentas yra sujungiamas su kitais atsitiktinai parinktais agentais. Iš agento siunčiamas pranešimas yra siunčiamas keturiems kitiems agentams (agento kaimynams), su kuriais yra susijungęs konkretus agentas. Kiti agentai pranešimą persiunčia dar kitiems keturiems agentams su kuriais yra susijungę, kol eksponentiškai pranešimas yra išsiunčiamas po visą BlockChain tinklą.

- Sugeneruojamas tam tikras skaičius agentų. Transakcijų erdvę nusako atitinkamas agento būsenos parametras, kuriame yra saugomas transakcijų erdvėje esamų transakcijų skaičius. Šiame darbe bus tiriamas duomenų struktūros išplatinimo laikas, todėl transakcijų erdvės reikšmė visada bus 1.

- Kai kurie iš agentų (pagal tikimybių parametrą) sukuria ir paleidžia į tinklą transakciją.

- Agentai persiunčia gautas transakcijas. Persiuntimas prasideda nuo transakciją sukūrusio agento kaimynų, kurie ją persiunčia savo kaimynams ir t.t. Vienas persiuntimas vyksta per vieną simuliacijos žingsnį (angl. – *step*). Šiame modelyje daroma prielaida, kad visos siunčiamos transakcijos yra teisingos ir, kad transakcijos persiuntimas baigiasi ties paskutiniu tinklo dalyviu.

- Modelyje sąvoka „siuntimas“ reiškia tai, kad įvykus vienam simuliacijos žingsniui agento N keturių kaimynų atitinkami transakcijos arba grandinės aukščio parametrai pasikeičia iš 0 į 1. Tai reiškia, kad agentas priėmė transakciją ar pridėjo bloką.

- Agentai M siunčia blokus. Šiame modelyje duomenų struktūra „Blokas“ skiriasi nuo duomenų struktūros „Transakcija“ tuo, kad Blokas yra didesnis ir tinkle yra išplatinamas ilgiau nei transakcija.

Kiekvienu simuliacijos žingsniu turėtų būti atliekami šie veiksmai:

- Gaunamos ir persiunčiamos kitų agentų sukurtos transakcijos;
- Transakcijų arba blokų erdvių reikšmės pakeičiamos iš 0 į 1. Taip sindikuojant duomenų struktūros priėmimą;
- Jei gaunamos duomenų struktūros jau buvo pridėtos prie N ir M agentų erdvių ankščiau, persiuntimas sustabdomas. Kitaip sakant jei transakcijų ar blokų erdvė yra lygi 1, persiuntimas yra nutraukiamas.

3.2. Agentų būsenos

M ir N agentų būsenos yra apibrėžiamos iš loginės ir techninės pusės. Nors BlockChain pagrindinė paskirtis yra apsikeitimas tam tikrais valiutos vienetais, bet kadangi šiame darbe yra tiriamos techninės BlockChain technologijos savybės, agentai neturi saugomų valiutos vienetų parametrų. Iš techninės pusės kiekvienas agentas savo pusėje turi BlockChain tinklo kopiją. Agentai tarpusavyje bendrauja siųsdami vienas kitam duomenų struktūras (transakcijas ir blokus). Dėl to keičiasi jų pusėje saugomų transakcijų ir blokų erdvių reikšmės iš 0 į 1. 0 – agentas neinformuotas apie atitinkamą duomenų struktūrą (Transakciją/Bloką). 1 – informuotas.

Tinklo vaizdas kiekviename iš agentų priklauso nuo to kaip greitai yra perduodama informacija BlockChain tinklu ir konkretaus agento prisijungimo prie tinklo laikas ir kokybė. Kiekvienas agentas priklausomai nuo gaunamos informacijos atitinkamai koreguoja savo pusėje turimas erdvių reikšmes. Pagal tai kaip greitai (per kiek žingsnių) agentas yra informuojamas apie transakciją galima spręsti apie tinklo būseną.

3.3. Modeliavimo aplinka

Aplinka šiuo atveju yra BlockChain tinklas. Kadangi tinklas yra *peer-to-peer* tipo, visa verslo logika yra realizuojama pačiuose agentuose. Kaip buvo minėta anksčiau aplinkos būsena yra įvertinama vertinant kiekvieno agento atskirai ir visų agentų visumos erdvių parametrus. Bet kokie kiti viso modelio parametrai tokie kaip transakcijų persiuntimo greitis yra priskirti patiems agentams. Dėl BlockChain specifikos N ir M agentų parametrai įtakoja visos BlockChain platformos veikimą.

3.4. Agentų parametrai

Šiame skyriuje yra pateikiami agentų parametrai, kurie yra naudojami N ir M tipo agentų. Darbe modeliuojant duomenų struktūrų platinimo BitCoin tinkle procesą N ir M agentai turės vienodus parametrus. Ateityje plečiant modelį, dėl skirtingos N ir M agentų specifikos, vieno tipo agentų parametrai skirsis nuo kito tipo agentų parametrų.

$N_{K.}$ M_K – Kaimynų sąrašas. Yra nurodomi BlockChain tinkle esantys keturi agentai, su kuriais yra susijungęs konkretus agentas. Išsiųstos transakcijos ar kita informacija nueis šiems keturiems kaimynams, kurie ją persiųs toliau;

N_{ST}, M_{ST} – transakcijų skaičius agento transakcijų erdvėje. Šiame darbe galimos reikšmės 0 arba 1;

$N_{SB,}$ M_{SB} – agento pusėje saugomos BlockChain aukštis. Šiame darbe galimos reikšmės 0 arba 1;

$N_{SR,}$ M_{SR} – Signalas dėl įeinančios transakcijos. Šiuo parametru yra modeliuojamas transakcijos gavimo įvykis;

$N_{SL,}$ M_{SL} – Signalas dėl įeinančio bloko. Šiuo parametru yra modeliuojamas bloko gavimo įvykis.

3.5. Modelio pritaikymo galimybės

Mastelio keitimas

Kuo daugiau vietos kilobaitais užima transakcija, tuo jų mažiau telpa į tam tikru dydžiu apribotą bloką. Kuo didesnis bloko dydis yra nustatomas BlockChain tinkle, tuo ilgiau jis yra persiunčiamas tinklu ir tuo ilgiau užtrunka visiems tinklo dalyviams sužinoti apie naujus blokus ir atitinkamai atsinaujinti savo pusės BlockChain bazę. Vienas iš galimų modelio pritaikymo variantų yra galimybė ištirti kaip pailgėjus bloko generavimo, bloko persiuntimo tinklu laikui, keičiasi N ir M agentų supratimas apie aktualią BlockChain situaciją ir konsensuso mechanizmą. Šiame darbe bus tiriamos galimybės modeliuoti duomenų struktūrų plitimo BlockChain tinkle greitį, keičiantis duomenų struktūrų, šiuo atveju, bloko dydžiui.

BlockChain sukūrimas ir plėtimas

Modelis gali būti kuriamas atsitiktiniu būdu kiekvienam iš N ir M agentų suteikiant tam tikras $_{ST}$ ir $_{SB}$ būsenos parametrų reikšmes. Taip pat galima modeliuot BlockChain tinklo kūrimą nuo pačios pradinės transakcijos ir pačio pradinio bloko. BitCoin tinkle pats pirmasis 2009 metais sukurtas blokas yra vadinamas Pradžios bloku (angl. – *Genesis block*) [Nak09]. Inicijuojant modelį būtų sugeneruojamas atitinkamas kiekis N ir M agentų, turinčių atitinkamus blokus. Modelio iniciavimo metu yra sukuriama atitinkama BlockChain struktūra su joje esančiomis transakcijomis ir blokais, kurie toliau redaguojami ir keičiami sąveikaujant N ir M agentams. Tobulinant ir plečiant modelį, būtų modeliuojamos blokų nuorodos vienas į kitą, taip būtų galima sumodeliuoti Pradinį bloką, kuris turėtų pasižymėti visomis bet kurio kito BlockChain tinklo bloko savybėmis. Tik jis neturėtų nuorodos į prieš tai esantį bloką.

Dvigubo išleidimo problema

Problema kyla kai N agentas *X* nori pervesti tą pačią iš tų pačių, 2.3.1. skyriuje minėtų, *UTXO* sudarytą lėšų sumą agentams *Y* ir *Z*. Į tinką yra paleidžiamos dvi transakcijos *X->Y* ir *X->Z* abi bandančios išleisti tas pačias lėšas, kitaip sakant yra vykdoma dvigubo išleidimo operacija. Kai kurie tinklo dalyviai gali gauti transakciją *X->Y*, o kai kurie *X->Z*. Skirtingi agentai patvirtins (ir atmes) skirtingas transakcijas, priklausomai nuo to kokia transakcija, kurį agentą pasieks pirma. Jei konkretų agentą per tinklą pirma pasieks transakcija *X->Z*, vėliau pasiekus transakcijai *X->Y* ji jau bus atmesta kaip netinkama, nes konkretus agentas patikrinęs

norimus pervest *UTXO* pamatys, kad jie jau išnaudoti transakcijoje *X->Y*. Kai kurie agentai atmes vienas, o kiti kitas transakcijas. Šis disputas bus išspręstas tik tada, kai kažkurią iš šių transakcijų, kuris nors M agentas įtrauks į naują bloką, o naujas blokas bus įtrauktas į blokų grandinę. Kuri būtent transakcija bus įtraukta į grandinę priklausys nuo tokių parametrų kaip transakcijų sukūrimo eiliškumas, laiko tarpas, transakcijų dydis ir panašiai. Realiose BlockChain veikimą realizuojančiuose tinkluose su dvigubo išleidimo problema tvarkosi tinklo dalyvio pusėje veikianti programinė įranga. Turėdama viso tinklo vaizdą ar tik su dalyvio raktais susijusių transakcijų sąrašą (*SPV* atveju), sąsaja užtikrina, kad konkretūs *UTXO* būtų panaudojami tik vieną kartą. Tačiau jei modelyje yra daroma prielaida, kad kliento dalimi negalima pasitikėti, dėl to tinklas turi užtikrinti, kad būtų išvengta dvigubo išleidimo problemos. Darbe [Sil16] aprašant protokolo būsenas tarp įvairių komponentų gaunamų užklausų atmetimo varianto taip pat yra aprašytas variantas:

„Žinoma x tikimybė, jog įeinanti užklausa neatitinka protokolo, todėl ji turi būti atmesta. Šioje situacijoje, svarbu saugoti tikimybę, jog užklausa neatitinka protokolo bei įgyvendinti užklausos patikrinimą. Minėtąją tikimybę laikysime protokolo būsena. Ji gali būti saugoma agento, vaizduojančio komponentą, parametre K_{bp}. Tikimybė nurodoma intervale [0,1]. Agento elgsenoje, turėtų būti aprašyta protokolo neatitikimo tikimybės patikrinimas. Jei tikimybė patenkinama, tuomet užklausa turėtų būti atmesta. Tokią situaciją patogu modeliuoti tada, kai nežinomas modeliuojamos sistemos užklausų pobūdis, tačiau galima įvertinti užklausos neatitikimo protokolui tikimybę.“

Tai gali būti kaip vienas iš galimų sprendimų transakcijų tikrinimo ir neteisingų transakcijų atmetimo modeliavimui. Gali būti sukurti papildomi parametrai kaip:

- $N_{E,}M_E$ – tikimybė, kad agentas siunčia nekorektišką (technine prasme) transakciją. Reikšmių intervalas (0,1]. Nekorektišką transakciją turi atmesti tinklas (kiti dalyviai).
- $N_{D,}M_D$ – tikimybė, kad agentas siunčia nekorektišką (logine prasme) transakciją. Reikšmių intervalas (0,1]. Bandoma dar kartą panaudoti jau panaudotus *UTXO*.

BlockChain išsišakojimas (angl. – *Fork*)

Atsiradus pakeitimams BlockChain platformos principiniuose veikimo mechanizmuose (konsensuso mechanizme, duomenų struktūrų dydžių ribojimuose ir pan.) agentas gali nesužinoti apie pakeitimus. Tokiu atveju agentas visus naujus blokus laikys neteisingais ir dirbs su ta grandinės versija, kuri neturi naujų, naujomis savybėmis pasižyminčių blokų. Taigi kol agentas

neatsinaujins, jis ilgiausia grandine laikys iš tikro trumpesnę grandinę. BlockChain platformų bendruomenėse tai laikoma nepriimtina situacija, nes tokiu atveju atsiranda dvi atskiros grandinės, kurios niekada nesusijungs (tam naudojamas angliškas terminas *Fork*). Yra sprendimas pakeitimus įdiegti nevykdant tokių kardinalių veiksmų. Kai kuriais atvejais yra diegiamos griežtesnis tikrinimo taisyklių rinkinys naujiems tinklo dalyviams. Tokiu atveju agentai nespėję atnaujinti programinės įrangos pripažįsta teisingais visus blokus, o agentai atsinaujinę informaciją – tik dalį. Šio sprendimo esmė yra ta, kad svarbu, jog kritinė masė agentų atsinaujintų taisykles ir galėtų primesti jų atsinaujinimo būtinybę ir kitiems dalyviams.

Šis veiksmas yra įvardijamas kaip vienas labiausiai nepageidaujamų situacijų kalbant apie BlockChain duomenų bazes. Tačiau ne visada senos grandinės yra paskelbiamos neaktualiomis. Ethereum tinklo atveju, kai po įvykdytos atakos blokų grandinė buvo išskirta į naują šaką (kurioje buvo veikiama taip lyg ataka niekada nebūtų įvykusi), senoji grandinės šaka buvo naudojama toliau. Senoje šakoje buvo sukurta nauja valiuta Ethereum Classic (ETC) ir dėl investuotojų susidomėjimo, praėjus 6 mėnesiams po jos sukūrimo 2016 metais ji išlaiko stabilų kursą kripto valiutų keityklose nors ir nėra naudojama jokiuose žymesniuose projektuose.

Išskyrimo operacijos būna dviejų tipų [Cas17]:

- Minkšta (*angl. – soft fork*);
- Kieta (*angl. – hard fork*).

Jų skirtumą lengviausia iliustruoti bloko maksimalaus dydžio pakeitimo pavyzdžių. Iš paaiškinimo matysis, kad bloko dydžio sumažinimas reikštų minkštą, o bloko dydžio padidinimas kietą išskyrimą. Kartu verta prisiminti, kad bloko dydžio maksimalus dydis yra nurodytas vientisumo taisyklėse, kurias turi ir patikrina kiekvienas tinklo dalyvis [Ant14][Nak09].

1 pavyzdys – Maksimalus bloko dydis sumažinamas iki 0,5 MB: informacija apie pasikeitusį dydį pranešama visiems tinklo dalyviams. Tačiau ne visi tinklo dalyviai iš karto gauna apie tai informaciją. Rezultate: tinklo dalyviai neturintys informacijos apie pasikeitusias taisykles, toliau tvirtina ir priima blokus, kurių dydis >0,5MB ir <1Mb.

2 pavyzdys – Maksimalus bloko dydis padidinamas iki 2 MB. informacija apie pasikeitusį dydį pranešama visiems tinklo dalyviams. Tačiau ne visi tinklo dalyviai iš karto gauna apie tai informaciją. Rezultate: tinklo dalyviai neturintis informacijos apie pasikeitusias taisykles, tvirtina ir priima tik blokus, kurių dydis <1 MB ir atmetinėja (nes neturi naujausios informacijos) visus blokus, kurių dydis >1Mb.

Išskyrimų operacijos modeliavimas nėra šio darbo apimtyje, tačiau tai galėtų būti tyrimo objektas kituose BlockChain modeliavimo agentais darbuose.

4. Modelio realizacija

Šiame ir kituose darbo etapuose modelis yra realizuojamas *Repast Symphony* (toliau – *Repast*) programinio paketo pagrindu. *Repast* tai JAVA programavimo kalba paremta modelių kūrimo priemonė. Tai populiarus [MNo09] sprendimas, plačiai naudojamas akademinėje aplinkoje. Ši priemonė pasižymį, ne tik modeliavimo ir vizualizacijos, bet taip pat ir integracijos galimybėmis su populiariausiomis analizės priemonėmis.

Repast buvo pasirinktas dėl jau minėto JAVA palaikymo, o taip pat, kad sprendimas yra sukurtas *Eclipse* paketo pagrindu, kas labai palengvina darbą, kuriant agentais grįstus modelius pažįstamoje aplinkoje.

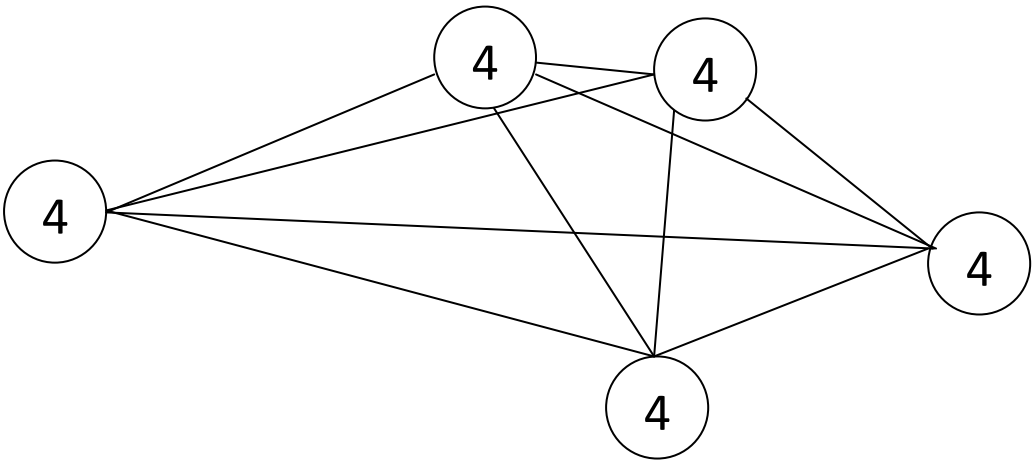
Šiame darbe *Repast* priemonėmis yra sukuriamas bazinis BlockChain modelis, patvirtinantis prielaidas dėl galimybės modeliuoti BitCoin pagrindu veikiančių BlockChain tinklus tam panaudojant *Repast* sprendimus. Tam, kad būtų pateiktos principinės tokio modelio sukūrimo galimybių išvados, modelis šiame etape dar neapims daugumos BitCoin tinklų dalyvių parametrų, tačiau iliustruos pagrindinius BlockChain tinklo kūrimo principus bei bazinį mastelio keitimo modeliavimą ir atitinkamas išvadas.

Kuriant agentais grįstus modelius *Repast* (taip pat ir kituose) karkasuose ypatingai svarbu yra parinkti tinkamą modeliavimo erdvę. [Sil16] darbe identifikuojami skirtingi erdvių tipai:

- Sriuba. Modelis be erdvės, jame agentas neturi atributų išreiškiančių jo padėti erdvėje.
- Tinklelis. Agentų ryšiai vaizduojami kaip tinklelis. Tokiame modelyje agentai gali bendrauti tik su savo kaimynais.
- Euklidinė erdvė. Agentai išdėstomi plokštumoje arba trimatėje erdvėje.
- Geografinės informacinės sistemos. Agentai susieti su realistiniu geografiniu reljefu ir juda juo.
- Tinklai. Tinklai gali būti statiniai ir dinaminiai. Statiniai tinklai, tai tokie tinklai kuriems ryšiai nustatomi iš anksto, o dinaminių tinklų ryšiai yra kintantys laike.

Kaip ir [Mik14] [Sil16] šiame darbe modeliavimas yra atliekamas Tinklo tipo erdvėje. Kaip buvo minėta ankščiau, BitCoin pagrindu veikiančiuose BlockChain tinkluose dalyviai paprastai būna sujungti su 3 – 4 kaimynais. Nors BitCoin tinklas ir neturi aiškios topologijos, šiame modelyje dėl paprastumo BitCoin tinklas yra vaizduojamas kaip grafas G=(V,4). Čia V yra tinko

dalyvių skaičius, kurių kiekvienas turi 4 kaimynus. Tokio tinklo pavyzdys gali būti iliustruojamas taip (11 pav.):

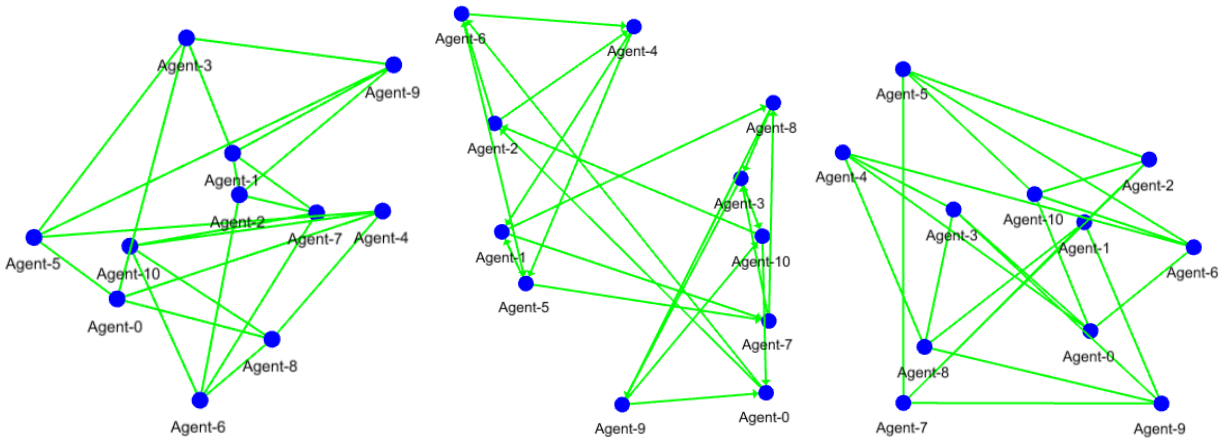


11 pav.: G=(5,4) grafo pavyzdys

Pagal pavaizduotą grafą veikiantis tinklas būtų labai trivialus ir jokių ženklių išvadų neapteiktų. Šiame etape yra generuojami tinklai su keliomis dešimtimis, šimtais ir tūkstančiais dalyvių.

4.1. Tinklo modeliavimas

Repast siūlo įvairius tinklų kūrimo metodus. Šiame darbe yra pasinaudojama generatoriumi *WattsBetaSmallWorldGenerator*. Šis generatorius leidžia sukurti grafą, turintį fiksuotą skaičių briaunų. Kaip jau buvo minėta, šiame modelyje yra laikoma, kad kiekvienas tinklo dalyvis turi 4 kaimynus. Tai reiškia, kad yra generuojamas 4 briaunas turintis grafas. Agentai generuojamame tinkle gali išsidėstyti skirtingai. Kaip pavyzdys 11 dalyvių tinklas gali atrodyti taip (12 pav.):



12 pav.: *Repast* generuojami, skirtingi G=(11,4) tinklą reprezentuojantys grafai

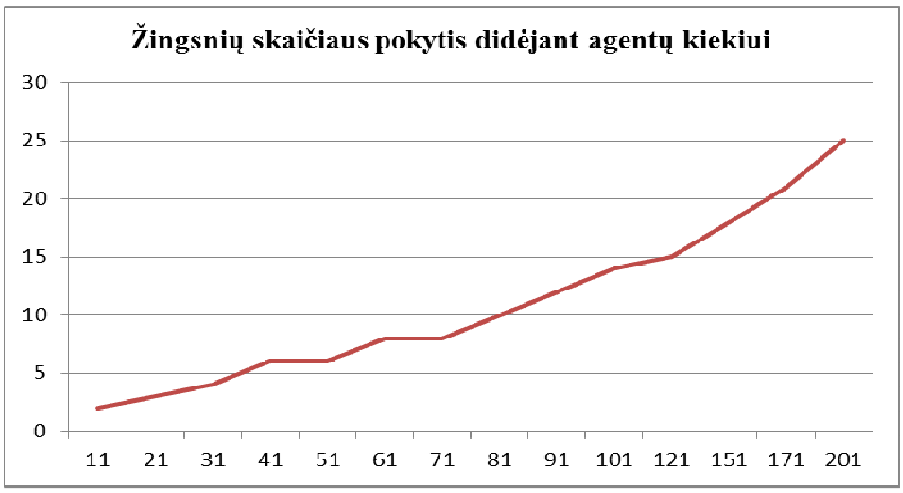
Pavyzdyje tinklai generuojami su 11, 21, 31, 41, 51, 61, 71, 81, 91, 101, 151, 201 agentais. Toks agentų skaičiaus išdėstymas buvo parinktas tam, kad būtų galima palyginti kaip didėjant agentų skaičiui tinkle ilgėja visų tinklo dalyvių informavimas apie naują transakciją.

Techniškai šiame pavyzdyje modelio veikimas atrodo taip:

- Sukuriamas n agentų tinklas, kuris iš tikro yra n viršūnių grafas su 4 briaunom.
- Atsitiktiniu būdu yra parenkamas vienas agentas iš visų n agentų, kuris konkrečioje simuliacijoje generuos transakcijas.
- Pradedama simuliacija, pirmas žingsnis. Kiekvienas agentas pasitikrina savo transakcijų pokyčio reikšmę. Pirmojo žingsnio metu tik atsitiktinai parinktas transakciją siunčiantis agentas turi šią pasikeitimo reikšmę. Jis informuoja savo keturis kaimynus apie naujas transakcijas. Šioje simuliacijoje informavimas vykdomas, kaimynams nustatant atitinkamas reikšmes ir padidinant transakcijų erdvės parametro skaičių vienetu. Dabar keturi agento parametrai turi pasikeitimo reikšmes.
- Žingsnis vykdomas toliau, *Repast* variklis toliau sukasi per visus sugeneruotus agentus. Jeigu tik dabar pasiekiamas ką tik transakciją sugeneravusio agento kaimynas, jis turės ką tik jam nustatytą pokyčio reikšmę, kas reikš, jog jis savo ruožtu toliau informuos jau savo turimus keturis kaimynus (išskyrus jį informavusi ar ankščiau informuotą kaimyną) to pačio žingsnio metu.
- Žingsnis baigiasi kai dalis agentų jau turi atsinaujinę informaciją, t.y. padidinę savo transakcijų erdvę vienetu.
- Žingsnis vykdomas dar kartą, agentai toliau informuoja dar neatsinaujinusius savo kaimyninius agentus. Žingsniai vykdomi tol, kol visų agentų transakcijų erdvėse yra reikšmė 1.
- Užfiksuojama kiek žingsnių turėjo būti įvykdytą lyginant skirtingų agentų skaičiaus tinklus.

4.2. Simuliacijos rezultatai

Pagal gautus rezultatus kaip ir galima buvo tikėtis gaunamas eksponentinė grafiko kreivė (13 pav.). Kuo daugiau agentų tinkle, tuo ilgiau užtrunka viso tinklo informavimas. 11 agentų tinklui susipažinti su nauja transakcija reikia vidutiniškai 2, o 201 agento vidutiniškai 24 žingsnių. Kituose darbo skyriuose bus detaliau ištirta kaip agentų kiekis, taip pat ir blokų dydis, agentų prisijungimo charakteristikos daro įtaką tinklo mastelio keitimo savybėms, bet šiame etape matosi, kad sparčiai didėjant agentų skaičiui, staigiau didėja ir viso tinklo informavimo laikas:



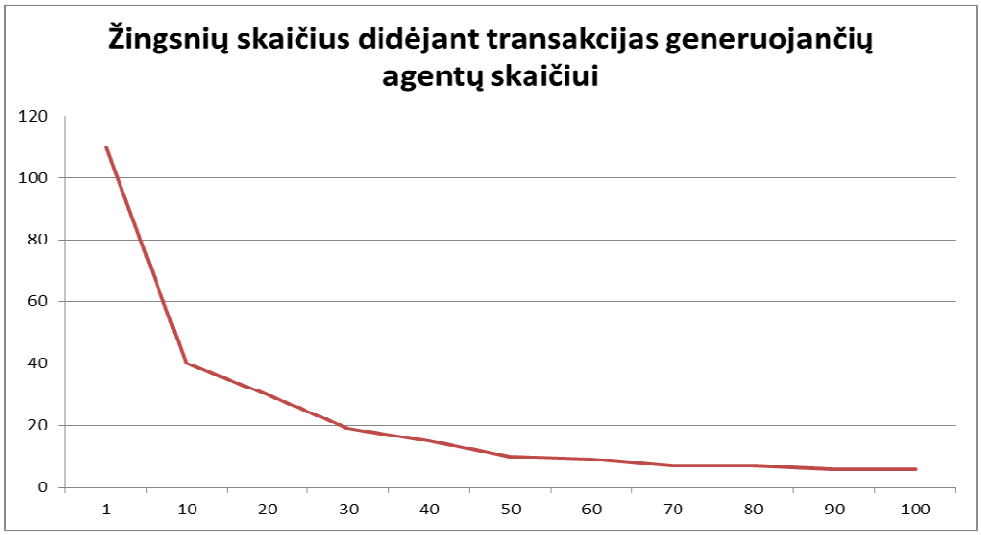
13 pav.: Žingsnių, reikalingų išplatinti transakciją po visą BlockChain tinklą priklausomybė nuo tinklo dalyvių skaičiaus. X ašis – agentų skaičius, Y ašis – žingsnių kiekis

4.3. Modelio praplėtimas

Iš aprašyto bazinio veikimo matosi, kad transakcijų išplatinimo proceso modeliavimas BlockChain tinkluose yra įmanomas. Šiame skyriuje bus išnagrinėtos modelio pritaikymo galimybės mastelio keitimo funkcionalumo modeliavimui. Iš pradžių bus pateiktos galimybės išplėsti transakcijų modelį, o vėliau jis bus papildytas blokų modeliavimo funkcionalumu.

Ankščiau aprašytoje simuliacijoje transakcijas visada siunčia tik vienas agentas. BitCoin (ar kitose sistemose) BlockChain tinkle transakcijas vienu metu siunčia keli tūkstančiai tinklo dalyvių [Nor16][Ant14]. Tam, kad šio modelio veikimas būtų taip pat priartintas prie realiame pasaulyje veikiančių BlockChain sprendimų, pridedama galimybė dviem ir daugiau agentų generuoti transakcijas. Tokiu atveju įmanoma ištirti kaip daugėjant transakcijas generuojančių agentų skaičiui trumpėja viso tinklo užpildymo transakcijomis greitis. Realiame BitCoin tinkle skirtingų agentų generuojamos skirtingos transakcijos skirsis viena nuo kitos savo duomenų struktūromis. Skirtingos transakcijos niekada nebus vienodos, vien todėl, kad jos niekada neturės vienodų identifikacinių hash reikšmių [Ant14]. Šiuo atveju bus laikoma, kad visi agentai siunčia vienodą transakciją. Transakcija bus laikoma ne lėšų pervedimas BitCoin tinkle, bet tam tikros duomenų struktūros platinimas. Tai yra aktualu modeliuojant kitus BlockChain panaudojimo variantus. Pavyzdžiui, kai kuriose privačiose BlockChain platformose galima realizuoti balsavimo funkcionalumą [Nxt14]. Tokiu atveju visi tinklo dalyviai siųstų vienodos struktūros pranešimus, su skirtingomis duomenų reikšmėmis. Balsavime būtų svarbus viso tinklo užpildymo konkrečiais pranešimais laikas, o ne tik 51% tinklo kaip yra BitCoin atveju [Ant14].

Testuojant 1000 dalyvių BlockChain tinklą (Grafą G=(1000,4)) vykdomi bandymai su transakcijas generuojančių agentų skaičiumi nuo 1 iki 100. Simuliacijos rezultatų kreivė pavaizduota apačioje, 14 pav.:



14 pav.: Žingsnių skaičius reikalingas išplatinti transakciją 1000 agentų modelyje, didėjant transakcijas generuojančių agentų skaičiui nuo 1 iki 100. X ašis – generuojančių agentų skaičius. Y ašis – reikalingų žingsnių skaičius

14 pav. matos, kad lyginant su 1 generuojančiu agentu, 10 generuojančių agentų transakcijos (duomenų struktūros) išplatinimą sutrumpina daugiau nei du kartus. Tačiau toliau didėjant generuojančių agentų skaičiui yra pasiekiamas 7–8 žingsnių laikas, kuris toliau sparčiai nemažėja. Tai galima paaiškinti, kad generuojančių agentų skaičiui artėjant prie 500 (pusės visų tinklo dalyvių) daugelis iš agentų turi 1, 2 ar daugiau savo kaimynų grafe, kurie turi būti informuojami apie naują transakciją (duomenų struktūrą). Kaip generuojančių agentų skaičius pasiekia 500 tada žingsnių kiekis yra 2, o kai tampa daugiau nei 500 tada visada užtenka 1 žingsnio užpildyti visą tinklą.

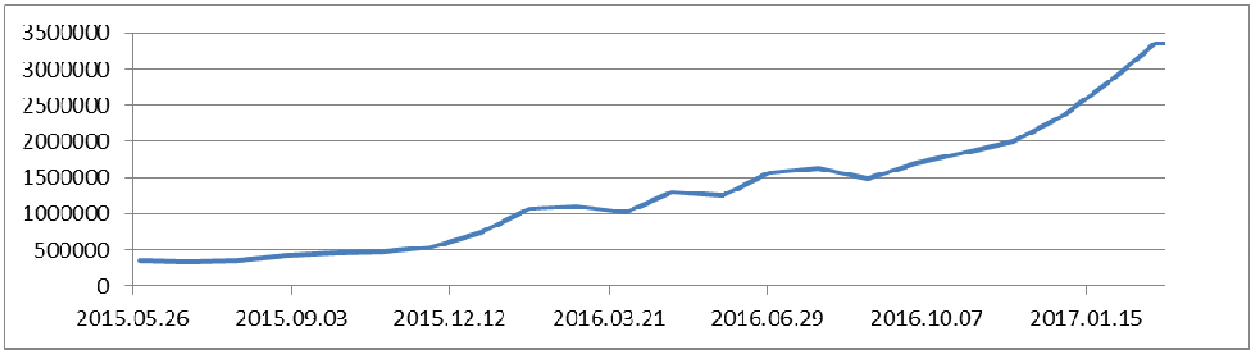
Šiame modelio pritaikyme skirtingi agentai siunčia tą pačią transakciją. BlockChain tinkluose skirtingi agentai siunčia skirtingas transakcijas. Visos siunčiamų transakcijų duomenų struktūros (nesvarbų ar jas siunčia skirtingi ar vieną po kitos tas pats dalyvis) yra skirtingos [Nak09]. Visada skiriasi transakcijų ID, hash reikšmės. Todėl simuliacijoje agentų funkcionalumą galima papildyti taip, kad kiekvienas agentas generuotų skirtingas transakcijas. Šiuo atveju terminas „skirtingas“ reiškia, kad vienas transakcijas generuojantis agentas kaimyninio agento transakcijų erdvėje nustatys reikšmę 1, kitas agentas – 2, kitas – 3 ir panašiai, priklausomai nuo agentų skaičiaus. Šiame simuliacijos etape būtų galima įvertinti kaip greitai tam tikro agento transakcijos plinta BlockChain tinkle. Tai būtų paprasta įvertinti, nes užtektų įvertinti kokių sveikųjų skaitmenų susijusių su atitinkamas transakcijas pateikusiais agentais, yra

daugiausiai, tinkle dalyvaujančių agentų transakcijų erdvėse. Toks tyrimas nėra šio darbo apimtyje, bet tai galėtų būti plačiau išnagrinėta ateities darbuose.

Taip pat, kaip bus nurodyta vėliau, yra aktualu pridėti tam tikrus funkcionalumo ir tinklo infrastruktūros parametrus, darančius įtaką duomenų struktūrų (transakcijų, blokų) išplatinimo greičiui tinkle. Šiuo atveju tai reiškia, kad turės būti modeliuojami ir kiekvieno iš tinklo dalyvių prisijungimo prie tinklo trukdžiai. Šių parametrų realizavimas leis labiau priartėti prie realaus pasaulio funkcionalumo.

4.4. Modelio pritaikymas realaus pasaulio uždaviniams

Iki šiol buvo modeliuojami tik N agentai platinantys transakcijas BitCoin tinkle. Šiame skyriuje modelis yra papildomas M agentais, kurie tinkle platins blokus. Modelis su veikiančiais N ir M agentais turi papildomus (lyginant tik su transakcijų moduliu) parametrus. Svarbiausias iš jų yra M agentų skaičius. Vadovaujantis realiai veikiančiu BitCoin tinklu yra sunku pasakyti koks turėtų būti N ir M agentų santykis. BitCoin tinkle nėra žinoma kiek šiuo metu yra aktyvių išgavėjų [Nak09]. Yra žinoma tik bendras BitCoin tinklo skaičiavimo pajėgumas. Nuo BitCoin sukūrimo pradžios, didėjant BitCoin populiarumui šis dydis tolygiai didėja kaip parodyta 15 pav. (pagal http://www.blockchain.info informaciją):



15 pav.: skaičiavimo pajėgumų didėjimas nuo 2015 metų gegužės iki 2017 metų kovo. Y ašyje – skaičiavimų galingumas Th/s

15 pav. esančiame grafike y ašyje skaičiavimų galingumas yra išreiškiamas dydžiu *Th/s*. Šis dydis reiškia kiek trilijonų (10^{12}) skaičiavimo operacijų yra atliekamą per vieną sekundę. 2017 metų vasario pabaigoje šis dydis buvo pasiekęs rekordinį dydį ir perkopė 3 900 000 *Th/s*, tai reiškia, kad bendras BitCoin tinklo skaičiavimo pajėgumas artimiausiu metu peržvengs 4 kvintilijonų ($4x10^{18}$) operacijų per sekundę ribą. Kaip buvo minėta ankščiau didėjant skaičiavimo pajėgumams, kartu didėja ir *Proof-of-work* matematinio uždavinio sudėtingumas, tam, kad būtų išlaikytas transakcijos patvirtinimo laikas ~ 10 minučių. Todėl šiuo metu nėra

jokių sprendimų įvertinti kiek aktyvių išgavėjų veikia BitCoin tinkle. Šiame darbe, tam, kad būtų galima tik palyginti skirtingų dydžių blokų išplatinimo laikus blokus generuos vienas M agentas.

Šiame agentais grįstame modelyje, transakcijų išplatinimo trukmė yra skaičiuojama simuliacijos žingsniais. Pagrindinis skirtumas tarp atlikto eksperimento [DWa14] ir šio agentų modelio yra tai, kad šiame modelyje yra modeliuojamas skirtingas agentų skaičius, o eksperimentas yra su apibrėžtu 4000 tinklo dalyvių kiekiu. Todėl toliau šiame darbe bus taip pat dirbama su 4000 agentų tinklu tam, kad modelis būtų atitinkamai sukalibruotas su vykdytų eksperimentų rezultatais.

Šio agentais grįsto modelio veikimo rezultatus veikiant 4000 agentų, pateikus tokiu pačiu formatu kaip 1 lentelėje gaunama:

Dalis	**žingsniai**
50%	135
75%	210
90%	400
95%	450
99%	528

2 lentelė: agentais grįsto modelio transakcijų vidutinė išplitimo trukmė

2 lentelėje esantys rezultatai nurodo vidutinį tam tikros transakcijų dalies išplitimo greitį žingsniais. Modelis buvo leistas 10 kartų ir pateikti veikimo vidurkiai. Palyginus 1 ir 2 lenteles matosi, kad eksperimente 50% transakcijų yra išplatinama 100 kartų greičiau nei 99%, o šiame modelyje tik 4 kartais greičiau nei 99% (Paprastumo dėlei laikykime, kad 99% == 100%)

Tam, kad modelis būtų priartintas prie eksperimento sąlygų, reikalinga įtraukti tinklo trukdžio parametrus. Kad būtų simuliuotos tikro tinklo sąlygos, trukdžiai turėtų priklausyti nuo tinklo dalyvių ir transakcijų ar blokų platinimo šaltinio atstumo. Tokiu būdu būtų modeliuojamos situacijos kai toliausiai grafe esantys dalyviai vėliausiai sužino apie išplatintą informaciją. Šiuo atveju terminas „atstumas“ reiškia fizinį atstumą tarp nutolusių tinklo taškų. Kuo toliau vienas nuo kito fiziškai yra tinklo dalyviai tuo ilgiau tarp jų keliauja pranešimas ir tuo daugiau tinklo trukdžių gali pasitaikyti kelyje. Fizinis atstumas yra išreiškiamas ne atstumo matavimo vienetais (metrais, kilometrais), bet tarp dviejų tinklo dalyvių esančių kitų dalyvių skaičiumi. Pavyzdžiui, jeigu tinklo dalyvis esantis grafo viršūnėje nr. 1253 siunčia transakciją į tinklą, tinklo dalyvis esantis grafo viršūnėje nr. 1258 gaus pranešimą, po to kai apie jį sužinos esantys kiti 4 dalyviai (1254, 1255, 1256, 1257). Toks veikimas yra simuliuojamas šiame modelyje, bet būtų nebūtinai vienintelis teisingas tikrame BitCoin tinkle. Dėl tinklo greitaveikos niuansų dalyvis nr. 1258 apie iš nr. 1253 išsiųstą transakciją galėtų sužinoti ir iš dalyvio nr. 1259 ir iš nr. 1260. Šiame modelyje yra modeliuojamas veikimas kai pranešimai iš šaltinio yra paskleidžiami tolygiai jiems plintant nuo epicentro (šaltinio). Toliau tobulinant ir vystant modelį tai padėtų modeliuoti

situacijas kai arčiau esantys dalyviai gali pradėti vykdyti dvigubo išleidimo atakas prieš ankščiau, bet toliau grafe paleistas transakcijas. Pavyzdžiui, grafo viršūnėje nr. 5 esantis M agentas formuoja bloką į kurį, šalia kitų transakcijų, įdeda vieną transakciją (ID: 587123) kuria siunčia BTC į M agento kontroliuojamą adresą. Suformavęs bloką agentas M neplatina naujo bloko tinkle, bet kitam agentui N esančiam grafo viršūnėje nr. 2681 siunčia apmokėjimą už tam tikras prekes ir paslaugas už tuos pačius BTC iš kurių sudaryta naujame Bloke esanti transakcija ID: 587123. Paleidęs transakciją, agentas M pradeda naujo bloko platinimą tinkle. Visi M esantys agentai sužino apie naują bloką ir įtraukia į savo pusėje turimas blokų grandines. Gali susidaryti situacija kai agentas N iš karto išsiunčia prekes ar paslaugas, bet visi kiti tinklo dalyviai priimdami naują bloką į savo blokų grandines paverčia transakciją ID: 587123, negaliojančia. Taip pat tai padėtų modeliuoti situacijas kai skirtingi tinklo dalyviai gauna informaciją apie skirtingus į vieną ir tą pačią vietą blokų grandinėje pretenduojančius blokus.

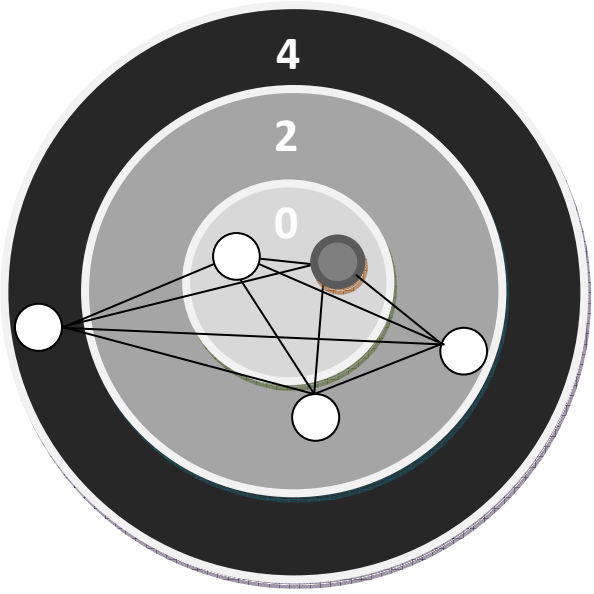
Tuo tikslu N ir M agentai yra papildomi parametru *TINKLO_TRUKDIS*. Šis, sveikųjų skaičių reikšmes galintis įgauti parametras nusako kiek simuliacijos žingsnių turi praleisti tinklo dalyvis, kol galės priimti transakciją arba bloką. Tokiu būdu simuliuojami tinklo trukdžiai. Skirtingiems dalyviams šis parametras yra parenkamas priklausomai nuo to kaip toli jie yra nuo transakcijas ar blokas inicijuojančių tinklo dalyvių (Tokius dalyvius vadinsime *Šaltiniais*). Kiekvieną kartą kai N ar M agentas gauna informaciją apie transakciją, jis patikrina ar parametras *TINKLO_TRUKDIS* yra 0, jei ne, šis parametras yra sumažinamas vienetu ir kitu simuliacijos žingsniu šis konkretus agentas vėl gauna pranešimą apie naują bloką ar transakciją. Kai parametras *TINKLO_TRUKDIS* tampa 0 tada transakcija ar blokas yra pridedamas į konkretaus N ar M agento transakcijų erdvę (transakcija) ar grandinę (blokas). Pradedant simuliaciją yra nustatoma pradinė parametro *TINKLO_TRUKDIS* reikšmė. Ši pradinė reikšmė nusako kiek žingsnių turėtų praleisti arčiausiai Šaltinio esantys N ir M agentai. Toliau esančių N ir M agentų tinklo trukdžiams aprašyti reikalinga įvesti papildomus parametrus nusakančius kaip turėtų didėti parametras *TINKLO_TRUKDIS* agentams tinkle tolstant nuo Šaltinio. Papildomi parametrai:

TRUKDZIO_DELTA – kiek žingsnių turėtų padidėti *TINKLO_TRUKDIS*;

TRUKDZIO_ZINGSNIS – kas kiek agentų tolstant turėtų padidėti *TINKLO_TRUKDIS* dydžiu *TRUKDZIO_DELTA*.

Kiekviename agente parametrai *TRUKDZIO_DELTA* ir *TRUKDZIO_ZINGSNIS* yra konstantos viso modelio veikimo metu. Inicijuojant simuliaciją kiekviename agente atitinkamai užsipildo parametrų *TINKLO_TRUKDIS* reikšmės. Pavyzdžiui, simuliacijoje nustačius

parametrus: *TINKLO_TRUKDIS* =0; *TRUKDZIO_DELTA* = 2; *TRUKDZIO_ZINGSNIS* = 100. Simuliacijoje su 4000 agentų iš kurių transakcijas generuoja agentas N nr. 1509. Tokiu atveju agentai diapazone [1409....1609] *TINKLO_TRUKDIS* reikšmes turės lygias nuliui; diapazonuose [1309...1409) ir (1609...1709] *TINKLO_TRUKDIS* = 2; diapazonuose [1209...1309) ir (1709...1809] *TINKLO_TRUKDIS* = 4 ir panašiai. Tokį tinklo trukdžio pokytį, ankščiau 11 pav. pavaizduotame tinkle, iliustruoja žemiau pateiktas pavyzdys.:



16 pav.: BlockChain tinklo trukdžių išsidėstymo pavyzdys

16 pav. pavyzdyje transakciją arba bloką siunčia pažymėtoje viršūnėje esantis agentas. Nuo šaltinio nutolę agentai šviesiai pilkoje, pilkoje ir juodoje zonose turi *TINKLO_TRUKDIS* reikšmes atitinkamai: 0; 2; 4.

Norint šio agentais grįsto BlockChain modelio veikimo rezultatų sugretinti su [DWa14] rodomais rezultatais, svarbu rasti minėtų tinklo trukdžių parametrų reikšmes, su kuriomis modelis grąžintų eksperimente pateiktas transakcijų ir blokų išplatinimo trukmių proporcijas.

Atlikus eksperimentus buvo gauti tokie, žemiau pateikti rezultatai:

TRUKDZIO_DELTA = 100;

TRUKDZIO_ZINGSNIS = 10;

100% tinklo užpildymas pasiekiamas per: 35 000 žingsnių.

Šiuo atveju nors ir yra gaunama didesnė viso tinklo užpildymo trukmė, tuo pačiu padidėja ir 50% tinklo užpildymo laikas (50% užpildoma per ~ 10 000 žingsnių). Tai reiškia, kad nėra pasiekiamos [DWa14] nurodytos proporcijos. Toks pat neatitikimas yra ir su 75% bei 90% tinklo užpildymo proporcijomis. Pagal 1 lentelę parengta užpildymo lentelė atrodo taip:

Dalis	žingsniai
50%	11500
75%	25300
90%	30200
100%	35000

3 lentelė: tinklo užpildymo rezultatų lentelė

Tam, kad 3 lentelėje 50%, 75%, 90% ir 100% užpildymo skirtumai būtų priartinti prie 1 lentelėje esančių proporcijų, modelio veikimas yra papildomas skirtingu simuliacijos veikimu iki ir po 50% užpildymo. Daroma prielaida, kad 50% tinklo dalyvių, kurie pirmieji sužino apie išplatinamas duomenų struktūras, yra arčiau šaltinio nei likę kiti 50%. Tokiu atveju veikiant simuliacijai pirma pusė tinklo yra pildoma nenaudojant trukdžių parametrų. Trukdžių funkcionalumas yra įjungiamas apie duomenų struktūras informuojant likusius 50% tinklo dalyvių. Kad būtų gautos atitinkamos proporcijos šioje simuliacijoje pirmieji 2000 agentai yra informuojami be tinklo trukdžių, o likusiems 2000 yra pritaikomi atitinkami trukdžių parametrai. Tam, kad būtų išlaikytas didesnis viso tinklo užpildymo laikas yra vykdomi eksperimentai ir su per puse sumažinta parametro *TRUKDZIO_ZINGSNIS* reikšme, kuriai yra nustatoma reikšmė: 5. Taip pat svarbu pažymėti, kad šiuo atgebu yra dirbama jau ne su N (kaip buvo transakcijų siuntimo atveju), o su M agentais, nes siunčiamos didesnės duomenų struktūros – blokai. Nors šiame etape ir nėra esminio skirtumo tarp N ir M agentų veikimo, tačiau ruošiant modelį detalesniam BlockChain technologijos modeliavimu yra būtina aiškiai atskirti šių dviejų agentų tipų paskirtis. Su skirtingais trukdžių parametrais gaunami žemiau pateikti modelio veikimo rezultatai:

TRUKDZIO_DELTA = 50; TRUKDZIO_ZINGSNIS = 5;

Dalis	žingsniai
50%	135
75%	9200
90%	12600
100%	14770

TRUKDZIO_DELTA = 100; TRUKDZIO_ZINGSNIS = 5;

Dalis	žingsniai
50%	135
75%	16200
90%	26800
100%	29000

TRUKDZIO_DELTA = 200; TRUKDZIO_ZINGSNIS = 5;

Dalis	žingsniai
50%	135
75%	25200
90%	54600
100%	57000

TRUKDZIO_DELTA = 300; TRUKDZIO_ZINGSNIS = 5;

Dalis	žingsniai
50%	135
75%	46700
90%	84200
100%	87300

TRUKDZIO_DELTA = 50; TRUKDZIO_ZINGSNIS = 10;

Dalis	žingsniai
50%	135
75%	3500
90%	6500
100%	7200

TRUKDZIO_DELTA = 100; TRUKDZIO_ZINGSNIS = 10;

Dalis	žingsniai
50%	135
75%	9100
90%	12300
100%	14500

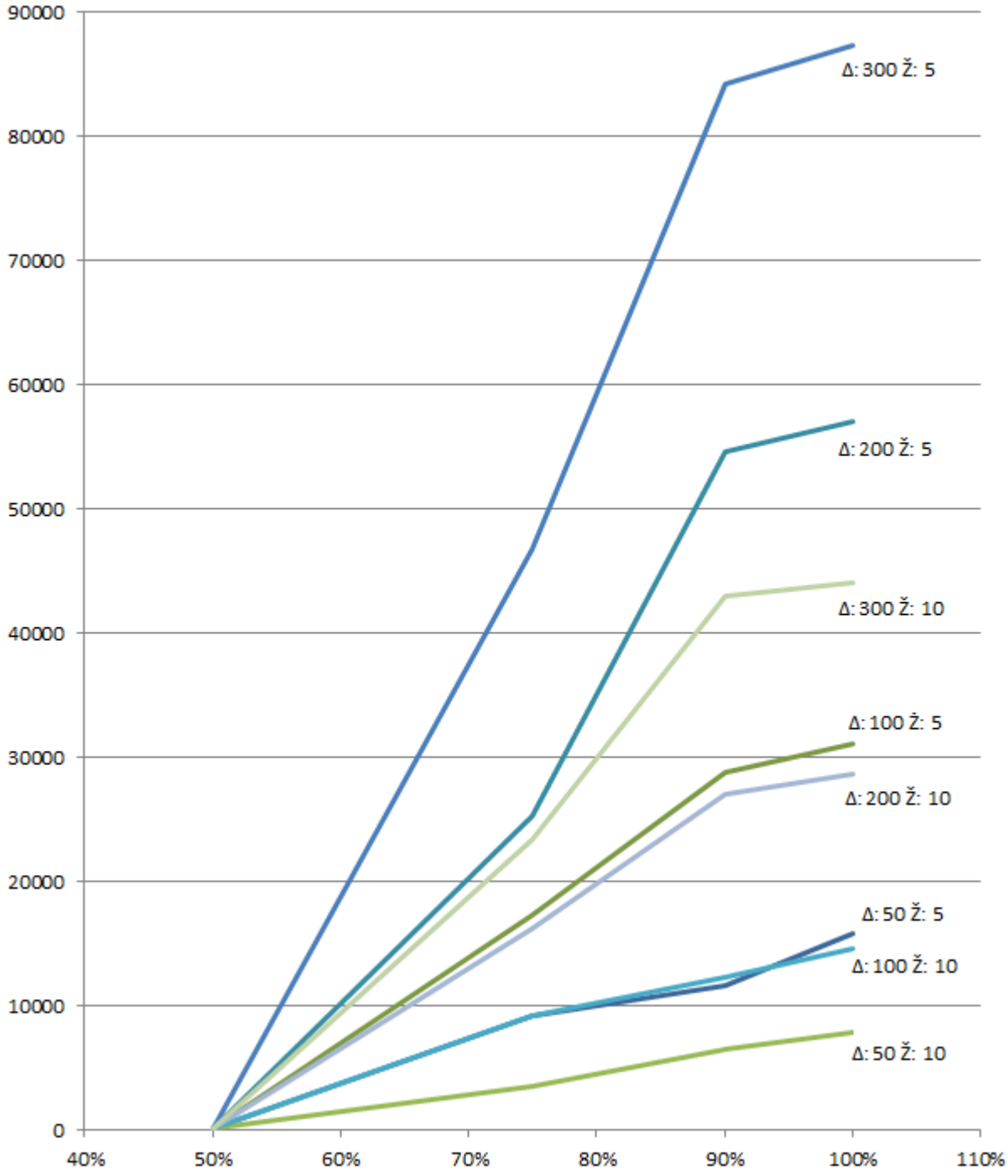
TRUKDZIO_DELTA = 200; TRUKDZIO_ZINGSNIS = 10;

Dalis	žingsniai
50%	135
75%	16200
90%	27000
100%	28600

TRUKDZIO_DELTA = 300; TRUKDZIO_ZINGSNIS = 10;

Dalis	žingsniai
50%	135
75%	23300
90%	43000
100%	44100

4 lentelė: 50, 75, 90, 100 procentų tinklo užpildymas su skirtingomis trukdžių parametrų reikšmėmis



18 pav.: 50, 75, 90, 100 procentų tinklo užpildymo su skirtingomis trukdžių parametrų reikšmėmis žingsnių skaičiaus palyginimo grafikai. Čia Δ – TRUKDZIO_DELTA, Ž – TRUKDZIO_ZINGSNIS

18 pav. modelio rezultatų suvestinėje matosi, kad rezultatai, arčiausi 1 lentelėje išvardintų [DWa14] eksperimento proporcijų tarp 50% ir 100% užpildymo yra gaunami tada kai trukdžių parametrai yra: *TRUKDZIO_DELTA = 50; TRUKDZIO_ZINGSNIS = 5 bei TRUKDZIO_DELTA = 100; TRUKDZIO_ZINGSNIS = 10.* Kadangi *TRUKDZIO_DELTA =*

100; TRUKDZIO_ZINGSNIS = 10 parametrų rezultatai buvo truputį artimesnį [DWa14] eksperimento rezultatams, toliau modelyje bus naudojami būtent šie trukdžių parametrai.

Transakcija, su kuria buvo vykdomas eksperimentas [DWa14] yra 20KB dydžio. Blokas, eksperimente yra 500KB dydžio. Akivaizdu, kad eksperimente vykdomos duomenų struktūrų platinamų tinkle operacijų trukmė priklauso ne tik nuo tinklo trukdžių, bet taip pat ir nuo duomenų struktūrų dydžių. Šiame, agentais grįstame modelyje taip pat yra būtina įvesti duomenų esybės dydžio sąvoką ir patikrinti ar ji daro atitinkamą įtaką veikimo rezultatams. Tam bus panaudojami parametrai *TX_DYDIS* ir *BLOKO _DYDIS*. Šiuo atveju, galima būtų išsiversti ir su vienu dydžio parametru, nes lyginant eksperimento rezultatus svarbiausia yra dydžių skirtumai. Tačiau, kad būtų galimybė ateityje praplėsti modelį detalesniu blokų modeliavimu, bus įtraukiami šie du parametrai. Šie parametrai panašiai kaip ir parametras *TINKLO_TRUKDIS* nurodo kiek žingsnių turi praleisti simuliacija, kol transakcija bus pridėta į transakcijų erdvę arba blokas pridėtas į blokų grandinę. Šiame modelyje yra daroma prielaida, kad 4 lentelėje nurodyti trukdžių parametrai grąžina tinkamus simuliacijos veikimo rezultatus, platinant vieną transakciją tada, kai *TX_DYDIS = 1*. Tam, kad simuliacijos rezultatai būtų palyginami su [DWa14] eksperimento rezultatais turi būti tokia parametro *BLOKO_DYDIS* reikšmė, su kuria simuliacijos 50% tinklo užpildymo rezultatų reikšmė būtų 3,5 karto didesnė nei simuliacijai veikianti be šio bloko dydžio parametrų. Taip yra dėl to, kad 1 lentelėje pavaizduotų eksperimento rezultatų 50% išplatinimo tinkle transakcijų ir blokų reikšmės atitinkamai yra 1,3s ir 4,5s. Šiuo atveju pagrindinis tikslas yra rasti tokias parametro *BLOKO_DYDIS* reikšmę, kuri pagal 4 lentelės esančius simuliacijos rezultatus užtikrintų jog, 50% tinklo užpildymo trukmė būtų 470 (~135 * 3,5) žingsnių. Atlikus eksperimentus, kai trukdžio parametrai: *TRUKDZIO_DELTA = 100; TRUKDZIO_ZINGSNIS = 10*, buvo gauta:

TRUKDZIO_DELTA = 100; TRUKDZIO_ZINGSNIS = 10;

Dalis	**BLOKO_DYDIS**	**žingsniai**
50%	1	135
50%	50	180
50%	100	230
50%	500	390
50%	700	470
50%	1000	710

5 lentelė: 50% tinklo užpildymo žingsnių pokytis keičiantis parametrui *BLOKO_DYDIS*

Iš 5 lentelės matosi, kad 1 lentelės atitinkančias proporcijas simuliacija (su statinėmis trukdžių reikšmėmis) pasiekia tada, kai parametro *BLOKO_DYDIS* reikšmė yra 700. Galima daryti išvada, kad modelis veikia maksimaliai priartėjęs prie realaus pasaulio sąlygų, kai jo parametrai yra: *TRUKDZIO_DELTA = 100; TRUKDZIO_ZINGSNIS = 10; BLOKO_DYDIS =*

700. Su tokiais parametrais veikianti simuliacija leidžia modeliuoti 500KB dydžio bloko platinimą BitCoin principais veikiančiame tinkle. Norint modeliuoti mastelio keitimo savybes kai yra keičiamas bloko dydis reikia tiesiog proporcingai pakeisti parametrą *BLOKO_DYDIS*. Pavyzdžiui, norint modeliuoti 1 MB bloko išplatinimą, parametrui *BLOKO_DYDIS* reikia nustatyti reikšmę 1400. Modeliavimo su skirtingais blokų dydžiais rezultatai pateikti lentelėje apačioje:

TRUKDZIO_DELTA = 100; TRUKDZIO_ZINGSNIS = 10;

Dalis	**Bloko dydis**	**BLOKO_DYDIS**	**Žingsniai**
100%	250 KB	350	14900
100%	500 KB	700	15500
100%	1 MB	1400	16100
100%	2 MB	2800	17900
100%	4 MB	5600	19900
100%	10 MB	14000	28000

6 lentelė: modeliavimo rezultatai kintant bloko dydžiui. Stulpelyje „Bloko dydis“ yra nurodomas BitCoin bloko dydis realybėje. Stulpelyje „BLOKO_DYDIS“ – modelio bloko parametras

Pagal 6 lentelę matosi, kad 500 KB blokas yra išplatinamas 6,9% lėčiau (kas yra lygu 1000 žingsnių) nei yra išplatinama transakcija. 1 lentelėje matosi, kad [DWa14] eksperimento metu 500 KB blokas buvo išplatintas 8,6 sekundės arba 6,8% lėčiau nei transakcija. Pagal šį agentais grįsto modelio suderinimą su vykusiu realiu eksperimentu, galima teikti, jog simuliacijos veikimas yra priartintas realioms veikimo sąlygoms.

Kiekvieno N ir M agento trukdžių parametrai šiame modelyje yra fiksuoti ir vieną kartą nustatyti agentui priklausomai nuo jo pozicijos šaltinio atžvilgiu, nekinta visos simuliacijos metu. Toks sprendimas buvo pasirinktas tam, kad greičiau ir efektyviau būtų pademonstruotos BlockChain sistemų modeliavimo agentais galimybės. Ateityje kuriant sudėtingesnius modelius būtų reikalinga realizuoti atsitiktinį trukdžių parametrų generavimo funkcionalumą, kad būtų galima gauti realistiškesnius rezultatus.

Rezultatai ir išvados

Rezultatai

- Sukurtas agentais grįstas BlockChain savybes tiriantis modelis.
- Lyginant su realybėje vykdytais BitCoin duomenų struktūrų platinimo BlockChain tinkle eksperimentais, matosi, kad agentais grįstame modelyje parinkus atitinkamus parametrus yra įmanoma gauti tokias pačias duomenų struktūrų laiko išplatinimo proporcijas.
- Modelio veikimo rezultatai rodo, kad padvigubinus šiuo metu egzistuojantį 1MB [Nak09] maksimalų bloko dydį jo išplatinimo trukmė vidutiniškai pailgėtų 5%. Tai nėra žymus padidėjimas turint omenyje įvertinimus kaip toks bloko dydžio padidinimas kardinaliai pagreitintų BitCoin sistemų transakcijų apdorojimą [Her17].

Iš šio modeliavimo rezultatų negalima teigti, kad bloko padidinimas iš principo yra teigiamas dalykas ir nesukeltų jokių rimtų trikdžių BitCoin BlockChain tinkle. Su bloko padidinimu yra susiję eilė kitų rizikų, tokių kaip grandinės išskyrimas, decentralizacijos savybių praradimas, dvigubo išleidimo atakų tikimybės padidėjimas [Her17] [Riz15].

Išvados

- BitCoin, o kartu ir BlockChain sistemų savybes tiriančius modelius yra įmanoma modeliuoti agentais grįstu modeliavimu. Atitinkamos programinės priemonės leidžia kurti BlockChain tinklus ir atskirus jų dalyvius simuliuojančias esybes;
- Bloko išplatinimo greitis BitCoin tinkle ženkliai nesulėtėtų, padidinus maksimalų bloko dydį du ar keturis kartus.

Sprendžiant iš to, kad yra įmanoma sukurti agentais grįsta transakcijų ir bloko platinimą simuliuojantį modelį, galima daryti prielaidą ir dėl kitų BlockChain pagrindu veikiančių sistemų procesų modeliavimo. Šiame darbe aprašyti agentų parametrai gali būti papildyti naujais atributais sudėtingesnių BlockChain savybių modeliavimui.

Ateities darbai

Šiame modelyje mastelio keitimo savybių tyrimas yra kaip pavyzdys to kokias BlockChain problemas ir esybes būtų galima tirti agentais grįstu modeliavimu. Bloko maksimalaus dydžio padidinimas tai vienas iš būdų spręsti BitCoin mastelio keitimo klausimus. BlockChain technologijos bendras klausimas įvairioms šią technologiją realizuojančioms platformoms tai didesnių duomenų kiekių saugojimas tinkle. BitCoin, Ethereum, Ripple platformos leidžia saugoti tik transakcijų informaciją kai vieno įrašo dydis yra iki kelių dešimčių ar vieno šimto kilo baitų. Skirtingų transakcijų dydžių modeliavimas galėtų būti taip pat viena iš modeliavimo užduočių tiesiogiai lemiančius ir blokų dydžių pokyčius.

Bloko padidinimas tai ne vienintelis mastelio keitimo būdas, kuriuo tikimasi pagerinti BitCoin sistemos veikimo savybes. 2017 metų pradžioje buvo iškelta ir didelio populiarumo sulaukė vadinamas „Atskirto liudininko“ (*angl. – Segregated Witness*) sprendimas [Lee17][Пол16]. Šio sprendimo esmė yra iškelti tam tikrą meta–informaciją iš transakcijos duomenų struktūros ir saugoti ją trečiosios pusės kontroliuojamose saugyklose. Tikimasi, kad šis sprendimas padėtų sumažinti transakcijos dydį, o tai reikštų, kad daugiau transakcijų tilptų į bloką ir taip pagreitėtų jų apdorojimas BitCoin tinkle. Atskirto liudininko sprendimas taip pat galėtų būti modeliuojamas agentais, sukuriant naują esybę simuliuojančia trečiosios šalies, saugančios transakcijų meta duomenis ir funkcionalumą.

Visos 3.6. skyriuje apžvelgtos rizikos taip pat galėtų būti modeliuojamos šiame darbe aprašyto modelio pagalba, panaudojus ankščiau pasiūlytus modelio parametrus. Modeliuojant situacijas susidarančias neteisingų transakcijų paleidimo, dvigubo išleidimo atakos vykdymo, staigaus BlockChain bazės didėjimo metu, ypač svarbus būtų einamąją BlockChain grandinės būseną nusakantis parametras. Papildžius modelį tokiu parametru, būtų galima modeliuoti įvairias minėtas situacijas, kurių veikimo pagrindiniai indikatoriai būtų grandinės aukštis, aukščiausio bloko ID, konkrečioje grandinės dalyje esančių blokų unikalumas. Paskutinis parametras yra ypač aktualus modeliuojant grandinių išsiskyrimo situacijas.

Papildžius atskirų agentų veikimo funkcionalumą, būtų galima modeliuoti neteisingų transakcijų skleidimą BitCoin tinkle. Bet kuris tinklo dalyvis gavęs transakciją ar bloką, kurio duomenys prieštarauja tinklo dalyvio turimai informacijai, turėtų sustabdyti tolimesnį tokios esybės persiuntimą.

Ateityje būtų galima kurti modelius simuliuojančius ne tik BitCoin, bet taip pat ir kitų BlockChain platformų veikimą. Išmaniųjų kontraktų pagrindu veikiančių BlockChain sistemų (Ethereum, HyperLedger ir kitų) modeliavimas būtų ypač aktualus pritaikant išmaniųjų

kontraktų logiką FinTech sektoriuje. Skirtingai nuo BitCoin, kurio sistemoje veikiantys procesai (kartu ir modeliavimo galimybės) yra aiškūs, išmaniųjų kontraktų funkcionalumas tiesiogiai priklauso nuo jų realizacijos ir veikimo srities. Tačiau 2016 metų vasarą prieš Ethereum tinklą įvykusi DAO ataka [Sie16], kurios metu pasinaudojus išmaniojo kontrakto klaidomis buvo pavogta 3,6 milijonų Ether valiutos vienetų (tuo metu vertų apie 70 milijonų eurų), perša mintį, kad prieš įdiegiant išmanųjį kontraktą į realią BlockChain sistemą, verta apsvarstyti jo veikimo modeliavimo agentais grįstame modelyje galimybę. Ethereum ekosistema būtų palanki terpė modeliavimui agentais ir dėl to, kad programinis išmaniųjų kontraktų kodas šioje platformoje visada turi iš anksto nustatytą maksimalų veikimo žingsnių kiekį [But14]. Taip yra todėl, kad būtų išvengta taip vadinamos Sustojimo (angl. – *Halting*) problemos kai iš programos parametrų ir savybių neįmanoma pasakyti ar yra rizika, jog programa veiks be pabaigos. Kiekvienas žingsnis yra įkainojamas Ether valiuta, taip išmaniųjų kontraktų kūrėjai yra skatinami rašyti kaip įmanoma efektyvesnį kodą. Jeigu išmanusis kontraktas baigia veikimą ankščiau nei pasiekiamas nustatytas žingsnių kiekis, nepanaudoti valiutos vienetai grąžinami atgal į kūrėjo sąskaitą. Jeigu išmanusis kontraktas išnaudoja visus numatytus žingsnius, bet dar nepasiekia savo veikimo pabaigos, tokiu atveju visi jo įvykdyti pakeitimai BlockChain platformoje yra atšaukiami [But14].

Preliminariai vertinant toks išankstinis žingsnių nustatymas koreliuoja su šiame darbe aprašytais simuliacijos žingsniais Repast Symphony modeliavimo agentais platformoje. Išmaniųjų kontraktų veikimo modeliavimas galėtų būti stiprus kandidatas kitiems modeliavimo agentais darbams.

2.3.6 skyriuje minėtas BTU įgyvendinimo klausimas taip pat yra tampriai susijęs su ankščiau, 3.5. skyriuje minėta BlockChain išskyrimo (*BlockChain fork*) situacija. 2017 metais tik 2,5% BTC tinklo dalyvių pasirinko naująją grandinę. Pagal minėtą paaiškinimą tai kietojo tipo išskyrimas. Modeliavimas agentais galėtų padėti įvertinti panašius scenarijus dar prieš priimant atitinkamus sprendimus.

Šaltinių sąrašas

[Ant14] Andreas M.Antonopoulos. Mastering BitCoin. 2014.

[Bei17] Ofir Beigel. Is Bitcoin Mining Profitable in 2017. 2017. [žiūrėta 2017-03-25] Prieiga per internetą: https://99bitcoins.com/bitcoin-mining-profitable-beginners-explanation/

[BHR13] Francoise Baude, Ludovic Henrio, Cristian Ruz. Programming distributed and adaptable autonomous components – the GCM/ProActive framework. 2013.

[BMV09] Stefania Bandini, Sara Manzoni and Giuseppe Vizzar. Agent Based Modeling and Simulation: An Informatics Perspective. 2009. [žiūrėta 2016-12-10]: http://jasss.soc.surrey.ac.uk/12/4/4.html

[But14] Vitaliy Buterin. Ethereum: A Next-Generation Cryptocurrency and Decentralized Application Platform. 2014.

[Caf15] Grace Caffyn. What is the Bitcoin Block Size Debate and Why Does it Matter? 2015.[žiūrėta 2016-10-26] Prieiga per internetą: http://www.coindesk.com/what-is-the- bitcoinblock-size-debate-and-why-does-it-matter/

[Cas17] Amy Castor. A Short Guide to Bitcoin Forks. 2017. [žiūrėta 2017-03-25] Prieiga per internetą: http://www.coindesk.com/short-guide-bitcoin-forks-explained/

[DWa14] Christian Decker, Roger Wattenhofer. Information Propagation in the Bitcoin Network. 2014.

[DSt16] Nicolas Dorier, Bill Strait. BlockChain Programming in C#. 2016.

[ESi14] Ittay Eyal and Emin G¨un Sirer. Majority is not Enough: Bitcoin Mining is Vulnerable. 2014.

[Gre16] Gideon Greenspan. Four genuine blockchain use cases. 2016. [žiūrėta 2016-10-15] Prieiga per internetą: http://www.multichain.com/blog/2016/05/four-genuine-blockchain-use-cases/

[Gun14]]N. J. Gunther. How to Quantify Scalability. The Universal Scalability Law (USL). 2014. [žiūrėta 2016-11-07] Prieiga per internetą: http://www.perfdynamics.com/Manifesto/USLscalability.html

[Her17] Alyssa Hertig. CoinDesk Explainer: The Bitcoin Unlimited Debate. 2017. [žiūrėta 2017-03-25] Prieiga per internetą: http://www.coindesk.com/coindesk-explainer-bitcoin-unlimited-debate/.

[Hyp16] Hyperledger Whitepaper. 2016.

[Lee17] Charlie Lee. Many People Do Not Know the Meaning of Segregated Witness. 2017. Prieiga per internetą: https://www.crypto-news.net/the-meaning-of-segregated-witness/

[Mik14] Algirdas Mikoliūnas. Programų sistemų architektūrų tyrimas taikant agentais grįstą modeliavimą. 2014.

[MNo09] Charles M. Macal Michael J. North. AGENT-BASED MODELING AND SIMULATION. 2009.

[Mwa16] Mabvuto Mwale. Modelling the Dynamics of the Bitcoin Blockchain. 2016.

[Nak09] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2009.

[Nor16] Steven Norton. CIO Explainer: What Is Blockchain? 2016.

[Nxt14] Nxt community. Nxt Whitepaper. 2014.

[Riz15] Peter R. Rizun. A Transaction Fee Market Exists Without a Block Size Limit. 2015.

[Pri16] Rob Price. The 18 companies that control bitcoin in 2016. 2016. [žiūrėta 2016-12-27] Prieiga per internetą: http://uk.businessinsider.com/bitcoin-pools-miners-ranked-2016-6/#16-bitminter--02-3

[Sco16] Allen Scott. 'The' Blockchain Vs. 'A' Blockchain: Setting the Record Straight. 2016 [žiūrėta 2016-10-25] Prieiga per internetą: https://news.bitcoin.com/blockchain-setting recordstraight/

[Shi16] Laura Shin. Central Banks Explore Blockchains: Why Digital Dollars, Pounds Or Yuan Could Be A Reality In 5 Years. 2016.[žiūrėta 2016-10-15] Prieiga per internetą: http://www.forbes.com/sites/laurashin/2016/10/12/central-banks-explore-blockchainswhy-digital-dollars-pounds-or-yuan-could-be-a-reality-in-5-years/?_ke=YW5hdG9saWpuQGdtYWlsLmNvbQ%3D%3D#15b67f8676d8

[Sie16] David Siegel. Understanding The DAO Attack. 2016. [žiūrėta 2017-03-27] Prieiga per internetą: http://www.coindesk.com/understanding-dao-hack-journalists/

[SJ05] A Vijay Srinivas, D Janakiram. A model for characterizing the scalability of distributed systems. 2005.

[Пол16] Евгений Половцев. Что такое Segregated Witness и как он может улучшить биткоин. 2016. [žiūrėta 2017-03-25] Prieiga per internetą: http://forklog.com/chto-takoe-segregated-witness-i-kak-on-mozhet-uluchshit-bitkoin/