|  |  |  |  |
| --- | --- | --- | --- |
| МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ  Кафедра програмних систем і технологій  **Курсова робота**  на тему  “Розробка блокчейн вузла у якості автоматизованої системи безготівкових платежів” | | | |
| **Виконав:** | Гоша Давід | **Перевірив**: |  |
| Група | ІПЗ-33 | Дата перевірки |  |
| Форма навчання | денна | Оцінка |  |
| Спеціальність | 121 |
| 2022 | | | |

# Анотація

Ця робота починається з короткого ознайомлення з основними поняттями біткоїн і блокчейн, після чого йде опис різних відомих атак на мережу біткоїн. Після цього поступово описуються основні складові блокчейну та архітектури мережі найбільш популярних проектах. У наступних пунктах відбувається математичне моделювання мережі з метою дослідження її основних принципів. На основі останнього проектуються власна автоматизована система безготівкових платежів.

## **Зміст**

## **Список малюнків**

**РОЗДІЛ 1**

# Вступ

### **Цілі**

Оскільки технологія біткоїн викликає великий інтерес навіть серед найбільших банків, існує необхідність ретельно проаналізувати систему та перевірити її безпеку. Враховуючи те, що проведення експериментів на реальній мережі біткойн є складним і модифікувати її неможливо, було створено симулятор мережі біткойн з відкритим кодом [15]. Основна мета цього проекту полягає в тому, щоб отримати інформацію та відповідні показники на рівні безпеки мережі з симулятора мережі Bitcoin і в той же час вивчити, як ця інформація може допомогти зрозуміти цю складну систему.

Як ми бачимо нижче, ми подумали, що було б цікаво включити перелік конкретних цілей проекту для кращого відстеження виконаної роботи:

1. Проаналізуйте структуру та роботу мережі Bitcoin P2P.
2. Вивчити протоколи зв’язку та системи безпеки, які має ця мережа.
3. Вивчити уразливості мережі P2P, щоб зробити оцінку безпеки мережі.
4. Вивчити, явище надходження блоків та змоделювати цей процес для оцінки нашого проекту з точки зору системи масового обслуговування.
5. Написання проекту з урахуванням попередніх пунктів аналізу та специфікація основних частин.
6. Аналіз результатів, висновки що до переваг та недоліків порівняно з головними конкурентами ринку.

### **Практичне застосування блокчейн технології в автоматизованих платіжних системах**

Розуміння громіздкості та застарілості поточної найпопулярнішої системи здійснення грошових транзакцій у світі дуже важливо, і перевага блокчейна однозначна. Такою системою є банківські платежі, а емітентами – Visa, Mastercard, Maestro. Для початку потрібно зрозуміти чому транзакція в мережі ethereum відбувається швидше за банківські. І тому необхідно вивчити ланцюжок посередників та призначення кожної ланки економіки банківського сектора.

Уявімо деяких абстрактних покупця і мерчанта, для відновлення всього ланцюжка транзакції. І проведемо аналіз, починаючи від прикладання банківської картки до терміналу, до підтвердження транзакції. І так, жодна юридична особа не може функціонувати без рахунку у банку. ІП – може. Уявімо, що наш мерчант це якийсь ТОВ. Для функціонування він відкрив рахунок у банку B. Останній у свою чергу видав платіжний термінал. Далі наш абстрактний покупець здійснює покупку в ТОВ, після докладання картки до терміналу відбувається кілька програмних дій, у свою чергу ми розглянемо лише формування транзакції. Примітив виду транзакції наступний, якийсь користувач банку «А хоче зробити переказ у розмірі 100 умовних одиниць валюти до банку продавця «Б), кілька реквізитів та МСС код. Після цього термінал направляє пакет даних у банк А. У свою чергу, якщо транзакція міжнародна, повинен звернутися до міжнародної платіжної системи, наприклад - VISA. Безпосередньо МПС Visa повинна передати запит банку клієнта щоб отримати відповідь про наявність потрібної суми та статусу покупця. Якщо все задовольняє умови здійснення транзакції для обох сторін, банк «А відправляє відповідь, якою рухається по тому ж ланцюжку. Банк «А - МПС - Банк «Б - термінал. І на жаль після всього обсягу виконаної роботи, гроші мерчант отримає лише за три дні. Далі наведемо діаграму специфікації еквайрингу. Де Банк Б – еквайєр, Банк А – емітент картки.

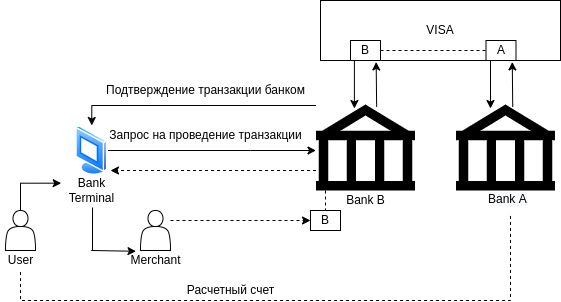


Рис 1.1: Специфікація транзакції у банківській мережі

### **Короткий зміст**

В даний час є кілька робіт, що стосуються мережі Bitcoin, деякі з них [10][11]. Вивчаючи вміст, ми бачимо, що мережа біткойн – це складний сценарій з декількома агентами, інтересами та дуже специфічним методом поширення інформації серед усіх користувачів. З точки зору зловмисників, існує тонкий баланс між наміром вкрасти та шахрайства в цій мережі та потребою безпеки, щоб підтримувати високий рівень довіри користувачів, а отже, і ціною біткойна.

Слід зазначити, що деякі інші криптовалюти використовують той самий вихідний код для мережі та P2P-зв’язку, що й біткойн. Цей факт допомагає розширити коло інтересів, оскільки будь-який недолік безпеки мережі в системі біткойн впливає на всі інші криптовалюти, які поділяють з нею код.

Ми виявили, що такі криптовалюти мають дуже схожий вихідний код для P2P-мережі з біткойнами:

* Litecoin <https://github.com/litecoin-project/litecoin>
* Dogecoin https://github.com/dogecoin/dogecoin
* Ethereum https://github.com/ethereum/go-ethereum
* Solana https://github.com/solana-labs/solana
* Cardano https://github.com/input-output-hk/cardano-node
* Polygon https://github.com/maticnetwork/matic-docs

Ми помітили, що деякі з цих криптовалют були розроблені шляхом створення гілки найновішої версії вихідного коду біткойна та застосування деяких змін для використання його як нової криптовалюти. Тим не менш, базова структура коду залишається вихідним кодом біткойна.

Хорошим прикладом цього є Dogecoin. Як ми бачимо на малюнку 1.2, вони використали вихідний код біткойнів як основу для застосування деяких змін і розробки своєї криптовалюти за потреби. Можливо, із застосованими змінами гіпотетична вразливість у коді біткойна не існувала б у новій криптовалюті. Тим не менш, важливо знайти будь-яку можливу проблему в мережі біткойн, щоб виправити її або перевірити, чи не зачеплені інші криптовалюти.

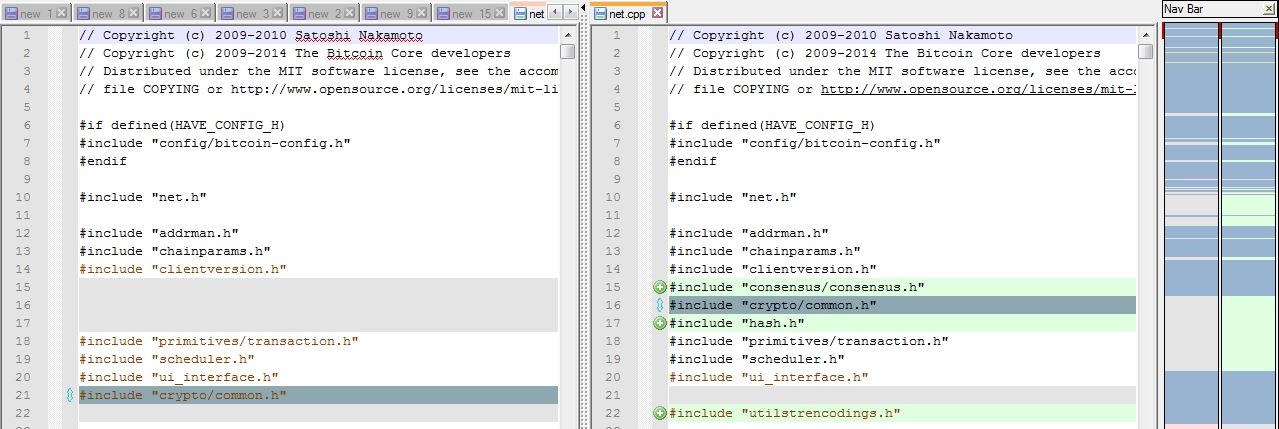


Рис 1.2: Порівняння вихідного коду net.cpp (ліворуч Dogecoin, праворуч Bitcoin)

Існує дуже мало повних рішень у сфері моделювання біткойнів. Найбільш релевантними роботами, знайденими в цій галузі, є: симулятор Python [29] і Simbit [12]. Симулятор Python розроблено спеціально для протоколу Bitcoin, але йому не вистачає можливості показувати детальні дані про симуляцію.

З іншого боку, Simbit — це загальний мережевий симулятор, який містить бібліотеку для імітації клієнта Bitcoin, але він все ще знаходиться в розробці. Зважаючи на це, ми вважаємо, що найкращим підходом для розробки цього проекту є використання Bitcoin Network Simulator [25], розробленого Віктором Морою, як головного фінального проекту для MISTIC. Симулятор був розроблений командою UAB [15] за підтримки Bitcoin Foundation. Протягом останніх місяців команда UAB розробляла різні модулі програмного забезпечення для моделювання, і з цим проектом ми додамо необхідну функціональність до симулятора, щоб видобувати відповідні дані, обробляти їх і показувати у зручний для людини спосіб.

### **Короткий зміст**

Зміст цього документа впорядковано за розділами таким чином:

* Розділ 1: Ознайомлення з проблемами та цілями роботи.
* Розділ 2: Основні поняття, які фігуруватимуть у роботі.
* Розділ 3: Формування боків та їх складова.
* Розділ 4: Топологія мережі, Gossip протокол, та її складові.
* Розділ 5: Доказ парадоксу Пуассона для біткоїну

**РОЗДІЛ 2**

# Основні поняття

У цій главі наведено деякі основні визначення, які використовуються в решті цього документа щодо біткоїна, а також відомі атаки на його мережу.

### **Баланс**

Уточнимо, що топологія нашої мережі – однорангова децентралізована або пірингова, оскільки приклад, що розглядається, це біткоїн. Отже ця архітектура мережі представляє відсутність центральної точки управління. Ця технологія породжує ряд неоднозначних моментів, таких як подвійне витрачання (double spend). Проблема подвійної трати була вирішена за допомогою ланцюжка цифрових підписів. Для розуміння описаного потрібно абстрагуватися від звичної моделі обміну фідуціарними грошима. У біткоїні відсутнє поле балансу або фізичні монети як такі. Існує один і лише один леджер, який є ланцюжком усіх транзакцій, на основі якого ми можемо математичним шляхом встановити баланс вузла. Тобто сказане раніше можна уявити, як ланцюжок передач прав на володіння якоюсь частиною загальної емісії валюти, а самі коїни перебувають у стані мономорфізму та існують лише безпосередньо для простоти людського сприйняття. Ніде не фігурує кінцеве поле балансів користувачів, це значення — поліморфне.

При створенні наступної транзакції якийсь сторонній вузол повинен перевірити леджер1, ланцюжок транзакцій, пов'язаних хешами попередніх транзакцій. Для того, щоб переконатися, що кількість входів ≥ кількості виходів. Фактично перевірка, чи вистачає відправнику коштів. Так, як у одного відправника може бути ряд вхідних транзакцій та ряд вихідних, виведемо таку формулу, яка позначить валідність суми переказу:

, де i = сума всіх вхідних переказів, o = сумою вихідних переказів

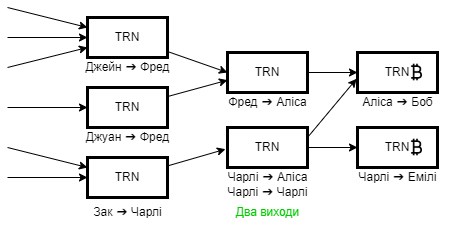


Рис 2.1: Спрощений ланцюжок транзакцій

Реалізувати блокчейн можна у різний спосіб, навіть якщо сфера застосування заздалегідь відома. Так наприклад, якщо програма, що розробляється на основі блокчейну є платіжною системою, то отримання балансу користувача вже можна реалізувати двома способами: детермінованим і недетермінованим2, не кажучи вже про те, з чого складатися блок, які обмеження блоку, які нагороди за майнінг і тд. Такий стан справ несе негативніший характер, оскільки безпека підсумкового продукту не визначатиметься загальноприйнятими стандартами, які пройшли відкритий та довготривалий аналіз. Щоб боротися з цим фактором прийнято дотримуватися стандартів де-факто, подібності біткоїну «Bitcoin» (який виступає як класична платіжна система) та ефіріуму «Ethereum» (як платформа смартконтрактів).

### **Транзакції та UTXO модель**

Механізм створення транзакцій без додаткового навантаження для підтвердження володінням коштів. Це означає, що при створенні переказу ми можемо розпоряджатися тільки повною частиною отриманих монет, згадуємо про роботу балансу. Оскільки фізичного поняття балансу немає, це лише функція, яка рекурсивно відновлює по ланцюжку транзакцій кількість зобов'язань перед об'єктом. Отже, витратити баланс можна тільки повністю. Часткова витрата призведе до розподілу гілки і не відповідності в ланцюжку блокчейну. Механізм переказу монет наступний, суб'єкт відправника має передати потрібну суму об'єкту одержувача, якщо ж суму меншу від балансу, суб'єкт повинен повернути залишок собі (див. рис 2.1).

Якщо ж реалізовано недетермінований протокол зі зберіганням станів у блоці, UTXO модель буде не потрібна. Оскільки зміни станів леджера фіксується у кожному блоці.

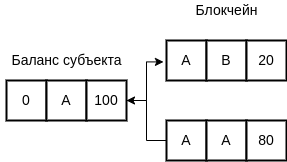
Вирішення проблеми подвійної витрати. Так як у мережі відсутній посередник або валідатор, якому за умовчанням довіряють усі ноди без винятків, виникає проблема зв'язкова з виробом транзакцій. Відправник А може відправити користувачеві Б транзакцію еквівалентно своєму балансу, потім відправити таку суму користувачу В. Мережа повинна мати певний алгоритм і ряд властивостей для запобігання таким типам атак. Для цього в кожній транзакції вказується посилання на попередню, тобто останню загальноприйняту на даний момент часу та власний унікальний ідентифікаційний номер, від якого в майбутньому відштовхуватимуться нові транзакції. У біткоїні такий Id генерується за допомогою хешу самої транзакції та оказії, таким чином формуючи орієнтований список та захищаючи транзакцію від змін у майбутньому.

Рис 2.2: Специфікація UTXO транзакції

Після чого відбувається процес надходження транзакції в mempol, місце, де вона вже потрапила в мережу, але ще не була записана не в один із блоків, отже, не була продекларована фактично. У такому стані транзакція знаходиться до додавання її в блок. Блок це один мегабайт транзакцій. Блоки потрібні для оптимізації роботи мережі, адже підтверджувати тисячу транзакцій одноразово менш ресурсозатратно та раціональніше. Формування блоку починається із створення леджера. Леджер це криптографічно доведене впорядкованих транзакцій для запобігання вразливості подвійної витрати. Після формування леджера, блок хешується за алгоритмом дерева меркла.

### **Конфіденційність**

Якщо в раніше розглянутих пунктах механізм роботи хоч і відрізнявся від консервативних, суті та цілей результату це не змінювало. Суть конфіденційності повністю перевернута з ніг на голову, на відміну від банківських платежів. Якщо в банківській програмі є якась точка довіри, тобто сам банк, який приховує нашу конфіденційну інформацію від інших учасників, хоч і повністю володіє їй. Принцип приховування в сучасних протоколах криптовалют зовсім інший. Вся інформація про суми та рахунки, їх транзакції публічно доступна кожному учаснику мережі. Мало того, вони повинні зберігати її в себе. Але зв'язати номер рахунку з його власником досить складно3. Це тягне за собою ряд деяких проблем, які порушують цю концепцію, і ряд рішень такі як міксера і створення нового гаманця для кожної транзакції. Добре, що створення гаманця обмежується генерацією пари ключів.

### **Асиметричне шифрування**

Для розуміння реалізації роботи мережі в умовно ворожому середовищі нам потрібно ознайомитися з основами асиметричного шифрування, ми не заглиблюватимемося в такі математичні аспекти як генерація випадкових байтів для пари ключів або самі алгоритми, які реалізують односторонні функції4 з доказом складно оборотності. Ми лише розглянемо протокол, на якому базується логіка шифрування та створення цифрових підписів.

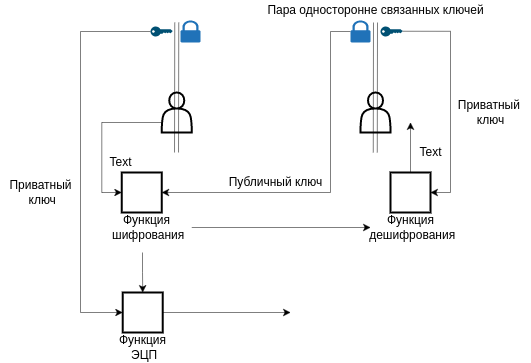


Рис 2.3: Специфікація взаємодії двох суб'єктів у явно ворожому середовищі

Як ми бачимо для шифрування даних, нам знадобиться два ключі, які пов'язані односторонньо. Є окремий розділ математики – модальна арифметика, що займається доказом односторонньої оборотності обох ключів. Що це означає. Що маючи приватний ключ ми з поліноміальною складністю отримаємо публічний ключ, а для отримання приватного ключа знаючи публічний нам знадобиться 2n, де n - це половина довжини ключа в байтовому еквіваленті. Якщо враховувати, що алгоритм дешифратора має зневажливо малий шанс передбачення в порівнянні з генерацією всіх значень, на отримання ключа підуть десятиліття, використовуючи всі актуальні потужності світу. Повідомлення, яке було зашифровано публічним ключем одержувача, може прочитати лише власник приватного ключа, тобто одержувач. Безумовно, є ряд атак спрямованих на аналіз шифротексту з метою злому, без отримання значення ключа. Але як правило стандартизовані алгоритми шифрування вже добре протестовані і максимально вразливі лише на перших ітераціях шифрування, тому до сьогодні найефективнішим методом дешифрації залишається терморектальний криптоаналіз5.

Так само функціонує ЕЦП – електронний цифровий підпис. У мережі вона потрібна для захисту від MITM6 атак. Працює схоже з шифруванням, але в метод подається приватний ключ відправника, після чого шифруються хешовані дані про транзакцію, будь-який користувач, хто має публічний ключ відправника може розшифрувати інформацію про транзакцію і переконатися, що тільки власник даного приватного ключ, тобто відправник, міг підписати транзакцію. Публічний ключ це загальнодоступна інформація, і з цього розшифрувати її може будь-який учасник мережі. Саме так працює симетричне шифрування. Як уже згадувалося вище, це необхідно для підтвердження відправника транзакції, щоб виключити посередників перехоплення повідомлень або відправок від третьої особи.

### **Proof-Of-Work**

У цьому підпункті описано реалізацію консенсусу блокчейн-системи біткоїну. Почнемо з того, що у whihe paper bitcion by Satosi Nakamoto відбувається посилання на роботи Адама Бека Hash-Cash, саме в ній вперше описаний алгоритм підтвердження роботи, як універсальна технологія захисту від спаму. Наприклад щоб перед підтвердженням блоку, вузлу довелося виконати певну обчислювальну роботу, імовірно йому було б не вигідно атакувати мережу засмічуючи тисячами транзакцій за секунду. Насправді технологія може використовуватися і в інших системах для спаму. Але в нашому випадку консенсус відіграватиме зовсім іншу роль. А саме — рівномірний розподіл емісії монет і протокол вибору лідера. Учасники намагатимуться знайти потрібний хеш та отримати право додати блок до блокчейну.

Для хешування використовується протокол sha256. Який приймає певний рядок всіх транзакцій у блоці та повертає хеш, що складається з 256 бінарних даних. Значення входять до діапазону масиву від 256 нулів до 256 одиниць. Якщо ж ми встановимо правило, за яким мережа, прийматиме хеш, у якому перші 10 значень – бінарні нулі, ми істотно збільшимо складність підбору такого хешу через зменшення діапазону відповідних хешів. Отже, обмеживши діапазон у 2 рази, приблизний час хешування збільшиться вдвоє. Ми можемо зменшувати діапазон до того часу, поки отримаємо потрібну нам швидкість видобутку хешу, за середній час6. Але згадаємо головну властивість хешування. Якщо подавати однакові блоки інформації, отримуватимемо однакові хеші. Тому для отримання різницю у виконанні роботи, а не генерації одного і того ж хешу всією мережою, було введено поняття nonce7. Тепер, подаючи кожен раз оказію до функції хешування, ми отримуватимемо різні хеші для одного блоку, і перебиратимемо їх, поки один з вузлів не отримає входження у вказаний діапазон.

Оскільки потужність мережі чи кількість обчислювачів, це динамічна властивість, що залежить від кількості учасників самої мережі, було передбачено алгоритм динамічного зміни складності мережі. Наприклад, сьогодні в мережі 10 учасників, обчислювальна потужність який дорівнює деякому значенню n. Тобто з цією потужністю підтвердження блоку відбувається в середньому раз на 10 хвилин. Якщо до мережі підключитися ще 10 майнерів і потужність зросте у 2n разів, блоки почнуть підтверджуватись у 2 рази швидше, що у свою чергу хоч і прискорить процес підтвердження, порушить концепцію емісії монет у мережі та призведе до подвоєного навантаження. Для запобігання цьому кожних 1440 блоків або ~1 місяць реального часу відбувається зміна актуальної складності. Тому, як правило, на початку місяця блоки підтверджуються раз на 15 хвилин, потім прискорюються до 7 хвилин.

Форк (Fork) – Розгалуження ланцюжка блоків. Існує можливість одночасного знаходження хешу для двох різних блоків з різним значенням оказії. У такій ситуації відбувається хардфорк і мережа поділяється на дві частини, кожна з яких продовжує якусь частину своєї гілки. Отже вирішується проблема дуже легко, хто наступний виграє у видобутку блоку, того гілка визнається головною через правило - найдовший ланцюжок перемагає. Тут працює принцип електронного голосування, що більше потужності майнить (голосує) за видобуток наступного блоку, тим більше вірогідність його підтвердження та долучення до мережі. Вузли, які програли, зобов'язані прийняти нову гілку блокчейну. Тому важливо очікувати більше 1 підтвердження від мережі для конфірма переказу.

### **Майнінг**

Майнінг дуже тісно пов'язаний із консенсусом proof-of-work. Оскільки в мережі немає центральної точки відмови, з якою за умовчанням погоджуються всі вузли, виникає задача під назвою «Візантійська угода». Проблема про прийняття рішення у свідомо ворожому середовищі. Це класична криптографічна задача, яка має кілька досить спірних рішень, але в біткоїні вона вирішується саме випадковим вибором вузла, блок якого визнає дійсним вся мережа. Учасники мережі, зокрема майнери. грають у криптографічну лотерею. Вони перевіряють блок на дійсність всіх транзакцій, порівнюючи його хеш з хешом будь-якого іншого вузла, якщо ж у блоці буде якась незначна зміна, хеш буде разюче відрізнятися і майнити далі буде просто безглуздо, тому що такий блок, навіть якщо вдасться знайти хеш, що задовольняє мережу. Блок, прийнятий не буде. Після перевірки форку, всі учасники намагаються за допомогою перебору nonce знайти підходящий хеш. Як тільки такий хеш буде знайдений, вузол отримає заохочення від сховища і право на встановлення та розповсюдження нового блоку, так як вгадування хешу досить ресурсозатратне обчислення, ми може припустити, що майнеру буде не вигідно обманювати мережу, так навіть якщо спробувати змінити якесь значення в блоці, інші вузли це помітять і не прийматимуть його блок. Нагороду за здобутий блок він отримує від сховища.

Сховище або coinbase, яке є якимось блокчейн користувачем, але не має приватного ключа. Це можна розглядати як виняток із правил, що формує нове правило лише для нього в контексті майнерів. Суть сховища зводиться до того, щоб воно виплачує зі своєї скарбниці гроші за виконану роботу майнерам. При цьому сховище може бути як самовідновлюваним, так і таким, що не відновлюється.

Якщо ми говоримо про сховища, що не відновлюються:

1. Майнер замайнув блок
2. Сховище платить гроші майнеру
3. Майнеру перераховується відсоток із транзакцій

Якщо ми говоримо про сховища, що самовідновлюються:

1. Майнер замайнив блок
2. У сховище перераховується відсоток із транзакцій
3. Сховище платить гроші майнеру

Якщо різним майнерам посилається одночасно транзакція, то у майнерів виникає конкуренція чи гонка з майнінгу. Шансів виграти більше у того, хто має великі потужності, але питання в іншому, як координуються майнери?

Адже теоретично майнери можуть майнити одні й самі значення, цим витрачаючи дарма час і енергію. Для такого випадку створюються пул-сервери, до яких підключається група майнерів, зацікавлених у вирішенні одного завдання та розподіл ресурсів між собою. Пул-сервера координують дію кожного окремого майнера таким чином, щоб вони не перетиналися в переборі однакових значень.

**РОЗДІЛ 3**

# Форування блоків

Почати розгляд блоку варто з самої мінімальної його версії, де він здатний залишатися і функціонувати як об'єкт в блокчейне. Так, для блоку досить тільки одного поля - хешу попереднього блоку. Ця урізана версія проста для розуміння, але також породжує теоретичну базу. Так, наприклад, давайте уявимо, що перед нами є певний блок В0, який представляє початок ланцюжка, в термінології це блок генезису. Суть його в тому, що він не має попереднього блоку і є єдиним прабатьком всіх наступних. Ви можете думати про блок B0 наступним чином:

B0 = <випадковий рядок>

Випадковий рядок в блоці генезису замінює хеш блоку і в теорії він може бути будь яким, навіть не випадковим, так як це ніяк не впливає на безпеку. Всі наступні блоки будуть генеруватися на основі наступної формули:

Bi = hash(Bi-1),

де hash()- криптографічна хеш-функція\*

Таким чином, виходить ланцюжок хешів, яку тільки із застереженнями можна назвати блоком. Щоб надати блоку «сенс», потрібно помістити дані (D) в блок для зберігання, але таким чином, щоб ці дані були зафіксовані в результаті хешування. Тепер блок ускладнився і замість одного значення (хешу) став зберігати два (дані і хеш), тим самим ставши масивом:

B0 = [<NULL>, <випадковий рядок>]

Bi  = [D, hash(D || Bi-1{2})]

У цьому прикладі була введена запис «||», що означає об'єднання рядків і запис «Bі-1{2}», що означає взяти другий елемент з масиву Bі-1 (індексація для масиву почнеться з одиниці).

Цей запис вже більше схожа на блокчейн, адже в ній зберігаються хоч якісь значення, нібито корисні користувачеві. Але може виникнути логічне питання: навіщо все це було зроблено? І відповідь - за безпеку (хоча на даному етапі проблем ще дуже багато). Суть полягає в тому, що більше неможливо змінити значення Bi{1} (або Bi-1{2}), не впливаючи на Bi{2}, інакше зміна буде виявлено, оскільки значення блоку не буде відповідати його хешу.

Ситуація для зловмисника буде плачевною в той момент, коли йому потрібно буде змінити значення блоку в середині або на початку ланцюжка, так як йому доведеться міняти всі наступні хеші, які надходять після зміни блоку. Але проблеми в цій схемі все ж залишаються.

Що робити, якщо грошові перекази зберігаються в блокчейні? Буде дуже прикро, якщо зловмиснику вдасться змінити ваше значення, що зберігається в блоці, на значення набагато більше, або згенерує новий блок, в якому він вказує вас як відправника.

Щоб вирішити цю проблему, необхідно ознайомитися з таким терміном, як цифровий підпис\*\*.

Цифровий підпис допомагає ідентифікувати творця блоку, тим самим усуваючи моменти, коли хтось може видати себе за іншу людину. Саме поняття цифрового підпису несе в собі ще два терміни - address (відкритий ключ) і wallet (закритий ключ).

Підпис слід наносити на хеш блоку, а не на його значення за двома ознаками. По-перше, хеш завжди має фіксовану довжину, відповідно, і швидкість підписання не буде змінюватися. По-друге, підпис також вплине на хеш-інформацію попереднього блоку, що повністю зв'яже підпис з попередніми блоками.

Блок ускладнився трьома елементами (значення, хеш, підпис):

B0 = [<NULL>, <випадковий рядок>,<NULL> ]

Bi = [D, hash(D || Bi-1{2}), sign(Bi{2})]

Тепер, якщо зловмисник хоче змінити значення Bi{1}, йому потрібно буде змінити значення Bi{2}, але якщо він змінить значення Bi{2} він не зможе змінити значення Bi{3} так що цей блок продовжує ідентифікуватися як до зміни. Або іншими словами, посередник зможе змінити блок, але тільки в тому випадку, якщо він поставить свій підпис на блоці і аналогічно змінить всі наступні блоки. Таким чином, зловмиснику доведеться міняти весь блокчейн, що звичайно є досить складним процесом, але можливим.

Навіть якщо ваш блок знаходиться на початку ланцюжка, а ланцюжок - мільйон блоків, сучасний комп'ютер зможе змінити значення в ланцюжку і потім генерувати всі наступні хеші з підписами зловмисника, тим самим стираючи всі реальні блоки. Генерація всіх блоків, які в даний момент існують в блокчейні біткоіни, займе не більше 10 хвилин на сучасному процесорі.

У цьому випадку на допомогу приходить доказ роботи\*\*\*. Це ще більше ускладнення блоку, до якого додається такий термін, як складність розрахунку. Ця складність заснована на математично складній задачі, подобі знаходження потрібного хешу. Звичайний комп'ютер може витрачати на такі завдання кілька годин, днів, а то й місяців і років, не кажучи вже про вміння людини знайти рішення своїми руками. Для регулювання складності була придумана задача обчислення не всього хешу, а тільки його частини. Припустимо, ви проходите через число і хешуєте його, вам потрібно знайти значення, що швидко псується, щоб воно починалося з чотирьох нульових байтів. Розмір хешу, а точніше кількість нульових байтів на початку називається ціллю. Якщо такий хеш знайдений, то проблема вирішена. Але тут однозначно є мінус, якщо хоча б одна людина знайде хешоване число, яке явно почнеться з великої кількості нулів, то доведеться змінити спосіб обчислення складної задачі, наприклад, перейти на одиниці, потім на двійки, трійки, четвірки і так далі. Це явно незручно, відповідно, був придуманий вдосконалений спосіб виконання цього завдання, що робити, якщо хешувати не просто збільшене число, а хешувати об'єднання збільшеного числа з поточним хешем блоку? Завжди буде отриманий випадковий результат і отримане інкрементне число не можна буде використовувати знову на іншому блоці. Пошук потрібного хешу ще називають Майнінг.

Блок тепер буде виглядати так:

B0 = [<NULL>, <випадковий рядок>,<NULL> ,<NULL> ]

Bi = [D, hash(D || Bi-1{2}), sign(Bi{2}), pow(Bi{2})],

, де pow повертає оказію x = Bi{4} в якому хеш(x || Bi{2}) <= TARGET.

Слід зазначити, що функція військовополонених призначена не тільки для підтвердження роботи, що сповільнило б генерацію блоків і розтягнуло б емісію монет в мережі. В основі мети доказу роботи лежить рішення двох математичних задач. Перший - це протокол відбору лідера, де лідер, а саме майнер, буде обраний гідним внести якусь інформацію в мережу. Другий - розподілене голосування, де вага голосу визначається комп'ютером учасника. У цьому випадку результат військовополоненого, число, яке доводить, що

1.Майнер використовував правильну ціль

2.Поточна складність майнінгу блоків

Також ряд неоднозначних питань виникає при об'єднанні учасників в пули і зіставленні майнера зі знайденим хешем. Тобто блок не містить ніякої інформації про те, хто замінував. Уявімо собі таку ситуацію, що ви самі не видобуваєте блок, а перекладаєте це завдання на групу інших людей, потужність комп'ютера яких в кілька разів перевищує ваш комп'ютер і за це ви платите певний відсоток від перерахованих коштів. Тоді одна людина з цієї групи нарешті знаходить нонс і кладе його в блок. Але в цій групі є нападник, він копіює нонс і пересилає його вам. Тепер ви отримали дві відповіді про знайдений хеш, кому довіряти? Можна сказати, хто приніс першим, той отримає винагороду, але можна зробити це і іншим, більш правильним способом. Нехай майнер сам створить блок, а ви відправите йому тільки транзакцію (а саме перші три поля блоку), потім майнер введе в блок свій хеш і підпис, а на своєму хеші виконає операцію військовополонених.

Блок ще більше ускладнюється:

B0 = [<NULL>,<NULL> , <випадковий рядок>,<NULL> ,<NULL> ,<NULL> ]

Bi = [[D, hash(D), sign(Bi{1}{2})], M, hash(M || Bi{1}{2}|| Bi-1{3}), signM(Bi{3}), pow(Bi{3})]

Коли майнер знайде оказію, число x = pow(Bi{3}), ніхто інший не зможе скопіювати це число і підставити його для свого результату, так як отримане значення не буде підходити під хеш іншого блоку, де буде вказано інший майнер. Але навіть незважаючи на це, необхідна транзакція надходить у блокчейн, зі значенням, хешем значення та хеш-підписом. Тобто все як годиться. Але і на цьому етапі є лазівка. Ви можете замінити знак(Bi{1}{2}) без наслідків і ніхто цього не помітить. Звичайно, питання може бути різним, для яких цілей існуючий підпис сторонньої особи захоче змінити на його підпис, при цьому підлаштовуючись під його хеш? Але на жаль, статися може все, що завгодно і тому його теж слід вважати вразливістю. У виправленні цієї вразливості може допомогти розкриття нутрощів магічного значення D.

І знову блок починає ускладнюватися:

B0 = [<NULL>, <NULL>, <випадковий рядок>,<NULL> ,<NULL> , <NULL>]

Bi = [[[S, R, V], хеш(Bi{1}{1}), знакS(Bi{1}{2})], M, hash(M || Бі{1}{2} || Bi1{3}), signM(Bi{3}), pow(Bi{3})]

При цьому S - відправник, R - приймач, V - значення (саме значення або кількість умовних відправлених одиниць). Також змінено вигляд знака(Bi{1}{2}) на signS(Bi{1}{2}), вказавши, чий це підпис. Тепер підпис не можна замінити, не зачепивши хеш. Також не можна змінювати творця підпису, адже потрібно буде змінити хеш-значення, що неминуче призводить знову до заміни підпису. Підпис майнера також проблематично змінити, адже його ім'я вказано в хеші. Доказ роботи також не вкрадуть, адже він прив'язаний до хешу. Здається, ніби ось вона, ідеальний захист! Але на жаль, немає, вразливості все ж є.

Пам'ятайте, що нікому не можна довіряти блокчейну? Власне, це і стосується шахтарів. Але як майнер може завдати шкоди? Адже була зроблена хеш-підпис, а хеш вказує на відправника, одержувача і значення, тобто нічого з цього змінити не можна, і якщо змінити його повністю, що аналогічно простому ігноруванню додавання транзакції в блок. Це все вірно, але є одна деталь, яка дуже дратує - угода ні до чого не прив'язана. Відповідно, майнер або хтось інший зможе скопіювати всю транзакцію і продублювати її знову, при цьому вона також буде дійсна, адже всі хеші і підписи правильні (адже вони дійсно реальні). Щоб позбутися від цієї вразливості, можна просто вказати хеш попереднього блоку в угоді.

Блок трохи ускладнився:

B0 = [<NULL>,<NULL> , <випадковий рядок>,<NULL> ,<NULL> ,<NULL>]

Bi = [[[S, R, V], хеш(Bi{1}{1} || Бі-1{3}), знакS(Bi{1}{2})], M, hash(M || Бі{1}{2} || Бі-1{3}), знакМ(Бі{3}), військовополонений(Бі{3})]

Тепер, навіть якщо майнер захоче продублювати транзакцію, вона буде перебувати під новим блоком, і хеш попереднього блоку більше не буде збігатися з хешем, який є в транзакції. І якщо майнер спробує залишити змінені значення, навіть незважаючи на невідповідність, то інші майнери просто відкинуть його блок як недійсний.

А тепер у нас блоки вишикувалися в ланцюжок, всі вони захищені від підміни, зміни, дублювання і одночасно від всіх користувачів. Але що робити, якщо майнерів кілька і у кожного з майнерів різний ланцюжок блоків? Як вони домовляться про загальний блокчейн? Pow - це різновид комп'ютерного голосування. Уявіть, що два або більше майнерів знайшли відповідний блок одночасно або майже одночасно, і поширили обидва блоки на мережу. У цей момент блокчейн розпаралелюється на кілька гілок, і перетворюється в дерево. Таким чином, UTXO перестає працювати і стає не зовсім зрозуміло, як перевіряти транзакції в такому деревному ланцюжку. Проблема вирішується за допомогою Майнінг. Зараз змагаються група вузлів, які взяли блок першого майнера, і група, яка взяла блок другого. Той, хто першим знайде блок, має право замінити і відкотити ланцюг протиборчої групи вузлів. Отже, чим більше комп'ютерна група, тим більше шансів, що вона першою знайде хеш. Тому, якщо більшість мережі продовжувала видобувати будь-який з ланцюгів, вони голосували за цю гілку блокчейну. Такі суперечливі ситуації називаються м'якою виделкою, коли блокчейн розділений на кілька гілок, але все ж є можливість вибрати тільки одну «правильну» ланцюжок.

А крім м'яких вилок, існують ще й так звані хард-форки. Тверді вилки, на відміну від м'яких, не з'являються самі по собі. Вони використовуються, коли необхідно змінити або поліпшити додаток, або створити новий блокчейн на основі існуючого. Це призводить до того, що нова гілка не буде «зливатися» зі старою, тим самим стаючи несумісною.

Також кожен блок повинен випускати монети в мережу. Таким чином, знаючи все це, необхідно змінити блок генезису:

B0 = [[[S, M, V], <NULL>, <NULL>], <NULL>, hash(B0{1}{1}), <NULL>, <NULL>,<NULL> ],

де S - сховище, M - добувач блоку генезису, V - винагорода блоку генезису. Як тільки блок генезису генерується, починає діяти сховище, яке має певний і обмежений запас ресурсів.

Начебто все вже сказано, але в блокчейні все ще є ряд тонкощів.

По-перше, якщо транзакція відправляється різним майнерам одночасно, то у майнерів є конкуренція або гонка на видобуток. Шанси на перемогу більші у того, хто має більше влади, але питання в іншому, як координуються шахтарі? Адже в теорії майнери можуть добувати одні й ті ж значення, тим самим витрачаючи час і сили. Для такого випадку створюється пул серверів, до якого підключається група майнерів, зацікавлених у вирішенні одного завдання і розподілі ресурсів між собою. Сервери пулу координують дії кожного окремого майнера, щоб вони не перетиналися в пошуку одних і тих же значень.

По-друге, транзакцій в блоці може бути кілька (і це найчастіше буває). В цьому випадку виникає ще одна вразливість, а саме дублювання угоди в одному блоці. Щоб цього не сталося, потрібно вставляти в кожну транзакцію випадкове число (r) і хешувати його разом з усією іншою інформацією.

По-третє, ви можете використовувати позначки часу (Time), які вкажуть час створення блоку, для його подальшого відстеження в історії та визначення поточної ємності блокчейну майнерами\*\*\*\*\*. Все це можливо тільки за умови грамотної синхронізації з сервером, який представляє час.

В результаті остання версія блоку буде виглядати так:

B0 = [[[[, S, M, V],<NULL> ,<NULL> ]],<NULL> ,<NULL> , hash(B0{1}{1}{1}),<NULL> ,<NULL> ,<NULL> ]

Bi = [[[[r, S, R, V], hash(Bi{1}{1}{1} || Bi-1{4}), signS(Bi{1}{1}{2})], [...], ... ], M, Time, hash(M || Time || (Bi{1}{1}{2} || Bi{1}{...}{2}) || Bi-1{4}), signM(Bi{4}), pow(Bi{4})]

\*Криптографічна хеш-функція — це одностороння функція, яка приймає рядок довільної довжини і повертає рядок фіксованої довжини. Властивість однобокості говорить про те, що з x [y = f(x)] легко отримати y, але важко отримати x з y [x = f-1(y)]. В якості криптографічних хеш-функцій рекомендується використовувати хеш-функції сімейства SHA-2 (sha224, sha256, sha384, sha512) або SHA-3 (keccak). [3, с.111] стверджує, що хешфукції «в сирому вигляді» мають недоліки, схожість подовження повідомлень і колізій. Позбутися від другого проблематично, через існування атаки на день народження, тим самим прийнято вибирати хеш-функції більшого порядку. Позбутися від першого недоліку, як приклад, можна, застосувавши до функції алгоритм HMAC. Але в блокчейні така атака буде марною, так як вимагає редагування вже існуючих даних і хешу даних, що тягне за собою необхідність редагування підпису, через подальші розбіжності між хешем і підписом.

\*\*Цифровий підпис є елементом асиметричної криптографії. В алгоритмах шифрування, таких як RSA, результатом є функція дешифрування 14, а перевірка підпису виконується за допомогою функції шифрування. Ідея полягає в тому, що функція дешифрування відома тільки творцеві закритого ключа, відповідно, тільки він здатний підтвердити свої дії за допомогою цієї властивості.

PoW (Proof-of-work) - доведення роботи, здійснюване шляхом вирішення складної математичної задачі (наприклад, знаходження шуканого хешу). Крім PoW, існують і інші методи доказування, такі як PoS (Proof-of-stake), PoA (Proof-of-authority), PoB (Proof-of-burn) і т.д.

Варто сказати, що якщо шахтарі зберуться в одну коаліцію і їх загальна потужність становить понад 50% від потужності всіх гірників, то можлива атака, при якій відбудеться досить довга м'яка вилка. Тобто накопичення блоків в різних ланцюжках буде відбуватися одночасно, тим самим розділивши загальний блокчейн на два табори. З цього явища можуть виступати спекулятивні можливості скасування угод за рахунок вибору необхідного ланцюжка майнерами.

Розрахувати поточну потужність майнерів можна за допомогою часових позначок. Якщо раніше видобуток займала в середньому 10 хвилин, а зараз 5, то можна стверджувати, що потужність видобутку зросла вдвічі. При цьому тенденція до зниження необхідного часу позначається несприятливим чином, так як почастішали випадки появи м'яких вилок. Через це блокчейни попередньо встановлюють функцію, яка регулює поточну складність блокчейна, підлаштовуючись під постійний час (наприклад, 10 хвилин).

**РОЗДІЛ 4**

# Топологія мережі, Gossip протокол та її складові

Чим більше ви вивчаєте мережу біткоїна, тим менше вона вам здається повністю децентралізованою. У цій мережі існують сервери, що вказують точний час, пул-сервери, які відповідають за координацію дій вузлів, а крім усього іншого, і самі вузли являють собою розподілений сервер, оскільки користувачі, які здійснюють транзакції, є клієнтами.

Суттю цього розділу є виявлення особливостей блокчейна з боку мережі та побудова зв'язків між її учасниками. Зокрема біткоїн це оверлейна пірингова мережа. Або не зовсім так. Давайте розберемо, оверлейна мережа - це загальний випадок логічної мережі, створюваної поверх іншої мережі. Вузли оверлейної мережі можуть бути пов'язані або фізичним з'єднанням, або логічним, для якого в основній мережі існують один або кілька відповідних маршрутів з фізичних з'єднань. А пірингова - оверлейна комп'ютерна мережа, заснована на рівноправності учасників. Часто в такій мережі відсутні виділені сервери, а кожен вузол (peer) як є клієнтом, так і виконує функції сервера. На відміну від архітектури клієнт-сервера, така організація дає змогу зберігати працездатність мережі за будь-якої кількості та будь-якого поєднання доступних вузлів. Учасниками мережі є всі вузли.

Так для початку варто сказати, що існує два основних види мережі - багаторангові та однорангові. Багаторангові (клієнт-серверні) мережі припускають існування двох об'єктів - клієнта і сервера, де клієнт здатний робити запит, а сервер видавати відповідь. Однорангові ж (peer-to-peer) мережі припускають рівноправність учасників у діях, або іншими словами, користувач у даній мережі є одночасно і клієнтом, і сервером (якщо виходити з термінології багаторангових мереж). Таких користувачів іменують вузлами. Так однорангові мережі є підмножиною децентралізованих мереж, які своєю чергою входять до множини розподілених.

Розподілених мережі включають в себе як однорангові з топологією всі до всіх, так і децентралізовані. Решта типів розподілених мереж, нас цікавити не будуть.

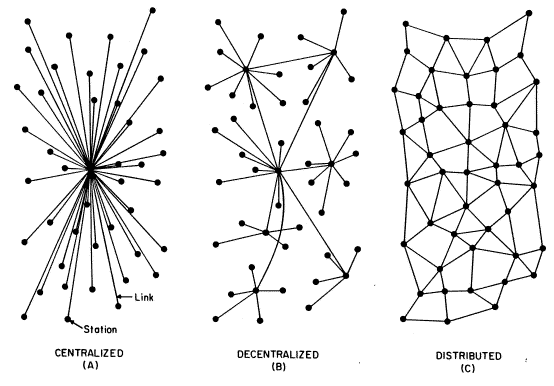


Рис 4.1: Графічне представлення комп'ютерних мереж

Але крім двох перерахованих вище типів мереж існує якийсь гібрид, який так і називається - гібридна мережа. Її основна суть полягає в тому, що існують, як і в багаторанговій мережі, два основних об'єкти - клієнт і сервер. Але особливість такої мережі полягає в тому, що сервером виступає однорангова мережа, або іншими словами мережа вузлів. На зображенні вище мережа А - багаторангова, В - гібридна, С - однорангова.

І це визначення гібридної мережі все ж може викликати питання. Наприклад, що якщо якась людина підніме сайт, зберігання даних якого буде розподілено між його серверами, тоді така мережа називатиметься гібридною? І чи можна буде назвати мережу гібридною, якщо існує тільки один вузол?

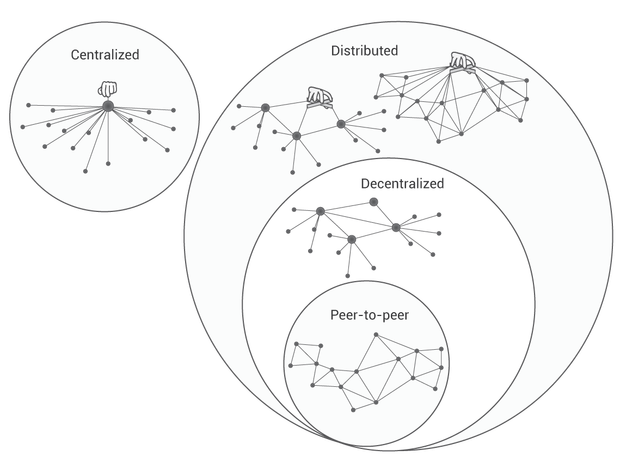


Рис 4.2: Графічне представлення ієрархії комп'ютерних мереж

Не відповівши на ці запитання, навряд-чи можна продовжувати міркування про цю тему. Відповідно, необхідно знайти в багаторангових і гібридних мережах принципову відмінність, і вона існує. Вузли в гібридних мережах, на відміну від серверів у багаторангових, не довіряють один одному, але й при цьому виконують спільну роботу. З цього випливає, що вузли не підпорядковуються якійсь конкретній людині або вузькому колу осіб і тим самим мають децентралізований характер. Але і з цього випливає те правило, що якщо існує тільки один вузол або вузли перебувають у руках однієї групи осіб, тоді така мережа втрачає властивості гібридної та переходить у фазу багаторангової.

У підсумку, існують такі зв'язки серед учасників блокчейну:

1. Клієнт -> [багаторангова] -> Вузол.
2. Вузол -> [однорангова] -> Вузол
3. Вузол -> [багаторангова] -> Пул-сервер
4. Вузол -> [багаторангова] -> Сервер часу

1. Клієнти надсилають вузлам запити щодо отримання балансу, блоків або занесення транзакції в блок.

2. Вузол зв'язується з іншими вузлами для зберігання загального ланцюжка. Вузол може посилати іншому вузлу запити про додавання нового блоку в блокчейн.

3. Вузол запитує в пул-сервера необхідний діапазон майнінгу.

4. Вузол запитує у сервера часу поточний стан часу.

Хоч кількість багаторангових зв'язків і переважає, проте важливість дій припадає на одноранговий зв'язок. Основне питання тут скоріше лежить у тому, чи погіршується відмовостійкість блокчейн-мережі в моменти появи багаторангового зв'язку? Візьмемо для початку приклад пул-сервера і уявімо зловмисника, який намагається його відключити від мережі методом DDoS атак. Сервер відключається, до нього доступ припинено для деяких вузлів. У підсумку, ця група вузлів може піти за двома сценаріями, або підключитися до іншого пул-сервера, або генерувати блоки окремо, виходячи з випадкового числа. У даному 17 випадку, мережа продовжує функціонувати за будь-якої обставини, хоч і з можливими застереженнями з приводу падіння продуктивності. Тепер же візьмемо сервер часу. Відмінність цього сервера від пулсервера полягає в тому, що його можна використовувати лише один раз перед запуском вузла. Інакше кажучи, запросити у сервера точний час, встановити цей час на локальній машині і запустити вузол. Таким чином, атаки на сервер завдаватимуть шкоди лише появі нових вузлів, тоді як вузли, що вже функціонують, продовжуватимуть роботу. Водночас варто враховувати, що серверів часу завжди кілька, і здійснити успішну DDoS атаку видається складним завданням. І навіть якщо всі сервери часу були відключені від мережі, в цей час продовжує функціонувати сам блокчейн, у вузлах яких зберігається поточний час. Відповідно, вузол, що формується, може запросити час не безпосередньо, у серверів часу, а побічно, через вузол, що вже працює в блокчейні. Якщо підсумувати все вищесказане, то відмовостійкість блокчейна залежатиме повністю від однорангової архітектури, водночас відключення від серверів може призвести лише до падіння продуктивності деякої групи майнерів, але ніяк не до зниження рівня відмовостійкості.

Гаразд, якщо блокчейн-мережа є за своєю суттю децентралізованою, тоді виникає логічне запитання: яким чином клієнти (і самі вузли) підключатимуться до інших вузлів? Звідки вони візьмуть початковий список необхідних адрес? Це насправді дуже часте питання в контексті децентралізованих мереж, оскільки кожна така мережа вирішує подібні проблеми різними способами. Найчастіше рішенням є використання сторонніх каналів зв'язку. Припустимо, створюється якийсь сервер, на якому буде розташовуватися список діючих вузлів. Виведення з ладу такого сервера не зруйнує саму мережу, але перекриє доступ до інформації про неї та її збільшення (що є проблемою лише під час початкового формування блокчейна). Розширення списку відбуватиметься самими ж вузлами (тобто вони вноситимуть свою адресу на сервер), оскільки зацікавлені в прибутку від майнінгу. І що частіше на різних серверах адреса майнера буде потрапляти, 18, то частіше до нього звертатимуться клієнти по допомогу в майнінгу (і самі майнери для додавання нових блоків).

Якщо в блокчейні враховувати додавання адрес, то кількість зв'язків у мережі збільшиться на два рядки:

1. Клієнт -> [багаторангова] -> Сервер адрес

2. Вузол -> [багаторангова] -> Сервер адрес

Також варто враховувати, що сам майнер здатний видавати список усіх інших майнерів, до яких він під'єднаний, тож здебільшого навіть не потрібно відвідувати сервери. Але може з'явитися одне питання: через конкуренцію майнерів у знаходженні потрібного хеша, чи не буде майнер намагатися видавати урізаний список адрес, щоб збільшити свої шанси на майнінг? Може і буде, але суть тут в іншому, якщо він не встигне замайнити блок і блокчейн оновиться за рахунок іншого майнер, то блок, який майнив "майнер-шахрай" виявиться невалідним. У підсумку, йому доведеться або створити свій ланцюг блоків, ігноруючи при цьому весь інший блокчейн (що досить ризиковано, оскільки це буде аналогічно hard fork'у, з подальшим залученням майнерів на свій блокчейн), або погодитися з іншими майнерами та прийняти той факт, що транзакції від клієнтів, які надсилалися йому, виявилися простроченими. З цього випадку клієнти зрозуміють, що якщо транзакції не потрапили в новий блок, то, найімовірніше, майнер видав не весь список діючих вузлів. Таким чином, клієнтам краще підключатися одразу до кількох вузлів і отримувати від них адреси інших майнерів, відсіваючи ті, що повторюються. І є ще третій спосіб, завдяки якому можна отримати список вузлів. Автори програми блокчейн-клієнта (або блокчейн-вузла) можуть за замовчуванням внести до програми (або до конфігураційного файлу) список довірених вузлів, які показали себе як такі, що працюють протягом тривалого часу. І в міру оновлення програми буде також оновлюватися цей список.

З боку збереження потужності і доступності блокчейн-мережі слід застосовувати відразу всі три способи знаходження адрес. Так, наприклад, якщо популярність цього блокчейна збільшилася на кілька порядків, то в теорії перший спосіб знаходження адрес (за допомогою сервера) можна не використовувати, але тоді буде виникати вразливість, коли довірені вузли (третього способу знаходження майнерів) скооперуються і не видаватимуть адреси всіх інших вузлів, тим самим знижуючи потужність блокчейн-мережі. Це позначиться на більш успішному поповненні балансу "майнерів-шахраїв", при цьому інші майнери навіть не отримають шансу формувати новий блок, виходячи з транзакцій. При цьому, якщо прибрати третій спосіб, то буде завжди виникати необхідність звернення до сервера на етапі запуску застосунку, що ініціює його запуск. Якщо прибрати перший і другий способи, то шанси занесення транзакції в новий блок будуть зниженими (за умови того, що існує кілька клієнтів з різним списком адрес). Якщо прибрати другий спосіб, то можливість отримувати список нових адрес буде покладено тільки на сервер. Ну а якщо прибрати перший і третій способи, то така блокчейн-мережа втрачатиме легкість використання, через необхідність ручного налаштування з'єднання.

Обмін даними в біткоїні відбувається за допомогою широкомовлення, а саме за допомогою Gosiip-протоколу. Це група протоколів в одноранговій комп'ютерній комунікації, в яких поширення інформації відбувається у спосіб, схожий на спосіб поширення епідемій, і зводиться до того, що кожен або деякі з вузлів можуть передавати оновлювані дані відомим цьому вузлу сусідам. Оскільки в мережі біткоїна, нам не відома точна кількість вузлів, це напевно найкращий варіант комунікації вузлів. Існує 2 раунди обміну:

1. обмін транзакціями

2. обмін блоком

Етап обміну транзакціями відбувається протягом ~10 хвилин, доки не буде знайдена оказія одним із майнерів. Після чого формується блок і відбувається етап обміну цим блоком.

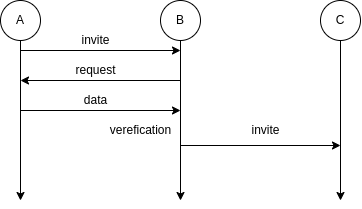


Рис 4.3: Процедура обміну блоком

Сам алгоритм розповсюдження блоку виглядає наступним чином і складається з 5ти етапів. Де на першому відбувається запрошення вузлом А до обміну з вузлом В. Запит за отримання блоку від вузла В. А відправляє свій блок. В валідує його і перевіряє на дійсність, тому що А міг його змінити або порушити структуру. Тільки після цього В ініціює запит обміну зі своїми вузлами-сусідами.

Цей алгоритм породжує низку вразливостей, пов'язаних зі швидкістю розповсюдження блоків

мережею. На кожному переході в широкомовній передачі повідомлення піддається затримці поширення. Затримка розповсюдження являє собою комбінацію часу передавання і локальної перевірки блоку або транзакції. Час передачі включає в себе оголошення у формі повідомлення запрошення, запит від сторони, що приймає, і доставку. Хоча повідомлення invite і data мають відносно невеликий розмір (61B у більшості випадків, тому що негайні повідомлення містять тільки один блок або транзакцію, про яку оголошують), повідомлення блоку може бути дуже великим - до 500 кБ на момент написання статті. Перш ніж блок буде передано сусідам вузла, його перевіряють. Перевірка блоку включає перевірку кожної транзакції в блоці. Перевірка транзакцій, своєю чергою, вимагає випадкового доступу до даних, що зберігаються на дисках. Нехай ti,j - це різниця в часі між першим оголошенням джерела в мережі та часом, коли вузол j отримує елемент i. Якщо вузол o є джерелом елемента даних i, тобто або творцем блоку, або вузлом, який створив транзакцію, то ti,o = 0. Часи ti,j, у які вузли дізнаються про існування вузли дізнаються про існування елемента даних, слідують подвійному експоненціальному закону. Подібно до рандомізованого поширення чуток , поширення елемента даних можна поділити на дві фази: початкова фаза експоненціального фаза зростання, в якій більшість вузлів, що отримують повідомлення inv повідомлення, будуть запитувати відповідний елемент даних, оскільки в них іще не мають його, і фаза експоненціального зменшення, в якій більшість вузлів, що отримують оголошення, вже мають відповідний елемент даних. вузлів у мережі. Наша реалізація поводиться так само, як звичайний вузол, з одним застереженням: вона не передає повідомлення, транзакції або блоки. повідомлення, транзакції або блоки. Він відстежує, як транзакції та блоки поширюються мережею, прослуховуючи оголошення про їхню доступність у формі inv-повідомлень. повідомлення. Як тільки вимірювальний вузол отримує повідомлення inv, що містить посилання на блок, ми знаємо, що вузол, який відправив повідомлення, отримав і перевірив блок. Вимірювальний вузол зібрав інформацію про час із блокчейна висотою 180'000 для 10'000 блоків. Тимчасова інформація містить хеш блоку, IP того, хто анонсує.

Для вимірювання затримки розповсюдження ми реалізували протокол мережі біткоїна і підключилися до великої вибірки IP вузлів і локальну мітку часу, коли оголошення було отримано. Оцінка для ti,j дається шляхом віднімання мітки часу першого оголошення блоку з усіх оголошень для цього елемента даних. На малюнку показано нормалізовану гістограму tb,j для всіх блоків b у вимірюваному інтервалі. Нормалізація дає нам змогу використовувати це як апроксимацію для функції щільнісної ймовірності швидкості, з якою вузли дізнаються про блок. Зверніть увагу: що ми не робимо відмінностей між розмірами блоків і замість цього агрегуємо за всіма блоками. Медіанний час до отримання блоку становить 6,5 секунд, тоді як середнє значення становить 12,6 секунди. Довгий хвіст розподілу означає, що навіть після 40 секунд все ще є 5% вузлів, які ще не отримали блок.

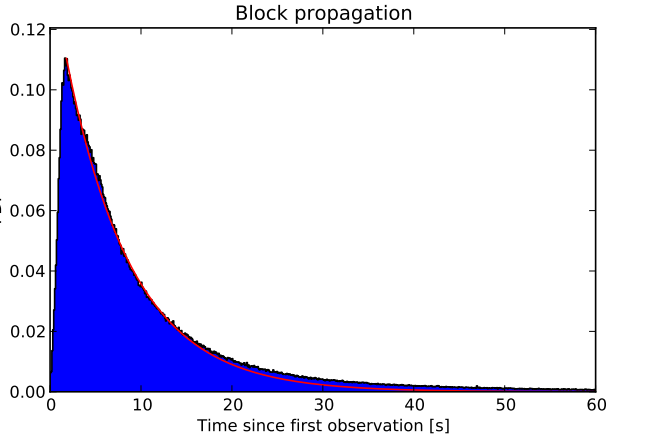


Рис 4.4: Специфікація поширення блоку у мережі біткоїну

Такий тип уразливості можуть використовувати як для передчасного отримання підтвердження блоку, так і для створення повторних транзакцій. Другий тип атаки більш небезпечний. Зловмисник намагається надіслати транзакцію жертві, де щільність вузлів набагато менша. Паралельно створюючи другу транзакцію, яку відправляє в місці великого скупчення потужностей, де з більшою часткою ймовірності буде підтверджено наступний блок. Тим самим коли другу транзакцію прийняло більше відсотків мережі, першу, відправлену жертві, мережа ігнорує, і вона не отримує підтвердження.

Вирішення проблеми розповсюдження блоків базується на оптимізації валідації блоку або транзакції. Розберемо пункт 4 - verification. Він ділиться на 3 підпункти:

1. Dificulty check

2. Transactions

3. Done

Dificulty check - перевірка накопиченої складності, яка відображає обчислювальні потужності в розрахунку хешів блоків цього ланцюжка.

Решта кроків, як-от перевірка транзакції на валідність, є більш тимчасово витратними, тому пропонується надсилати ініціювальне повідомлення на обмін після пункту Dificulty check.

Існує другий підхід обміну даними. Щойно до нас надходить inv на обмін, разом із запитом на отримання даних, розсилається запрошення всім сусіднім вузлам, будується черга пріоритизації відправки транзакції, після перевірки складності об'єкт розсилається згідно з побудованою чергою. Так само запропоновано і третій підхід, що пропонує змінити топологію мережі на star sub networ. Цей метод веде до повного порушення головної концепції криптовалют і децентралізованих мереж загалом. Через це є, хоч і досить продуктивною, так як кожна транзакція буде проходити через 1 вузол, що сприяє практично моментальному обміну, але досить сильно зменшує відмовостійкість мережі.

**РОЗДІЛ 5**

# Доказ парадоксу Пуассона для біткоїну

1. **Гіпотеза** – Більшість користувачів очікують підтвердження блоку більше ніж 10 хвилин, не дивлячись на те, що в середньому воно становить 10 хвилин.
2. **Технічне завдання на розроблення моделі**

Блок, моделюється як процес Пуассона зі швидкістю λ. Тут середній час між блоками, , встановлюється на основі цільової складності в операції майнінгу PoW; для Bitcoin = 10 хвилин. Пуассонівський процес — це процес, у якому нові події (або надходження) відбуваються через випадкові проміжки часу після експоненціального розподілу. Крім того, інтервали між будь-якими двома подіями не залежать один від одного та статистично ідентичні. Нагадаємо, що експоненціальна випадкова величина X з параметром λ має розподіл

Також нагадаємо, що пуассонівська випадкова величина Y з параметром λ має розподіл

У пуассонівському процесі кількість подій на інтервалі довжиною T є пуассонівською випадковою величиною з параметром λT. Крім того, кількість подій у непересічних проміжках часу не залежить. Якщо ми розглядаємо малі інтервали (), то в інтервалі з ймовірністю λT є одна подія і жодної іншої. Таким чином, процес Пуассона можна імітувати шляхом «гри в орлянку» де послідовність (незалежних) підкидань монети має дуже малу вірогідність випаду «орла». Тепер ми розуміємо, чому процес майнингу має таку властивість.

Уявіть, що кількість майнерів у системі та загальна обчислювальна потужність у їхньому розпорядженні є постійною протягом певного періоду часу. Припустімо, що кожен з комп’ютерів

майнерів безперервно перебирає хеші, і це єдине обчислення, яке вони виконують. Тоді можна сказати, що загальна кількість хешів, які обчислюються (еквівалентно, загальна кількість оказій, які перебираються) за одиницю часу є приблизно постійною. Скажімо, щосекунди обчислюється мільярд хешів.

Крім того, припустимо, що рівень складності хеш-головоломки для кандидатного блоку дуже висока. Наприклад, скажимо, що перші тридцять п’ять бітів хешу мають бути нульовими, щоб блок був дійсним. Імовірність того, що певний nonce відповідатиме цьому критерію, становить 2−35 ≈ 3 × 10−11. Таким чином, ймовірність того, що хеш-головоломка буде розв’язана будь-яким майнером за певну секунду часу, становить 0,03 , невелике число (ми припустили, що кожну секунду обчислюється мільярд хешів). Те, чи знайдено хеш підтвердження роботи в певну секунду, не впливає на те, чи буде він знайдений у наступну секунду. Причина цього в тому, що хеш-функція, по суті, є випадковим оракулом; хеш-значення для різних вхідних даних не залежать одне від одного. Таким чином, процес видобутку добре моделюється як процес Пуассона. При моделюванні процесу видобутку як пуассонівського процесу ми зосереджуємося лише на моменті створення нових дійсних блоків.

Модель процесу Пуассона актуальна незалежно від кількості майнерів, їхньої індивідуальної обчислювальної потужності, того, чи працюють різні майнери над одним або різними блоками, і коли різні користувачі отримують щойно видобуті блоки. Параметр λ процесу майнінгу, який називається швидкістю майнінгу, дорівнює середній кількості блоків, видобутих за одиницю часу. У біткоїнах λ дорівнює 1/(600 с), тобто один блок кожні 600 секунд (десять хвилин).

З варіаціями загальної обчислювальної потужності припущення про фіксовану швидкість майнінгу спростовуються. Насправді загальна обчислювальна потужність не змінюється раптово. Таким чином, протягом невеликого періоду часу швидкість є приблизно постійною. Регулювання параметра складності через регулярні проміжки часу допомагає підтримувати швидкість видобутку на одному рівні.

1. **Математичний опис моделі**

Чи є надходження блоків Пуассонівським процесом. Спробуємо довести це. Відразу відмітимо що незалежно від того, чи моделюється глобальний хешрейт H(t). емпірично або параметрично, для кожного входження блоку моделі в цьому моделюванні хешрейт визначається до вибірка випадкового процесу Xi(t).

* Для моделей із детермінованим налаштуванням складності, складність коригується в детермінований час моменту yn, які не відповідають випадковим моменти надходжень блоків та швидкість надходження блоків λ(t) не залежить від надходжень блоку в попередньому сегменти. Якщо немає затримки, то на кожному інтервалі модель є неоднорідним процесом Пуассона.
* Для моделей із випадковим налаштуванням складності, складність регулюється у випадкові моменти часу, після кожного 2016 сегменту блокчену використовуючи рівняння. Якщо немає затримки розповсюдження, тоді кожен сегмент процесу є неоднорідним пуассонівським зі швидкістю, заданою як λ(t) = H(t)/Di. Оскільки швидкість надходження блоку λ(t) залежить від першого та останнього надходження в попередній сегмент блокчейну, процес не Пуассонівський протягом послідовного періоду часу сегменту.
* Якщо присутня затримка поширення, то прибуття блоку процес навіть не є неоднорідним процесом Пуассона на одному сегменті. У наступному розділі ми порівняємо їх моделювання до даних про позначку часу з блокчейну біткоїна.

Тому ми знехтуємо динамічною складністью і моделюватиме тільки сегмент з 2016 блоків,

приблизно 1 місяць реального часу, або 20160 хвилин. Також припустимо, що розповсюдження блоків в мережі є моментальним та не викликає жодних затримок.

Точковий процес N є процесом Пуассона на , якщо він має наступні дві властивості.

1. Випадкова кількість точок N([a, b)) точкового процесу N, розташованих в обмеженому інтервалі [a, b) ⊂ R, є пуассонівською випадковою величиною із середнім Λ([a, b)), де Λ є невід’ємною мірою Радона.
2. Кількість точок точкового процесу N, розташованих на k інтервалах [a1, b1), . . . , [ak, bk) утворюють k незалежних пуассонівських випадкових величин із середніми Λ([a1, b1)), . . . ,Λ([ak, bk)).

Відтепер будемо записувати N([a, b)) як N(a, b) і Λ([a, b)) = Λ [a, b) для зручності. Перша властивість передбачає що

і , а друга властивість – це Основна причина придатності процесу точки Пуассона і зазвичай це основа статистичних тестів, які вимірюють адекватність моделей Пуассона. Розподіл Пуассона N(a, b) означає, що його дисперсія Var[N(a, b)] = Λ(a, b), факт який також використовується як статистичний тест. Міра Λ відома як міра інтенсивності або середнє значення міри процесу точки Пуассона. Припустимо, що a існує така функція λ(t), що

Тоді λ(t) визначена як функція швидкості. Якщо λ(t) є сталою λ > 0, то процес називається однорідним точковим процесом Пуассона. Інакше процес називають неоднорідним або **неоднорідним точковим процесом Пуассона**. Якщо обмежити нашу увагу інтервалом невід’ємних чисел [0, ∞), міра інтенсивності задається формулою

Для пуассонівського процесу N з мірою інтенсивності Λ ймовірність існування n точок в інтервалі [a, b) дорівнює

Час надходження та час між надходженнями: розглянемо точковий процес {X(i)}i≥1, визначений на невід’ємних дійсних числах із майже напевно кінцевою кількістю точок у будь-якому обмеженому інтервалі. Тоді ми можемо інтерпретувати точки процесу як часи добування нових блоків та розмістити їх у порядку зростання, X1 ≤ X2 ≤ . . .. Тоді відстані між сусідніми точками дорівнюють Ti := Xi − Xi−1 для i = 2, 3, . . . і T1 = X1. Випадкові величини Ti відомі як час очікування або час між надходженнями. Для однорідного процесу Пуассона зі швидкістю λ відповідні часи між надходженнями є незалежними та однаково розподіленими експоненціальними випадковими величинами із середнім значенням 1/λ

Де властивість експоненціального розподілу без пам’яті було використано. Це не стосується неоднорідного точкового процесу Пуассона з інтенсивністю λ(t), де перший час між надходженнями T1 = X1 має розподіл

За першого часу очікування T1 = t1 умовний розподіл другого часу очікування T2 є

і так далі для k ≥ 2

Можна показати, що k-й час надходження Xk має розподіл

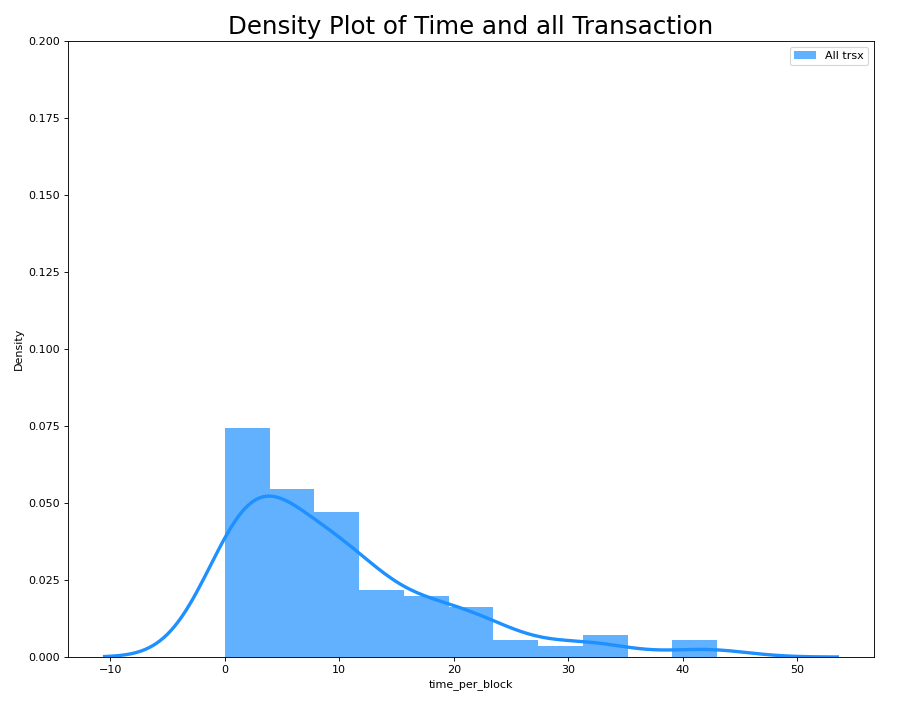
З щільністю

Умова на n точок пуассонівського процесу, що існує в деякому обмеженому інтервалі [0, t]. Ми називаємо ці точки умовним часом надходження блоку. Якщо процес Пуассона є однорідним, то умовні часи надходження рівномірно і незалежно розподілені, утворюючи n рівномірних випадкових величин на [0, t]. Ця різниця між часом очікування Ti та умовним часом надходження Ui відіграє роль у тесті Пуассона.

Для неоднорідного пуассонівського процесу кожна точка Ui незалежно розподілена на інтервалі [0, t] із розподілом

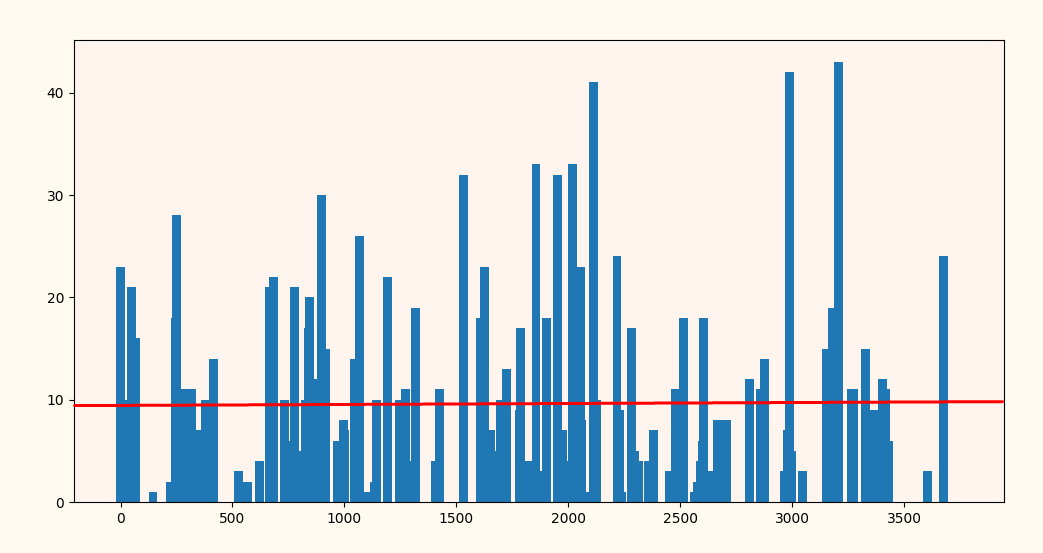
,

Якщо розподіл кожного Ui відомий і оборотний, то кожен Ui може бути перетворений в рівномірну випадкову величину на [0, 1], що призводить до n незалежних рівномірних випадкових величин. Іншими словами, Λ(t) перетворює процес Пуассона на однорідний процес Пуассона з густиною один на відрізку дійсних чисел. Отже, статистичні методи для неоднорідних процесів Пуассона часто передбачають перетворення даних перед виконанням аналізу.

1. ****Аналіз результатів**

Підсумовуючи, швидкість майнингу біткойна відповідає розподілу Пуассона, тобто більшість блоків знайдено протягом 10-хвилинного інтервалу. Але оскільки це випадковий процес, завжди є деякі повільні блоки, на пошук яких майнерам потрібно більше або менше часу. Тим не менш, середній час підтвердження транзакції має становити близько 10 хвилин, як показано вище.

Парадокс — це на перший погляд абсурдне твердження, яке при дослідженні виявляється обґрунтованим, але нелогічним. Отже, наприклад, припустимо, що більшість людей очікують довший час підтвердження транзакції, незважаючи на те, що середній показник становить 10 хвилин.

Було використано невелику вибірку з 140 блоків BTC від 759149 до 759289, щоб дослідити цю гіпотезу. Спочатку ми запускаємо перевірку наших даних.

Гістограма нижче показує, що час підтвердження транзакції насправді нагадує розподіл Пуассона. Більшість транзакцій, 78%, займає від 5 до 20 хвилин. А середній час знаходження блоку становить 9,9 хвилин.

Нижче наведено графік хвилин між блоками, зображений помаранчевою лінією. Сума транзакцій на блок показана синіми стовпцями. Ми виявили, щосередня кількість транзакцій на блок становить 1805.

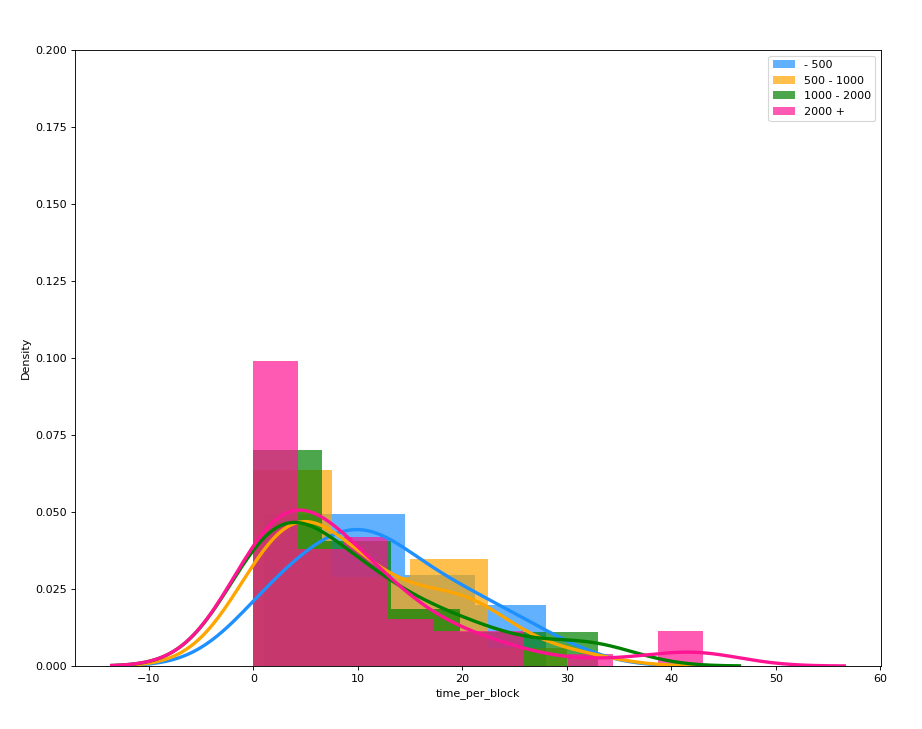


Синя лінія позначує час очікування підтвердження блоку. Отже можна побачити кореляцію, між очікуванням блоку та великою кількістю транзакцій у наступному блоці. Тобто, якщо підтвердження прийшло швидко, як о 10.47, то наступній блок буде порожнім і навіть якщо 2000 транзакцій підтвердились упродовж 5ти хвилин, блоки з ними були на-пів порожніми, що свідчить про те, що менша кількість користувачів отримала швидке підтвердження, ніж користувачі о 13.21 де 40 хвилин очікувало майже 11 тис транзакцій.

Таким чином, не обов'язково, що всі швидкі блоки малі за розміром. Наприклад, об 10:55 один блок займає 5 хвилин майнінгу, але підтверджує лише 200 транзакцій. У той час як об 15:28 пошук блоку займає 0 хвилин і містить 2982 транзакції. Таким чином, розмір блоку залежить від кількості транзакцій, які очікують підтвердження в mempool (перевантаження), а не від швидкості надходження блоку.

З рештою, це дослідження даних не доводить існування парадоксу Пуассона біткойна. Однак доказ насправді прихований у початковій гістограмі, тобто в довгому хвості.

Дійсно, більшість людей у нашій вибірці чекають на підтвердження довше 10 хвилин, хоча середнє значення становить 9,9 хвилин. Це тому, що розподіл Пуассона має довгий хвіст вправо. Тобто, існує більше можливостей для виявлення блоку між 10-40 хвилинами, оскільки часовий інтервал у чотири рази більший, ніж інтервал між 0-10 хвилинами. Дивіться нижче.



|  |  |
| --- | --- |
| Час підтвердження блоку | Відсоток вибірки |
| 0 – 10 хвилин | 40 % |
| 10 – 40 хвилин | 60 % |

Підсумовуючи, у 2/5 нашої вибірки трансакція підтверджується менш ніж за 10 хвилин, тоді як у 3/5 трансакція підтверджується більше ніж за 10 хвилин. Це парадокс Пуассона біткойна.

Є ще кілька причин, чому транзакції BTC можуть проходити повільніше або швидше, ніж 10 хвилин. Хоча це не стосується парадоксу Пуассона біткойна.

По-перше, майнери збирають і хешують транзакції з мемпулу в заголовок блоку, перш ніж шукати наступне значення nonce. Це може створити відставання. Наприклад, нові транзакції, не підібрані майнерами, залишаються бездіяльними в мемпулі, доки не буде видобуто попередній блок, а також час, необхідний для виявлення нового блоку(ів). Транзакції також можуть застрягти, якщо комісія занадто низька.

Однак, якщо транзакція зависає, її можна збільшити, змінивши комісію за допомогою Child-Pays-for-Parent (CPFP) або Replace-by-Fee (RBF). Зміна вартості комісії за транзакцію підвищує її пріоритет, тому майнери з більшою ймовірністю включать її у свій наступний блок. Крім того, деякі майнінгові пули можуть додавати транзакції до своїх власних блоків, що прискорює час транзакцій.

1. **Висновок**

У цьому розділі роботі ми представили набір моделей для точкового процесу епох майнінгу блоків у блокчейні біткойн і протестували його за допомогою моделювання та даних, доступних із самого блокчейну. Основними труднощами в цьому є:

1. Невідомий глобальний хеш-рейт, який керує швидкістю виявлення блоків;
2. Історично відоме, але випадкове значення складності майнінгу.

Крім того, дані про час надходження блоків є вичерпними, але не повністю надійними. Ми постулюємо модель точкового процесу, в якій процес **надходження блоку поводиться як неоднорідний процес Пуассона** в періоди між змінами складності зі швидкістю, пропорційною відношенню швидкості хешування до складності, але яка залежить від себе, коли враховуються зміни складності. Для заданої глобальної швидкості хешування час, коли змінюється складність (і, отже, змінюється швидкість надходження блоку), визначається шляхом вибірки процесу.

Тим не менш, глобальна швидкість хешування стабільно швидко зростає, а механізм зворотного зв’язку щодо труднощів має достатню затримку, тому швидкість надходження блоків була приблизно на 11,5% більшою за базову швидкість 6 блоків на годину. Це означає, що в цілому пропускна здатність транзакцій і загальний дохід майнера від бонусів вищі, ніж у базовому варіанті.

Крім того, час, коли винагорода за майнінг блоків зменшується вдвічі, і час, коли всі біткойни будуть створені, настане раніше, ніж це було б, якби блоки майнилися зі швидкістю шість на годину. Окрім надання моделі для процесу надходження блоків, ми вивели зв’язок між частотою надходження блоків і швидкістю експоненціального збільшення швидкості хешування. Ми також надали практичне наближення, яке демонструє граничну поведінку незалежно від початкових умов і збурень процесу надходження блоку, і перевірили це наближення за допомогою моделювання та вимірювань з блокчейну.

Також **було підтверджено гіпотезу стосовно парадоксу Пуассона** в очікуванні підтвердження транзакції. Через те що розподіл має довгий хвіст вправо більшість користувачів очікують підтвердження блоку довше за середній час, у той час коли менша кількість користувачів має пріоритет у швидкості.