ОЦІНЮВАННЯ ПАРАМЕТРІВ ІМІТАЦІЙНОЇ МОДЕЛІ KADEMLIA DHT

Мазурок І. Є., Єжкова А. Г., Царенко О. І.
Одеський національний університет ім. І. І. Мечникова

Стаття присвячена вивченню роботи DHT (Kademlia) та аналізу кількості переданих пакетів з метою оптимізації параметрів мережі.

Ключові слова: P2P, однорангові мережі, DHT, Kademlia, пакети, маршрутизація, C++

P2P, або однорангові мережі, часто використовуються для розповсюдження цифрових медіафайлів. В такому разі кожен комп’ютер виконує роль і сервера, і

клієнта — надаючи та приймаючи файли. Kademlia — це рання реалізація DHT (розподілена хеш-таблиця). Однією з її основних особливостей є метрика на основі XOR для побудови топології. Kademlia використовує чотири основні операції: PING, FIND_NODE, STORE і FIND_VALUE, з яких ми розглянемо FIND_NODE.

Для вивчення DHT нам потрібна симуляція подій із підрахунком пакетів. Наша мета – виявити найкращі параметри мережі. Однак цей метод також обмежений продуктивністю процесора.

Усі вузли отримують випадковий 160-бітний ідентифікатор. Вузли зберігаються як листя у бінарному дереві. Позиція кожного вузла визначається найкоротшим унікальним префіксом його ID. По-перше, ми генеруємо велику кількість однорангових вузлів (близько 10^5), які проходять бутстрап-процедуру. Згідно з Kademlia, кожен новий вузол повинен мати доступ до бутстрап-вузла та зареєструвати його у власному списку відомих вузлів (bucket, або бакет) [1]. Потім новий вузол ініціює пошук власного ідентифікатора. Коли інші вузли отримують запит FIND_NODE, вони зберігають ідентифікатор запитувача у своїх бакетах. Отримавши відповіді, ініціатор пошуку зберігає ідентифікатори відповідачів. Так заповнюється DHT. Щойно бутстрап завершується для всіх вузлів, кожен з них обирає випадковий ID та надсилає запит FIND_NODE для цього ідентифікатора. З кожним кроком DHT стає більш повною. У наших тестах цей крок повторюється 200 разів. Ми аналізуємо вузли з розміром бакета 5, 8, 20 і 40. Якщо пошук відбувається занадто довго, а саме кількість пакетів перевищує 3000, шуканий вузол оголошується мертвим для цього кроку. Мета полягає в тому, щоб мінімізувати кількість мертвих вузлів. Це означатиме, що маршрутизація через таблицю виконується правильно. Ми отримуємо найкращі результати з розміром бакета = 8: максимум 4 мертвих вузла за крок. Цей максимум досягається лише один раз. В більшості кроків отримуємо 0 мертвих вузлів. Вони з’являються частіше після кроку 80, коли таблиця наближається до повного стану. Одна з найважливіших систем, що використовує DHT, Bittorrent, зазначає у своїй офіційній документації: кожен бакет може містити лише K вузлів, наразі 8, перш ніж досягути «повноти»[2]. Ситуація погіршується при розмірі бакета = 20. Ми отримуємо більше мертвих вузлів, найчастіше 2. Вони з’являються раніше, починаючи з кроку 25. Максимальна кількість мертвих вузлів становить 8, максимум з’являється один раз ближче до кінця тесту. Ми пояснюємо таку поведінку більшою ймовірністю слідування за неправильним маршрутом. Коли вузол надсилає запит FIND_NODE, він запитує k = 3 вузли, найближчі за XOR до шуканого вузла, чи містять вони відповідний вузол. Якщо так, пошук закінчується. Якщо ні, вони, у свою чергу, шукають у своїх бакетах

вузли, найближчі до шуканого, і продовжують рекурсивний пошук. Під час цього процесу вузол-ініціатор і вузли-посередники реєструють один одного. У великих бакетах накопичується більше непотрібних вузлів. Реєстрація великої кількості віддалених вузлів призводить до помилок у подальшому пошуку, оскільки віддалені вузли можна помилково прийняти за найближчі до шуканого вузла. Ми бачимо ще гірший результат із розміром бакета = 40, де кількість мертвих вузлів найчастіше становить 4, а пошук поганий із самого початку тесту.

Ми розглянули алгоритм Kademlia DHT FIND_NODE і виявили найкраще значення параметра розміру бакета, а саме 8. Результати нашого симуляційного тесту відповідають офіційній документації Bittorrent.

Література

1. Maymounkov P., Eres D. Kademlia: A Peer-to-peer Information System Based on the XOR Metric. Lecture Notes in Computer Science. 2002. № 2429. pp 53–65. DOI:10.1007/3-540-45748-8_5.
2. Loewenstern A., Norberg A. DHT Protocol [Electronic resource]. Access mode: https://www.bittorrent.org/beps/bep_0005.html (date of access: 10.04.2023).