КОМП'ЮТЕРНІ НАУКИ

УДК 004.75
DOI https://doi.org/10.32782/2521-6643-2026-1-71.5

Гуменюк А. О., здобувач вищої освіти
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
ORCID: 0009-0007-3202-8154

Отрох С. І, доктор технічних наук, професор,
професор кафедри цифрових технологій в енергетиці
Національного технічного університет України «Київський
політехнічний інститут імені Ігоря Сікорського»
ORCID: 0000-0001-9008-0902

ПІДВИЩЕННЯ ДОСТУПНОСТІ ДАНИХ У ПІРИНГОВИХ СИСТЕМАХ З АДРЕСАЦІЮ ВМІСТУ ЗА РАХУНОК ПРОАКТИВНОЇ РЕПЛІКАЦІЇ

Сучасні децентралізовані системи, зокрема IPFS, здебільшого використовують підхід «pull-on-demand», за якого реплікація виникає лише як побічний ефект від вилучення даних. Це призводить до низької доступності у невеликих мережах, де кількість активних вузлів замала для формування достатньої кількості копій.

У статті розглянуто підхід до підвищення доступності даних у піринговых системах з адресацією вмісту шляхом проактивної реплікації та детермінованого розміщення блоків.

Розроблено та проаналізовано механізм, що забезпечує передбачувану доступність даних без суттєвого збільшення координації між вузлами та зберігає сумісність із DHT. Сформовано вимоги до протоколу розміщення даних у піринговых мережах, розроблено модель проактивної реплікації, узгоджену з розподілом блоків у просторі ключів, а також оцінено здатність запропонованого підходу до зменшення навантаження на окремі вузли та забезпечення вищої доступності даних.

Запропоновано підхід, який інтегрує маршрутизацію за допомогою DHT з механізмом розміщення, що базується на ідентифікаторах блоків, які визначають їх позицію у просторі ключів. Впроваджено дворівневу модель закріплень: жорсткі закріплення гарантують постійне зберігання блоків, тоді як м'які створюються автоматично під час реплікації та мають обмежений термін дії, який продовжується при повторному доступі; збирач сміття інтерпретує комбінацію закріплень, щоб вирішити, які блоки можна безпечно видалити, зберігаючи лише релевантні та часто використовувані дані.

Експериментально показано, що запропонований підхід дозволяє зменшити навантаження на окремі вузли в мережі та підвищити стійкість системи до відмов, що робить його перспективним для використання у малих приватних піринговых мережах із підвищеними вимогами до доступності.

Ключові слова: *пірингові мережі, адресація вмісту, DHT, проактивна реплікація.*

Humeniuk A. O., Otrokh S. I. Improving Data Availability in Peer-to-Peer Content-Addressed Systems through Proactive Replication

Modern decentralized systems, including IPFS, predominantly employ a "pull-on-demand" approach, in which replication occurs only as a side effect of data retrieval. This leads to low availability in small networks where the number of active nodes is insufficient to naturally form an adequate number of copies.

The paper examines an approach to improving data availability in peer-to-peer content-addressable systems through proactive replication and deterministic block placement.

A mechanism is developed and analyzed that ensures predictable data availability without substantially increasing coordination among nodes and while preserving compatibility with DHTs. The requirements for a data-placement protocol in peer-to-peer networks are formulated, a proactive replication model, aligned with block distribution in the keyspace is proposed; and the capability of the approach to reduce load on individual nodes and achieve higher data availability is evaluated.

Solution that integrates DHT-based routing with a placement mechanism driven by block identifiers that determine their position in the keyspace is proposed. Two-tier pinning model is introduced: hard pins guarantee persistent storage of blocks, whereas soft pins are created automatically during replication and have a limited lifetime that is extended upon repeated access.

A garbage collector interprets combinations of pins to determine which blocks may be safely removed, retaining only relevant and frequently used data.

Experimental results demonstrate that the proposed approach decreases the load on individual nodes and increases the system's resilience to failures, making it a promising option for small private peer-to-peer networks with stringent availability requirements.

Key words: *peer-to-peer networks; content addressing; DHT; proactive replication.*

Постановка проблеми. У міру збільшення обсягу даних, потреба в розподілених системах та механізмах, що вони пропонують стає все більш нагальною. Традиційні централізовані сховища даних швидко досягають ліміту пропускної здатності та ємності. У свою чергу, розподілені – дозволяють без особливих труднощів масштабувати ці параметри.

Звісно, децентралізованість системи додає цілу низку нових проблем, вирішення яких є необхідним для їх ефективної та коректної роботи. Основним питанням є забезпечення передбачуваної доступності. Хоча сучасні системи і намагаються його вирішити, проте здебільшого зосереджуються на моделі «pull on demand» (вилучення при потребі), тобто в першу чергу виявленні провайдерів, а не розміщенні контенту, що спричиняє реплікацію даних лише як побічний ефект при їх вилученні. Такий підхід розкриває свій потенціал лише при масштабованому розгортанні, де частий доступ до даних природно призводить до їх реплікації. У контексті малих приватних мереж пасивна реплікація призводить до малої кількості провайдерів, що своєю чергою призводить до концентрації даних на окремих вузлах та використанню сторонніх централізованих сервісів та блокчейну для покращення доступності даних.

Аналіз останніх досліджень та публікацій. Пірингові системи з адресацією вмісту, організовують пошук і передавання даних за допомогою розподіленої хеш-таблиці та унікальних ідентифікаторів вмісту. За відсутності проактивної реплікації, дані фактично зберігається на одному або невеликій кількості вузлів, доки інші учасники мережі не звернуться за ними. Вихід такого вузла з мережі безпосередньо збільшує ймовірність втрати доступу до даних.

Дослідження, наведене в роботі [1], показує, що навіть відкрита мережа IPFS характеризується дуже обмеженим рівнем автоматичного дублювання: лише 2,7 % об'єктів мають понад п'ять копій на різних вузлах. Вбудований механізм ґрунтується переважно на ручному закріпленні (pinning), коли оператор вузла свідомо позначає дані як такі, що необхідно зберігати. Більш систематичним рішення є IPFS Cluster – доповнення до інфраструктури IPFS, що групує та координує декілька вузлів для синхронізованого закріплення даних. Варто зазначити, хоча це в більшості й усуває питання єдиної точки відмови, проте не є рішенням на рівні протоколу.

Інші системи, безпосередньо, реалізують механізми активної реплікації шляхом копії даних на певні вузли вже під час початкового завантаження. Наприклад, Swarm копіює кожен фрагмент файлу до сусідніх вузлів. Водночас такий підхід потребує окремого механізму координації, здатного гарантувати, що при виході одного учасника з мережі відповідальність за зберігання його частини даних буде автоматично перерозподілена між іншими вузлами.

Додатковою проблемою є вартість підтримки самої DHT. Щоб таблиця залишалася актуальною та коректно відображала стан мережі, вузли мають періодично обмінюватися службовими повідомленнями, оновлювати маршрути та перевіряти доступність один одного. Результати, наведеними в роботі [3], демонструють, що продуктивність IPFS під час операцій читання та запису в приватних мережах суттєво поступається централізованим клієнт-серверним рішенням.

Альтернативним напрямом є використання блокчейн-технологій, таких як Filecoin або Arweave. Ці системи поєднують ідеї розподіленого реєстру (distributed ledger) та децентралізованого сховища файлів. Теоретично вони мають забезпечувати високу доступність даних, оскільки сама природа блокчейну стимулює реплікацію історії транзакцій усіма учасниками мережі за допомогою різних алгоритмів узгодження.

Мета та задачі дослідження. Дана робота зосереджена на розробці та аналізі підходу, спрямованого на підвищення продуктивності й надійності піринговых систем з адресацією вмісту. Запропоноване рішення не заперечує використання DHT, як основи для маршрутизації та пошуку даних, а доповнює її механізмом проактивної реплікації у поєднанні з гнучкою моделлю закріплення блоків.

У межах дослідження розглядається, чи здатна активна реплікація даних разом із чітко визначеною політикою вилучення зменшити навантаження на окремі вузли та підвищити загальну доступність вмісту в мережі. Буде сформульовано загальні вимоги до протоколу, запропоновано модель закріплення даних та видалення даних, а також зібрано необхідні метрики для оцінки поведінки та ефективності запропонованої системи.

Результати дослідження і їх обговорення. Запропонована система є розширенням подібних піринговых мереж на основі DHT. Для ідентифікації даних використовується концепт адресацію за вмістом, де кожен блок є хешем його вмісту:

$$ID(B) = H(B), \tag{1}$$

де B – блок даних, H – обрана хеш-функція.

Блоки об'єднуються у граф(Merkle DAG), посилаючись на ідентифікатори вмісту: листям графу є блоки з даними, що об'єднуються у вузли які своєю чергою є ідентифікаторами рівнем нижче. Запис провайдера – елемент, що поєднує DHT з адресацією вмісту та вказує на те, що вузол має дані пов'язані з шуканим ідентифікатором.

Процес вилучення файлу з мережі складається з наступних кроків:

1. Клієнт надсилає запит до DHT для пошуку провайдерів кореневого блоку;
2. DHT повертає множину кандидатів (один або кілька вузлів);
3. Клієнт паралельно запитує блоки у кількох провайдерів та обходить граф зверху вниз;
4. Кожен блок перевіряється на цілісність за хешем та у разі помилки клієнт перемикається на іншого провайдера.

Розміщення блоку у мережі визначено ідентифікаторами його вмісту, що гарно поєднується із використанням DHT. Оскільки хеш-функція рівномірно розподіляє значення у просторі ключів, кожен блок природним чином потрапляє в певний діапазон. Вузол з ідентифікатором Ni відповідає за блок, якщо виконується умова:

$$d\left(ID\left(B\right),N_i\right)=ID\left(B\right)\otimes N_i=\min_j\left(ID\left(B\right)\otimes N_j\right), \tag{2}$$

де B – блок даних, N_i – ідентифікатор поточного вузла, N_j – ідентифікатор будь-якого іншого вузла в мережі

Таким чином, уникається локалізація даних на окремих вузлах і досягається більш рівномірний розподіл навантаження на мережу.

Локальне розміщення блоку на конкретному вузлі визначається відповідно до низки узгоджених політик, які регламентують, у яких випадках вузол має прийняти блок у локальне сховище та довго його зберігати та за яких умов він може бути видалений.

Локальне розміщення блоку визначається за наступними політиками:

– Під час розміщення не досягнуто мінімальної кількості копі через невдалі запити чи відповідно до власних налаштувань вузла у відмові приймати сторонні блоки.

– Кореневі блоки завжди зберігаються локально в пам'яті вузла публікатора. Така політика дозволяє відновити дані з локального кешу за допомогою републікації, навіть якщо частина копій у мережі була втрачена.

– Вузол приймає фіксовану частку блоків, у яких ідентифікатор в числовій формі має значення менше за визначене відповідно до власних налаштувань. Формально політику можна описати як:

$$\frac{ID\left(B\right)}{2^k}<p_i,p_i\in\left(0,1\right],ID\left(B\right)\in\left\{0,1,\ldots,2^k-1\right\},, \tag{3}$$

де p_i – фракція блоків, яку вузол готовий зберігати локально, B – блок даних, k – кількість бітів в ідентифікаторі або розмір простору ключів мережі.

На практиці це дозволяє досягнути пропорційного розподілу блоків розосереджених у мережі та тих, що скупчуються на окремих вузлах. Це дозволяє децентралізувати дані, при цьому даючи контроль вузлам над власними ресурсами.

Збереження файлів у мережі забезпечується через механізм закріплення блоків. Закріплення слугує ознакою того, чи дозволено збирачеві сміття видаляти відповідний блок. У системі передбачено два типи закріплень – жорсткі (hard pins) та м'які (soft pins), які відрізняються як способом створення, так і правилами свого життєвого циклу.

Жорсткі закріплення можуть бути визначені лише користувачем та не мають терміну. Вони не мають терміну дії та інтерпретуються як вимога зберігати відповідний блок, доки користувач не скасує закріплення. Жорстке закріплення не дозволяє видалити блок збирачем сміття при жодних умовах.

М'які закріплення, навпаки, є рекомендацією щодо тимчасового збереження блоку. Вони створюються автоматично при збереженні чужого блоку. Для кожного закріплення визначається термін дії (TTL), після завершення якого блок вважається кандидатом на видалення. Кожне успішне вилучення блоку оновлює пов'язане з ним закріплення, подовжуючи його термін дії, тобто активні блоки постійно перепублікуються, а неактивні поступово втрачаються й можуть бути безпечно видалені.

Покриття закріплення може бути як прямим, тобто блок позначається індивідуально, так і рекурсивним. Рекурсивне закріплення охоплює весь підграф, починаючи з кореневого блоку, тобто усі нащадки автоматично вважаються захищеними.

Для кожного блоку вузол підтримує індекс маркерів, що відображають усі наявні закріплення: прямі чи рекурсивні, м'які чи жорсткі. Збирач сміття інтерпретує їх комбінації відповідно до правил, наведених у таблиці 1.

Принцип полягає в тому, що жорстке закріплення має вищий пріоритет за м'яке, а рекурсивне – за пряме: якщо блок хоч охоплений жорстким або рекурсивним закріпленням, він залишається на вузлі. М'які та прямі закріплення розглядаються як менш сильні та можуть бути скасовані після завершення терміну дії.

Таблиця 1

Захист закріплень від збирача сміття

Вид закріплення	Покриття	Захист від збирача сміття
Жорсткий	Пряме	Повинен зберегти; нащадки можуть бути видалені
Жорсткий	Рекурсивне	Повинен зберегти повний граф
М'який	Пряме	Повинен зберегти при умові незакінченого терміну дії

Модель закріплень виступає шлюзом між протоколом реплікації та локальною політикою кешування – нові копії створюються протоколом розміщення даних, а м'які закріплення фіксують факт прийняття вузлом ролі провайдера на певний період. Якщо вузол бажає зменшити свою участь у реплікації, він може понизити TTL для нових закріплень або пришвидшити інтервал роботи збирача сміття, не змінюючи глобальної логіки протоколу. З іншого боку, користувач завжди може підвищити важливість окремого файлу, перевівши відповідні блоки з м'яких закріплень у жорсткі, тим самим гарантуючи їх збереження.

Збирач сміття відповідає за вивільнення місця у локальному сховищі вузла коштом видалення блоків, які більше не вважаються потрібними. На відміну від моделі закріплення даних, що визначає логічний статус блоку, збирач сміття реалізує конкретний алгоритм прийняття рішень: які блоки можуть бути видалені, в якому порядку та за яких умов.

Алгоритм роботи збирача сміття базується на інформації про закріплення, яка під час проходу інтерпретується до правил, наведених у таблиці 1. Збирання сміття виконується у двох режимах: автоматичний, тобто з певним інтервалом часу відбувається перевірка локального сховища та ручний.

Щоб коректно обробити об'єкти, представлені у вигляді графу, застосовується підхід «позначити й видалити» (mark-and-sweep). На першому етапі збирач сміття формує множину коренів – блоків з жорсткими рекурсивними та м'якими прямими закріпленнями. Починаючи з цих коренів, збирач сміття обходить відповідні підграфи та позначає всі досяжні блоки як потенційно потрібні. На другому етапі блоки, які не були позначені й не мають закріплень, вважаються кандидатами на видалення.

Метою обчислювальних експериментів є порівняння запропонованої моделі проактивної реплікації з моделлю «pull-on-demand», подібною до IPFS та надання відповіді на наступні дослідницькі питання:

– чи підвищує запропонована система доступність даних;

– як змінюється час відповіді (затримка) для типових операцій вилучення.

Для оцінювання ефективності запропонованої системи було сформовано експериментальне середовище, яке надає можливість тестування процесу завантаження та вивантаження файлів. Мережа складається з 25 вузлів, кожен з яких має інформацію про випадкову частину мережі. Комунікація між вузлами моделюється з урахуванням мережевої затримки, тобто запити між вузлами отримують штучну затримку у діапазоні 2–20 мс. Для відтворення поведінки реальної пірингової мережі та створення навантаження, тестове середовище періодично виключає випадкові вузли з мережі на час 200 мс – 1,5 с з інтервалом 250 мс.

Сценарій випробування включає розміщення файлу до мережі та його вивантаження випадковим вузлом, відмінним від першого. Аналогічні тести запускаються з використанням моделі pull-on-demand. Для досягнення статистично значущих результатів кожен сценарій запускається, як серія зі 150 незалежних випробувань. Між окремими випробуваннями мережа не перезавантажується, що дозволяє оцінити її стабільність в умовах тривалої роботи.

Для кожної конфігурації збираються наступні показники:

– доступність, тобто частка успішних вилучень файлів до загальної кількості випробувань;

– затримка запиту, тобто час від початку операції вилучення до отримання останнього блоку, агрегований у вигляді середнього значення та метрик p50, p95, p99.

У запропонованій системі параметр keep-local визначає, яку частку блоків вузол погоджується зберігати локально, відповідно до моделі розміщення даних. Як можна помітити у таблиці 2, розподілене розміщення стабільно показує вищу доступність. Навіть при найменшому значенні keep-local=0.1 додаткові репліки вже помітно зменшують частку невдалих запитів.

Таблиця 2

Результати вимірювань для різних значень параметра keep-local

Модель реплікації	keep-local	Успішні запити / всього	Доступність, %	p50, мс	p95, мс	p99, мс	Середня затримка, мс
Пасивна	–	141 / 150	94.00	54.91	68.83	71.82	54.72
Проактивна	0.10	146 / 150	97.33	47.49	62.67	68.64	46.50
Проактивна	0.25	147 / 150	98.00	45.53	68.63	76.29	45.54
Проактивна	0.50	149 / 150	99.33	50.28	69.22	81.86	49.00
Проактивна	0.70	148 / 150	98.67	45.04	67.01	70.98	44.90
Проактивна	0.90	147 / 150	98.00	43.69	63.10	79.31	44.12

Максимальну доступність у 99.33 % було досягнуто при keep-local=0.5, що вказує на наявність корисного діапазону значень, при яких система отримує найбільший виграш від розподіленого розміщення.

Медіанна затримка запиту(p50) знаходиться в межах приблизно 43–50 мс, а середнє значення – 44–49 мс. Це пояснюється наявністю кількох провайдерів для більшості блоків, тому клієнт з високою ймовірністю знаходить близьку репліку, навіть якщо частина віддалених вузлів тимчасово недоступна. Значення p95 та p99 є трохи вищими при проактивній реплікації, що вказує на те, що в поодиноких випадках клієнт змушений спускатись до більш повільних копій та робити додаткові запити при недоступності першочергових вузлів.

Наступні експерименти спрямовані на оцінку того, як на поведінку системи впливають параметри:

– розмір даних у блоці (chunk);

– загальний розмір файлу.

При різному розмірі файлу та частини на які він поділяється, створюється різна кількість блоків – менший розмір частини збільшує кількість блоків, які необхідно успішно отримати, щоб вилучити файл і, відповідно, зі збільшенням частини кількість зменшується.

Результати у таблиці 3 показують передбачуваний компроміс між доступністю та затримкою. Зі зменшенням розміру блоку доступність падає, оскільки успішне вилучення потребує отримання більшої кількості блоків, що збільшує ймовірність втрати хоча б одного блоку.

Таблиця 3

Результати вимірювань для різних *розмірів файлу*

Модель реплікації	Розмір файлу, МБ	Розмір частини, КБ	Доступність, %	p50, мс	p95, мс	p99, мс	Середня затримка, мс
Пасивна	2	256	86.67	54.55	68.00	71.24	53.87
Проактивна	2	256	95.33	47.22	65.36	94.49	48.49
Пасивна	2	512	89.33	53.61	68.30	71.88	53.31
Проактивна	2	512	96.67	46.30	65.64	74.19	45.71
Пасивна	2	1024	91.33	56.54	69.45	73.21	56.24
Проактивна	2	1024	97.33	44.12	70.61	82.33	45.65
Пасивна	5	256	88.67	63.32	76.08	78.39	63.65
Проактивна	5	256	88.67	56.13	91.72	119.88	57.57
Пасивна	5	512	92.67	57.09	69.30	71.60	56.97
Проактивна	5	512	92.67	48.10	67.84	73.89	48.83
Пасивна	5	1024	93.33	59.04	71.53	74.68	58.35
Проактивна	5	1024	96.00	51.20	74.44	90.41	52.43

При цьому виграш у затримці є обмеженим: при базовому підході медіанна затримка практично не покращується, а при розподіленому – залишається в одному діапазоні для всіх комбінацій розміру блоку та файлу. Найкращий баланс досягається для блоків середнього розміру, які зменшують кількість запитів на файл, але не збільшують час передачі кожного блоку настільки, щоб це помітно погіршувало p50.

На (рис. 1) зображено стовпчикову діаграму залежності доступності від кількості блоків одного файлу. Можна помітити, що зі збільшенням числа блоків доступність монотонно знижується для обох підходів до розміщення, проте розподілений стабільно демонструє вищі значення при однаковій кількості блоків.

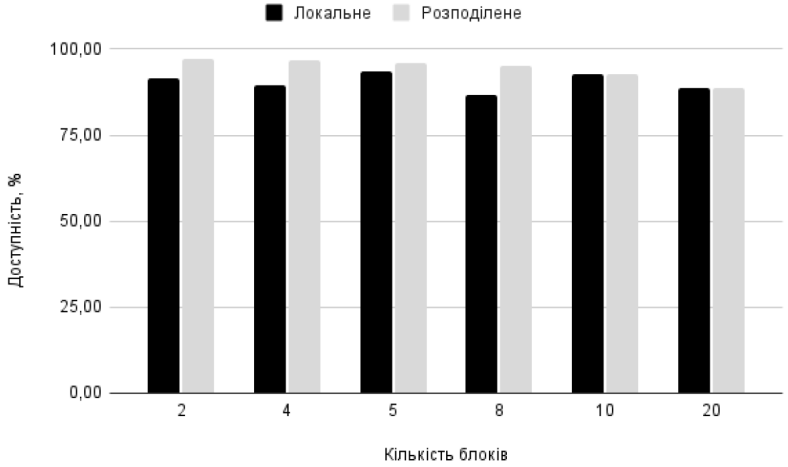

Рис. 1. Стовпчикова діаграма залежності доступності від кількості блоків

Для більш фрагментованих файлів (10 та 20 блоків) обидва підходи дають однакову доступність, що вказує на те, що за великої кількості блоків домінуючим фактором стає ймовірність втрати хоча б одного фрагмента, і перевага додаткових реплік частково нівелюється.

Це підтверджує, що основним фактором ризику є саме фрагментація файлу на велику кількість частин, а запропонований механізм розподіленого розміщення хоча й зменшує, проте не усуває цю проблему повністю.

Висновки. У роботі проаналізовано проблему забезпечення доступності даних у пірингових системах з адресацією вмісту та розроблено підхід, який поєднує проактивне розміщення блоків за їх ідентифікаторами та чіткі політики закріплення даних. Запропонована система підвищує рівень доступності даних без необхідності введення складних або ресурсомістких механізмів координації. Результати дослідження демонструють, що такий підхід є доречним для застосування в малих мережах, де надійність та доступність даних є критичною.

Список використаних джерел:

1. Centralization in the Decentralized Web: Challenges and Opportunities in IPFS Data Management / R. Shi et al. *WWW '25: The ACM Web Conference 2025*, Sydney NSW Australia. New York, NY, USA, 2025. P. 4068–4076. https://doi.org/10.1145/3696410.3714627 (date of access: 13.11.2025).

2. Daniel E., Tschorsch F. IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks. *IEEE Communications Surveys & Tutorials*. 2022. Vol. 24, no. 1. P. 31–52. https://doi.org/10.1109/comst.2022.3143147 (date of access: 13.11.2025).

3. Abdullah Lajam O., Ahmed Helmy T. Performance Evaluation of IPFS in Private Networks. *DSDE '21: 2021 4th International Conference on Data Storage and Data Engineering*, Barcelona Spain. New York, NY, USA, 2021. https://doi.org/10.1145/3456146.3456159 (date of access: 13.11.2025).

References:

1. Shi, R., Cheng, R., Fu, Y., Han, B., Cheng, Y., & Chen, S. (2025). Centralization in the Decentralized Web: Challenges and Opportunities in IPFS Data Management. У *WWW '25: The ACM Web Conference 2025* (с. 4068–4076). ACM. https://doi.org/10.1145/3696410.3714627

2. Daniel, E., & Tschorsch, F. (2022). IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks. *IEEE Communications Surveys & Tutorials, 24*(1), 31–52. https://doi.org/10.1109/comst.2022.3143147

3. Abdullah Lajam, O., & Ahmed Helmy, T. (2021). Performance Evaluation of IPFS in Private Networks. У *DSDE '21: 2021 4th International Conference on Data Storage and Data Engineering*. ACM. https://doi.org/10.1145/3456146.3456159

Дата першого надходження статті до видання: 24.11.2025
Дата прийняття статті до друку після рецензування: 16.12.2025
Дата публікації (оприлюднення) статті 27.01.2026