УДК 004.021

Киричек Г.Г.
Запорізький національний технічний університет

Рудьковський О.Р.
Запорізький національний технічний університет

Тимошенко В.С.
Запорізький національний технічний університет

ДЕЦЕНТРАЛІЗОВАНА СИСТЕМА ПІДТРИМКИ ОБЧИСЛЕНЬ

У роботі проведено аналіз наявних децентралізованих систем, здатних підтримувати роботу сервісних додатків, наведено етапи реалізації системи з підтримки обчислень у децентралізованих мережах, а також етапи розробки протоколу передачі й обробки інформації для децентралізованих мереж різного призначення.

Ключові слова: *децентралізація, обчислення, мережа, алгоритм, шифрування, ізольоване середовище.*

Постановка проблеми. Розвиток комп'ютерної інженерії та інформаційних технологій досяг рівня, що дає змогу майже будь-який предмет підключити до мережі. Тому питання доступності інформації, підвищення швидкодії її отримання та використання, підвищення ефективності та якості проведення обчислень є дуже актуальними натепер. У сучасному світі складно уявити виконання багатьох задач без використання сервісів. З розвитком технологій все гостріше постають питання подальшого розвитку мереж та забезпечення їх безпеки. Вирішенням усіх вище зазначених проблем є децентралізація мереж та додатків.

Аналіз останніх досліджень і публікацій. Останнім часом набули популярності різні децентралізовані додатки та мережі. Найбільш популярними є Manet, ZigBee, Ethereum, EOS. Manet та ZigBee є фізичними децентралізованими мережами, а Ethereum та EOS – оверлейні мережі з підтримкою роботи децентралізованих додатків [1; 2]. Оскільки більша частина децентралізованих мереж оверлейні, вони використовують уже наявні протоколи для передачі інформації, ними зазвичай є транспортні протоколи: Transmission Control Protocol (TCP) та User Datagram Protocol (UDP). Різні децентралізовані мережі використовують різні протоколи. Деякі підтримують роботу, використовуючи обидва протоколи. Майже всі мережі не використовують власні алгоритми шифрування та хешування, а використовують вже наявні методи, надійність яких доведена вченими [3–7]. Такий підхід підвищує надійність готового продукту та зменшує вірогідність помилки у разі розробки нових методів. Тому для реалізації мереж використовують наявні бібліотеки, методи та функції. З вищезазначеного зрозуміло, що є багато технологій децентралізованих мереж та сервісних додатків різного призначення. Кожен з них використовує власні методи та алгоритми у реалізації специфічних завдань [1; 2]. Виходячи з вимог конкретної мережі або додатка, обираються алгоритми. Але майже всі рішення не здатні задовольнити потреби нових сервісів, що постійно реалізуються в мережі. Тому в цій роботі запропоновано вирішення цих проблем шляхом реалізації децентралізованої системи підтримки роботи сервісних додатків.

Наявні децентралізовані мережі, які підтримують роботу сервісних додатків, мають власні мови програмування. Найпопулярнішою є Solidity, яка є інтерпретованою, об'єктно орієнтованою, розроблена для криптовалюти Ethereum та підтримує функції, змінні, цикли та умовні оператори. До традиційних мов відносяться Java, C++, C#, Python та ін [8; 9]. Для кожної з них є безліч бібліотек та готових рішень. Їхня швидкість роботи вища, ніж у Solidity, вони мають додаткові утиліти для перевірки та оптимізації сирцевого коду. Крім цього, всі ці мови мають доступ до заліза і мережі, дають змогу створювати складні додатки. З огляду на це зрозуміло, що слід використовувати одну з традиційних мов – C++, Java, C# або Python.

Постановка завдання. Мета роботи – проведення досліджень та реалізація методу підтримки обчислень у децентралізованих мережах шляхом розробки протоколу децентралізованої мережі з підтримкою роботи сервісних додатків. Об'єктом дослідження є процес підтримки обчис-

лень у децентралізованих мережах. Предметом дослідження є моделі, методи та інструментальні засоби підтримки ефективної взаємодії кінцевих пристроїв у децентралізованому середовищі. Основним завданням роботи є проведення досліджень та на їх основі розробка моделей, методів та засобів підтримки ефективної взаємодії кінцевих пристроїв у децентралізованій мережі з можливістю передавати, зберігати, обробляти дані як із використанням сервісних додатків, так і без них. Під час розробки програмного забезпечення (ПЗ) необхідно вирішити такі задачі: реалізувати децентралізовану систему з підтримки роботи сервісів; провести тестування децентралізованої системи на працездатність. Система реалізується як протокол передачі та обробки інформації для децентралізованих мереж різного призначення. Для роботи з мережею користувач використовує стороннє ПЗ або розроблене власноруч на основі чіткого алгоритму роботи окремих частин мережі.

Виклад основного матеріалу. Одним з основних завдань є реалізація децентралізованої мережі, яка підтримує: децентралізований обмін даними; децентралізоване зберігання інформації та запуск сервісів у децентралізованій мережі. За результатами попереднього аналізу та проведених досліджень виявлена необхідність у децентралізованій мережі, яка здатна підтримувати роботу сервісних додатків. У результаті першого етапу її реалізації розроблено та наведено модель децентралізованої системи, загальними елементами якої є фізичні елементи мережі та логічні зв'язки між ними (рис. 1).

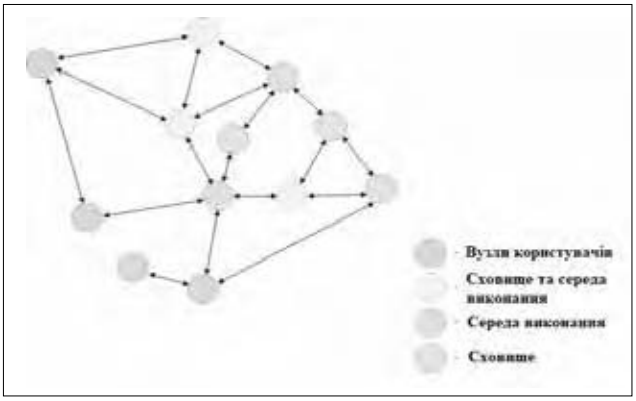

Рис. 1. Модель децентралізованої системи

Як видно на рисунку, в реальній системі є декілька типів вузлів: користувацький, сховище, середа виконання та вузли, що поєднують одночасно декілька функцій. Окрім того, всі вузли системи є транзитними, тому будь-хто може підключитись до системи через ці вузли. Система у кожен момент часу є централізованою, децентралізованою або розподіленою; інформація зберігається на всіх пристроях або на окремих, а додатки виконуються на будь-якому вузлі системи або на спеціальних вузлах. За великої кількості пристроїв система частіше всього є розподіленою, а під час роботи додатків вона є частково децентралізованою, тому що є вірогідність існування одного або декількох центральних елементів, які керують додатками. За наявності великої кількості пристроїв у мережах різних типів система є повністю розподіленою під час пересилання та зберігання даних.

Реалізований протокол описує оверлейну децентралізовану мережу, яка здатна передавати, зберігати та обробляти дані користувачів. Передача виконується децентралізовано без прямого встановлення зв'язку між клієнтами. Дані зберігаються розподілено та публічно. Обробка даних виконується за рахунок підтримки роботи сервісних додатків у рамках мережі. Протокол описує взаємодію модулів мережі. Кожен модуль відповідає за певні функції в рамках мережі. Такий підхід дає змогу швидко змінювати складові частини додатка, які реалізує клієнт мережі, не впливаючи на роботу інших модулів. Усього протокол описує дев'ять модулів: мережі; повідомлень; хешування; шифрування; команд; пам'яті; маршрутизації; серіалізації; додатків та розширень (рис. 2). Додаток, що реалізує протокол, може використовувати не всі модулі, а лише необхідні для його роботи. Кожен модуль виконує одну дію і це зменшує кількість помилок та допомагає тестувати окремі частини мережі. Далі детально розглянемо всі модулі.

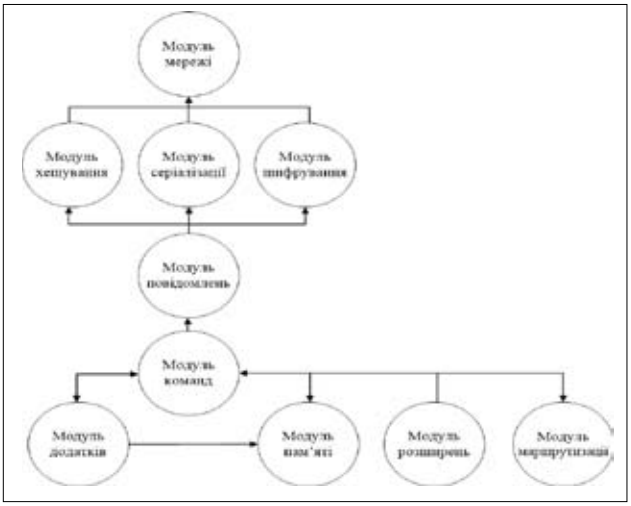

Рис. 2. Взаємозв'язок модулів протоколу

Додатково протокол описує структуру пакетів у рамках мережі, алгоритми та формат адрес і

даних, що передаються. Додаткове ПЗ, необхідне для роботи мережі, обирається програмістом, який інтегрує мережу у свій додаток. Протокол описує взаємодію двох типів пристроїв у мережі: кінцеві та транзитні. Транзитним є будь-який вузол, що не є відправником або отримувачем повідомлення. Один фізичний пристрій може одночасно бути як транзитним вузлом, так і кінцевим, оскільки кожен додаток, що використовує мережу є незалежною одиницею та має власну адресу. Сервісні додатки та дані виконуються на транзитних вузлах. Транзитні вузли передають дані та не мають доступу до вмісту даних. Будь-який транзитний вузол може отримати будь-яку інформацію, що зберігається у мережі. Не всі рівні моделі використовуються під час роботи протоколу. Деякі додатки можуть обмежуватись певним необхідним рівнем, наприклад транзитні вузли можуть реалізовувати рівні до модуля повідомлень включно, оскільки інші рівні в їхньому разі не використовуються. Додатково кожен з модулів є незалежним і може використовуватись з будь-якого іншого модуля, наприклад модуль команд може шифрувати та дешифрувати дані напряму без звертання до модуля повідомлень. Дані у мережі передаються у текстовому вигляді. Перед передачею даних вони серіалізуються. Як алгоритм серіалізації обрано формат даних JSON, що широко використовується у сучасних додатках. Для передачі інформації використовуються пакети. Пакет має два рівні: транзитний та клієнтський. Транзитний рівень є відносно публічним і будь-хто може отримати до нього доступ. Клієнтський рівень призначений для отримувача пакета і передається у зашифрованому вигляді.

Модуль мережі реалізує метод, який відповідає за встановлення з'єднання між клієнтами. Він керує з'єднанням, передачею та отриманням інформації за допомогою протоколу TCP, а також будує певний абстрактний шар між децентралізованою мережею та наявною фізичною мережею. За допомогою цього методу створюється оверлейна мережа, яка працює з будь-яким фізичним з'єднанням та протоколом. Модуль мережі є найнижчим модулем. Він отримує дані, які необхідно надіслати, та передає їх необхідному вузлу і працює лише з вузлами, що підключені до нього.

Модуль команд призначено для створення додатків, які використовують мережу. Завдяки командам розширюються можливості мережі та додаються нові функції. Вони не є сервісами у мережі, а є їх заміною у певних випадках. Типовим використанням команди є передача спеціалізованого повідомлення, запит на підключення, додавання або отримання даних. За допомогою команд підтримується працездатність мережі. Додаток може добавити власні команди до наявних. У разі отримання команди додаток перевіряє, чи може він її виконати. Якщо виконати команду не можна, додаток її відкидає. Інакше вона виконується. Командою є спеціальний об'єкт. Дані команди зберігаються у цьому об'єкті. Команда є надбудовою над повідомленням та не відрізняється від повідомлення зовні. Вона відрізняється тим, що може виконувати будь-які дії, відсилати відповіді та реагувати на них. Інакше кажучи, метод дає змогу створювати двонаправлену взаємодію. Приклад методу наведено у лістингу 1. Метод отримує повідомлення та надсилає відповідь. Команда приймає рядок. Відповідь містить вхідний рядок великими літерами, поточну дату та час.

Лістинг 1 – Приклад реалізації:

```
import command
import simple_package
import simple_response_package
import datetime
class SimpleCommand(command.Command):
      def handle(self, package: simple_
package.SimplePackage):
       response = simple_response_package.
SimpleResponsePackage()
        response.date = datetime.datetime.
now()
             response.message = package.
message.capitalize()
         self.send_response(response).
```

Модуль повідомлень формує, передає та отримує нові повідомлення (пакети). Він працює з модулями мережі, шифрування, хешування та серіалізації. Цей модуль розбиває дані на сегменти, формує пакети та передає їх модулю мережі для передачі. Також він отримує дані від модуля мережі. Після отримання даних з мережі він десеріалізує їх та зчитує транзитний рівень. Якщо додаток є вузлом, якому відправлено дані, то він виконує необхідні дії для обробки пакета (рис. 3). Якщо вузол є транзитним, то він формує новий ланцюг та надсилає наступному вузлу. Якщо поточний вузол не є кінцевим або транзитним, то пакет видаляється.

Модуль шифрування виконує такі дії: шифрує дані; дешифрує дані; генерує спільний ключ. Він працює з такими алгоритмами шифрування: Elliptic Curve (ECC), Elliptic Curve Diffie-Hellman (ECDH) та AES [5–7,10].

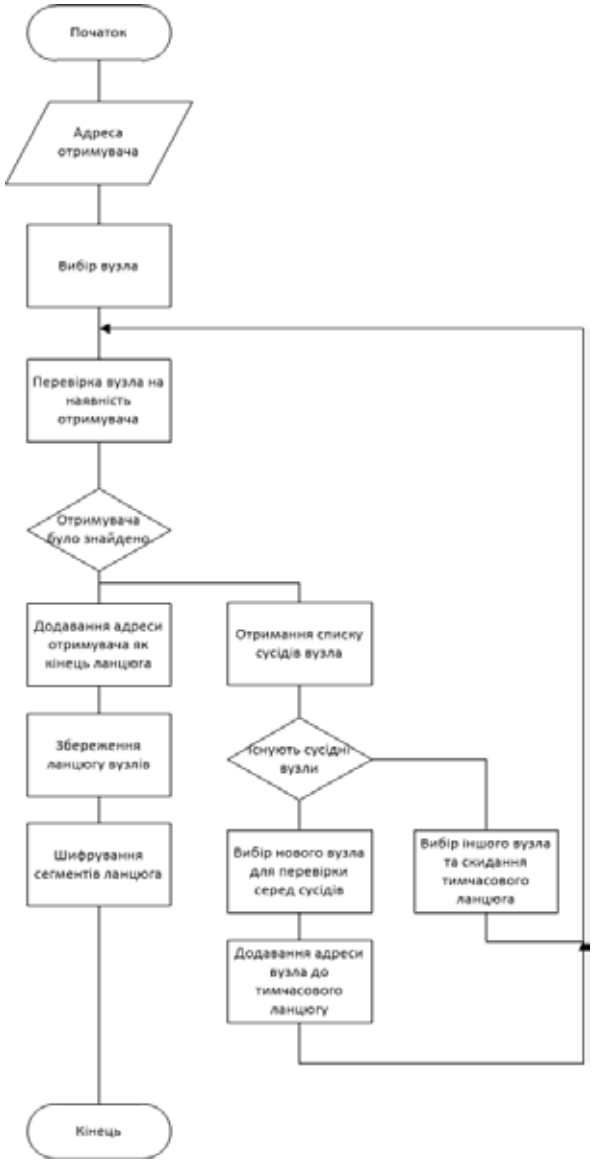

Рис. 3. Алгоритм побудови ланцюга вузлів для передачі інформації

Модуль хешування виконує дві операції: хешує дані та перевіряє хеш. Модуль серіалізації (лістинг 2) перетворює структури даних на рядки у форматі JSON. Він десеріалізує дані з рядка в об'єкти мови програмування. Оскільки додаток розробляє будь-хто, він може містити нестандартні повідомлення та структури даних. Тому вирішено передавати дані без прив'язки до мов програмування, а додати ідентифікатори.

Лістинг 2 – Модуль серіалізації:

```
import sys, os
import json
from serialization import Encoder
sys.path.insert(0,    os.path.dirname(os.
getcwd()))
class Serializer(object):
      __instance = None
      @staticmethod
      def get_instance():
          if Serializer.__instance is None:
                    Serializer.__instance =
Serializer()
           return Serializer.__instance
      def __init__(self):
          self.types = {
              # messages types
          }
          self.encoder = Encoder.Encoder()
      def serialize(self, obj):
        return json.dumps(obj, default=self.
encoder)
      def deserialize(self, message):
                return  json.loads(message,
default=self.encoder)
      def get_type(self, obj):
             return obj.__class__.__bases__
[0].__name__
      def add_type(self, obj, tag):
          obj_type = self.get_type(obj)
          self.types[obj_type] = tag.
```

Модуль розширень додає нові типи даних, що підлягають серіалізації та команди. Без цього клієнти можуть не знати, як правильно перетворити серіалізовану послідовність на об'єкти. Модуль розширень зв'язує додаткові ідентифікатори з типами даних, яким вони відповідають.

Лістинг 3 – Модуль розширень:

```
from serialization import Serializer
class Extensions(object):
     __instance = None
     @staticmethod
     def get_instance():
         if Extensions.__instance is None:
             Extensions.__instance =
Extensions()
         return Extensions.__instance
     def __init__(self):
         self.serializer = Serializer.
Serializer()
     def add_type(self, obj, tag):
         self.serializer.add_type(obj, tag).
```

Модуль пам'яті виконує роль розподіленої бази даних. Він відповідає за зберігання даних з мережі у вигляді розподіленої хеш-таблиці. Додаток може обмежувати обсяг даних, які він зберігає. Модуль маршрутизації – один з найважливіших у мережі. Саме він відповідає за побудову маршруту від відправника до отримувача. За основу маршрутизації в роботі обрано алгоритм цибулевої маршрутизації з однією відмінністю – шифрується не увесь пакет, а лише ланцюг адрес. Такий підхід дає змогу безпечно та неочікувано передавати інформацію, оскільки маршрут передачі заздалегідь визначено клієнтом, а сам маршрут надійно зашифровано. Модуль додатків є най-

вищим модулем у мережі і відповідає за роботу сервісних додатків. Його функції: створення запиту на виконання додатка; запуск додатка; комунікація між додатком та вузлом; комунікація між додатком та клієнтом; завершення роботи додатка. Оскільки запуск сторонніх додатків є небезпечною процедурою, додатки виконуються у спеціальному ізольованому просторі. Модель системи під час виконання додатка зображена на рисунку 4. Додатки працюють за децентралізованою схемою. Додаток не має доступу до мережі безпосередньо, оскільки в ізольованому середовище він заблокований.

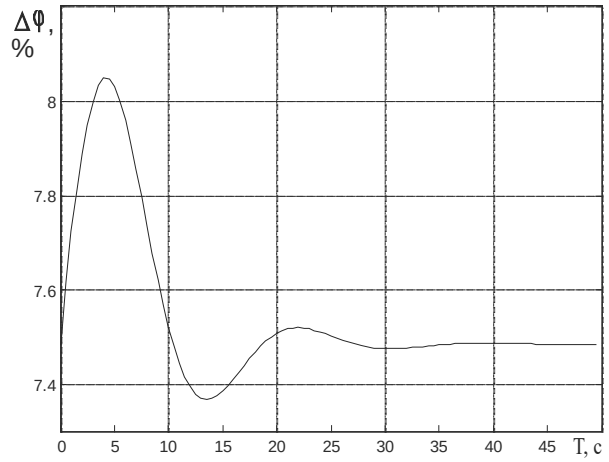

Рис. 4. Модель системи з віртуальним каналом

Для виконання запиту до мережі додаток має довести право на володіння цим ресурсом. Тому під час публікації додатка необхідно вказати, до яких ресурсів він мусить мати доступ. Вузол, що публікує додаток, перевіряє володіння ресурсом, зчитуючи попередньо задану послідовність символів з попередньо вказаного файлу. Якщо файл прочитати не вдалося або послідовність інша заявка на публікацію відхиляється. У разі успішного підтвердження додаток публікується разом з переліком підтверджених ресурсів. Для того щоб зробити запит додаток надсилає запит вузлу, на якому працює. Вузол перевіряє, чи є ресурс підтвердженим. Якщо підтверджено – запит додається у чергу запитів до ресурсів. Кожну секунду вузол може виконати обмежену кількість запитів, тому запити виконуються у режимі FIFO. Після виконання запиту вузол надсилає його додатку.

Висновки. У цій роботі вдосконалено процес роботи сервісних додатків у децентралізованих системах, створено модель децентралізованої мережі, а також розроблено протокол системи підтримки обчислень у децентралізованих мережах. Авторами реалізовано методи взаємодії частин системи та компонентів додатків. Під час подальших досліджень планується вдосконалення розроблених методів та доопрацьовування протоколу з метою підтримки більшої кількості мережних та сервісних функцій для збільшення кількості сценаріїв використання.

Список літератури:

1. Mobile Ad hoc Networking (MANET). URL: http://tools.ietf.org/html/rfc2501 (дата звернення: 10.10.18).
2. A Next-Generation Smart Contract and Decentralized Application Platform. URL: https://github.com/ethereum/wiki/wiki/White-Paper (дата звернення: 10.10.18).
3. Ахметов Б.С., Корченко А.Г., Сиденко В.П. Прикладная криптология: методы шифрования. Алматы: КазНИТУ им. К.И. Сатпаева, 2015. 496 с.
4. Eastlake D., Hansen T. US Secure Hash Algorithms. 2006. URL: https://tools.ietf.org/html/rfc4634 (дата звернення: 10.10.18).
5. Ship M., Yiqun L. Cryptanalysis of Twofish (II). URL: https://www.schneier.com/twofish-analysis-shiho.pdf (дата звернення: 10.10.18).
6. Rivest R., Shamir A., Ademan L. Cryptographic communications system and method. URL: https://patentimages.storage.googleapis.com/49/43/9c/ b155bf231090f6/US4405829.pdf (дата звернення: 10.10.18).
7. Rescorla E. Diffie-Hellman Key Agreement Method. URL: https://tools.ietf.org/html/rfc2631 (дата звернення: 10.10.18).
8. Хайнеман Д., Поллис Г., Селков С. Алгоритмы. Справочник с примерами на C, C++, Java и Python. 2017. М: Альфа-книга. 434 с.
9. Быков А.Ю. Решение задач на языках программирования Си и Си++. 2017. М: МГТУ им. Н. Э. Баумана. 248 с.
10. Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. 2007. URL: https://nvlpubs.nist.gov/nistpubs/ Legacy/SP/nistspecialpublication800-56ar.pdf (дата звернення: 10.10.18).

ДЕЦЕНТРАЛИЗОВАННАЯ СИСТЕМА ПОДДЕРЖКИ ВЫЧИСЛЕНИЙ

В работе проведен анализ существующих децентрализованных систем, способных поддерживать работу сервисных приложений, приведены этапы реализации системы по поддержке вычислений в децентрализованных сетях, а также этапы разработки протокола передачи и обработки информации для децентрализованных сетей различного назначения.

Ключевые слова: *децентрализация, вычисления, сеть, алгоритм, шифрование, изолированная среда.*

DECENTRALIZED SYSTEM OF THE SUPPORT COMPUTATIONS

The work analyzes the existing decentralized systems capable to support the work of service applications, presents the stages of implementation the system of supporting the computing in decentralized networks, as well as stages of development transmission protocol and processing of information for decentralized networks of different purposes.

Key words: *decentralization, computation, network, algorithm, encryption, isolated environment.*