Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
(повна назва інституту/факультету)

Кафедра телекомунікацій
(повна назва кафедри)

«На правах рукопису»
УДК _____

До захисту допущено
В.о. завідувача кафедри

_____ Явіся В.С.___
(підпис) (ініціали, прізвище)

“___”_____2018_р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 172 Телекомунікації та радіотехніка,
(код і назва)

спеціалізація Мобільні телекомунікації

на тему: Алгоритми балансування навантаження в мережі доставки контенту.

Виконав: студент _2_ курсу, групи ТМ-71мп
(шифр групи)

Корнійчук Юрій Володимирович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник _професор, д.т.н. Лисенко О.І. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018_ рік

ЗМІСТ

Оглавление

Анотація

С збільшенням кількості користувачів проблема доставки об'ємного контенту в інтернеті стає все більш актуальною. Виникає необхідність розташовувати сервера з даними якомога ближче до користувачів, для зменшення затримок і зниження навантаження на магістральні канали. Особливо це актуально для контенту,
який потрібно одночасно роздати велику кількість користувачів.

Таким чином великим постачальникам контенту потрібно розподілена мережа серверів, звана Мережею Доставки Контенту (англ. Content Delivery Network - CDN). При побудові CDN можуть виникати різні проблеми - оптимізація трафіку всередині CDN, оптимізація розподілу контенту по серверам.

Метою магістерської дисертації є вивчення та розробка алгоритмів розподілу трафіку всередині CDN на основі модельної мережі доставки контенту. Розроблені алгоритми повинні дозволити знизити навантаження на мережеву інфраструктуру всередині CDN.

Вступ

Мережі доставки контенту (CDN) - активно розвивається напрямок. Все більше інтернет-компаній потребують інструмент, що дозволяє швидко і надійно доставляти інформацію клієнту.

На швидкість завантаження веб-сторінки і її вмісту сильно впливає віддаленість

користувача від сервера. Це трапляється тому, що при застосуванні

технології TCP / IP, яка використовується для поширення інформації в мережі

Інтернет, задержки при передачі інформації залежать від

чисельності маршрутизаторів, що знаходяться на маршруті між джерелом

споживачем контенту. Розташування контенту між кількома серверами

засобами CDN зменшує мережевий маршрут передачі даних і робить завантаження

сайту скоріше з точки зору користувача.

Використання CDN знижує кількість переходів між вузлами мережі, що істотно збільшує швидкість скачування контенту з мережі Інтернет. Кінцеві користувачі почувають меншу затримку при завантаженні контенту, відсутність різких змін швидкості завантаження і високу добротність потоку даних. Пропускна здатність дозволяє операторам CDN доставляти відеоконтент високої роздільної здатності (1080p, 4K), забезпечувати швидке завантаження файлів великих розмірів або створювати відеовещанія з високою якістю сервісу (QoS) і низькими витратами на мережу.

Технологія CDN здатна запобігти затримки при передачі даних, можливі переривання зв'язку і втрати на перевантажених каналах і стиках між ними. Керування навантаженням при передачі мережевого трафіку дозволяє розгрузити великі сполучні канали і вузли мережі, розподіливши виникає навантаження між розділеними серверами.

Розташування серверів в прямій близькості від кінцевих користувачів може підвищити вихідну пропускну спроможність всієї системи. Зокрема, наявність одного порту 100 Мбіт / с не має на увазі дану швидкість на всіх ділянках мережі, так як вільна пропускна здатність сполучного каналу в момент передачі може бути 10 Мбіт / с. В тому випадку, коли використовуються 10 розділених серверів, сумарна пропускна здатність може скласти 10 × 100 Мбіт / с.

Сучасні мережі доставки і дистрибуції контенту спроможні створювати автоматичний контроль цілісності даних на кожному з серверів мережі. При цьому гарантується 100% відкритість контенту для кінцевого користувача в разі втрати зв'язку між вузлами мережі, поламки центрального або віддаленого сервера.

Найбільш розвинені CDN дають статистичний контроль процесів доставки та розповсюдження контента. Контент-провайдер в режимі онлайн може отримати всю необхідну інформацію про завантаження, доступності і популярності свого контенту в кожному регіоні присутності.

Постановка задачі

Метою магістерської дисертації є дослідження існуючих рішень в області мереж доставки контенту.

На основі проведеного дослідження запропонувати модель розподіленої мережі доставки контенту

С використанням моделі розробити алгоритми управління трафіком всередині мережі CDN для мінімізації навантаження на канал.

Ключові особливості

- Розподілена мережа граничних серверів
- Один або кілька серверів, на яких розташовується інтерфейс управління. За допомогою цього інтерфейсу можна вручну управляти маршрутизацією в мережі і розподілом копій контенту.
- Додаток-агент на граничних серверах (модуль системи, який встановлюється на кожен з серверів, з яких будуть віддаватися дані). Дозволяє управляти маршрутизацією в мережі, управляти локальними копіями контенту, проводити моніторинг стану мережі.
- Веб-інтерфейс підтримує різні способи додавання файлів в CDN
- Два рівня балансування - DNS і HTTP

Дослідження предметної області

Побудова мережі доставки контенту - велика задача. Навіть при розробці модельної мережі використовується велика кількість різних технологій. У наступних розділах розглядаються існуючі мережі доставки контенту, а також технології, які найбільш часто використовується при вирішенні подібних завдань. Також розглянуті дослідження в області розподілу запитів всередині мережі доставки контенту.

В наступних розділах з використанням розглянутих технологій і на підставі проаналізованих алгоритмів буде розроблена модельна мережу доставки контенту і запропоновано алгоритми балансування і розподілу всередині мережі.

1. Аналіз існуючих мереж доставки контенту і готових рішень для їх створення

1.2 Akamai CDN

The Akamai Network [2] [11] - одна з найбільших в світі розподілених обчислювальних платформ. Це мережа з більш ніж 105,000 серверів із спеціальним програмним забезпеченням, розташованих в 78 країнах. Сильною стороною мережі Akamai є алгоритми розподілу навантаження всередині мережі. Сервери знаходяться приблизно в 1,900 різних мережах, здійснюючи моніторинг Інтернет в реальному часі - збираючи інформацію про трафік, перевантаженнях, і проблемних точках. Akamai використовує зібрану інформацію для оптимізації маршрутів і динамічної реплікації, щоб доставляти контент швидше і надійніше.

Основні способи оптимізації доставки контенту:

- Мінімізація довжини маршрутів, за допомогою реплікації і доставки контенту з серверів, що знаходяться максимально близько до користувача.
- Оптимізація маршрутів шляхом побудови шляхів через Інтернет, що обходять проблемні місця, стиснення контенту, і реплікації пакетів.
- Переміщення обчислень ближче до кінцевих користувачів, щоб уникнути затримок. (EdgeComputing). [1] [8]

1.3 Amazon CloudFront

Amazon CloudFront - веб-сервіс для доставки контенту. Amazon CloudFront інтегрується з іншими Amazon Web Services. Суть сервісу - надати розробникам і підприємствам простий спосіб поширювати контент для останніх користувачів з найменшими затримками, підвищенною швидкістю передачі даних.

Данний сервіс не є вільним для користування і входить в інфраструктуру сервісів Amazon Web Services. [25]

1.4 CacheBoy

Cacheboy надає відкриту платформу для здійснення ефективної доставки контенту. Дзеркальна інфраструктура відкритого проекту підтримується сервіс-провайдерам по всьому світу. Проект перенаправляє користувача на найближчий дзеркало. Cacheboy поширюється під ліцензією GPLv3.

1.5 OpenCDN

OpenCDN - CDN рівня додатки, який реплицирует контент і розділяє трансляції та записаний контент. OpenCDN написаний на Perl і використовує технологію Relay для поділу медіа пакетів. Зв'язок між вузлами і джерелами здійснюється по засобом XML - RPC. Поширюється під ліцензією Perl Artistic. [18]

1.6 P2P-Next

P2P-Next - програмне забезпечення наступного покоління для доставки контенту, яке може бути використане для одночасної передачі мільйонам людей. Воно використовує мультікастінга для передачі даних мільйонам користувачів одночасно. Потік даних розподіляється по локальних серверів і прямий ефір може бути ретранслювати локальним глядачам. Так як IP роутери не підтримують мультікастінга, оптимізовані механізми uni-cast, broadcast і multicast були адаптовані для P2P.

1.7 CloudFlare Railgun

Один із шляхів зниження навантаження всередині CDN - кешувати якомога більше контенту. Але в реальності тримати в кеші можна тільки близько 66% об'єктів, а решта 34% припадає заново запитувати з сервера в разі поновлення. Для стиснення цього трафіку і був створений Railgun.

В Railgun використовуються приблизно такі ж алгоритми, як при обробці послідовності кадрів в відео високої чіткості. Високий ступінь стиснення в відеокодек досягається за рахунок стиснення не окремих кадрів, а відмінностей між сусідніми кадрами. Це дозволяє стиснути кадр розмірів мільйони пікселів в кілька кілобайт. Теоретично можна його стиснути взагалі в один байт, якщо він нічим не відрізняється від попереднього кадру. При зміні веб-сторінки змінюється тільки невелика частина вмісту, і досить передати зміна між актуальною версією і тієї, яку клієнт отримав в попередній раз.

Висновок до розділу

Отже, основні способи оптимізації доставки контенту:

- Мінімізація довжини маршрутів, за допомогою реплікації і доставки контенту з серверів, що знаходяться максимально близько до користувача.
- Оптимізація маршрутів шляхом побудови шляхів через Інтернет, що обходять проблемні місця, стиснення контенту, і реплікації пакетів.
- Переміщення обчислень ближче до кінцевих користувачів, щоб уникнути затримок. (EdgeComputing).

Технічно, Railgun складається з двох компонентів: відправника (sender) і одержувача (listener). Відправники встановлені в кожному дата-центрі CDN-мережі. Одержувач - програмний компонент, який встановлюється на прикордонні вузли мережі. Між відправником і отримувачем встановлюється TCP-з'єднання, захищене TLS, за яким бінарний протокол Railgun здійснює асинхронну передачу HTTP-запитів. Для веб-клієнта система Railgun виглядає як проксі-сервер, хоча насправді це спеціалізована система зі специфічними функціями. Одна з них - стиснення контенту, який неможливо кешувати, за рахунок синхронізації версій сторінок. При оновленні версії сторінки по мережі передається тільки зміна між попередньою і новою версією.

Дослідження показали, що максимальне стиснення досягається на новинних сайтах з великою відвідуваністю. Наприклад, бінарне порівняння головної сторінки сайту reddit.com показує зміну в середньому 2,15% протягом 5 хвилин і 3,16% протягом години. Головна сторінка The New York Times бінарному змінюється на 0,6% за п'ять хвилин і на 3% протягом години. Для головної сторінки BBC News ці показники становлять 0,4% і 2%, відповідно. У загальному випадку, для найпопулярніших сайтів при повторному запиті змін різниця займає один пакет TCP.

2.Огляд технологій використовуваних для побудови CDN

При побудові мережі доставки контенту необхідно вирішити такі основні завдання:

- Віртуалізація мережевого рівня
- Контроль цілісності мережі
- Управління прикордонним трафіком

2.1 Програмно-конфігуровані мережі (SDN)

Для полегшення масштабованості мережевої інфраструктури CDN можливо використовувати принципи програмно-конфігуруються мереж SDN (Software Defined Networks). [9] [12

В SDN рівні управління мережею і передачі даних поділяються за допомогою перенесення функцій керування (комутаторами, маршрутизаторами і т. п.) в додатки, що працюють на окремому сервері (контролері). Данна ідея була сформульована фахівцями університетів Стенфорда і Берклі ще в 2006 році, а запроваджені ними дослідження були підтримані не тільки в академічних кругах, але і були активно прийняті передовими виробниками мережевого обладнання, що створили в березні 2011 року консорціум Open Networking Foundation (ONF). Його творцями були Google, Deutsche Telekom, Facebook, Microsoft, Verizon і Yahoo. Склад ONF швидко розширюється, в неї вже увійшли такі компанії, як Citrix, Brocade, Dell, Oracle, Ericsson, IBM, HP, Marvell, NEC і ряд інших. Однією з перших практичну реалізацію SDN запропонувала компанія Nicira, яка увійшла недавно до складу VMware. [

Зацікавленість IT-компаній в SDN пояснюється тим, що дані технології дозволяють підняти ефективність мережевого обладнання на 25-30%, знизити на 30% витрати на експлуатацію мереж, переробити управління мережами з

мистецтва в інженерію, підняти рівень безпеки та надати користувачам можливість

програмно створювати нові сервіси та оперативно завантажувати їх в мережеве обладнання. [27] [28]

Основні розробки в області SDN ведуться учасниками програми GENI (Global Environment for Network Innovations) дослідження майбутнього Інтернету, силами об'єднаного центру Стенфорда і Берклі (Open Network Research Center), що виконує дослідження і розробки в області Internet2; а також з Сьомої рамкової програми досліджень Європейського СоюзуOfelia і проектом FEDERICA.

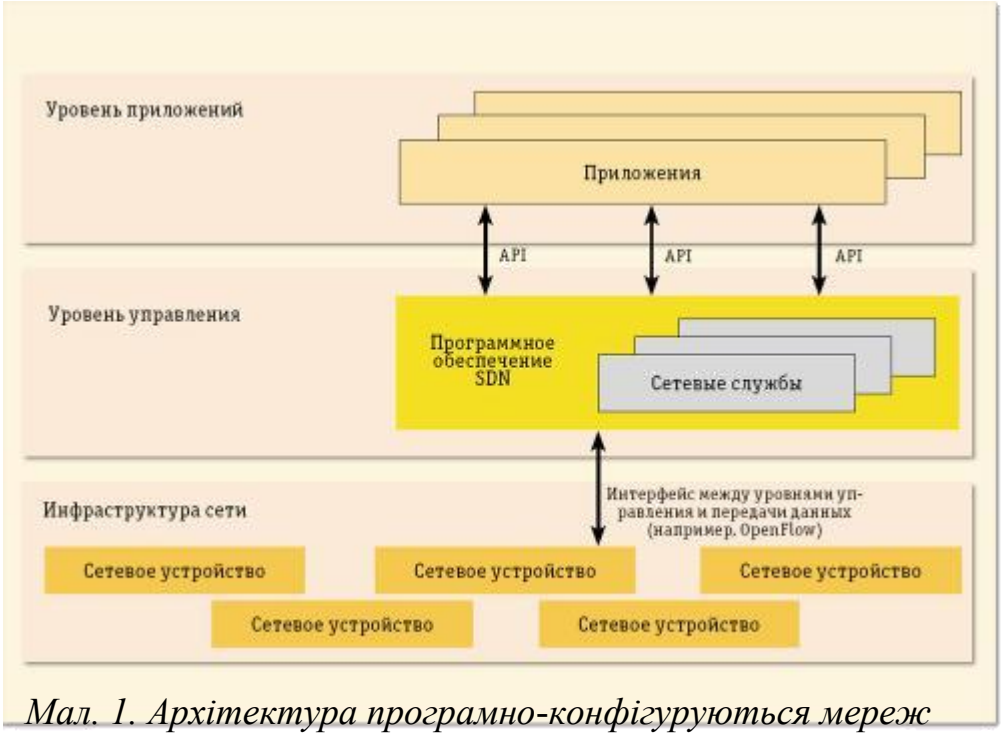
Основні завдання SDN:

- розподіл процесів передачі та керування даними;
- єдиний, унікальний, незалежний від постачальника інтерфейс між рівнями управління і передачі даних;
- логічно централізоване управління мережею, здійснюване з допомоги контролера з встановленої мережевою операційною системою і реалізованими поверх мережевими додатками;
- віртуалізація фізичних ресурсів мережі.

2.1.1Архітектура SDN

В архітектурі SDN можна виділити три рівні:

1. *рівень інфраструктури*, що надає набір мережевих пристроїв (каналів передачі даних і комутаторів);
2. *рівень управління*, що включає в себе мережеву операційну систему, яка виконує забезпечення додаткам мережеві сервіси та програмний інтерфейс для управління мережевими пристроями і мережею;
3. *рівень мережевих додатків* для гнучкого й ефективного управління мережею.

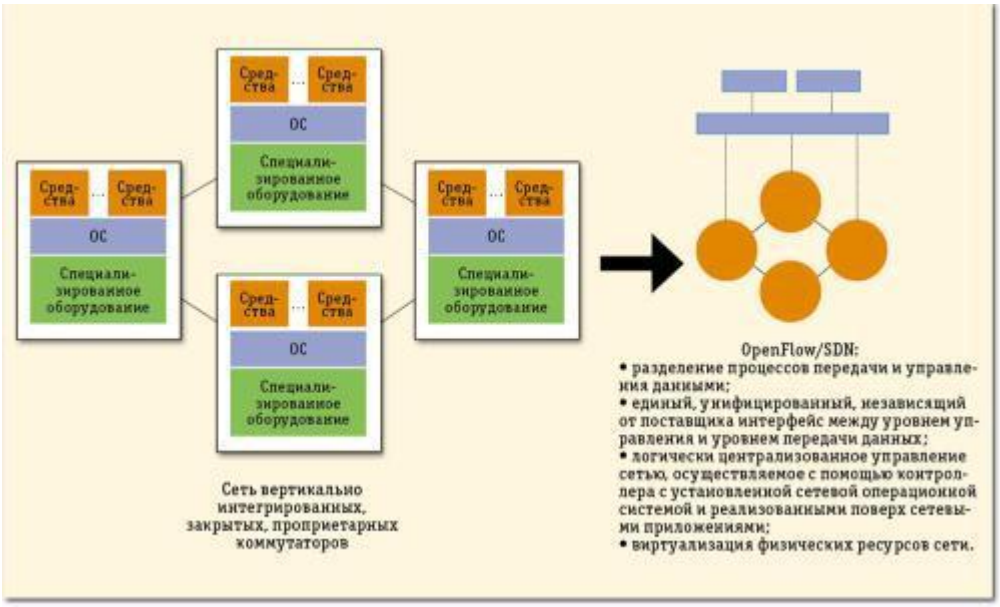


Мал. 1. Архітектура програмно-конфігуруються мереж

Найбільш перспективним і активно розвиваються стандартом для SDN є OpenFlow - відкритий стандарт, в якому описуються вимоги, що пред'являються до комутатора, що підтримує протокол OpenFlow для віддаленого управління.

С допомогою сучасних маршрутизаторів зазвичай вирішуються дві основні задачі: передача даних (forwarding) - просування пакета від вхідного порту на певний вихідний порт і управління даними - обробка пакету і прийняття рішення про те, куди його передавати далі, на основі поточного стану маршрутизатора. Це відповідає рівню передачі даних, на якому зібрані кошти передачі (лінії зв'язку, каналоутворювального обладнання, маршрутизатори, комутатори), і рівнем управління станами засобів передачі даних. Розвиток маршрутизаторів досі йшло по шляху зближення цих рівнів, проте з ухилом на передачу (апаратне прискорення, вдосконалення ПО і впровадження нових функціональних можливостей для збільшення швидкості прийняття рішення по маршрутизації кожного пакета), тоді як рівень управління залишався досить примітивним і спирався на

складні розподілені алгоритми маршрутизації і хитромудрі інструкції по конфігурації і налаштування мережі. Зрозуміло, ПО маршрутизаторів, що реалізує рівень управління, було пропрієтарним і закритим.



Мал. 2. Традиційні мережі і SDN

Згідно зі специфікацією 1.3 стандарту OpenFlow, взаємодія контролера з комутатором виконується при допомозі протоколу OpenFlow - кожен комутатор має містити одну або декілька таблиць потоків (flow tables), групову таблицю (group table) та підтримувати канал (OpenFlow channel) для зв'язку з віддаленим контролером - сервером. Специфікація не регламентує архітектуру контролера і API для його додатків. Кожна таблиця потоків в комутаторі має певний перелік записів (flow entries) про потоки або правила. Кожен такий запис складається з полів-ознак (match fields), лічильників (counters) і інструкції (instructions).

Механізм роботи комутатора OpenFlow досить простий. У кожного пакету, що прийшов «вирізується» заголовок (бітовий рядок певної довжини). Для цього бітового рядка в таблицях потоків, починаючи з першого, вишукується правило. Знайдене правило повинно максимально близько відповідати заголовку пакета. При знаходженні збігу, над пакетом і його заголовком виконуються деякі перетворення, які класифіковані інструкцією, зазначеною

в знайденому правилі. Інструкції пов’язані з кожним записом таблиці, відображають дії, пов'язані з передачею пакета, видозміною його заголовка, обробкою в таблиці груп, обробкою в конвеєрі і передачею пакета на певний порт комутатора. Інструкції конвеєра обробки дають змогу надсилати пакети в слідуючі таблиці для подальшої обробки і у вигляді метаданих пересилати інформацію між таблицями. Інструкції також показують нам правила модифікації лічильників, які можуть бути застосовані для збору різних видів даних.

Якщо необхідного правила в першій таблиці не знайдено, то пакет інкапсулюється та відправляється контролеру, котрий формулює відповідне правило для пакетів даного типу і налаштовує його на комутаторі (або на наборі керованих ним комутаторів), або пакет може бути скинутий (в залежності від властивостей комутатора).

Запис про потік може давати вказівку про пересилання пакета в потрібний порт (звичайний фізичний порт або віртуальний, призначений комутатором, або зарезервований віртуальний порт, установлений специфікацією протоколу). Резервні віртуальні порти можуть визначити загальні дії пересилання, а саме - пересилання контролера, широковещательная (лавинна) розсилка, відправка без OpenFlow. Віртуальні порти, певні комутатором, можуть точно визначати групи агрегування каналів, тунелі або інтерфейси зі зворотним зв'язком.

Записи про потоки можуть також указувати на групи, в яких висвітляється додаткова обробка. Групи є наборами дій для багатомовної розсилки, а також набори дій надсилання з скалднішою семантикою, наприклад швидка видозміна маршруту або агрегування каналів. Механізм груп надає право ефективно міняти загальні вихідні дії для потоків. Таблиця груп зберігає записи про групи, що містять список контейнерів дій з особливою семантикою, що має залежність від типу групи. Дії в одному або багатьох контейнерах дій вживаються до пакетів, що надсилаються в групу.

Розробники комутаторів можуть бути вільні у реалізації їх внутрішньої начинки, однак процедура перегляду пакетів і семантика інструкцій повинні бути для всіх однакові. Наприклад, в той час як потік може використовувати всі групи для пересилки в деякий безліч портів, розробник комутатора може вибрати для реалізації цього єдину бітову маску всередині апаратної таблиці маршрутизації. Інший приклад - це процедура перегляду таблиць: конвеєр фізично може бути реалізований за допомогою різної кількості апаратних таблиць. Встановлення, видалення та оновлення правил в таблицях потоків комутатора виконуються контролером. Правила можуть визначатися реактивно (у відповідь на що прийшли пакети) або проактивно (заздалегідь, до приходу пакетів).

2.1.2 Openflow

Openflow (відкритий потік)[32] - протокол (і технологія) управління процесом обробки даних, що передаються по комп'ютерах мережі маршрутизаторами і комутаторами.

Управління даними в OpenFlow здійснюється не на рівні окремих пакетів,

а на рівні їх потоків. Правило в комутаторі OpenFlow встановлюється за участю контролера тільки для першого пакету, а потім всі інші пакети потоку його використовують.

Наявні на сьогоднішній день фізичні комутатори SDN відповідають поки специфікації OpenFlow 1.0 і містять тільки одну таблицю потоків.

Протокол використовується для управління мережевими комутаторами

(Маршрутизаторами) з центрального пристрою - контролера мережі (наприклад, з сервера або навіть персонального комп'ютера). Це управління замінює або доповнює собою працюючу на комутаторі (маршрутизаторі) пропрієтарних програму (яка здійснює побудова маршруту, створення карти

комутації і т. Д.). Контролер застосовується для керування таблицями потоків комутаторів, на підставі яких впроваджується рішення про трансфер прийнятого пакета на певний порт комутатора. Таким чином в мережі утворюються прямі мережеві з'єднання з мінімальними затримками передачі даних і необхідними параметрами

Версії мікропрограм з підтримкою Openflow розроблені для пристроїв багатьох виробників, включаючи Cisco, Juniper, HP, IBM, NEC. [1]

На даний момент протокол має версію 1.2 (прийнято 5 грудня 2011 року).

2.2Архітектура

Інформація про шляхи проходження даних (datapath) представляє з себе таблиці потоків (flow table) і дій, призначених для кожного запису. Самі таблиці можуть ставитися як до Ethernet (або інших протоколів канального рівня), так і к іншим протоколам вищих рівнів (IP, TCP). Точний список дій може варіюватися, але основні це: перенаправлення (пересилання PDU (пакета, фрейма) в заданий порт), пересилання PDU на контролер через безпечний канал для подальшого дослідження, відкидання PDU (drop). Для пристроїв, які суміщають Openflow і звичайну обробку пакетів засобами вбудованого пристрою, додається четвертий тип дії: обробка PDU 'звичайними' засобами. Устаткування, що підтримує ці чотири дії є пристроями першого типу (Type0).

Пристрій OpenFlow складається, як мінімум, з 3 компонентів:

- таблиці потоків (англ. flow table);
- захищенного каналу (англ. secure channel), що створюється для управління комутатором зовнішнім «інтелектуальним» пристроєм (контролером);
- Підтримки протоколу OpenFlow protocol, що використовується для управління. Використання цього протоколу дозволяє уникнути необхідності писати програму для керованого пристрою;

Кожен запис в таблиці потоків має три поля: заголовок PDU, який дозволяє визначити відповідність PDU потоку, дія і поле зі статистикою

(Число байтів і PDU, відповідне потоку, час, проходження останнього відповідного потоку PDU).

Тема може складатися з безлічі полів різного рівня (наприклад, MAC-адрес відправника і одержувача, полів з заголовка IP-пакета, полів з заголовка TCP-сегмента). Кожне поле може мати особливе значення (зірка), що означає відповідність будь-якому значенню відповідного поля в PDU. [24]

Пристрої другого типу (Type1), які будуть забезпечувати трансляцію адрес (NAT), підтримку класів і пріоритетів, заплановані, але їх специфікація поки не визначена.

Контролери забезпечують наповнення таблиці потоків, отримання пакетів через безпечний канал від пристрою. Вони можуть бути реалізовані як найпростіший алгоритм, що нагадує поведінку комутатора, що розділяє пакети по віртуальним мережам. За допомогою контролерів можливо реалізовувати складну динамічну логіку, яка впливає на проходження пакетів виходячи з зовнішніх чинників (права доступу, завантаження серверів, пріоритети по обслуговуванню і т. Д.).

2.3Мережева операційна система

Логічно-централізоване управління даними в мережі передбачає винесення всіх функцій управління мережею на окремий фізичний сервер, званий контролером, який знаходиться у віданні адміністратора мережі. Контролер може управляти як одним, так і декількома OpenFlow-комутаторами і містити мережеву операційну систему, яка додає мережеві сервіси по низкорівневому управлінню мережею, сегментами мережі і станом мережевих елементів, а також додатки, які виконують високорівневе управління мережею і потоками даних.

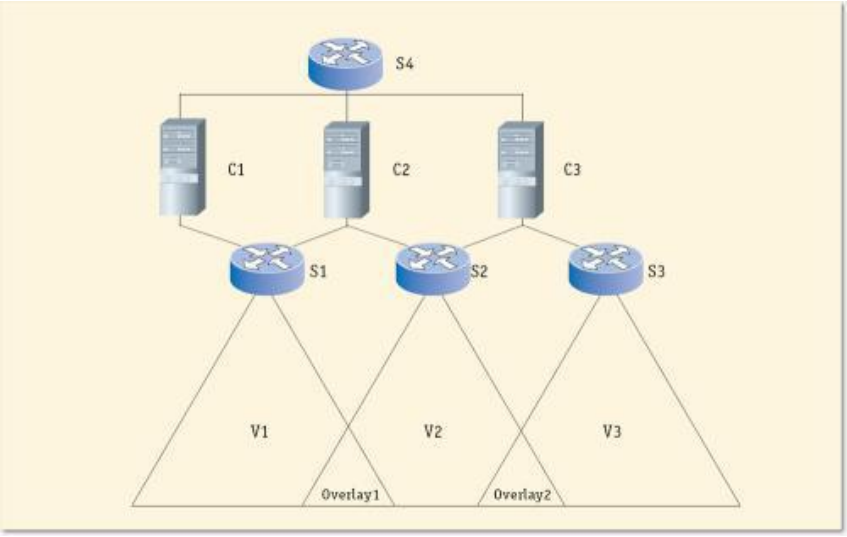
Мережева ОС (СОС) надає додаткам доступ до керування мережею і безперервно слідкує за конфігурацією засобів мережі. На відміну від традиційного тлумачення терміна ОС, під СОС мається на увазі програмна система, що виконує моніторинг, доступ і управління ресурсами всієї мережі, а не її певного вузла.[13]

Подібно традиційної операційній системі, СОС надає програмний інтерфейс для додатків керування мережею і відтворює механізми керування таблицями комутаторів: додавання, видалення, видозмінення правил і збір різноманітного роду статистики. Отже, вирішення задач управління мережею відбувається за допомогою додатків, виконаних на основі API мережевої операційної системи, що дають змогу створювати додатки в термінах високорівневих абстракцій (наприклад, ім'я користувача та ім'я хоста), а не низькорівневих параметрів конфігурації (наприклад, IP- і MAC адрес). Це надає змогу виконувати керуючі команди попри базову топології мережі, однак вимагає, щоб СОС підтримувала відображення між високорівневими абстракціями і низькорівневими конфігураціями.

В кожному контролері є хоча б один додаток, який керує комутаторами, напряму з'єднаними з цим контролером, і формулює уявлення про топології фізичної мережі, що знаходиться під керуванням контролера, тим самим робить управління центральним. Подання топології мережі має в собі

топологію комутаторів, розташування користувачів і хостів та інших елементів і обслуговування мережі. Подача також містить в собі прив'язку між іменами та адресами, ось чому однією з найважливіших завдань, що вирішуються СОС, є постійний моніторинг мережі. Таким чином, СОС дозволяє створювати додатки у вигляді централізованих програм, що використовують високорівневі імена, на основі таких алгоритмів, як, наприклад, алгоритм Дейкстри пошуку найкоротшого шляху в графі, замість складних розподілених алгоритмів на кшталт алгоритму Беллмана - Форда, в термінах низькорівневих адрес, які використовуються в сучасних маршрутизаторах.

Для контролерів в SDN дуже значущим є вимога того, щоб усі програми одного контролера в кожен момент часу мали ідентичне уявлення про топологію мережі. Однак перехід від розподіленого управління мережею до централізованого має і ряд недоліків. Наприклад, зниження надійності, відмовостійкості, масштабованості.



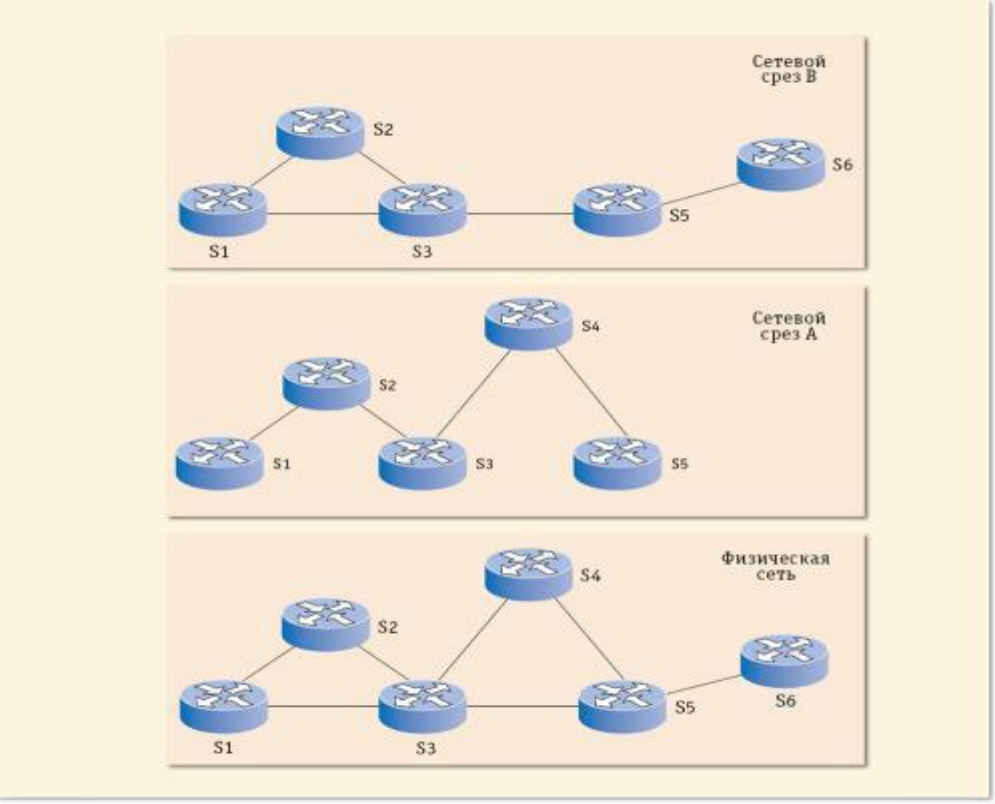
Мал. 3. Альтернативний підхід до побудови розподіленого масштабованого контролера

Сьогодні отримали розвиток кілька підходів до побудови розподіленого масштабованого контролера: HyperFlow, Onix і Kandoo.[27] Однак, найбільш перспективним вважається альтернативний підхід (рис. 3)[28].

Оскільки кожен контролер може бути з'єднаний з декількома комутаторами, а кожен комутатор - з декількома контролерами, то контролери, що управляють одним і тим же комутатором, можна об'єднати в груповий контролер (ГК). Всі контролери одного і того ж ГК повинні мати узгоджену уявлення о топології тієї частини мережі, до якої вони забезпечують доступ. Як видно з рис. 3, C1 - C3 - контролери, S1 - S4 - комутатори, а V1 - V3 - фрагменти мережі, до яких забезпечує доступ комутатор S1, S2, S3 відповідно. Тоді ГК1 утворюють контролери C1 і C2, ГК2 - C2 і C3, а всі програми в ГК1 повинні мати узгоджену уявлення про топології V1 і V2, всі програми в ГК2 – о топології V2 і V3. У разі виходу з ладу, наприклад, контролера C1 його може замінити C2, взявши на себе управління V1. Подання про стан відповідної частини мережі контролери можуть погоджувати або через комутатор S4, або через S1, S2 і S3.

Такий підхід до побудови розподіленого контролера вирішує проблему масштабованості і підвищує відмовостійкість SDN.

2.4Віртуалізація в SDN



Мал. 4. Віртуалізація в SDN

Одна з ідей, стрімко розгортається в рамках SDN, - це віртуалізація мереж з метою більш ефективного використання мережевих ресурсів (рис. 4). Під віртуалізацією мережі мається на увазі відокремлення мережевого трафіку - угруповання (мультиплексування) кількох потоків даних з несхожими показниками в рамках однієї логічної мережі, яка може ділити єдину фізичну мережу з іншими логічними мережами або мережевими кусками (network slices). Кожен такий «кусок» має право використовувати власну адресацію, власні алгоритми маршрутизації, керування якістю сервісів і т. д.

Віртуалізація мережі дає наступні переваги: підвищення ефективності розподілу мережевих ресурсів і збалансування навантаження на них; ізолювання потоків різних користувачів і додатків коло однієї фізичної мережі; адміністраторам різних «кусків» використання своєї політики маршрутизації та правил керування потоками даних; проведення експериментів у мережі, застосовуючи справжню фізичну мережеву

інфраструктуру; застосування в кожному «куску» тільки тих сервісів, які необхідні конкретним додаткам

Одним із прикладів віртуалізації ресурсів SDN, поділу мережі на зрізи і управління ними є FlowVisor - програма-посередник (proxy), що діє на рівні між OpenFlow- комутаторами і різними контролерами SDN. За допомогою FlowVisor можна створювати логічні сегменти мережі, що використовують різні алгоритми управління потоками даних, забезпечуючи ізоляцію даних мереж друг від друга. Це означає, що кожен контролер управляє тільки своєю логічною мережею і не може впливати на функціонування інших. Для контролера, що взаємодіє з обладнанням OpenFlow через FlowVisor, весь обмін повідомленнями виглядає так само, як якби контролер взаємодіяв зі звичайною мережею SDN. Всю необхідну модифікацію повідомлень, що вимагається для підтримки різних ізольованих сегментів мережі, виконує FlowVisor.[19]

Переваги SDN

Завдяки зняттю з комутаторів навантаження по обробці тракту управління, SDN дозволяє цим пристроям направити всі свої ресурси на прискорення переміщення трафіку, що істотно підвищує продуктивність. При цьому за рахунок віртуалізації управління мережею знижуються витрати на їх побудову і супровід. За результатами тестів, проведених на базі найбільших провайдерів США, використання SDN дозволяє на 20-30% збільшити утилізацію ресурсів ЦОД і в кілька разів знизити експлуатаційні витрати.

Програмні засоби SDN дозволяють адміністраторам додавати нову функціональність до вже наявної мережевої архітектури. При цьому нові функції будуть працювати на багатьох платформах - їх не доведеться реалізовувати заново у вбудованому програмному забезпеченні комутаторів кожного постачальника.

На централізованому контролері SDN системний адміністратор може спостерігати всю мережу в єдиному поданні, за рахунок чого підвищуються

зручність управління, безпеку і спрощується виконання ряду інших завдань. Дійсно, оскільки адміністратор бачить все потоки трафіку, то йому легше помічати вторгнення, призначати пріоритети різним типам трафіку і розробляти правила реагування мережі при заторах і проблеми з обладнанням.

Теоретично необмежені можливості мереж SDN до розширення дозволяють будувати реальні хмари, масштабовані в залежності від розв'язуваних завдань. При цьому мережа володіє необхідною «інтелектуальністю», необхідної, зокрема, для керування роботою великих груп комутаторів.

2.5Розподіл контенту всередині CDN

Окремою проблемою при побудові мережі доставки контенту є поширення даних усередині мережі. Дані завантажуються на один з вузлів мережі і потім повинні бути розподілені по іншим вузлам, або по деякому їх підмножині. При великій кількості вузлів, які скачують дані, що роздає вузол може бути перевантажений і процес розподілу контенту може відбуватися неприпустимо довго.

2.5.1 BitTorrent

BitTórrent[31] - пірінговий (P2P) мережевий протокол для колективного обмінювання файлами через Інтернет.

Файли надсилаються частинами, кожен torrent-клієнт, отримуючи (закачавши) дані частини, в той самий час віддає (надсилає) їх іншим клієнтам, що зменшує навантаження та залежність від кожного клієнта-джерела і гарантує надмірність даних.

2.5.2 Принцип роботи протокол

Перед початком закачування клієнт конектиться до трекеру за адресою, яка вказана в торрент-файлі, сповіщає йому свою адресу та хеш-суму торрент-файлу, в той самий час клієнт отримує адреси інакших клієнтів, що закачують або роздають цей самий файл. Потім клієнт з періодичністю сповіщає трекер про те, як йде процес та приймає оновлений список адрес. Такий процес має назву оголошення (англ. Announce).

Клієнти конектяться один з одним та роблять обмін сегментами файлів без прямої участі трекера, який тільки і робить, що зберігає інформацію, яка була отримана від приєднаних до обміну клієнтів, безпосередньо список клієнтів та інші статистичні дані. Для результативної роботи мережі BitTorrent необхідно, щоб як можна більше клієнтів були спроможні приймати вхідні з'єднання. Помилкове налагоджування NAT або брандмауера мають змогу перешкодити цьому.

При поєднанні клієнти в той самий час діляться інформацією про присутні у них сегменти. Клієнт, який має бажання скачати сегмент (лічер), відправляє запит та якщо інший клієнт підготовленний до віддачі, має змогу отримати даний сегмент. Після даної дії клієнт робить перевірку контрольної суми сегменту. У випадку, коли вона збіглася з тією, що занотована в торрент-файлі, то сегмент можна вважати вдало скачаним та клієнт починає оповіщати всіх приєднаних пирів про можливу присутність у нього даного сегменту. Коли ж контрольні суми відрізняються, то сегмент розпочинає завантажуватись знову.

Отже, об'єм службової інформації (а саме розмір торрент-файлу та обсяг повідомлень з повним списком сегментів) напряму в залежності від кількості, а з цього випливає, що і від розміру сегментів. Звідси слідує, що при виборі сегмента треба додержуватися балансу: з одного боку, при значному розмірі сегмента розмір службової інформації буде менше, але якщо відбудеться помилка, необхідно виконати перевірку контрольної суми та будеш змушений завантажувати ще раз надлишкову інформацію. З іншої сторони, при малому обсязі помилки не на стільки шкідливі, так як є потреба завантажити ще раз вже менший розмір, проте розмір торрент-файлу та самих повідомлень щодо існування сегментів збільшується.

2.5.3 Алгоритм обміну даними

Кожен користувач має змогу блокувати віддачу другому користувачу, на якійсь проміжок часу. Все це робиться, для того, щоб канал віддачі, використовувався ефективно. Окрім цього, під час вибору хто буде розблокований, перевага буде надаватись пірам, які передали найбільше

сегментів цьому клієнту. Отже, піри, які мають гарну швидкість віддачі, таким чином заохочують друг друга за принципом «я – тобі, ти - мені».

Принцип за яким відбувається обмін є тим самим – «я – тобі, ти - мені» симетрично у дві сторони.

Користувачі оповіщають один одного про відомі сегменти під час підключення, а також при отриманні нових, звідси кожен користувач може зберігати інформацію, про наявні сегменти у інших пірів. Обмін виконується у наступному порядку: спочатку проходить обмін серед користувачів з рідкісними сегментами, а потім інші, завдяки цьому буде підвищуватись відкритість файлів у самій роздачі.

У той же час вибір сегмента серед найбільш виняткових випадковий, і тому можна оминути ситуації, коли всі клієнти розпочинають завантажувати один і той же самий винятковий сегмент, що негативно б відобразилося на продуктивності.

Обмін даними стартує, коли обидві сторони в ньому заінтересовані, тобто, кожна сторона має сегменти, які відсутні в іншої. Кількість надісланих сегментів підраховується, і якщо одна зі сторін підмічає, що надсилає в середньому більше, ніж отримує, вона блокує на певний час віддачу іншій стороні. Таким чином, в протокол закладений захист від лічерів.

Сегменти діляться на блоки за розміром 16-4096 кілобайт, і кожен клієнт дає запит саме ці блоки. Одночасно можуть бути подані на запит блоки з різних сегментів. Також, деякі клієнти підтримують скачування блоків одного сегмента у різних пірів. В даному випадку вище описані алгоритми і механізми обміну застосовані і до рівня блоків.

При отриманні готового файлу клієнт переходить в особливий режим роботи, в якому він тільки повинен віддавати дані (стає Сідом). Далі сід періодично дає інформацію трекеру про зміни в стані торрентів (скачувань) і оновлює списки IP-адрес.

2.5.3.1 Загальні особливості

- Немає черги на скачування.
- Файли завантажуються маленькими фрагментами; чим менше фрагмент, тим частіше він буде передаватись. Таким чином, наявність в мережі «сіда» з готовим файлом для закачування необов'язково - система розподіляє сегменти між «пірами», щоб в подальшому вони могли обмінятися відсутніми сегментами.
- Клієнти (peers) мають змогу обмінюватись сегментами між собою, за принципом «я – тобі, ти - мені».
- Закачені фрагменти стають доступні без усіляких затримок іншим клієнтам.
- Контролюється кожен фрагмент, для зберігання цілісності.
- На фрагменти розбивається вся роздача цілком, а не окремі файли, тому у «ліча», який захотів завантажити лише деякі файли з роздачі, для того, щоб не постраждала цілісність фрагментів нерідко може зберігатися також невеликий обсяг зайвої (для нього) інформації.
- Об’єктом роздачі можуть бути декілька файлів (наприклад, вміст каталогу).

2.5.4 Трекер і робота без трекера

Трекер - спеціалізований сервер, що працює по протоколу HTTP. Трекер потрібен для того, щоб клієнти мали змогу знайти один одного. Тобто, на трекері зберігаються IP-адреси, вхідні порти клієнтів та хеш-суми, особливим чином ідентифікують об'єкти, які приймають участь у закачуванні. За стандартом, імена файлів на трекері не мають зберігатись, і відслідкувати їх по хеш-сумам не можна. В той час на практиці трекер часто окрім своєї базової функції виконує і функцію немаленького веб-сервера. Даний сервер зберігає файли метаданих і опису файлів,що передаються, надає статистику завантажень по різних файлах, відображає кількість підключених пірів на даний момент часу та ін.

В нових версіях протоколу була винайдена бестрекерна системи, яка вирішує певні попередні проблеми. Поламка трекера в даних системах не призводить до автоматичного призупинення всієї мережі.

Починаючи з версії 4.2.0 офіційного клієнта, в ньому винайдена функція бестрекерной роботи, що основується на DHT Kademlia. У даних системах трекер доступний децентралізовано, на клієнтах, у вигляді розподіленої хеш-таблиці.

2.5.4.1Multicast

Multicast - спеціалізована форма широкого мовлення, за допомогою якої мережевий пакет надсилається одночасно певній підмножині адресатів - не одному (unicast), та й не всім (broadcast). [5] [21]

Поряд з додатками, які утворюють зв'язок між джерелом і одним отримувачем, бувають такі, де потрібно, щоб джерело посилало інформацію напряму до групи одержувачів. У випадку таких додатків можна

навести приклад дистанційного навчання, розсилку внутрішньої інформації, копії баз даних та даних з веб-сайтів і багато іншого. При стандартній технології IP-адресації необхідно кожному отримувачу інформації надіслати свій пакет даних, тобто одна і та ж інформація обмінюється багато разів. Технологія групової адресації відображає собою розширення IP-адресації, що дозволяє відправити одну копію пакета водночас всім одержувачам. Нескінченна кількість одержувачів визначається належністю кожного з них до певної групи. Розсилку для певної групи одержують тільки члени даної групи.

Технологія IP Multicast дає ряд істотних переваг у порівнянні із стандартним підходом. А саме - додавання нових користувачів не призводить до необхідного збільшення пропускної здатності мережі. Значно скорочується навантаження на сервер який відправляє файл, йому більше не потрібно підтримувати безліч двосторонніх сполук. Використання групової адресації надає змогу забезпечити доступ корпоративних користувачів до інформації і сервісів, які раніше були недоступні, так як для їх доступу за допомогою стандартної адресації потрібні були б великі мережеві ресурси.

В останні роки широкого поширення та масштабності набули мультимедіа трансляції та відеоконференцзв'язок. При використанні стандартної технології пропускної здатності дійсних каналів ледь вистачає лише для встановлення зв'язку з дуже маленьким числом одержувачів. Групова адресація повністю знімає це обмеження і одержувачів може бути стільки, скільки людей захоче подивитись дану трансляцію.

У даний момент IP Multicast є широко забезпеченим мережевим стандартом. Все найсучасніше мережеве програмне забезпечення та апаратне устаткування підтримує цей стандарт. Для застосування групової IP- адресації є необхідність підтримки від локальної мережі. Що стосується глобальної мережі, в деколи допустимо використання «тунелювання» для подолання ділянок, цю адресації не підтримують.

Для реалізації групової адресації в локальній мережі необхідно мати: підтримку групової адресації стеком протоколу TCP / IP; програмну підтримку протоколу IGMP для посилання запиту про приєднання до групи і одержанні групового трафіку; підтримка групової адресації мережевою картою; додаток, який використовується для групової адресації, наприклад відеоконференція. Для розширення даної можливості на масштабну мережу додатково потрібна підтримка всіма проміжними маршрутизаторами групової адресації і пропускання групового трафіку мережними екранами. У локальній мережі є змога добитися ще більшої оптимізації, застосовуючи комутатори з фільтрацією групового трафіку, які мають змогу автоматично налаштовати на передавання трафіку тільки одержувачам.

Технологія IP Multicast застосовує адреси з 224.0.0.0 до 239.255.255.255. Підтримується статична і динамічна адресація. Наприклад статичною адресою є 224.0.0.1 - адреса групи, що містить в собі всі вузли локальної мережі, 224.0.0.2 – усі маршрутизатори локальної мережі. Діапазон адрес з 224.0.0.0 по 224.0.0.255 призначений для протоколів маршрутизації і других низькорівневих протоколів сапорту групової адресації. Також, всі інші адреси час від часу використовуються додатками.

Для розпізнавання членства мережевих пристроїв в неоднакових групах локальної мережі маршрутизатор застосовує протокол IGMP. Один з маршрутизаторів підмережі час від часу опитує вузли підмережі, для того, щоб дізнатися, які групи були використані додатками вузлів. Кожна група має тільки одну відповідь в підмережі. Для того, щоб стати членом наступної (нової) групи, вузол того, хто приймає робить запит на маршрутизатор локальної мережі. Мережевий інтерфейс вузла-одержувача підлаштовується на одержання пакетів з цим груповою адресою. Кожен вузол має функцію самостійно відстежувати свої активні групові адреси, а коли зникає потреба

відноситись до даної групи, зупиняє відправляти підтвердження на IGMP-запити. Результати IGMP-запитів вживаються протоколами групової маршрутизації саме для надсилання інформації про членство в групі на найближчі маршрутизатори і так далі.

Основна ціль групової маршрутизації є те, що маршрутизатори, під час обміну один з одним інформацією, створюють шляхи дистрибуції пакетів до всім необхідним підмережам без повторювання і петель. Кожен з маршрутизаторів передає отримуваний пакет на один або кілька інших маршрутизаторів, тим самим уникаючи повторного надсилання одного і того ж пакету по одному каналу і відправляючи його всім одержувачам групи. Так як склад групи з часом може змінюватися, тим самим ті члени групи, які з'являються та вибувають динамічно враховуються в побудові шляхів маршрутизації.

2.6 Статичний і динамічний контент

CDN може проектуватися як для роздачі статичного контенту (медіа-файли, статичні файли js і css, великі файли будь-якого типу) так і динамічного (прискорення віддачі динамічних веб-сторінок, edge-computing, потокові дані). Проблеми виникають при побудова цих двох типів CDN різних, хоча і частково перетинаються. Алгоритми і рішення розглядаються в даній роботі в першу чергу відносяться до мереж із статичним контентом, але деякі з них (вибір оптимального шляху всередині CDN) можуть бути застосовані і для мереж з динамічним контентом. [

2.7 Балансування на рівні DNS

Round robin DNS - один з методів розподілу навантаження, або відмовостійкості за рахунок надмірності кількості серверів, за допомогою управління відповідями DNS-сервера відповідно до якоїсь статистичною моделлю. Зазвичай застосовується до таких інтернет-протоколів, як веб-сервери, FTP-сервери. [15] [6]

В найпростішому випадку Round robin DNS працює, відповідаючи на запити не тільки одним IP-адресою, а списком з декількох адрес серверів, що надають ідентичний сервіс. Порядок, в якому повертаються IP-адреси зі списку, заснований на алгоритмі round-robin. З кожним відповіддю послідовність ір-адрес змінюється. Як правило, прості клієнти намагаються встановлювати з'єднання з першою адресою зі списку, таким чином різним клієнтам будуть видані адреси різних серверів, що розподілить загальне навантаження між серверами.

Немає стандартної процедури для визначення того, які адреси будуть використовуватися запитуючою додатком - деякі сервери намагаються змінити порядок списку, приділяючи пріоритетну увагу чисельно більше «близьким» мереж. Деякі настільні клієнти намагаються отримати альтернативні адреси після того, як не вдалося встановити з'єднання протягом 30-45 секунд.

Кругова система DNS часто використовується для розподілу навантаження територіально розподілених веб-серверів. Наприклад, у компанії є один домен і три ідентичних веб-сайту, розташованих на трьох серверах з трьома різними адресами. Коли один користувач отримує доступ до головної сторінки, він буде направлений на першу адресу IP. Другий користувач, який звертається до головної сторінки, буде відправлений на наступну адресу IP, а третій користувач буде відправлений на третій адресу IP. У кожному разі, коли IP-адреса видається, він відправляється в кінець списку. Четвертий користувач, отже, буде відправлений знову на першу адресу IP, і так далі.

2.7.1 Недоліки

Хоча Round robin DNS (RR DNS) легко реалізувати, все ж цей алгоритм має кілька проблематичних недоліків, пов'язаних з кешуванням запису в ієрархії RR DNS самого себе, а також з кешуванням на стороні клієнта, виданого адреси і його повторного використання, поєднання яких важко керовано . RR DNS не спирається на доступність послуг. Наприклад, якщо сервіс на одній з адрес недоступний, RR DNS буде продовжувати роздавати цю адресу і клієнти будуть як і раніше намагатися з'єднатися з непрацюючим сервером.

Крім того, воно не може бути кращим вибором для балансування навантаження на самого себе, оскільки він лише замінює порядок адрес кожен раз, коли ім'я сервера запитується. Не існує обліку відповідності IP-адреси користувача і його географічного розташування, часу виконання, навантаження на сервер, перевантаження мережі і т.д. Кругова система DNS навантаження найкраще підходить для послуг з великою кількістю рівномірно розподілених з'єднань з серверами еквівалентної потужності. В іншому випадку він просто робить розподіл навантаження.

Існують методи, щоб подолати такі обмеження. Наприклад, модифіковані DNS-сервера (такі, як lbnamed) можуть регулярно опитувати дзеркала серверів для перевірки їх доступності та завантаженості. Якщо сервер не відповідає в міру необхідності, сервер може бути тимчасово вилучений з пулу DNS, поки він не повідомить, що знову працює у відповідності зі специфікацією.

Деякі проблеми DNS балансування, такі як швидкість перемикання на новий сервер, можна частково вирішити за допомогою IPVS. IPVS реалізує балансування на транспортному рівні всередині ядра Linux. IPVS є частиною Linux Virtual Server, в рамках якого працює в якості балансувальника перед кластером реальних серверів. IPVS може перенаправляти запити на реальні сервера так, що для зовнішнього спостерігача сервіси реальних серверів будуть перебувати на одному IP-адресу.[16]

2.8 BGP

BGP (англ. Border Gateway Protocol, протокол граничного шлюзу) - головний в Інтернеті протокол динамічної маршрутизації.[30]

Протокол BGP призначений для обміну інформацією про досяжності підмереж між автономними системами (АС), тобто групами маршрутизаторів під єдиним технічним управлінням, що використовують протокол внутрідоменной маршрутизації для визначення маршрутів всередині себе і протокол междоменной маршрутизації для визначення маршрутів доставки пакетів в інші АС. Передана інформація включає в себе список АС, до яких є доступ через дану систему. Кращі маршрути обираються згідно правил, які були прийняті в даній мережі.

BGP сапортить адресацію яка має назву «безкласова» та застосовує висновки з маршрутів для поменшення таблиць маршрутизації. З 1994 року є чинна четверта серія протоколу, усі попередні серії, вже не актуальні.

Одним із головних механізмів, які забезпечують безперервну функцію Інтернету є BGP та DNS

BGP це протокол прикладного рівня, який виконує функціонування над протоколом транспортного рівня TCP. Після установки з'єднання передається інформація про всі маршрутах, призначених для експорту. Надалі передається тільки інформація про зміни в таблицях маршрутизації. При закритті з'єднання видаляються всі маршрути, інформація про яких передана іншим боком. [28] [26]

2.9 Моніторинг стану мережі

Керуюча система повинна мати повну інформацію про стан мережевого обладнання та серверів підтримують її функціонування. Для цього використовуються такі технології.

2.9.1 SNMP

SNMP (англ. Simple Network Management Protocol – протокол мережового керування) – інтернет протокол який створений по стандартам, для керування пристроями в IP- мережах, які основані на архітектурі UDP / TCP. Комутатори, сервери, маршрутизатори, модемні стійки, принтери, робочі станції та інші, відносяться до пристроїв, які підтримують SNMP. Загалом, даний протокол використовується у системах мережного управління саме для здійснення перевірки підключених до даної мережі пристроїв, яким необхідний контроль адміністратора. Сам SNMP був визнаний інженерною радою інтернету, як частина TCP / IP. SNMP містить у собі набор стандартів для мережевого керування, а саме схему баз даних, протокол прикладного рівня та набір об’єктів даних.[23]

SNMP дає дані для керування під видом змінних, які обмальовують налаштування даної системи. Данні змінні можуть бути викликані (іноді вказані базово) додатками управління.

При застосуванні SNMP один або декілька комп’ютерів (в яких працюють програмні засоби, що називаються менеджерами) роблять відстеження або керування групою хостів або девайсів в комп’ютерній мережі. Кожна керована система, повинна бути пов’язана з постійно запущеною програмою, яка має назву агент, котра через SNMP дає менеджеру інформацію.

Дані про функціонування та конфігурацію керованих систем обробляються агентами SNMP, та перетворюють їх у внутрішній формат, який є зручним саме для підтримки даного протоколу. Цей протокол дає доступ до активних завдань керування, а саме – зміну та використання нової конфігурації із-за віддаленої зміни цих змінних. Змінні, які доступні через SNMP, організовані у ієрархії. Саме ці ієрархії, в той самий час, як і інші дані (по типу та опису

змінної), будуть описуватись базами управлінської інформаціїї (менеджерські бази).

Мережі складаються з 3 основних компонентів, які керуються протоколом SNMP:

- Пристрій, що керується;
- Агент – це програмне забезпечення, яке працює на керованому пристрої або на пристрої, що підключений до інтерфейсу керування даного пристрою.;
- Система мережевого управління (Network Management System, NMS) - програмне забезпечення, яке виконує взаємодію з менеджерами для саппорту складеної структури даних, що вказує нам стан мережі.

Керований пристрій це такий елемент мережі (пристрій або програма), що відтворює інтерфейс керування (можуть бути і відмінні від SNMP), що дає двонаправлений або однонаправлений (доступ для читання тільки) доступ до певної інформації про даний елемент. Керуючі пристрої обмінюються цією інформацією з менеджером. Керовані пристрої можуть ставитися до будь-якого виду пристроїв: маршрутизатори, сервери доступу, комутатори, мости, концентратори, IP-телефони, IP-відеокамери, комп'ютери-хости, принтери і т.п.

Агентом називається програмний модуль мережевого управління, що розташовується на керованому пристрої, або на підключеному до інтерфейсу управління керованого пристрою. Агент має локальним знанням керуючої інформації і переводить цю інформацію в специфічну для SNMP форму або з неї (медіація даних).

В склад Системи мережевого управління (NMS) входить додаток, що відстежує і контролює керовані пристрої. NMS забезпечують основну частину обробки даних, необхідних для мережевого управління. У будь-який керованої мережі може бути одна і більше NMS.

2.10 Алгоритми розподілу запитів

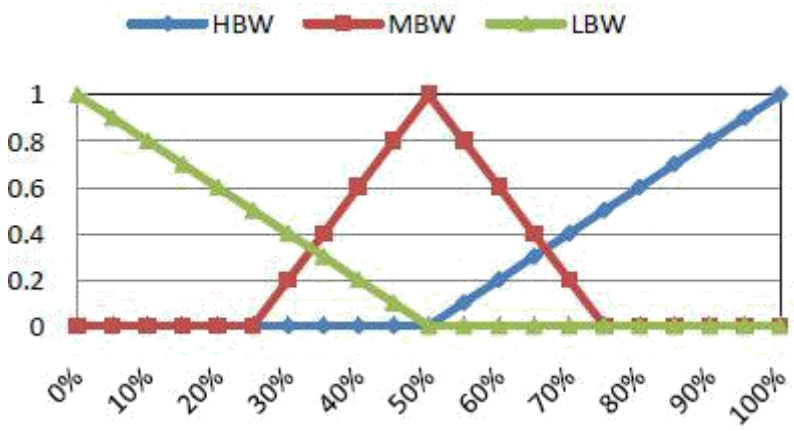
Існують різні алгоритми, які можуть бути використані для вибору сервера при маршрутизації запиту. Найбільш простими в реалізації є випадковий вибір (англ. Random Choice - RC) і послідовний вибір (англ. Round Robin - RR). Крім цих двох алгоритмів існують алгоритми, які вибирають найбільш підходящий сервер виходячи з деяких параметрів, таких як доступна пропускна здатність каналу (англ. Bandwidth availability - BW), доступність носія інформації (англ. Hard disk availability - HD) і кількість поточних підключень до сервера або доступність з'єднання (англ. connection availability - CON). [12] [17]

Коли система направляє запит користувача до одного з серверів з використанням одного з п'яти наведених алгоритмів пропускна здатність каналу від клієнта до сервера, завантаженість носія інформації і кількість підключень до сервера є трьома важливими факторами, за якими визначається, чи може сервер обслужити запит клієнта. Якщо хоча б по одному з цих параметрів сервер перевантажений, запит буде відхилено, а це значить що CDN не зможе обробити запит клієнта. В архітектурі мереж доставки контенту мінімізація відсотка відхилених запитів найбільш важливий фактор при порівнянні продуктивності різних алгоритмів.

2.10.1 Алгоритм прийняття рішень з використанням нечіткої логіки

Для даного алгоритму BW, HD і CON є вхідними параметрами. Покладаючись на ці три параметри алгоритм використовує механізм нечіткої логіки для прийняття рішення про вибір сервера.

На першому етапі вхідні параметри перетворюються в відповідні значення нечіткої логіки відповідно до функцій приналежності. Визначаються функції приналежності для кожного вхідного параметра. Для параметра доступності каналу (BW) ця функція буде виглядати наступним чином. HBW - висока значення приналежності для пропускної здатності каналу, MBW - середнє значення і LBW - низьке значення відповідно. [20] [22]



Значення приналежності HHD, MHD, LHD для носія інформації і HCON, MCON, LCON для кількості з'єднань визначаються аналогічно.

На другому етапі відбувається обчислення правил. Є 9 значень приналежності (HBW, MBW, LBW, HHD, MHD, LHD, HCON, MCON, LCON). Існує 27 можливих комбінацій цих значень. Для кожної з цих комбінацій ми визначаємо нечітке рішення з чотирьох можливих

- Настійно рекомендується сервіс - Так (Yes, Y)
- Рекомендований сервіс - Можливо так (Probably yes, PY)
- Чи не рекомендований сервіс - Можливо немає (Probably no, PN)
- Нізащо не рекомендований сервіс - Ні (No, N)

Кожне значення нечіткого рішення FD є мінімумом з трьох значень приналежності.

$$FD = \min \{HBW/MBW/LBW, HHD/MHD/LHD, HCON/MCON/LCON\}$$

BW	***HD***	***CON***	***FD***
HBW	*HHD*	*HCON*	*Y*
HBW	*HHD*	*MCON*	*Y*
HBW	*HHD*	*LCON*	*PY*
HBW	*MHD*	*HCON*	*Y*
HBW	*MHD*	*MCON*	*Y*

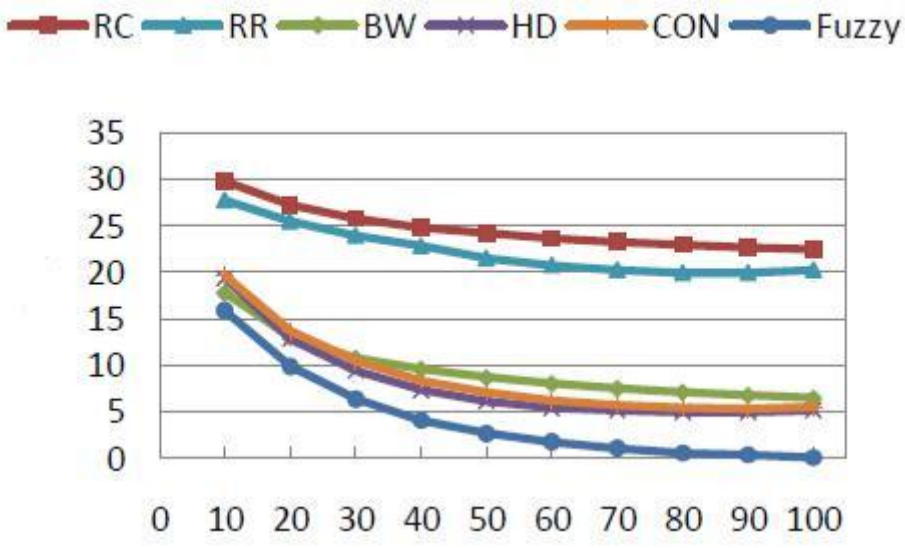
HBW	MHD	LCON	PY
HBW	LHD	HCON	PY
HBW	LHD	MCON	PY
HBW	LHD	LCON	PN
MBW	HHD	HCON	PY
MBW	HHD	MCON	PY
MBW	HHD	LCON	PY
MBW	MHD	HCON	PY
MBW	MHD	MCON	PN
MBW	MHD	LCON	PN
MBW	LHD	HCON	PY
MBW	LHD	MCON	PN
MBW	LHD	LCON	PN
LBW	HHD	HCON	PY
LBW	HHD	MCON	PN
LBW	HHD	LCON	PN
LBW	MHD	HCON	PN
LBW	MHD	MCON	N
LBW	MHD	LCON	N
LBW	LHD	HCON	PN
LBW	LHD	MCON	N
LBW	LHD	LCON	N

Таким чином виходить 27 значень належать до чотирьох груп. З кожної групи береться мінімум. В результаті виходить чотири значення FD (Y), FD (PY), FD (PN) і FD (N).

На третьому етапі алгоритму чотирьом значенням рішення призначається набір зважених значень. Кожне значення представляє різний набір ваг. Підсумкове "чітке" значення (англ. Crisp Value - CV) обчислюється на основі зважених значень і значень нечітких рішень.

Кожен сервер обчислює власне CV. Система розподілу запитів може використовувати CV для визначення найбільш підходящого сервера для обробки запиту. Сервер з найвищим CV є кращим для досягнення оптимальної продуктивності CDN.

Симуляція проводилася в порівнянні з алгоритмами RC, RR, BW, HD, CON. За вертикальної осі графіка кількість відхилених запитів в процентах. По горизонтальній осі середній час між запитами в мілісекундах.



Так як алгоритми RC і RR не використовують ніяких параметрів для розподілу запитів, для них відсоток відхилених запитів високий. При використанні тільки одного параметра (BW, HD, CON) відсоток відхилених запитів нижче ніж у RC і RR. Описаний алгоритм показує кращі результати так як враховує відразу все три параметра.

Розглянутий алгоритм можна розвинути ввівши додаткові параметри, такі як затримка (latency) або завантаженість процесора. Також можливий розвиток в напрямку комбінування з алгоритмами знаходження шляху в зваженому графі, де ваги дуг будуть визначаться з використанням розглянутого

Висновок до розділу:

Отже в архітектурі SDN можна виділити три рівні:

1.рівень інфраструктури, що надає набір мережевих пристроїв (каналів передачі даних і комутаторів);

2.рівень управління, що включає в себе мережеву операційну систему, яка виконує забезпечення додаткам мережеві сервіси та програмний інтерфейс для управління мережевими пристроями і мережею;

3. *рівень мережевих додатків* для гнучкого й ефективного управління мережею.

Openflow (відкритий потік)[32] - протокол (і технологія) управління процесом обробки даних, що передаються по комп'ютерах мережі маршрутизаторами і комутаторами.

Управління даними в OpenFlow здійснюється не на рівні окремих пакетів,

а на рівні їх потоків. Правило в комутаторі OpenFlow встановлюється за участю контролера тільки для першого пакету, а потім всі інші пакети потоку його використовують.

Пристрій OpenFlow складається, як мінімум, з 3 компонентів:

- таблиці потоків (англ. flow table);
- захищенного каналу (англ. secure channel), що створюється для управління комутатором зовнішнім «інтелектуальним» пристроєм (контролером);
- Підтримки протоколу OpenFlow protocol, що використовується для управління. Використання цього протоколу дозволяє уникнути необхідності писати програму для керованого пристрою;

3. Реалізація

Характеристика модельної мережі доставки контенту

Модельна мережу доставки контенту створювалася на основі віртуальних машин знаходяться на двох фізичних хост-машинах. Це дозволило моделювати різні топології мережі, а також варіювати продуктивність віртуальних машин і пропускну здатність каналів. Кількість віртуальних машин варіювалося від 5 до 15 в різних тестах. Віддача контенту здійснювалася за допомогою веб-сервера nginx. Управління розподілом копій контенту і маршрутизацією запитів здійснювалася за допомогою зміни конфігураційних файлів nginx керуючим модулем системи.

Таким чином програмна складова модельної мережі складається з двох основних компонентів - адміністративний інтерфейс управління, реалізований з використанням фреймворку Django і модуль управління вузлом мережі, заснований на фреймворку Twisted. Така архітектура дозволяє моделювати всі необхідні для перевірки алгоритмів операції (зміна маршрутів всередині мережі, створення додаткових копій контенту на вузлах) через управління кешом nginx.

3.1 Інструменти і технології

Далі описані найбільш важливі технології і програмні продукти, які були використані для створення модельної мережі.

3.1.1. Django

Фреймворк - Це каркас програмної системи (або підсистеми). Може включати допоміжні програми, бібліотеки коду, мова сценаріїв.

В як основу для реалізації адміністративного інтерфейсу був обраний фреймворк django, так як дозволяє з мінімальними затратами реалізувати необхідний функціонал.

Django - веб-фреймворк на мові Python. Однією з основних архітектурних особливостей фреймворка є подключаемості і отчуждаемость додатків

на базі яких створюється сайт. Кожна програма має власну модель даних ніяк не залежить від інших компонентів. Модель даних описується в вигляді класів на мові Python. Відображення моделі в базу даних здійснюється фреймворком автоматично, таким чином, втручання в структуру бази даних з боку розробників не потрібно.

Django має стандартну модель User для зберігання профілів користувачів і аутентифікації, в даній роботі стандартна аутентифікація використовується для входу в панель адміністрування.

Фреймворк має вбудовану настроюється панель адміністрування сайту яка дозволяє в зручній формі керувати програмами, даними додатків і профілями користувачів. Ця можливість дозволяє сильно спростити адміністрування сайту, і позбавляє від необхідності розробки власної адміністративної панелі.

3.1.2. Twisted

Twisted – це діяльно-зорієнтований мережевий фреймворк, який був дистрибутований за допомогою ліцезії MIT та розроблений на мові. Архітектура і концепції twisted оптимально підходять для модуля управління вузлом мережі. Всі операції виконуються модулем є реакцією на якісь зовнішні події. Також модуль виконує велику кількість операцій пов'язаних з мережевим взаємодією і асинхронність виконання цих операцій є необхідним атрибутом. У twisted вже реалізована велика кількість мережевих протоколів, зокрема необхідні для даного завдання SNMP і HTTP.

3.1.3. Nginx

Nginx - швидкий і надійний сервер, що володіє всім необхідним для побудови модельної мережі функціоналом.

Основні особливості:

- автоматичне створення списку файлів, а також обслуговування індексних файлів, кеш дескрипторів, статичних запитів відкритих файлів
- кидок запитів без кешування, розподіл навантаження і відмовостійкість
- підтримка кешування при прискореному проксінг і FastCGI
- пошвидшена підтримка FastCGI та memcached серверів, приклад простого розподілу навантаження та відмовостійкість
- фільтри, модульність, в тому числі стиснення (gzip), byte-ranges (докачка), часткові відповіді, HTTP-аутентифікація, SSI-фільтр
- кілька підзапитів на одній сторінці, оброблювані в SSI-фільтрі через проксі або FastCGI, виконуються паралельн
- підтримка SSL
- підтримка PSGI, WSGI

В Nginx робочі процеси обслуговують одночасно безліч з'єднань, мультіплексіруя їх асинхронними викликами операційної системи такими як select, epoll. Робочі процеси виконують цикл обробки подій від дескрипторів. Отримані від клієнта дані, за допомогою кінцевого автомата розбираються. Ланцюжок модулів обробляє розібраний запит, який попередньо заданий конфігурацією. В буферах, що зберігають дані які вказують на відрізок файлу або в пам’яті, формується відповідь клієнту. Послідовність, за якої дані будуть надіслані клієнтові, визначається за допомогою буферів, що об’єднуються в ланцюжки.

Конфігурація HTTP-сервера nginx розділяється на віртуальні сервери (Директива server). Віртуальні сервери поділяються на окремі розташування (location). Адреса та порти для віртуального серверу, будуть виконувати фінкцію прийому, а також можуть мати назви, які включають зірочку (*) щоб показати, що ця послідновність довільна в першій та останній частинах або задаватися регулярним виразом.

Location можуть задаватись трьома видами URI: точним, частиною або регулярным виразом. location можуть бути налаштовані для обслуговування запитів з статичного файлу, проксінг на fastcgi / memcached сервер.

Для ефективного управління пам'яттю nginx використовує пули. Пул - це послідовність попередньо виділених блоків динамічної пам'яті. Довжина блоку варіюється від 1 до 16 кілобайт. Спочатку під пул виділяється тільки один блок. Блок розділяється на не зайняту область та зайняту. Шляхом пересування покажчика на незайняту область, виконується виділення дрібних обє'ктів з урахуванням вирівнювання. Якщо незайнятої області у всіх блоках не вистачає для виділення нового об'єкта, то виділяється новий блок.

Таким чином, дрібні об'єкти виділяються дуже швидко і мають накладні витрати тільки на вирівнювання.

nginx містить модуль географічної класифікації клієнтів по IP-адресою. В його основу входить база даних відповідності IP-адрес географічних регіонів, представлена у вигляді Radix tree (стислий префіксне дерево або стиснений бор) в оперативної пам'яті. Nginx заздалегідь ділить перші декілька рівнів дерева, так, щоб вони мали змогу займати тільки 1 сторінку пам'яті. Це дає гарантії, що під час пошуків IP-адреса для перших вузлів, при трансляції адреси, бути завжди знайдений запис в TLB.

3.1.4 Numpy

Для математичного моделювання алгоритмів використовувалася бібліотека Numpy.

NumPy – це додадок, до мови Python, який робить більше підтримку для великих, багатовимірних масивів та матриць, поряд з широкою бібліотекою математичних функцій, які є високорівневими щодо операцій з даними масивами.

Попередня версія Numpy, Numeric, був спочатку створений Jim Hugunin.

Python – це введена мова, у якій набагато повільніше працюють математичні алгоритми ніж у мовах, які компілюються (C, Java). NumPy, для великої кількості даних алгоритмів, намагається вирішити дану проблему, в той самий час забезпечити підтримку для багатовимірних масивів та операторів і функцій для операційної роботи з ними. Отже, довільний алгоритм, може бути виражений як в основному послідновність операцій над масивами і матрицями, працює так само швидко, як еквівалентний код, що виконується в MATLAB, а після спеціальної оптимізації швидкість може досягти швидкості компільованих мов типу Сі.

NumPy можна розглядати як хорошу свободну копію MATLAB, томущо зовні, Matlab нагадує NumPy: обидва вони інтерпретуються, і обидва дозволяють користувачам писати швидкі програми, поки більшість операцій виробляються з матрицями або масивами, але не з скалярами. Велика кількість доступних додаткових тулбоксів, включаючи пакет Simulink, надає перевагу Matlab.

SciPy – це бібліотека, яка робить NumPy більше функцій, як у Matlab;

Matplotlib – це пакет, що створює графіку у стилі Matlab; є базовими пакетами, які доповнюють NumPy. Всередині NumPy та Matlab засновані на базі бібліотеки LaPack, яка була призначена для розв’язування базових задач лінійної алгебри.

3.2 Інфраструктура

Так як модельна система складається з декількох незалежних модулів, розташованих на різних віртуальних машинах, процес установки і настройки модулів системи потрібно максимально автоматизувати. також

в випадку застосування напрацювань на більш масштабних мережах доставки контенту, без автоматизації підтримка мережі стане неможливою.

3.2.1. Автоматизація установки і зборки

Для автоматизації установки був використаний механізм установки пакетів, вбудований в операційну систему. Процес побудови та встановлення можна описати наступними кроками

- Пакет збирається на машині ідентичною цільової (Версія ОС, розрядність ОС, версії системних бібліотек). Модуль викачується з репозиторію, під час визначення всіх залежності, викачуються всі необхідні бібліотеки, збирається python virtualenv.
- Зібраний пакет завантажується в репозиторій, доступний з усіх цільових машин
- Пакет встановлюється або оновлюється на цільовій машині за допомогою системного менеджера пакетів
- При необхідності вносяться необхідні зміни в конфігураційні файли модуля.

virtualenv - інструмент дозволяє створювати стерпні преконфігурірованние python-оточення. Дозволяє прискорити процес установки пакета на цільову машину, ізолювати оточення різних модулів в разі установки на одну машину.

3.2.2. Інструменти розробки

PyCharm - інтегроване середовище розробки модульних кроссплатформенних додатків. Середовище розробки орієнтована на Python. Також присутній розширена підтримка фреймворку django, що дозволяє спростити багато рутинних операцій, що виникають при розробці з використанням django. Є вбудована підтримка багатьох систем контролю версій, в тому числі Git, яка використовувалася в даному проекті. Середовище розробки має можливість автоматизувати процес відправки оновленого або виправленого проекту на тестовий сервер, що спрощує і прискорює процес тестування і розробки

Під час розробки будь-якого досить великого проекту неможливо обійтися без системи контролю версій. В якості системи контролю версій був використаний Git.

Git - розподілена система керування версіями файлів. Git підтримує швидке поділ і злиття версій, включає інструменти для візуалізації та навігації по нелінійної історії розробки. Надає кожному розробнику локальну копію всієї історії розробки, зміни копіюються з одного сховища в інший.

В відміну від централізованих систем, розподілені системи контролю версій дозволяють повноцінно вести розробку навіть за відсутності з'єднання з репозиторієм. Також наявність повної локальної копії сильно прискорює всі операції з ним.

У проекті також застосовувалося модульне тестування.

Модульне тестування або юніт-тестування - процес, що дозволяє перевірити на коректність окремі модулі вихідного коду програми. Ідея полягає в тому, щоб писати тести для кожної нетривіальною функції або методу. Це дозволяє досить швидко перевірити, чи не призвело чергову зміну коду до регресії, тобто до появи помилок в уже оттестировать місцях програми, а також полегшує виявлення і усунення таких помилок.

Для модульного тестування був використаний входить в стандартну бібліотеку Python фреймворк unittest, а також вбудований тестовий фреймворк django.

3.3 Оптимальний розподіл копій контенту всередині CDN

Змоделюємо топологію мережі у вигляді зваженого ненаправленного графа $G = \{V, E\}$. Безліч вершин - безліч вузлів мережі, а кожна дуга в безлічі E являє фізичне з'єднання між вузлами і зважено згідно деякої метриці, наприклад кількість хопов від одного вузла до іншого або інша більш складна функція яка бере до уваги пропускну здатність каналу або завантаженість вузла. Визначимо два підмножини V_a та V_R безлічі вузлів мережі. V_a - безліч вузлів, на які потрапляють запити від користувачів. V_R безліч вузлів мережі містять контент.

Нехай існує C наборів даних. Користувачі можуть надсилати запит на доступ до даних одного з C наборів через вузли доступу з V_a, дані з цих наборів можуть розташовуватися на кожному з серверів з V_R. Кожен вузол доступу може обслужити не більше ніж V_A^{MAX} запитів. Запити до заданого контенту обслуговуються найбільш підходящим вузлом з даними. Віддача контенту через вузол на відстані більшій ніж поріг d_{max} вважається незадовільною.

Кожен вузол з даними може обробити не більше K запитів. Не більше V_R^{MAX} ніж копій контенту може бути на одному вузлі даних.

Конфігурація запитів і вузлів з даними описується вектором стану розміру $C(|V_a| + |V_R|)$:

$$x = (a, r) = (a_1^1, a_2^1, \ldots, a_{|V_a|}^1, a_1^2, \ldots, a_{|V_a|}^C, r_1^1, r_2^1, \ldots, r_{|V_R|}^C)$$

Кожна з змінних a_i^C відображає кількість запитів до контенту c Є {1,…,C} на вузлі та Є V_a и r_j^c кількість копій контенту c Є {1,…,C} розташованих на вузлі j Є V_R

Якість сервісу оцінюється функцією $A(x)$, яка визначається як сума відстаней між вузлом доступу і вузлом з даними по всім запитам користувачів. Функція $M(x)$ оцінює вартість утримання копій і підтримки їх в актуальному стані та визначається як

$$M(x) = C_{Maint} \sum_{j \in V_R, c=1,\dots,C} r_j^c$$

где C_{Maint} константа.

Мінімізація описаних функцій і визначає завдання динамічного розподілу копій і розподілу запитів. Потрібно виробити стратегію, яка створює і видаляє копії на серверах в залежності від зміни стану мережі, запитів користувачів мінімізуючи вартість і час обробки запиту.

Для вирішення цього завдання був застосований Марковський процес прийняття рішень [22] [34] [35]. Стан процесу описується вектором, визначеним вище. Безліч станів визначається як:

$$\omega = \{x = (a, r): \sum_{k=1}^{C} a_i^k \le V_A^{MAX}; \sum_{k=1}^{C} r_j^k \le V_R^{MAX}; a_i^k, r_j^k \ge 0, \forall i \in V_A, \forall j \in V_R\}$$

Моделюємо агрегований запит до контенту k на сервері і Є V_a як процес народження смерті з коефіцієнтом народжуваності λ_i^k і коефіцієнтом смертності μ_i^k

Залежно від стану системи може бути прийнято рішення - видалення копії контенту. Простір дій описується як:

$$D = \left\{ \left(d_1^{1+}, d_2^{1+}, \dots, d_{|V_R|}^{1+}, \dots, d_1^{C+}, \dots, d_{|V_R|}^{C+}, d_1^{1-}, \dots, d_{|V_R|}^{1-}, \dots, d_1^{C-}, \dots, d_{|V_R|}^{C-}\right), d_i^{kx} \right.$$
$$\left. \in \{0,1\}, \sum_{\forall i,k,x} d_i^{kx} \leq 1;\ i = 1, \dots, |V_R|; x = +/- \right\}$$

Індикатор $d_i^{k+} = 1$ відображає рішення додати копію контенту k на сервер i Є V_R, $d_i^{k-} = 1$ відображає рішення забрати копію контенту k с сервера i. Якщо всі індикатори дорівнюють нулю, то розташувань копій залишається незмінним. Для спрощення моделі тільки одна копія може бути додана або видалена в кожен з моментів прийняття рішення.

Простір дій також обмежена умовою, що рішення про додавання копії допустимо тільки в разі якщо кількість копій на обраному сервері менше V_R^{MAX}, а рішення про видалення може бути прийнято тільки якщо на сервері є хоча б одна копія контенту. Нехай система в стані x = (a,r) $\in \omega$ та вибрано дію $d \in D$, тоді час життя стану x одно $\tau(x,d)$, де

$$\tau(x,d) = \left[\sum_{i=1}^{|V_A|} \sum_{k=1}^{C} \left(\lambda_i^k + a_i^k \times \mu_i^k\right) \right]^{-1}$$

Переходи, які можуть виникнути в цій моделі з вихідного стану в фінальне стан можуть бути обумовлені зростаючою кількістю запитів у вигляді "народження" або зменшується кількістю запитів у вигляді "смерті" в багатовимірному процесі народження-смерті. Імовірність переходу з стану $x = (x_A, x_R)$ в будь-який стан $y = (y_A, y_R)$ за рішенням d, приймає одне з наступних виразів, де ідентифікує вектор з одиничним елементом в позиції $(c \times |V_A| + i)$.

Переходи пов'язані з приходом запиту на контент c на сервр i:

У стані $(y_A, y_R) = (x_A + e_i^c, x_R + e_j^k)$ з ймовірністю $p_{xy}^d = \lambda_i^c \times d_j^{k+} \tau(x, d);$

У стані $(y_A, y_R) = (x_A + e_i^c, x_R - e_j^k)$ з ймовірністю $p_{xy}^d = \lambda_i^c \times d_j^{k-} \tau(x, d);$

У стані $(y_A, y_R) = (x_A + e_i^c, x_R)$ з ймовірністю $p_{xy}^d = \lambda_i^c \tau(x, d) \times [1 - \sum_{j=1}^{|V_R|} \sum_{k=1}^{C} (d_j^{k+} + d_j^{k-})];$

Зміни повязані з відправкою відповіді на запит на контент c сервера i:

У стані $(y_A, y_R) = (x_A - e_i^c, x_R + e_j^k)$ з ймовірністю $p_{xy}^d = (x_A)_i^c \times \mu_i^c \times d_j^{k+} \tau(x, d);$

У стані $(y_A, y_R) = (x_A - e_i^c, x_R - e_j^k)$ з ймовірністю $p_{xy}^d = (x_A)_i^c \times \mu_i^c \times d_j^{k-} \tau(x, d);$

У стані $(y_A, y_R) = (x_A + e_i^c, x_R)$ з ймовірністю $p_{xy}^d = (x_A)_i^c \mu_i^c \tau(x, d) \times [1 - \sum_{j=1}^{|V_R|} \sum_{k=1}^{C} (d_j^{k+} + d_j^{k-})];$

Всі зміни стану крім перерахованих вище мають ймовірність 0. Далі формулюється цільова функція. Функції вартості пов'язані зі станом A (x) та M (x) обчислюються за кожну одиницю часу, в яку система

знаходиться в стані x. Вартості стягуються тільки в момент додавання або видалення копії контенту, тобто в момент відповідного переходу стану x.

Вартість C+ та C- беруться тільки в момент додавання або видалення копії контенту, тобто в момент відповідного переходу стану.

Таким чином формулюється неоднорідна функція вартості:

$$r(x, d) = [A(x) + M(x)] \times \tau(x, d) + \sum_{j=1}^{|V_R|} \sum_{k=1}^{C} (d_j^{k+} C^+ + d_j^{k-} C^-)$$

Використовуючи техніки приведення до однорідності [33] [34] отримуємо еквівалентний Марковський процес в якому тимчасові проміжки генеруються пуассоновским процесом з рівномірною частотою. Перехід зі стану в стан описується Марківського ланцюжком, яка дозволяє фіктивні переходи зі стану в саме себе.

Норма Γ однорідного процесу визначається наступним чином.

$$\Gamma = \sum_{i=1}^{|V_A|} \sum_{k=1}^{C} \left(\lambda_i^k + V_A^{MAX} \mu_i^k \right)$$

Функція вартості також наводиться до однорідного виду:

$$\tilde{r}(x,d) = \frac{r(x,d)}{\tau(x,d) \times \Gamma} = \frac{1}{\Gamma} \{[A(x) + M(x)] + \frac{1}{\tau(x,d)} \times \sum_{j=1}^{|V_R|} \sum_{k=1}^{C} \left(d_j^{k+} C^+ + d_j^{k-} C^- \right)\}$$

Оптимальне рішення описується за допомогою змінної прийняття рішення відбиває ймовірність системи прийняти рішення d будучи в стані x.

Завдання лінійного програмування для даного Марківського процесу мінімізує витрати в довгостроковій перспективі формулюється так:

Мінімізувати $\sum_{(x,d) \in S} \tilde{r}(x,d) \times \pi_{xd}$ з обмеженнями:

$$\pi_{xd} \geq 0 \text{ при } (x,d) \in S$$

$$\sum_{(x,d) \in S} \pi_{xd} = 1$$

$$\sum_{d \in B} \pi_{jd} = \sum_{(x,d) \in S} \tilde{p}_{xj}^{d} \pi_{xd} \text{ при } j \in \omega$$

Де S це звичайно безліч всіх можливих пар векторів (стан, рішення). Дане завдання може бути вирішена засобами відомих методів дослідження операцій [35].

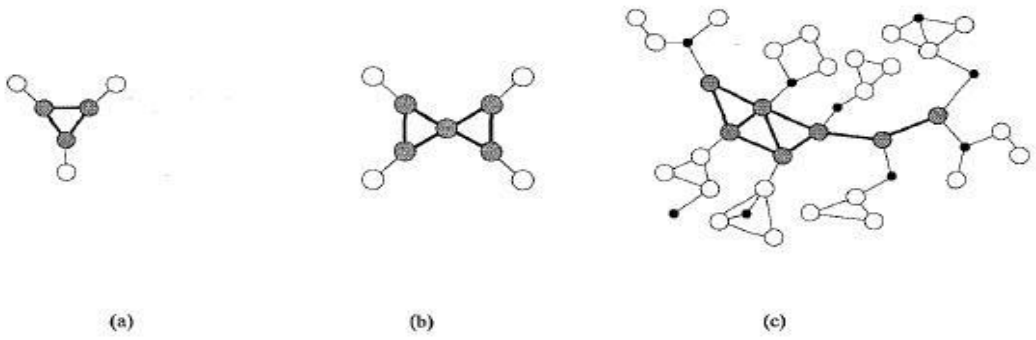
3.4. Методика використання

Рішення оптимізаційної проблеми має занадто високу складність для застосування на практиці. Були побудовані евристичні правила вибору дії $d \in D$ шляхом аналізу поведінки оптимального рішення. Нехай функція вартості має на увазі наступні пріоритети (в порядку убування) для підсумкових правил: Можливість обслуговувати запити; Мінімізація кількості копій; Мінімізація відстані між користувачами і контентом.

Це досягається за рахунок вибору вартості утримання значно вище максимального ваги дуги в графі. Таким чином оптимальне рішення буде використовувати мінімальну кількість копій контенту, при цьому залишаючи деякий запас ємності. При моделюванні алгоритм виконував попереднє копіювання контенту, на випадок зростання кількості запитів, але в той же час і видаляв більш непотрібні копії.

На підставі цих спостережень пропонується наступний евристичний алгоритм. На кожному кроці алгоритм визначає чи може поточна конфігурація впорається з можливим збільшенням кількості запитів. Залежно від результату, алгоритм перевіряє, чи можливо видалити або додати копію.

Оптимальний алгоритм був симулювати на математичній моделі, евристичний алгоритм запущений на модельній мережі (100 симуляцій для кожного випадку). У симуляції були використані наступні топології.



• - універсальний вузол ○ - вузол доступу ● - вузол з даними

В результаті симуляції були отримані наступні результати.

Топология а

C=1	Средняя дистанция	Среднее кол-во копий	% отклоненных запросов
Оптимальное решение	2.047	1.969	0
Эвристика	2,160 ± 0,162	1,945 ± 0,213	0
C=2	Средняя дистанция	Среднее кол-во копий	% отклоненных запросов
Оптимальное решение	2,362	2,720	0,021
Эвристика	2,392± 0,109	2,65 ± 0,176	0.010 ± 0,012

Топология б

C=1	Средняя дистанция	Среднее кол-во копий	% отклоненных запросов
Оптимальное решение	2.435	2,401	0
Эвристика	2,441± 0,095	2,537 ± 0,283	0
C=2	Средняя дистанция	Среднее кол-во копий	% отклоненных запросов
Оптимальное решение	-	-	-
Эвристика	2,480± 0,119	3,939 ± 0,192	0.038 ± 0,004

Для топології С розрахунок оптимального рішення не виконано ввиду розрахованої важкості.

Топология с

C = 1	Средняя дистанция	Среднее кол-во копий	% отклоненных запросов
$d_{Max} = \infty$	$5{,}010 \pm 0{,}148$	$10{,}438 \pm 0{,}313$	0
$d_{Max} = 10$	$5{,}199 \pm 0{,}169$	$10{,}414 \pm 0{,}430$	0
$d_{Max} = 8$	$4{,}708 \pm 0{,}158$	$11{,}279 \pm 0{,}662$	0
C = 2	**Средняя дистанция**	**Среднее кол-во копий**	**% отклоненных запросов**
$d_{Max} = \infty$	$5{,}233 \pm 0{,}165$	$11{,}180 \pm 0{,}574$	0
$d_{Max} = 10$	$5{,}286 \pm 0{,}180$	$11{,}185 \pm 0{,}628$	$0{,}06 \pm 0{,}05$
$d_{Max} = 8$	$4{,}737 \pm 0{,}105$	$12{,}770 \pm 0{,}240$	$0{,}07 \pm 0{,}17$
C = 4	**Средняя дистанция**	**Среднее кол-во копий**	**% отклоненных запросов**
$d_{Max} = \infty$	$5{,}407 \pm 0{,}111$	$11{,}252 \pm 0{,}428$	0
$d_{Max} = 10$	$5{,}406 \pm 0{,}102$	$11{,}522 \pm 0{,}682$	$0{,}06 \pm 0{,}07$
$d_{Max} = 8$	$4{,}833 \pm 0{,}103$	$14{,}637 \pm 0{,}476$	$1{,}36 \pm 0{,}79$

Результати симуляцій показують, що евристичний алгоритм майже не поступається оптимального рішення на простих топологиях. На складних топологиях пряме порівняння неможливо, але можна помітити, що результати близькі до оптимального рішення. Ми можемо обчислити середню кількість запитів з кожного вузла доступу рівне 0,4. Множення на кількість серверів дає 19.2 запитів в середньому. Кожна копія може обслуговувати K = 2 запитів. Таким чином, для C =1 і $d_{max} = \infty$ очікується не менше 10 копій. Результати симуляції евристичного алгоритму близькі до 10. При цьому його обчислювальна складність допускає використання в реальних системах.

Висновок до розділу

Django - веб-фреймворк на мові Python. Однією з основних архітектурних особливостей фреймворка є подключаемості і отчуждаемость додатків на базі яких створюється сайт.

Twisted – це діяльно-зорієнтований мережевий фреймворк, який був дистрибутований за допомогою ліцезії MIT та розроблений на мові. Архітектура і концепції twisted оптимально підходять для модуля управління вузлом мережі. Всі операції виконуються модулем є реакцією на якісь зовнішні події. Також модуль виконує велику кількість операцій пов'язаних з мережевим взаємодією і асинхронність виконання цих операцій є необхідним атрибутом. У twisted вже реалізована велика кількість мережевих протоколів, зокрема необхідні для даного завдання SNMP і HTTP.

Nginx - швидкий і надійний сервер, що володіє всім необхідним для побудови модельної мережі функціоналом.

Процес побудови та встановлення можна описати наступними кроками

- Пакет збирається на машині ідентичною цільової (Версія ОС, розрядність ОС, версії системних бібліотек). Модуль викачується з репозиторію, під час визначення всіх залежності, викачуються всі необхідні бібліотеки, збирається python virtualenv.
- Зібраний пакет завантажується в репозиторій, доступний з усіх цільових машин
- Пакет встановлюється або оновлюється на цільовій машині за допомогою системного менеджера пакетів
- При необхідності вносяться необхідні зміни в конфігураційні файли модуля.

virtualenv - інструмент дозволяє створювати стерпні преконфігурірованние python-оточення. Дозволяє прискорити процес установки пакета на цільову машину, ізолювати оточення різних модулів в разі установки на одну машину.

PyCharm - інтегроване середовище розробки модульних кроссплатформенних додатків.

Висновок

В магістерської дисертації отримані наступні основні результати:

- Розроблено модель мережі доставки контенту.
- Розроблено алгоритм дозволяє рівномірно розподіляти навантаження всередині мережі доставки контенту і знизити відсоток відхилених запитів в момент пікових навантажень.
- Алгоритм реалізований в рамках модельної мережі доставки контенту
- Алгоритм випробуваний на модельних даних, результати підтверджують зниження відсотка відхилених пакетів і рівномірність розподілу запитів по серверам

Література

1. J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally Distributed Content Delivery," IEEE Internet Computing, pp. 50-58, September / October 2002.
2. Akamai Technologies, Inc., www.akamai.com, 2007
3. Y. Rekhter, and T. Li, "A Border Gateway Protocol 4," Internet Engineering Task Force RFC 1771, March 1995.
4. J. Ni, and DHK Tsang, "Large Scale Cooperative Caching and Application-level Multicast in Multimedia Content Delivery Networks," IEEE Communications, Vol. 43, Issue. 5, pp. 98-105, May 2005.
5. Internet Content Distribution: Developments and Challenges. Adrian Popescu, David Erman, Dragos Ilie, Doru Constantinescu
6. *On the Optimal Placement of Web Proxies in the Internet* . Li B., Golin MJ, Italiano GF, Deng X. and Sohraby K., IEEE INFOCOM 1999 року, New York, USA, March 1999
7. *ROVER .Routing in Overlay Networks*. Popescu A., Kouvatsos D., Remondo D. and Giordano S., EuroNGI project JRA.S.26, 2006
8. M. Esteve, G. Fortino, C. Mastroianni, CE Palau, and W. Russo, "CDN-supported Collaborative Media Streaming Control," IEEE Multimedia, 2007.
9. S. Scellato, C. Mascolo, "Track Globally, Deliver Locally: Improving Content Delivery Networks by Tracking Geographic Social Cascades"
10. H. Yin, X. Liu, T Zhan, "Design and Deployment of a Hybrid CDN-P2P System for Live Video Streaming"
11. Akamai Technologies, Inc. A Distributed Infrastructure for e-Business - Real Benefits, Measurable Returns.23

12. Amit Aggarwal and Michael Rabinovich. Performance of dynamic replication schemes for an internet hosting service. Technical Report HPL-2001-8, AT & T Labs, October +1998.
13. William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In Proceedings of the 17th ACM Symposium on Operating System Principals, December 1999.
14. Yair Bartal. Probabilistic approximation of metric space and its algorithmic applications. In 37th Annual IEEE Symposium on Foundations of Computer Science, October 1996.
15. M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm. Enhancing the web infrastructure - from caching to replication. IEEE Internet Computing, pages 18- 27, Mar-Apr 1997.
16. Rajkumar Buyya, Al-Mukaddim Khan Pathan, James Broberg and Zahir Tari. A Case for Peering of Content Delivery Networks
17. G. Peng, CDN: Content Distribution Network, Technical Report TR-125, Experimental Computer Systems Lab, Department of Computer Science, State University of New York, Stony Brook, NY 2003. http://citeseer.ist.psu.edu/peng03cdn.html
18. OpenCDN project home page. http://labtel.ing.uniroma1.it/opencdn/
19. A. Vakali and G. Pallis, Content Delivery Networks: Status and Trends, IEEE Internet Computing, IEEE Computer Society, pp. 68-74, November-December 2003.
20. A Fuzzy-based Decision Approach for Supporting Multimedia Content Request Routing in CDN. Jian-Bo Chen, Shao-Jun Liao
21. Jian-Bo Chen, Tsang-Long Pao, "Designing the Burst Web Load Balancing Architecture Using Fuzzy Decision," Trans. International Journal of Innovative Computing, Information and Control, Vol. 5, No. 6, pp. 1689-1698, June. 2009.

22. Broberg, J., Zeephongsekul., P., and Tari, Z. Approximating bounded general service distributions, In Proc. of IEEE Symposium on Computers and Communications, Aveiro, Protugal, Jul. 2007.
23. Day, M., Cain, B., Tomlinson, G., and Rzewski, P. A model for content internetworking. IETF RFC 3466, Feb. 2003.
24. Architecture and Performance Models for QoS-Driven Effective Peering of Content Delivery Networks. Mukaddim Pathan, Rajkumar Buyya
25. Amazon CloudFront. http://aws.amazon.com/cloudfront/
26. *oStream: Asynchronous Streaming Multicast in Application-Layer Overlay Networks*. Cui Y., Li B. and Nahrstedt K.,, IEEE Journal on Selected Areas in Communications, Vol 22, No 1, January 2004
27. Martin Casado, Tal Garfinkel, Aditya Akella, Michael J. Freedman Dan Boneh, Nick McKeown, Scott Shenker. SANE: A Protection Architecture for Enterprise Networks, 15-th Usenix Security Symposium, Vancouver, Canada, August 2006.
28. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S.Shenker, J. Turner. Openflow: Enabling innovation in campus networks. SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69-74, 2008.
29. RFC 3170, IP Multicast Applications: Challenges and Solutions
30. RFC 4271, A Border Gateway Protocol 4 (BGP-4); RFC 4274, BGP-4 Protocol Analysis
31. Arnaud Legout, Guillaume Urvoy-Keller: Understanding BitTorrent: An Experimental Perspective, PLANETE (INRIA Sophia Antipolis), 2005
32. Openflow project site, http://www.openflow.org/
33. ATBharucha-Raid. Elements of theory of Markov processes and their applications. McGraw-Hill, 1960
34. J.Keilson. Markov chain models. Rarity and exponentiality. Springer-Verlag, New York, 1979

Глосарій

SDN (Software Defined Network) - програмно конфігурується мережу, мережу використовує технології віртуалізації мережевого обладнання.

Хоп - перехід від одного вузла мережі до іншого при маршрутизації запиту

Нод(Англ. Node) - сервер входить в мережу доставки контенту

Пір - в протоколі BitTorrent клієнт, який бере участь в роздачі.

Ліч - в протоколі BitTorrent пір, який продовжує скачування, та не має всіх сегментів на даний момент.

Сід - в протоколі BitTorrent пір, що містить усі сегменти файлу, що знаходиться на роздачі. Це або перша людина, що роздає цей файл, або той що вже скачав цей файл та почав роздавати його іншим.