УДК 004.75

МЕТОДИ ПОБУДОВИ СИСТЕМ ДОСТАВКИ КОНТЕНТУ З ЗАСТОСУВАННЯМ РОЗПОДІЛЕНИХ ХЕШ-ТАБЛИЦЬ

А. В. Сергєєв, К. О. Ільченко
Київський національний університет імені Тараса Шевченка
вул. Ломоносова, 81, 03022, Київ, Україна. E-mail: a.serhieiev@gmail.com

Описано основні проблеми, що виникають у сучасних комп'ютерних мережах в умовах великих навантаженнях та у випадку розподілу динамічного контенту. Показано доцільність створення гібридних систем доставки контенту, побудованих на основі розподілених хеш-таблиць – систем, що поєднують переваги класичних систем доставки контенту і розподілених хеш-таблиць, але реалізація яких є надзвичайно складною. Детально досліджені та проаналізовані існуючі теоретично розроблені та практично апробовані моделі гібридних систем доставки контенту. Виділено переваги та недоліки даних систем та проведене їх порівняння. Розглянуто перспективу впровадження теоретично розроблених моделей.

Ключові слова: децентралізована мережа, оверлей, розподілена хеш-таблиця, система доставки контенту.

МЕТОДЫ ПОСТРОЕНИЯ СИСТЕМ ДОСТАВКИ КОНТЕНТА С ИСПОЛЬЗОВАНИЕМ РАСПРЕДЕЛЁННЫХ ХЕШ-ТАБЛИЦ

А. В. Сергеев, К. О. Ильченко
Киевский национальный университет имени Тараса Шевченко
ул. Ломоносова, 81, 03022, Киев, Украина. E-mail: a.serhieiev@gmail.com

Описаны основные проблемы, возникающие в современных компьютерных сетях в условиях больших нагрузок, а так же в случаях распределения динамического контента. Показана целесообразность создания гибридных систем доставки контента, построенных на основе распределенных хеш-таблиц – систем, объединяющих преимущества классических систем доставки контента и распределенных хеш-таблиц, но реализация которых является чрезвычайно сложной. Подробно исследованы и проанализированы существующие теоретически разработанные и практически апробированные модели гибридных систем доставки контента. Выделены преимущества и недостатки данных систем и проведено их сравнение. Рассмотрены перспективы внедрения теоретически разработанных моделей.

Ключевые слова: децентрализованная сеть, оверлей, распределённая хеш-таблица, система доставки контента.

АКТУАЛЬНІСТЬ РОБОТИ. Популяризація комерційних мереж та веб-сайтів відбувається за допомогою комбінації збалансованого навантаження серверів, швидкого мережевого з'єднання та систем доставки контенту – CDN (Content Delivery Network), мереж, що будуються за принципом реплікації даних з основного до периферійних серверів, з метою зменшення затримки при пошуку та доставці необхідної користувачам інформації. Відсутність перелічених компонентів призводить як до технічних ризиків – перевантаження та вихід системи з ладу через непередбачувані і нестабільні об'єми трафіку у мережі, так і до економічних – дистриб'ютори мають обмежений розмір цільової аудиторії і лімітовані типи контенту. У випадку статичного контенту окреслена проблема вирішується шляхом його розповсюдження більшій кількості клієнтів, ніж дистриб'ютори можуть обслуговувати самостійно, учасниками-волонтерами, які віддзеркалюють дані на своїх ресурсах. Незважаючи на ненадійність такого методу, останнім часом існує достатньо прикладів, коли окремі користувачі з широкою розподільною смугою, перерозподіляють контент, яким вони самі користуються. Складнішою є ситуація з динамічним контентом, стан якого постійно змінюється і процес розповсюдження серед волонтерів якого значно ускладнюється.

Використання гібридної оверлейної CDN-DHT (Distributed Hash Table – Розподілена Хеш-Таблиця) мережі потенційно є ефективною технологією для полегшення масштабного та надійного розподілу контенту та має порівняно низьку вартість розгортання. Але, враховуючи автономність та змінюваність учасників мережі, її побудова є нетривіальною задачею, вирішення якої можливе при дотриманні кількох умов. По-перше, розподіл завдань та контенту між вузлами не повинен конфліктувати з інтересами цих вузлів для надання належного стимулу для співпраці. По-друге, так як однорангові мережі абстрагують всю топологічну інформацію про фізичну структуру, що лежить у їх основі, структура CDN-DHT повинна враховувати фактор локальності для впровадження швидкого пошуку копій необхідних даних, які знаходяться на найменшій відстані від користувача. По-третє, система повинна бути надійною, тобто відмовостійкість в умовах помилок та динамічних змін системи, які трапляються у однорангових мережах. На сьогодні питання побудови оптимальних систем доставки контенту не є вирішеним, тому ця стаття розглядає питання методів їх реалізації, що базуються на використанні розподілених хеш-таблиць.

Метою даної роботи є дослідження та аналіз існуючих гібридних систем доставки контенту з метою виявлення переваг та недоліків для подальшої розробки удосконаленої моделі гібридної системи доставки контенту із урахуванням описаних результатів.

МАТЕРІАЛ І РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ. Наразі існує декілька спроб реалізації гібридних систем доставки контенту, з яких лише одна пройшла емпіричну апробацію. Спочатку необхідно розгля-

нути теоретичні моделі, які не мають практичної реалізації.

Найбільш вузько спеціалізованою є система G-CP2P [1], що призначена для потокових сервісів, таких як відео- чи аудіо-стрімінг. У її структурі основна увага приділяється таким проблемам, як мережева затримка та складність налаштування з'єднання між учасниками мережі. Ці проблеми є результатом суб'єктивного вибору вузлів, без врахування їх місцезнаходження та контенту, яким вони обмінюються або зберігають, що призводить до суттєвого погіршення у роботі системи. Для вирішення цих проблем авторами пропонується гібридна архітектура, яка базується на виборі елементів за рахунок їх розташування та зацікавленості у розподілі контенту. В даній моделі деякий супервузол виконує вибір вузлів на основі розташування, використовуючи асоціативну мережу (content addressable network – CAN) для поширення інформаційного каналу. Він же керує вузлами за інтересами до контенту, формуючи групу вузлів з однаковим каналом, у вигляді суб-оверлею (логічної топології мережі, що побудована поверх іншої мережі).

У G-CP2P вузол, що фізично близько розташований до сервера CDN виконує роль супервузла, який зберігає список ID (унікальних ідентифікаторів) елементів, інформацію про канал за яким вузол спостерігає і інформацію про статус вузла. У якості DHT використовується CAN (Content Addressable Network) [2], так як він використовує техніку зв'язку [3], що базується на фізичному положенні елемента.

Вузли, що слідкують за одним аудіо- чи відеоканалом та розташовані у одній місцевості формують деякий суб-оверлей, кожним з яких опікується супервузол і забезпечує роботу сервісу потоку. Якщо елемент хоче отримати список вузлів для запиту, він контактує з найближчим супервузлом, замість запиту до основного серверу. Якщо супервузол не має списку вузлів для шуканого каналу, елемент відправляє запит до супервузла, що розташований у іншому суб-оверлеї через CAN.

Техніка приєднання за ознакою розташування виглядає наступним чином. На першому кроці вузол вимірює свої RTT (round-trip time - час, за який запит проходить від клієнта до сервера і назад), які вимірюються за деякими наперед заданими орієнтирами. Після цього значення RRT впорядковуються за зростанням. У CAN вузли випадковим чином приєднуються до зон, а отже, суміжні вузли CAN не знаходяться топологічно близько на рівні IP. Це призводить до неефективного процесу маршрутизації, так як вузли, які контактують між собою можуть знаходитися географічно віддалено. Вищезазначена техніка приєднання може бути використана для побудови CAN, яка відповідає IP топології [4], що лежить у її основі.

У архітектурі G-CP2P припускається, що місцезнаходження супервузла при побудові DHT оверлею, базується на сервері CDN. Супервузли розташовуються у CAN-зонах відповідно до виміряних значень RTT.

Структура даних системи побудована наступним чином. Супервузол обслуговує базу даних вузлів, що знаходяться у одному суб-оверлеї, що відповідає його каналу. Кожен елемент у суб-оверлеї додає сутність, що зберігає ID каналу, ID вузла та статус вузла до бази. У момент, коли вузол відправляє запит, йому вертається список супервузлів. Ефективний вибір супервузла визначається спеціальним механізмом зберігання даних для алгоритму CAN. Супервузол зберігає функцію відображення з ID каналу до ID супервузла, додаючи кількість вузлів, які дивляться канал у даний момент:

$$mapping(Channel - ID,\ SuperPeer - ID + Number\ of\ Peers),$$

де $Channel - ID$ – ідетифікатор шуканого каналу; $SuperPeer - ID$ – ідентифікатор супервузла; $Number\ of\ Peers$ – кількість вузлів, що дивляться канал.

До недоліків даної моделі можна віднести орієнтованість на динамічний контент, при цьому не приділено уваги до роботи зі статичним контентом, а також використання у якості основи CAN, зі всіма її перевагами та недоліками [5].

Наступною моделлю гібридної системи доставки контенту є Flower-CDN [6] - багаторівнева система, що в теорії дозволить некомерційним сайтам ефективно розподіляти контент з допомогою спільноти, що зацікавлена у ньому. Головною ідеєю є зв'язок вузла з деяким оверлеєм, що складається з елементів які мають спільний інтерес та знаходяться близько один до одного. Елементи у ньому зберігають та розповсюджують контент певного сайту. Вузол, що шукає інформацію, може знайти свій локальний оверлей за допомогою спеціального службового каталогу, названого *D-ring*, який є реалізацією локально-орієнтованої розподіленої хеш-таблиці. Підтримка оверлею є низьковартістною у плані розрахункових ресурсів і забезпечується учасниками мережі.

Модель Flower-CDN призначена для підтримки W сайтів, кожен з яких має набір веб-сторінок та документів. Система використовує бажання клієнта співпрацювати у плані поширення цікавого для нього контенту (у цьому плані вона схожа на G-CP2P). Сайт ws додається до множини W за ініціативою самого сайту, або ж користувача, який у ньому зацікавлений та є учасником системи.

Для реалізації зв'язку, що базується на локальності, припускається, що вся мережа Інтернет розбивається на певні сегменти локальності за допомогою *landmark-based* техніки [6]. Кожен учасник системи знає кількість локальностей k і може визначити до якої локальності loc сам відноситься, вимірюючи затримку до кожної з них. Вузли, що належать до однієї локальності та зацікавлені у одному сайті, створюють оверлейну мережу, що визначається, як $contant - overlay\ (ws, loc)$. Елементи $c_{\mathrm{ws,loc}}$ цієї мережі знаходяться у локальності loc і зберігають, керують та розповсюджують інформацію ws, що значно зменшує навантаження на сервер. З кожного *contant-overlay* вибирається один вузол, який має всю інформацію про всі $c_{\mathrm{ws,loc}}$ та позначається $d_{\mathrm{ws,loc}}$. Множина усіх $d_{\mathrm{ws,loc}}$ утворює *D-ring* - роз-

поділену хеш-таблицю для підтримки запитів нових користувачів та пошуку інформації на сайтах з множини W.

Таким чином Flower-CDN опирається на гібридну архітектуру, що складається з набору незалежних *contant-overlay*, що з'єднані через *D-ring* на верхньому рівні. Замість того, щоб навантажувати сервер ws, новий клієнт, розташованій у локальності loc, відправляє запит до *D-ring*, який переправляється до $d_{\text{ws,loc}}$. Після цього, запит направляється до $c_{\text{ws,loc}}$, який містить шуканий об'єкт і відповідний об'єкт відправляється напряму до користувача. Після цього, клієнт може приєднатися до $contant - overlay\ (ws, loc)$ у вигляді нового учасника $c_{\text{ws,loc}}$, якщо він має бажання брати участь у розподілі даних системи. Для майбутніх запитів $c_{\text{ws,loc}}$ шукає прямо у $contant - overlay\ (ws, loc)$ замість нового запиту до *D-ring*.

Це означає, що у мережі всі вузли які бажають підтримувати деякий сайт $ws \in W$ стають частиною одного з оверлеїв *ws* і допомагають розподіляти контент. Такий набір вузлів виглядає наступним чином:

$$\forall\, ws \in W:\ P_{ws} = \cup_{0 \le loc \le k}\, contant - overlay\ (ws, loc).$$

Для забезпечення швидкого доступу до системи, *D-ring* може бути побудована за допомогою будь-якої структурованої DHT (таких як Chord [7], Pastry [8] і т.д.). Для будь-якого веб-сайту $ws \in W$ оверлей каталогу дозволяє k учасникам з P_{ws}, де k - кількість локальностей, стати $d_{\text{ws,loc}}$ для *ws*, таким чином, щоб кожна локальність *loc* була покрита $d_{\text{ws,loc}}$ для реалізації перенаправлення запитів, що базується на локальності. Для знаходження $d_{\text{ws,loc}}$ йому необхідно призначити ідентифікатор, котрий якимось чином був би зв'язаний з сайтом та локальністю. У даній системі цей *ID* визначається, як конкатенація ідентифікаторів сайту та локальності, де:

- *ID* локальності – молодший сегмент бітів довжини m_1 (у ідентифікаторі $d_{\text{ws,loc}}$). m_1вибирається таким чином, щоб $2^{m_1} \ge k$;
- *ID* сайту - старший сегмент бітів довжини m_2. Отримується у результаті хешування URL сайту ($hash(ws)$).

$d_{\text{ws,loc}}$ у одній локальності мають однаковий *ID* локальності, з одного сайту - однаковий *ID* сайту. Таким чином вони мають послідовні ідентифікатори і є сусідами у *D-ring*. При цьому шуканий $d_{\text{ws,loc}}$ може бути тимчасово недоступним, або він міг ще не встигнути приєднатися до *D-ring*. В таких випадках, інший $d_{\text{ws,loc}}$ того ж веб-сайту *ws* повинен обробити запит. Втім, можливий випадок, у якому сусідній $d_{\text{ws,loc}}$ буде відноситися до іншого сайту. Для гарантії правильного перенаправлення, припускається модифікація пошукового сервісу у DHT.

Модифікація алгоритму пошуку виглядає наступним чином. Операція $route(key, msg)$ призначена для відправки повідомлення *msg* до вузла з *ID* рівним, або чисельно близьким до ключа key. Нехай вузол p виконує алгоритм пошуку, використовуючи свою таблицю маршрутизації. Якщо вона визначає, що p найближчий до ключа вузол, то повідомлення *msg* досягло пункту призначення і процедура закінчується. В іншому випадку, p вибирає зі своєї таблиці маршрутизації вузол p', *ID* якого найближчий до ключа. Якщо звичайний локальний пошук виконується у p, *ID* сайту вузла p' звіряється з ідентифікатором ключа. Після цього запускається додатковий пошук: він шукає чисельно найближчий вузол до ключа, з тим самим *ID* сайту як ключ, про який знає p (p' може мати інший *ID* локальності ніж ключ). Якщо такий вузол не знаходиться, результатом вважається p'.

Щоб впоратися із запитами, $d_{\text{ws,loc}}$ використовує дві локальні структури:

1. $directory - index(ws, loc_i)$ – каталог індексів, що індексує контент *ws*, який зберігається у $contant - overlay\ (ws, loc_i)$. У каталозі зберігаються сутності у яких містяться по 3 поля для кожного $c_{\text{ws,loc}_i}$:
   - Інформація про адресу $c_{\text{ws,loc}_i}$ (зазвичай, IP адреса);
   - Поле «віку», призначене для виявлення часу відмови або виходу з мережі;
   - Список ідентифікаторів об'єктів, що описують контент, який зберігається у $c_{\text{ws,loc}_i}$.
2. Набір $directory - summaries(ws, loc_j)$ - перелік directory-index, що обслуговуються іншими $d_{\text{ws,loc}_j}$, $i \ne j$.

Нехай o_{ws} - шуканий об'єкт даних, а $query(o_{ws})$ - відповідний запит. Коли новий клієнт робить запит $query(o_{ws})$, D-ring доставляє його до $d_{\text{ws,loc}}$, що відповідає за *ws* у локалі клієнта loc_i, пошуковий ключ генерується, використовуючи loc_i та *ws*. Спочатку $d_{\text{ws,loc}_i}$ шукає свій каталог індексів для шуканого об'єкту o_{ws}. Якщо $directory - index(ws, loc_i)$ показує, що o_{ws} зберігається деяким $c_{\text{ws,loc}_i}$, $d_{\text{ws,loc}_i}$ перенаправляє $query(o_{ws})$ до $c_{\text{ws,loc}_i}$, перед тим перевіривши чи існує він у системі. Інакше, $d_{\text{ws,loc}_i}$ робить запит до *directory-summaries*, щоб перевірити чи деякий $d_{\text{ws,loc}_j}$ має шуканий об'єкт у своєму каталозі індексів. Якщо $d_{\text{ws,loc}_j}$ буде знайдений, $query(o_{ws})$ буде перенаправлено до $d_{\text{ws,loc}_j}$ і почнеться процес обробки запиту $query(o_{ws})$. Якщо $d_{\text{ws,loc}_j}$ знайдено не буде, запит буде направлено до сайту *ws*.

Після того, як елемент перетворюється у $c_{\text{ws,loc}}$, будь-який запит до веб-сайту *ws*, вже використовує $contant - overlay\ (ws, loc)$ замість *D-ring*. Таким чином *D-ring* використовується тільки при першому заході елемента до системи, допомагаючи новому учаснику з локальністю *loc*, зацікавленому у сайті *ws*, знайти його $contant - overlay\ (ws, loc)$.

Необхідно також розглянути побудову самого $contant - overlay\ (ws, loc)$. $d_{\text{ws,loc}}$ - точка відліку цієї структури. Після того, як $d_{\text{ws,loc}}$ розгорнувся у мережі, інші вузли у локальності *loc*, які бажають підтримувати сайт *ws*, приєднуються до оверлею у вигляді $c_{\text{ws,loc}}$. На практиці це відбувається, коли вузол p робить свій перший пошук деякого об'єкта

o_{ws} на сайті ws. Таким чином, p спочатку отримує доступ до $d_{\text{ws,loc}}$ за допомогою вищеописаного сервісу пошуку, а після цього отримує копію o_{ws} для подальших запитів. p стає $c_{\text{ws,loc}}$ і додається до $directory - index(ws, loc)$.

$c_{\text{ws,loc}}$ може забажати доступу до об'єктів ws, які не знаходяться у його локальному сховищі. Щоб уникнути зайвих запитів, вузли $c_{\text{ws,loc}}$ обмінюються, у межах свого оверлею, переліком контенту ws, що зберігається у їх сховищах. Отже, $c_{\text{ws,loc}}$ може переглядати зміст свого $contant - overlay\ (ws, loc)$, щоб побачити де може зберігатися копія потрібного йому об'єкта даних.

Обробляючи запити, Flower-CDN проводить копію об'єкта по всьому $contant - overlay\ (ws, loc)$. Таким чином, навантаження, що виникає у результаті перенаправлення запитів буде, як правило, розподілено рівномірно по всьому набору $c_{\text{ws,loc}}$, що зберігають копії шуканого об'єкта.

$d_{\text{ws,loc}}$, як учасник $contant - overlay\ (ws, loc)$, також приймає участь у керуванні оверлеєм. Для цього кожний $c_{\text{ws,loc}}$ зберігає *слід* поточного $d_{\text{ws,loc}}$ і підтримує спеціальну сутність для $d_{\text{ws,loc}}$, яка зберігає лише його адресу та вік. $c_{\text{ws,loc}}$ періодично підвищує вік сутності $d_{\text{ws,loc}}$. При будь-якому обміні інформацією між вузлами, $c_{\text{ws,loc}}$ посилає сутність, що належить до $d_{\text{ws,loc}}$. Цей процес поширює інформацію про оновлення $d_{\text{ws,loc}}$ через весь $contant - overlay\ (ws, loc)$, в першу чергу, маючи на меті відновлення після можливої помилки.

У випадку, якщо шуканий вузол від'єднався, або ж не зміг обробити запит, $d_{\text{ws,loc}}$ шукає іншого учасника, від $c_{\text{ws,loc}}$ до сервера, до тих пір, поки доступна копія шуканого об'єкту не буде знайдена. Система мінімізує кількість відмов при перенаправленні запитів, підтримучи каталоги індексів з нещодавно оновленими сутностями. Для цього використовуються так звані *keepalive messages* (повідомлення, що перевіряють активність), які періодично відправляютья для перевірки з'єднання між вузлами. Таким чином, $c_{\text{ws,loc}}$ регулярно посилають *keepalive messages* до $d_{\text{ws,loc}}$. Після прийняття такого повідомлення, $d_{\text{ws,loc}}$ обнуляє вік сутності $c_{\text{ws,loc}}$. Більше того, $d_{\text{ws,loc}}$ постійно перевіряє вік кожного каталогу і видаляє його, якщо вік перевищує деяке порогове значення, що позначається, як T_{dead}. Не можна виключати й випадку при якому відмовляє $d_{\text{ws,loc}}$. Звичайна DHT замінює вузли, перебудовуючи систему і відповідно перерозподіляючи дані. У Flower-CDN своя стратегія заміни для збереження працездатності *D-ring*.

Заміна $d_{\text{ws,loc}}$ проводиться вузлами з $contant - overlay\ (ws, loc)$, тому що вони розділяють інтерес у одному веб-сайті та знаходяться у одній локальності. Коли $d_{\text{ws,loc}}$ по своєму бажанню виходить з системи, він вибирає вузол із $contant - overlay\ (ws, loc)$, відповідно до певного, наперед заданого алгоритму. Якщо ж $d_{\text{ws,loc}}$ виходить з ладу, деякі вузли $c_{\text{ws,loc}}$ виявляють це, відправляючи *keepalive messages*. Кожен $c_{\text{ws,loc}}$ котрий виявляє, що $d_{\text{ws,loc}}$ вийшов з ладу, намагається його замінити наступним чином: він використовує загальний ключ, прив'язаний до $d_{\text{ws,loc}}$ і намагається приєднатися до *D-ring*, використовуючи спеціальну процедуру [7]. Повідомлення про приєднання в кінці кінців досягає найближчого до шуканого *ID* $d_{\text{ws,loc}}$ у *D-ring*. Якщо позиція $d_{\text{ws,loc}}$ вже була зайнята, то повідомлення приходить до нього, отже $c_{\text{ws,loc}}$, який намагається приєднатися до *D-ring* «знайомиться» зі своїм $d_{\text{ws,loc}}$ та інформує про своє існування інші $c_{\text{ws,loc}}$. Новий $d_{\text{ws,loc}}$ поступово будує каталог, отримуючи нові й нові повідомлення.

Для забезпечення масштабованості системи, припускається, що від кожної пари (сайт, локальність) може бути делеговано більше одного $d_{\text{ws,loc}}$. Кожен з цих вузлів керує своїм особистим оверлеєм. Для досягнення цього на практиці, ID вузла потрібно розширити, додавши b додаткових бітів у його кінці для збереження ідентифікації локальності та сайту.

Ще одна проблема полягає у тому, що локальності вузлів можуть змінюватися, через те, що мережа є динамічною. Таким чином, деякі вузли повинні будуть змінювати свій $contant - overlay\ (ws, loc)$. Система долає ці проблеми таким же чином, як і різні відмови: вузол p, що змінює свою локальність, будь це $d_{\text{ws,loc}}$ чи $c_{\text{ws,loc}}$, приєднується до свого нового $contant - overlay\ (ws, loc)$ як новий учасник і після цього оновлює інформацію $d_{\text{ws,loc}}$, до якого належить. Коли вузол з попереднього $contant - overlay\ (ws, loc)$ p' контактує з p, йде обмін інформацією і p видаляється із контактів цього оверлею.

Емуляція моделі системи показала наявність деяких недоліків. По-перше це проблеми масштабованості - при збільшенні кількості елементів мережі в деяких випадках значно збільшується час пошуку необхідної інформації. По друге, у випадку різкого скорочення кількості елементів мережі у моменти пошуку, його ефективність падає практично до нуля, і повертається до нормального стану тільки при наступному процесі пошуку. Очевидно, що дана система потребує вдосконалення, так як вказані недоліки є критичними у випадку повсякденного використання.

Система PetalUp-CDN [9], є логічним продовженням Flower-CDN, в якій розглядається побудова гібридної системи з акцентом на масштабованість та надійність.

Архітектура системи в цілому базується на архітектурі Flower-CDN з деяким удосконаленням. У цій системі $contant - overlay\ (ws, loc)$ визначається як $petal\ (ws, loc)$, а сама вона створена таким чином, що при розгортанні нової *пелюстки* ($petal\ (ws, loc)$) навантаження на $d_{\text{ws,loc}}$ залишалося у нормі. Так як і у Flower-CDN до *D-ring* може входити більше одного $d_{\text{ws,loc}}$ з кожного $petal\ (ws, loc)$.

Для керування масштабованістю до вищезгаданого *ID* $d_{\text{ws,loc}}$, який є конкаттенацією *ID* сайту та *ID* локальності, додаються m додаткових бітів, що називаються *ID* масштабованості. Для кожної пари (ws, loc) отримується 2^m послідовних ідентифіка-

торів. Отже можна отримати до 2^m $d_{\text{ws,loc}}$, які визначаються як $d^i_{ws,loc}(0 \le i \le 2^m)$. Всі $d_{\text{ws,loc}}$ одного сайту й локальності мають послідовні *ID* і є сусідами у *D-ring*. Кожен вузол $d^i_{ws,loc}$ керує частиною каталогу індексів $directory - index(ws, loc_i)$. Наявність декількох $d_{\text{ws,loc}}$ означає, що відмова кількох з них не приведе до повної втрати інформації про каталог і це дозволяє системі продовжувати роботу, хоч й із дещо пониженою пропускною здатністю.

Конструкція PetalUp-CDN включає у себе розширення *D-ring* та його пелюсток. Розширення *D-ring* полягає у наступному: новий $d_{\text{ws,loc}}$ створюється для $petal\ (ws, loc)$ тоді, коли існуючі $d^i_{ws,loc}$ не можуть впоратися з кількістю $c_{\text{ws,loc}}$. Іншими словами, $d_{\text{ws,loc}}$ системи $petal\ (ws, loc)$ створюються послідовно, починаючи з $d^0_{ws,loc}$. Таким чином, у PetalUp-CDN, запит, що шукає $petal\ (ws, loc)$ сканує всі існуючі $d^i_{ws,loc}$, щоб знайти той недовантажений вузол, який зможе обробити запит і взяти нового учасника до свого оверлею у вигляді $c_{\text{ws,loc}}$. Якщо такий елемент не знаходиться, останній створений $d^i_{ws,loc}$ ініціює приєднання нового $d^{i+1}_{ws,loc}$. Запит, що проводиться через *D-ring*, використовує ключ, у якому ідентифікатори сайту та локальності відображають інформацію клієнта. Для визначення *ID* масштабованості припускається, що оптимальний шлях той, який запит повинен пройти для сканування $d_{\text{ws,loc}}$ шуканої пелюстки. Для знаходження такого шляху необхідно по-перше, мінімізувати кількість перенаправлень запиту для обмеження часу відповіді на нього, а по-друге, жоден $d_{\text{ws,loc}}$ не повинен бути перевантажений через перенаправлення запитів. І справді, якщо кожен новий клієнт буде контактувати з $d^i_{ws,loc}$, то останній може перевантажитись, навіть якщо він тільки перенаправляє запити до інших вузлів. Отже $d^i_{ws,loc}$ повинні не тільки займатися керуванням інформацією каталогу, а й брати участь у процесі обробки нових запитів. Таким чином, оптимального шляху можна досягти у випадку, коли кожен клієнт може досліджувати кількість $d_{\text{ws,loc}}$, що були створені для його пелюстки, і випадковим чином вибирати одного, з яким буде контактувати. Якщо така схема не доступна, для *ID* масштабованості вибирається випадкове значення між нулем та його середньою величиною у мережі.

Розширення $petal\ (ws, loc)$ полягає у тому, що як тільки запит клієнта буде оброблено, сам він стає $c_{\text{ws,loc}}$ у пелюстці і більше не використовує *D-ring* для маршрутизації своїх запитів. Для можливості передачі даних через кожний $petal\ (ws, loc)$, $c_{\text{ws,loc}}$ контактує з будь-яким іншим $c_{\text{ws,loc}}$ зі своєї пелюстки.

Слід також розглянути відмовостійкість системи під час надзвичайних ситуацій. Всі механізми системи працюють за рахунок зв'язку між *D-ring* та пелюстками. Отже в першу чергу, слід приділяти увагу до обслуговуванні зв'язку у динамічному середовищі. Так як кілька $d^i_{ws,loc}$ співіснують у одній пелюстці, треба підтримувати зв'язок між ними всіма , а також між ними та набором вузлів у $directory - index(ws, loc_i)$. Кожен вузол з $petal\ (ws, loc)$ обмежує свої зв'язки до вузла $d^i_{ws,loc}$ через який елемент приєднався до $petal\ (ws, loc)$. $c_{\text{ws,loc}}$ підтримує каталог *dir-info,* який зберігає інформацію про $d^i_{ws,loc}$: вік, адресу та ID. Вік - значення, що $c_{\text{ws,loc}}$ періодично збільшує та обнуляє у момент кожного контакту з $d^i_{ws,loc}$ для перевірки його доступності. Процес перевірки та утримування зв'язку між $d^i_{ws,loc}$ та його $c_{\text{ws,loc}}$ проходить тим же чином, що і для Flower-CDN – за допомогою *keepalive messages.*

$c_{\text{ws,loc}}$, що контактують між собою, також обмінюються своїми *dir-info.* Якщо отримане *dir-info* сберігає той же ID вузла, то вузли належать одному й тому ж $d^i_{ws,loc}$. В такому випадку, вони зберігають *dir-info* з меншим віком, що означає більш свіжу інформацію про $d^i_{ws,loc}$. Отже, коли $d^i_{ws,loc}$ покидає систему, деякі з його $c_{\text{ws,loc}}$, що виявляють це, розподіляють інформацію до інших зацікавлених вузлів, для того, щоб вони змогли оновити свої *dir-info.*

Завдяки вдосконаленню архітектури системи, авторам вдалося подолати проблему слабкої масштабованості, але нажаль недолік пов'язаний із пошуком у момент відтоку вузлів залишився. Третя ітерація моделі, вже анонсована авторами, має позбутися цієї проблеми.

На даний момент, єдиною практично апробованою гібридною системою доставки контенту залишається CoralCDN [10]. Ця мережа перебувала у експлуатації у 2005-2012 роках, але деякі критичні недоліки не дозволили їй витримати конкуренцію з мережами інших типів.

З точки зору кінцевого користувача, використання приєднання до мережі є надзвичайно простим - необхідно тільки додати суфікс «nyud.net» до оригінального URL. Структурно CoralCDN складалася з трьох модулів: оверлею проксі серверів, які відповідали на запити клієнтів; оверлею DNS, що під'єднував клієнтів до найближчих проксі-серверів та інфраструктури індексування, яка була основою всієї системи.

Процес пошуку інформації відбувався наступним чином. Спочатку користувач робив запит до свого локального сервера, який у свою чергу за допомогою деякого DNS серверу намагається обробити ім'я шуканого хоста. Після цього, DNS сервер знаходив RTT клієнта з локальним сервером і на основі цього визначав, чи існують поблизу до локального серверу проксі-сервери. У випадку їх відсутності, вибирався випадковий набір серверів. Наступним кроком користувач робив HTTP запит до знайденого проксі серверу, у тому випадку, якщо проксі зкешував файл локально, процедура пошуку завершується, у іншому – проксі сервер шукав потрібний URL у основного серверу. Якщо система знаходив адресу вузла, що вже зкешував об'єкт, то проксі доставляв його до клієнта, при цьому кешуючи його. У іншому випадку, запит робився до основного серверу. У кінці процедури проксі зберігав посилання на себе у системі, записуючи факт того, що він тепер зберігає дані шуканого URL.

Інфраструктура індексування базувалася на так званій нечіткій розподіленій хеш-таблиці (distributed sloppy hash table – DSHT). На відміну від класичної

DHT, у DSHT під одним ключем може зберігатися декілька значень. DSHT використовувалася для пошуку серверів Coral, що знаходилися топологічно близько до мережі користувача, для знаходження проксі-серверів, що кешують об'єкти, а також для знаходження близьких вузлів для мінімізації затримки пошуку відповіді на запит. Базою даної DSHT була розподілена хеш-таблиця *Kademlia* [11].

На відміну від глобального оверлею [12–14], кожен вузол у Coral належав до кількох окремих DSHT, що називалися *кластерами*. Кожний кластер характеризувався максимальним RTT (*діаметром)*. Система параметризована фіксованою ієрархію діаметрів, що звуться *рівнями.* Кожен елемент системи був членом однієї DSHT на кожному рівні. Группа вузлів могла сформувати кластер рівня i, у випадку, якщо попарне RRT нижче порогу діаметру рівня i.

Суфікс nyucd.net є псевдонімом [15], та використовувався для спрощення повної адреси посилання http.L2.L1.L0.nyud.net.

Статистичний аналіз роботи мережі, що проводився під час її експлуатації, показав існування досить критичних недоліків, які у підсумку привезли до згортання системи. По перше, це – великий відсоток (більш ніж 6 %) з'єднань, у яких втрачались пакети [16]. По-друге – нестійкість в умовах слешдот-ефекту [17], тобто різкого збільшення навантаження. Таким чином, мережа була у змозі ефективно працювати тільки у випадку стабільного навантаження і не могла впоратися з різким напливом користувачів, а також не могла забезпечити комфортну роботу для клієнтів, втрачаючи критичну кількість пакетів при обміні інформацією між вузлами.

У результаті дослідження існуючих моделей гібридних систем доставки контенту доцільно провести їх порівняльну характеристику та розглянути перспективи впровадження для нереалізованих на даний момент систем.

Таблиця 1 – Порівняння існуючих моделей гібридних систем доставки контенту

Система	Основні недоліки	Основні переваги	Перспективи впровадження
G-CP2P	Непристосованість до статичного контенту; Використання у якості основи CAN	Зменшення затримки при доставці даних за рахунок ефективного вибору вузлів при взаємодії	Відсутні на даний момент
Flower-CDN	Низька результативність при пошуку інформації за умови значного відтоку вузлів мережі; Досить слабка масштабованість системи	Використання змішаної структурованої моделі, яка базується на виборі вузлів за критерієм місцезнаходження, результатом чого є значне зменшення часу при пошуку інформації	- (перша ітерація PetalUp-CDN)
PetalUp-CDN	Низька результативність при пошуку інформації за умови значного відтоку вузлів мережі	Покращений алгоритм пошуку, що дозволяє проводити пошуку лише у певному сегменті базової розподіленої хеш-таблиці; Збільшення кількості вузлів призводить до збільшення ефективності пошуку, при цьому ресусоємність обслуговування майже не змінюється	Реалізація планується при наступній ітерації розробки системи
CoralCDN	Велика кількість втрачених пакетів; нестійкість до слешдот - ефекту	Ефективна робота при відносно невеликому навантаженні; Простота у користуванні для клієнтів; Використання Kademlia у якості основи системи; Перша спроба реалізації гібридної системи доставки контенту	Вийшла з експлуатації у 2012 році

Так як більшість розглянутих систем досі практично не апробована, емпірично дослідити переваги і недоліки можливо лише для CoralCDN, в той час як інші системи можливо дослідити лише аналітичним шляхом, досліджуючи структуру їх моделей.

Кожна з розглянутих моделей має різну архітектуру (крім Flower-CDN та PetalUp-CDN, шо відрізняються лише у деталях) й відповідно свої індивідуальні недоліки. Для успішного введення в експлуатацію та ефективності використання з точки зору користувачів, ці недоліки мають бути усунуті.

ВИСНОВКИ. У статті розглянуті та проаналізовані розроблені на цей час гібридні системи доставки контенту, виділені їх переваги та недоліки, досліджено можливість впровадження моделей, що не були реалізовані до цього часу.

Підсумовуючи слід зазначити, що незважаючи на перспективу подолання багатьох проблем, пов'язаних у першу чергу з ресурсоємністю і надійністю сучасних мереж, процес побудови ефективної гібридної системи доставки контенту знаходиться на початковому етапі створення математичної моделі і потребує більшої уваги у сфері наукових досліджень.

ЛІТЕРАТУРА

1. Kim T.N., Jeon S., Kim Y.A CDN-P2P Hybrid Architecture with Content / Location Awareness for live Streaming Service Networks // Proc. IEEE 15th Int. Symposium on Consumer Electronics – Aug. 2011. – P. 438–441.

2. A scalable content-addressable network / S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker // Proc. of ACM SIGCOMM 2001 – August 2001 – P. 161–172.

3. «Topologically-aware overlay construction and server selection» / S. Ratnasamy, M. Handley, R. Karp, S. Shenker // Proc. of IEEE INFOCOM – June 2002 – P. 1190–1199.

4. El Dick M., Pacitti E., Kemme B. Flower CDN: a hybrid P2P overlay for efficient query processing in CDN‖ // ACM EDBT – 2009 – P. 427–438.

5. Сергеев А.В. Технологии поиска данных в одноранговых сетях, построенных на основе распределенных хеш-таблиц // Управляющие системы и машиныю – №2, 2016. – С. 41–48.

6. Mechanisms of protection of information in networks, based on distributed hash tables / S. Ratnasamy, M. Handley, R. Karp, S. Shenker // Proc. INFOCOM, New York, USA – Jun. 2002 – P. 32–36.

7. Chord: A scalable p2p lookup service for interent applications / I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan // Proceedings of the ACM SIGCOMM '01 Conference – San Diego, California – August 2001 – P. 149–160.

8. Antony I., Rowstron T., Druschel P. Pastry: Scalable, decentralized object location, and routing for large-scale P2P systems // Middleware. – 2001. – P. 329–350.

9. El Dick M., Pacitti E., Kemme B. A highly robust p2p-cdn under large-scale and dynamic participation in Advances in P2P Systems, 2009 // AP2PS '09. First International Conference on 2009 – 2009 – P. 180–185.

10. Freedman M. J., Freudenthal E., Mazieres D. Democratizing content publication with Coral // In Proceedings of 1st USENIX/ACM Symposium on Networked Systems Design and Implementation – San Francisco, CA – March 2004. – P. 239–252.

11. Maymounkov P., Mazieres D. Kademlia: A peer-to-peer information system based on the xor metric // In IPTPS – Cambridge, MA – Mar 2002. – P. 53–65.

12. Wide-area cooperative storage with CFS / F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, I. Stoica // In SOSP – Banff, Canada – Oct. 2001 – P. 202–215.

13. OceanStore: An architecture for globalscale persistent storage / J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao // In ASPLOS – Cambridge, MA – Nov. 2000. – P. 190–201.

14. Rowstron A., Druschel P. Storage management and cachingin PAST, a large-scale, persistent peer-to-peer storage utility // In SOSP – Banff, Canada – Oct. 2001. – P. 188–201.

15. Crawford M. Non-Terminal DNS Name Redirection // RFC 2672 – 1999. – 243 p.

16. Identifying Performance Bottlenecks in CDNs through TCP-Level Monitoring / P. Sun, M. Yu, M.J. Freed-man, J. Rexford // ACM SIGCOMM Workshop on Measurements Up the Stack – August 2011. – P. 49–54.

17. Wendell P., Freedman M.J. Going viral: flash crowds in an open CDN // ACM Internet Measurement Conference (IMC) – 2011 – P. 549–558.

METHODS OF THE CONSTRUCTION OF CONTENT DELIVERY NETWORKS BASED ON DISTRIBUTED HASH TABLES

A. Serhieiev, K. Ilchenko

Taras Shevchenko National University of Kyiv

vul. Lomonosova, 81, Kyiv, 03022, Ukraine. E-mail: anton.serhieiev@gmail.com

Purpose. Goal of this article is to improve the quality of existing models of hybrid content delivery networks through comparing theirs architecture and finding advantages and disadvantages. **Results.** The basic problems in modern computer networks under heavy loads, as well as in cases of distribution of dynamic content were described. The expediency of creating a hybrid content delivery systems that are based on distributed hash tables, systems that combine the advantages of classical content delivery systems and distributed hash tables, but the implementation of which is extremely complex, was shown. The existing theoretically developed and practically tested models of hybrid content delivery networks were researched. Advantages and disadvantages of these systems were highlighted and their comparison was carried out. The perspectives of implementation of theoretically developed models were considered. **Originality.** For the first time full comparison of existing hybrid content delivery networks and their prospects for implementation were carried out. **Conclusions.** The analyses and research of existing hybrid content delivery networks shows the critical shortcomings, which interfere comfortable use of this systems. Further development of new networks or upgrade of existing, needs effective mathematical model, which currently is absent and which creation needs subsequent research. References 17, table 1.

Key words: decentralized network, overlay, distributed hash table, content delivery network.

REFERENCES

1. Kim, T. N., Jeon, S., Kim, Y. (2011), "A CDN-P2P Hybrid Architecture with Content/Location Awareness for live Streaming Service Networks," *Proc. IEEE 15th Int. Symposium on Consumer Electronics*, pp. 438–441.

2. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S. (2001) "A scalable content-addressable network," *Proc. of ACM SIGCOMM 2001*, pp. 161–172.

3. Ratnasamy, S., Handley, M., Karp, R., Shenker, S., (2002), "Topologically-aware overlay construction and server selection," *Proc. of IEEE INFOCOM*, pp. 1190–1199.

4. El Dick, M., Pacitti, E., Kemme, B. (2009), "FlowerCDN: a hybrid P2P overlay for efficient query processing", *In CDN‖, ACM EDBT 2009*, pp. 427–438.

5. Serhieiev, A. (2016), "Technologies of the data retrieval in peer-to-peer networks constructed on the basis of distributed hash tables", *Control System and Computers*, no. 2 (2016), pp. 41–48.

6. Ratnasamy S., Handley M., Karp R., Shenker, S. (2002), "Mechanisms of protection of information in networks, based on distributed hash tables," *Proc. INFOCOM'02, New York, USA*, pp. 32–36.

7. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H. (2001), "Chord: A scalable p2p lookup service for interent applications", *Proc. of SIGCOMM'01*, pp. 149–160.

8. Antony, I., Rowstron, T. Druschel, P. (2001), "Pastry: Scalable, decentralized object location, and routing for large-scale P2P systems", *Middleware*, 2001, pp. 329–350.

9. El Dick, M., Pacitti, E., Kemme, B. (2009), "A highly robust p2p-cdn under large-scale and dynamic participation," *Advances in P2P Systems, 2009. AP2PS* '09. First International Conference, pp. 180–185.

10. Freedman, M. J., Freudenthal, E., Mazieres, D. (2004), "Democratizing content, publication with Coral", *NSDI*, pp. 239–252.

11. Maymounkov, P., Mazieres, D. (2002), "Kademlia: A peer-to-peer information system based on the xor metric", *Lecture Notes in Computer Science*, vol. 2429, pp. 53–65.

12. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I. (2001), "Wide-area cooperative storage with CFS", *SOSP*, pp. 202–215.

13. Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B. (2000), "OceanStore: An architecture for globalscale persistent storage», *ASPLOS*, vol. 35, iss. 11, pp. 190–201.

14. Rowstron, A., Druschel, P. (2001), "Storage management and cachingin PAST, a large-scale, persistent peer-to-peer storage utility", *SOSP '01 Proceedings of the eighteenth ACM symposium on Operating systems principle*s, pp.188–201.

15. Crawford, M. (1999*), Non-Terminal DNS Name Redirection*, RFC 2672, Fermilab, Batavia, USA, 243 p.

16. Sun, P., Yu, M., Freedman, M. J., Rexford, J. (2011), "Identifying Performance Bottlenecks in CDNs through TCP-Level Monitoring", *SIGCOMM Workshop on Meas. Up the Stack*, pp. 49–54.

17. Wendell, P., Freedman, M. J. (2011), "Going viral: flash crowds in an open CDN," *ACM Internet Measurement Conference (IMC)*, pp. 549–558.

Стаття надійшла 30.08.2016.