УДК 004.382.2

DOI: http://dx.doi.org/10.20535/2219-3804162017102683

А. М. Марусик[1], *бакалавр*

БАЛАНСУВАННЯ НАВАНТАЖЕННЯ У КЛАСТЕРІ ПРОГРАМНО-КОНФІГУРОВАНОЇ МЕРЕЖІ

En With the trend of amplifying in network scale and application, traditional network protocols and functions are becoming increasingly large and complex. The core idea of SDN is to decouple network control from data transmission. OpenFlow switches implement data transmission function, so as to simplify the design of switches, and control functions are provided by controllers. In order to improve the performance of data transmission in SDN, this paper proposes a load balance solution scheme by taking advantage of the global network view of SDN. By updating topology information of global network, SDN controller can discover all paths between each source node to each destination node. For the purpose of choosing real-time least loaded path, load balancer immediately calculate the integrated load condition of multiple path when receiving the path information transmitted from SDN controller. The contrast experiment results show that load balancing strategy proposed in this paper can select more rational transmission path for data-flow.

Ru Рассмотрена задача балансировки нагрузки в кластере серверов. В основе предложенной системы балансировки лежит программно определяемая сеть, что обеспечивает сбалансированную нагрузку на пути кластера.

[1]*НТУУ «Київський політехнічний інститут ім. Ігоря Сікорського», факультет інформатики та обчислювальної техніки*

Вступ

Програмно-конфігурована мережа — це відносно нова концепція мережевої архітектури, основна ідея якої полягає у відділенні рівня управління мережею від пристроїв передачі даних і реалізується програмно[1].

Дані передаються відповідно до таблиць маршрутизації, що зберігаються на апаратних системах, як і раніше. Але ці таблиці централізовано управляються віддаленою системою, у зв'язку з чим адміністратору не потрібно змінювати таблиці на кожному комутаторі. У ідеальному випадку всі мережеві компоненти повинні управлятися і налаштовуватися у ході однієї операції. Спільна робота компонентів програмно обумовленої мережі може бути заснована на стандарті *OpenFlow* (підтримується фондом *Open Networking Foundation*, *ONF*).

Для того, щоб забезпечити балансування навантаження між декількома серверами, розробнику потрібно вирішити дві задачі — яким чином направляти запити клієнтів на конкретний сервер і по яким критеріям розподіляти навантаження на сервери.

Існує багато різних алгоритмів і методів балансування навантаження. Вибираючи конкретний алгоритм, потрібно виходити зі специфіки конкретного проекту та із цілей, яких планується досягти.

Постановка задачі

При балансуванні навантаження необхідно зважати на такі критерії[2]:

- справедливість: потрібно гарантувати, щоб на обробку кожного запиту виділялися системні ресурси і не допустити виникнення ситуацій, коли один запит обробляється, а всі інші чекають своєї черги;
- ефективність: всі сервери, які обробляють запити, повинні бути зайняті на 100%; бажано не допускати ситуації, коли один із серверів простоює у очікуванні запитів на обробку;
- скорочення часу виконання запиту: потрібно забезпечити мінімальний час між початком обробки запиту (або його постановкою у чергу на обробку) і його завершення;
- скорочення часу відгуку: потрібно мінімізувати час відповіді на запит користувача.

У той же час, алгоритм балансування має володіти такими властивостями:

- передбачуваність: у яких ситуаціях і за яких навантажень алгоритм буде ефективним для вирішення поставлених завдань;
- рівномірне завантаження ресурсів системи;
- масштабованість: алгоритм повинен зберігати працездатність у разі збільшення навантаження.

Пропонована архітектура

Нижче буде розглянуто модель реалізації кластера у програмно-конфігурованій мережі.

Хоча традиційні маршрутизатори зберігають таблиці маршрутизації, останні містять тільки інформацію про вузли призначення і маршрутизатори можуть обчислювати тільки інформацію про мережі наступного переходу без глобального бачення мережі. Однак контролер *SDN* отримує можливість показати глобальне уявлення мережі на початку побудови мережі. Оновлюючи інформацію про топологію глобальної мережі, контролер *SDN* може виявити всі можливі шляхи між кожним вихідним вузлом і до кожного вузла призначення. Використовуючи цю перевагу глобального бачення мережі *SDN*, можна оцінити стан навантаження кожного глобального шляху.

Мережева архітектура пропонованої системи балансування навантаження *SDN* показана на рис. 1.

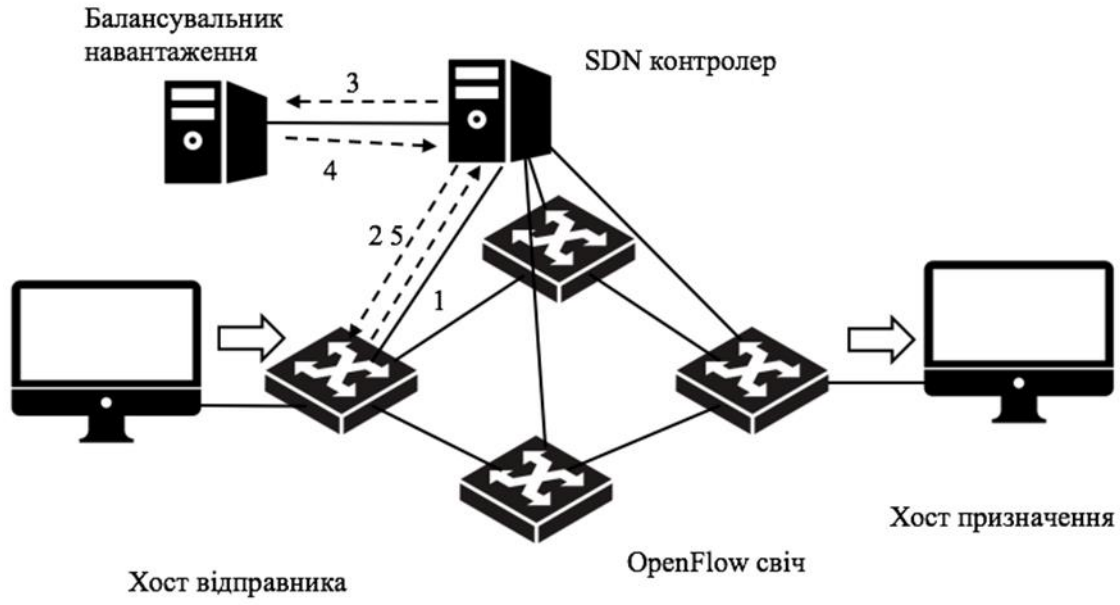

Рис. 1. Архітектура запропонованої системи балансування

Для того, щоб полегшити навантаження на *SDN* контролер, запропонована архітектура використовує окремий сервер для балансування навантаження на кожному шляху. Для того, щоб в режимі реального часу вибирати найменш завантажений шлях, балансувальник миттєво обчислює зібрану інформацію про стан завантаження шляхів, отриманої від *SDN* контролера. Таким чином, у пропонованій схемі, контролер періодично передає інформацію про навантаження кожного шляху балансувальнику навантаження. І коли *SDN* контролер виконав функцію балансування навантаження, балансувальник навантаження повертає найменш завантажений шлях контролеру. Після того, як контролер *SDN* отримує шлях для передачі, він

виділить флоу-таблицю для *OpenFlow* світчерів щоб виконати план передачі потоку даних.

Процедура керування даними у запропонованій системі показана на рис. 2:

1. Коли новий потік даних передається у домен *SDN*, *OpenFlow* включає процес відповідає інформації заголовку пакета і флоу-таблиці. Якщо флоу-таблиця відповідає інформації заголовку пакета, цей потік даних буде передаватися *Action* полем у флоу-таблиці. А якщо немає флоу-таблиці, яка б відповідала цьому пакету, *OpenFlow* комутатори будуть передавати інформацію заголовку цього пакета на контролер *SDN*, щоб визначити шлях для передачі.
2. У разі знаходження тільки одного шляху передачі даних, контролер *SDN* створить нові флоу-таблиці і призначить їм *OpenFlow* світчери для активної передачі даних
3. За знаходження декількох шляхів передачі даних, контролер *SDN* буде передавати множину інформації навантаження шляху до балансування навантаження.
4. Вирівнювач навантаження обчислює навантаження інтегрований для кожного шляху і вибирає одну із них найменше завантажений шлях як результат, щоб повернутися назад до контролера *SDN*.
5. Контролер *SDN* отримує обраний шлях від балансувача навантаження і створює флоу-таблиці щоб призначить *OpenFlow* світчери.

Експериментальні результати та оцінка

У даній роботі ми використовуємо *mininet* [3] – мережевий інструмент моделювання і *Floodlight* [4] – *SDN* контролер для імітації середовища багато шляхової мережі *SDN* для оцінки ефективності пропонованої стратегії балансування навантаження. Змодельована *SDN* топологія показана на рис. 2:

На рис. 2 показано кілька топологій мережі шляху. Ця топологія містить три шляхи від хоста 1 до хоста 7:

1. Свіч 1-> Свіч 2-> Свіч 6;
2. Свіч 1-> Свіч 4-> Свіч 3-> Свіч 6;
3. Свіч 1-> Свіч 4-> Свіч 5-> Свіч 6.

У реальному мережевому середовищі мережевий трафік є хаотичним і невизначеним. Таким чином у даній роботі експеримент призначений для імітації мережевої середовища у реальному часі. Трафік буде передаватися випадковим чином між хостом 2, хостом 3, хостом 4, хостом 5, хостом 6 і хостом 7. Під час передачі трафіку, умови завантаження трьох шляхів від хоста 1 до хоста 7 будуть доволі серйозно відрізнятися. Потім ми починаємо контролювати хост 1 і посилати трафік хосту 7, який розглядається у

якості нового вхідного потоку даних у кластері *SDN*. Кінцева мета для експерименту, щоб вибрати один найменш завантажений шлях від хоста 1 до хоста 7 у якості нового вхідного каналу передачі, і для досягнення балансування навантаження в мережі.Для того, щоб оцінити запропоновану стратегію балансування навантаження, ми взяли дві інших стратегії балансування навантаження у якості порівняння. Перша це *DLB* (*dynamic load balance* алгоритм) і статичний *Round Robin*.

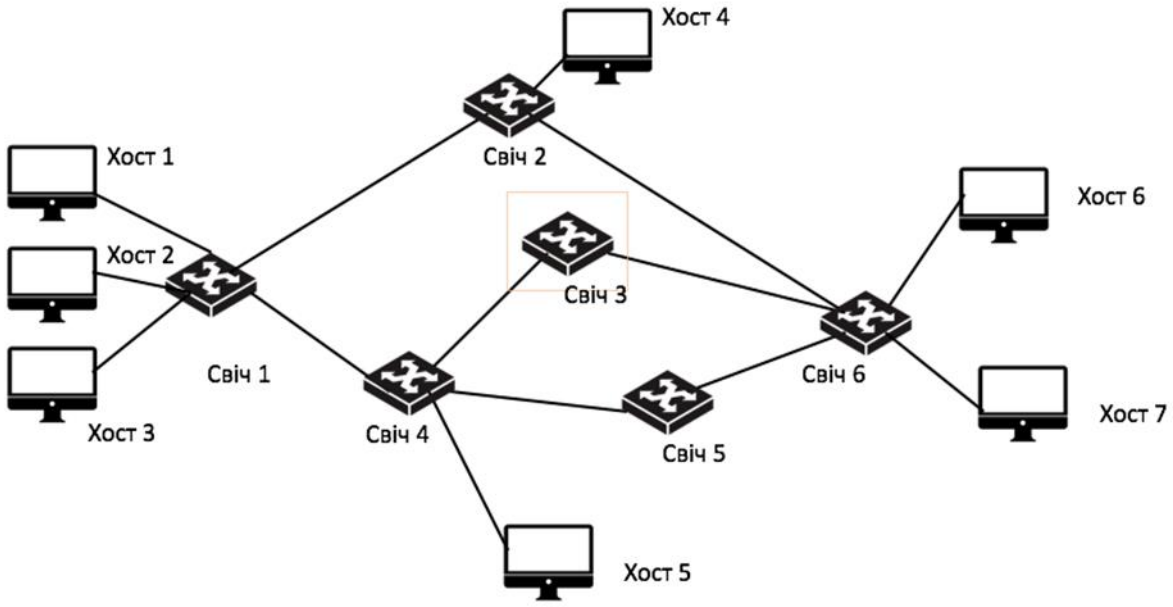

Рис. 2. Топологія *SDN* експерименту

Середній коефіцієнт використання пропускної здатності та середньоквадратичне відхилення використання пропускної здатності показані табл. 1.

***Таблиця* 1.**

Середній коефіцієнт використання пропускної здатності і середньоквадратичне відхилення використання пропускної здатності

	Пропонована архітектура	*DLB*	*RR*
Шлях 1	0,738	0,855	0,917
Шлях 2	0,450	0,744	0,398
Шлях 3	0,644	0,395	0,311
Середньоквадратичне відхилення	0,157	0,249	0,335

Як показано в табл. 1, середньоквадратичне відхилення запропонованої архітектури є найменшим серед усіх інших, що означає коефіцієнт використання пропускної здатності усіх шляхів близькими один до одного і сприятливий результат досягається за рахунок використання запропонованої архітектури. Оскільки стратегія *DLB* ігнорує глобальний стан навантаження шляху, шлях із найкращими умовами не обраний, і це несе погані наслідки. Але ця стратегія балансування навантаження все одно працює у якійсь мірі. *Round Robin* відноситься до статичної стратегії балансування навантаження, тому він не може аналізувати у режимі реального часу стан навантаження на шляхи. І результат цієї стратегії є найгіршим.

Висновок

Таким чином, запропонована система балансування користуючись глобальним баченням мережі вирішує проблему балансування. У такому разі вона основана на аналізі стану навантаження шляхів у реальному часі. Ця стратегія збирає коефіцієнт використання пропускної здатності, відсоток втрати пакетів, затримки і стрибки у передачі.

Стратегія балансування навантаження, запропонована у даній роботі застосовуються в імітаційному експерименті. У порівнянні зі статичною *Round Robin* стратегією, експериментальні результати показують, що запропонована у даній роботі стратегія, забезпечує більш високу продуктивність мережі й досягає 19,3% зниження латентності мережі.

Список використаної літератури

1. Openflow: Enabling Innovation In Campus Networks. / [N. Mckeown, S. Shenker, T. Anderson та ін.]. // Acm Sigcomm Computer Communication Review. – 2008. – №38. – С. 69–74.
2. *Марусик А. М.* Динамічна система балансування навантаження вебсерверів / А. М. Марусик // Інформаційні системи, механіка та керування. – 2015. – №12. – С. 25–32.
3. Mininet [Електронний ресурс] // http://mininet.org/ – Режим доступу до ресурсу: http://mininet.org/.
4. Floodlight [Електронний ресурс] // http://www.projectfloodlight.org/floodlight/ – Режим доступу до ресурсу: http://www.projectfloodlight.org/floodlight/.