УДК 004.77 DOI: 10.18372/2073-4751.74.17882

Кулаков Ю.О., д.т.н.,
orcid.org/0000-0002-8981-5649,
Череватенко О.В.,
orcid.org/0000-0001-9686-0555

МОДЕЛЮВАННЯ МАРШРУТИЗАЦІЇ З ВИКОРИСТАННЯМ ВІДОМИХ МАРШРУТІВ ЗА ДОПОМОГОЮ SDN-КОНТРОЛЕРА ONOS

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

ya.kulakov@gmail.com

Вступ

SDN – це оновлена мережева парадигма, яка дає надію здолати обмеження традиційної мережевої інфраструктури. Вона відокремлює площину керування від площини даних і реалізує кожну окремо. Сепарація площини керування та площини даних дозволяє мережевим комутаторам стати простими пристроями пересилання, а логіку керування реалізувати в централізованому контролері. Таким чином, підвищується гнучкість мережі, за рахунок розбиття керування мережею на частини. Цей підхід можна розвивати, вводячи нові абстракції та полегшуючи еволюцію й інновації мережі [1, 2].

Переважна більшість пакетів, які обробляє комутатор, керується лише площиною даних. Остання складається з портів, які використовуються для прийому та передачі пакетів, а також таблиці пересилання, яка вказує комутатору, як працювати з вхідними пакетами. Площина даних відповідає за буферизацію пакетів, планування пакетів, модифікацію заголовка та пересилання.

Деякі пакети не можуть бути оброблені площиною даних безпосередньо, оскільки, наприклад, їхня інформація ще не вставлена в таблицю пересилання. Такі пакети пересилаються в площину керування, яка лежить поверх площини даних. Площина керування включає багато дій, головним чином для модифікації таблиці пересилання площини даних. По суті, рівень керування відповідає за обробку різних протоколів маршрутизації, які можуть вплинути на таблицю пересилання [3, 4].

Програми для SDN мережі запускаються безпосередньо на контролері. Вони відповідають за управління таблицею потоків на мережевих пристроях (площина даних). Наприклад, виконують задачу маршрутизації пакетів через найкоротший шлях між двома кінцевими точками або балансують навантаження трафіку на кількох шляхах.

Перевагою централізації управління трафіком є те, що усувається необхідність обміну службовою інформацією між пристроями-ланками мережі, що завжди призводить до часових витрат у розподілених методах проектування трафіку. З іншого боку, оскільки вся важлива для управління інформація зберігається в одному місці, існує необхідність значно підвищити безпеку контролера задля уникнення несанкціонованого доступу до нього [5].

Актуальність проблем, вирішенням яких займаються автори в даній статті, можна зрозуміти, дивлячись на зростання розмірів сучасних мереж та кількості їхніх користувачів. Оскільки традиційні методи маршрутизації стають більш складними з огляду на збільшення масштабів, потрібен новий спосіб передачі інформації між кількома вузлами. Підтримка SDN також необхідна, оскільки ця технологія швидко поширюється та має корисне практичне застосування.

Об'єктом дослідження є створення програмної моделі мережевої інфраструктури, маршрутизація у якій відбувається на основі алгоритму багатошляхової маршрутизації з використанням відомих маршрутів, а управління мережею виконує

контролер на базі програмного забезпечення ONOS. Предметом дослідження є метрики топології, що будуть отримані на контролері в ході тестування моделі.

Мета цього дослідження – отримати практичні результати тестування, які продемонструють прикладне застосування обраного рішення маршрутизації на основі технології SDN та його доцільність при створенні локальної мережі з програмним управлінням.

Основна частина

Для проведення тестування створимо тестову топологію довільного типу. Для перевірки гнучкості алгоритму маршрутизації та обраного інфраструктурного рішення необхідно, щоб топологія не була одного з розповсюджених простих типів. Так матиметься змога переконатися в тому, що контролер готовий провести ефективну маршрутизацію за будь-яких умов.

Для моделювання віртуальної мережі скористаємось програмою mininet, яка дозволяє створити мережу, яку SDN контролер зможе виявити як реальну та побудувати копію її моделі в своїй пам'яті. У mininet топологія будується на основі python скрипту, де записані загальні дані про мережу (наприклад, версії протоколів, опис пристроїв, тощо) та безпосередні зв'язки між ланками на топології (links). Щоб не прописувати всі зв'язки вручну та зекономити час на проектування, скористаємось утилітою на основі mininet – Miniedit, де для побудови топології мережі можна використати графічний інтерфейс [6].

Для тестування була створена топологія з наступними характеристиками (рис. 1):

- містить чотири хости (hosts, комп'ютери користувачів), що об'єднані між собою у мережі з адресою 10.0.0.0/8;
- містить вісім комутаторів vSwitch (switches), що передають трафік на основі протоколу OpenFlow;
- містить один SDN контролер (controller), що виконує управління усіма комутаторами в мережі;
- працює на версії протоколу OpenFlow 1.3 для коректної сумісності з версією програмного забезпечення контролера ONOS [7].

В програмі також було налаштовано запуск командного інтерфейсу користувача (CLI) при запуску моделювання для виконання необхідних команд під час активної роботи моделі (рис. 2).

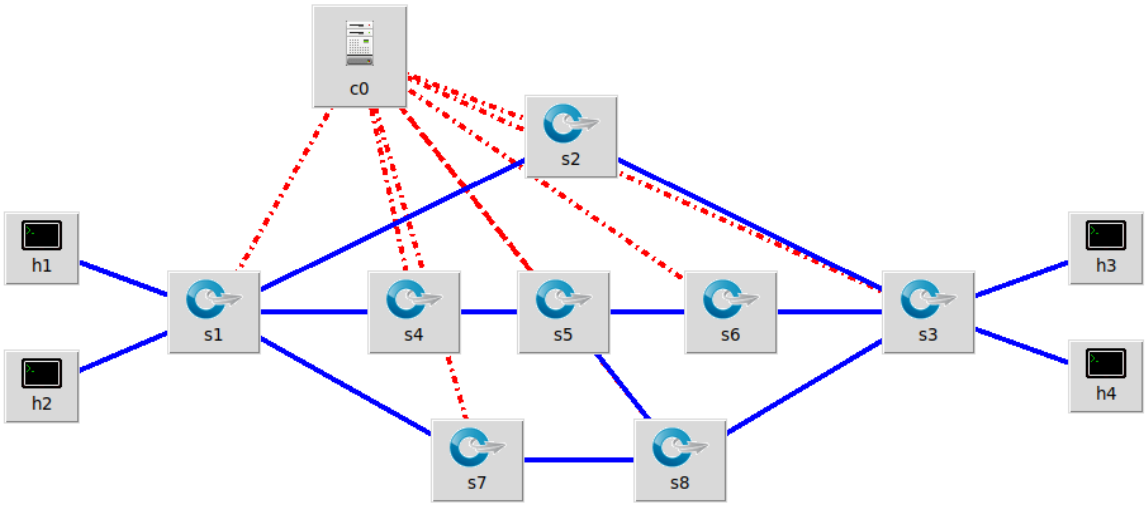

Рис. 1. Топологія мережі у mininet

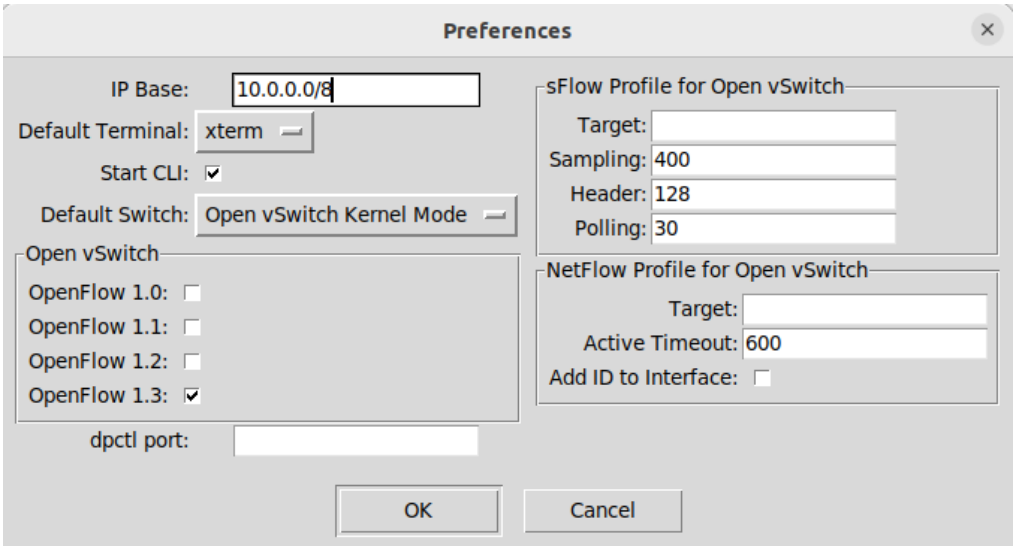

Рис. 2. Загальні налаштування мережевої моделі

Апаратним забезпеченням SDN контролера виступить звичайний комп'ютер, що працює під управлінням Ubuntu Linux. Від адміністратора потребується лише встановлення утиліти ONOS. Як уже зазначалося вище, на ONOS можна включати існуючі та додавати нові додатки – утиліти для розширення функціоналу контролера. В даному випадку ми користуємось двома додатками, що розробники ONOS включають в стандартний пакет установки:

- OpenFlow Provider Suite (org. onosproject.openflow) – додаток, що дозволяє контролеру працювати з протоколом OpenFlow;
- Reactive Forwarding (org.onosproject. fwd) – додаток, що дозволяє контролеру SDN виконувати маршрутизацію трафіку.

Для другого додатку авторами було змінене стандартне налаштування "SaveTopology", щоб мати можливість зберігати інформацію про всі маршрути на топології, які були досліджені контролером раніше. При такій конфігурації це дозволить практично реалізувати алгоритм маршрутизації з використанням відомих маршрутів [8].

Після запуску ONOS на комп'ютері та старту моделювання у mininet дві утиліти встановлюють зв'язок одна з одною, а машина, де запущено ONOS, представляється у топології як контролер c0 з побудованої нами топології. На контролері будується власне представлення топології мережевої інфраструктури. Її можна побачити, підключившись до системи ONOS, використовуючи графічний інтерфейс (рис. 3).

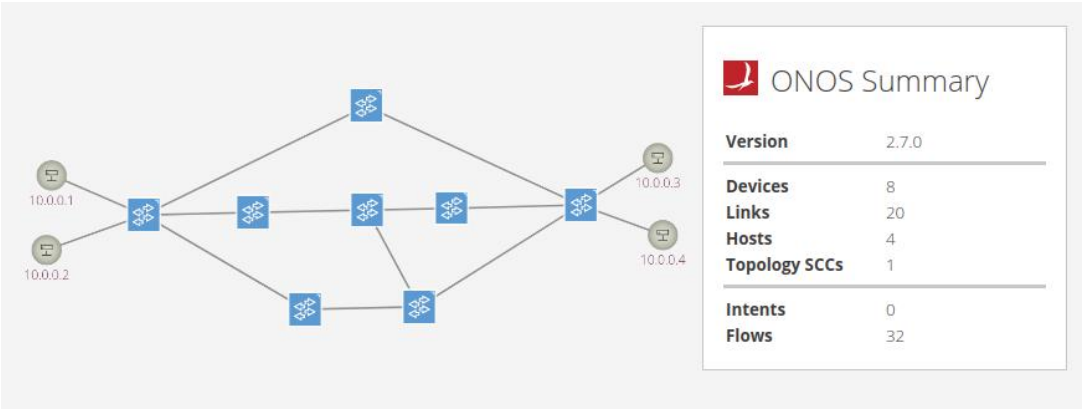

Рис. 3. Топологія мережі, побудована контролером ONOS на основі даних, отриманих з mininet

Спочатку топологія відображається без хостів *h1-h4*, визначених нами на етапі проектування топології. Це пов'язане з тим, що додаток OpenFlow Provider Suite визначає за замовчуванням лише OpenFlow пристрої у мережі і не визначає ті, що до них під'єднані. Для їхнього відображення виконаємо перевірку зв'язку між усіма хостами (pingall). Це дозволить перевірити не лише доступність хостів один для одного, але й дасть можливість контролеру провести дослідження маршрутів у мережі для подальшого запам'ятовування і використання.

В ході дослідження топології контролер визначив, що в мережі знаходяться вісім пристроїв і чотири хости, які об'єднані між собою двадцятьма зв'язками (links). Після перевірки зв'язку між хостами він також виділив тридцять два потоки (flows) – це дані, що додаються контролером для кожного комутатора та дозволяють останньому робити обробку трафіку. Кожен запис потоку має поле «селектор», яке визначає тип трафіку, що надходить на комутатор (в нашому випадку використовується селектор *ipv4*). Ці потоки є реалізацією таблиць маршрутизації у протоколі OpenFlow.

Перевіримо застосування алгоритму маршрутизації з використанням відомих маршрутів на контролері. Оскільки комутатори є пристроями другого рівня мережевої моделі OSI, вони не мають IP-адрес, як хости, тому не існує можливості застосувати команди для відслідковування пакетів IPv4 трафіку у створеній моделі. Проте стануть на пригоді можливості протоколу OpenFlow, на основі якого відбувається маршрутизація у нашій віртуальній інфраструктурі.

До контролера ONOS є можливість підключитись не лише за допомогою графічного інтерфейсу, а й через командний рядок (CLI). Скористуємось цим та проведемо тестування алгоритму маршрутизації.

Нехай задачею буде пересилка даних між двома хостами – *h1* (10.0.0.1) та *h3* (10.0.0.3). Щоб дістатись від першого хоста до другого, трафіку потрібно пройти через декілька Open vSwitch комутаторів. Якщо усі комутатори та зв'язки між ними знаходяться у робочому стані, існують п'ять оптимальних шляхів, щоб через кожну вершину трафік проходив не більше одного разу (табл.).

Таблиця. Вектори дистанцій між комутаторами *s1* та *s3*

$R_i(s,d)$	L_i	M_i
$R_1(1,3)$	*(s1 → s2 → s3)*	2
$R_2(1,3)$	*(s1 → s4 → s5 → s6 → s3)*	4
$R_3(1,3)$	*(s1 → s4 → s5 → s8 → s3)*	4
$R_4(1,3)$	*(s1 → s7 → s8 → s3)*	3
$R_5(1,3)$	*(s1 → s7 → s8 → s5 → s6 → s3)*	5

У таблиці 1 R є ідентифікатором маршруту, L – його описом, M – метрикою, що обрана для порівняння шляхів (кількість переходів, «хопів», між комутаторами). Маршрут з найменшою кількістю переходів алгоритм маршрутизації вважатиме основним, а найкоротший з тих, що залишились, буде резервним у випадку недоступності основного з певної причини. Перевіримо це безпосередньо в командному рядку контролера, використовуючи команду *paths*. Маємо на увазі, що контролер створив записи потоків для всіх комутаторів при активації додатку Reactive Forwarding, тому мусить мати інформацію про оптимальні маршрути.

Команда paths показує у своєму виводі не безпосередньо вузлові пристрої проходження пакетів, а зв'язки (links), які застосовуються для перебігу даних. Вводячи команду (використовуються назви пристроїв, які вони отримали на контролері ONOS, наприклад *s1* це *of:0000000000000001*), переконуємось, що контролер спрямував трафік через найкоротший шлях через комутатор s2, задіявши зв'язки *s1-s2* та *s2-s3*. Позначка *cost=2.0* в кінці виводу команди означає кількість

використаних зв'язків для досягнення адресата. Увівши команду з ключом --*disjoint*, маємо змогу побачити непересічні шляхи між джерелом та адресатом; контролер знає про резервний паралельний шлях, але не задіяв його через більшу метрику дистанції (рис. 4).

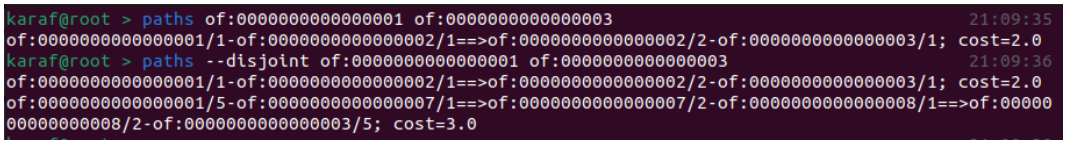

Рис. 4. Моделювання шляху *s1-s3* на початковій топології

Змоделюємо ситуацію виходу з ладу одного із зв'язків між комутаторами. Переходимо до командного рядку mininet (який ми передчасно налаштували на запуск одночасно з моделюванням мережі) та відключимо зв'язок між комутаторами *s1* та *s2*. Після цього знову перевіримо шлях, який обрав контролер для проходження трафіку після отримання інформації про відключення зв'язку. Ним став маршрут, який при минулій перевірці був резервним – через комутатори *s7* та *s8*, з кількістю переходів M=3 відповідно. У пам'яті в якості резервного присутній довший непересічний маршрут з M=4 (рис. 5).

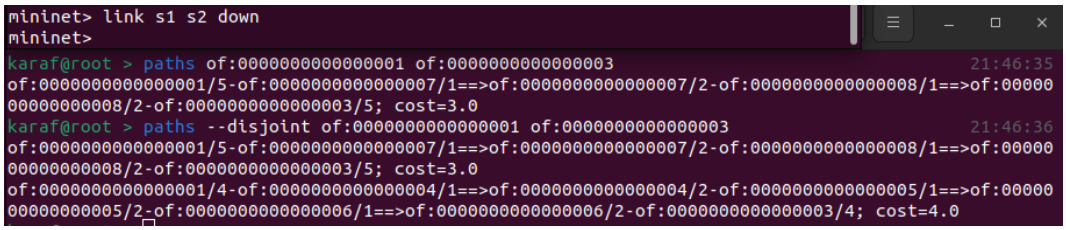

Рис. 5. Моделювання шляху *s1-s3* після відключення зв'язку *s1-s2*

Змоделюємо вихід з ладу одного з портів комутатора *s7*. За допомогою команди links для mininet дізнаємось, що він під'єднаний до комутатора s8 через інтерфейс *s7-eth2*. Відключаємо його, тим самим порушивши зв'язок між комутаторами. ONOS одразу отримує повідомлення про відключення порту на одному з пристроїв у мережі. Він реагує на це оновленням схеми топології та відправляє нові записи потоків сусіднім комутаторам. Аналізуючи записи логів ONOS, можна зробити висновок, що на ці операції він витратив 14 мілісекунд – такий час є абсолютно недосяжним для традиційного підходу до реконфігурації мереж, коли переналаштування портів доводилось би виконувати людині вручну (рис. 6).

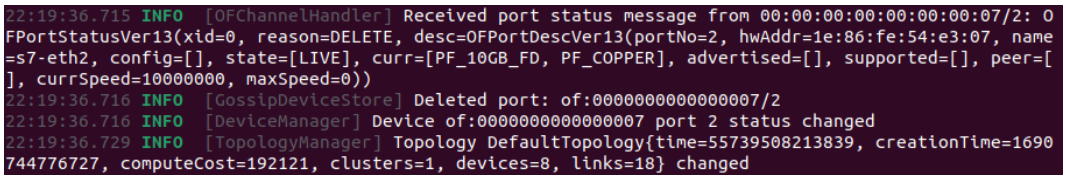

Рис. 6. Записи логів ONOS при відключенні порта комутатора

При перевірці основних маршрутів між комутаторами *s1* та *s3* в даній конфігурації мережевої моделі отримуємо очікувану картину: оскільки ті маршрути, що залишились робочими, мають однакову довжину 4, вони обидва були визначені контролером як основні та будуть використовуватись залежно від навантаження. ONOS має додатки для балансування трафіку та завантажить обидва маршрути відповідно заданому адміністратором відношенню [9]. Оскільки обидва маршрути мають загальну частину *s1* → *s4* → *s5*, вони не є непересічними, а отже команда paths --disjoint не продемонструє ніякого виводу (рис. 7).

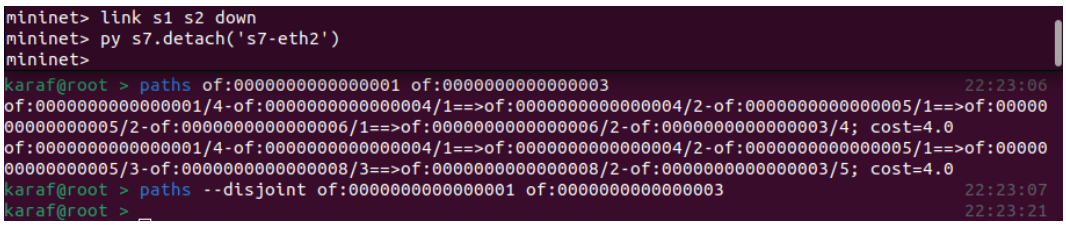

Рис. 7. Моделювання шляху *s1-s3* після відключення зв'язку *s1-s2* та порту *s7-eth2*

Висновки

Використання SDN контролера продемонструвало стабільність віртуальної мережевої моделі та забезпечило мінімальні витрати ресурсів та часу, необхідного для проведення усіх запланованих тестових активностей.

Було досягнуто мети роботи: отримані прикладні результати тестування, які довели, що розроблений авторами алгоритм маршрутизації на основі відомих маршрутів може бути втілений при роботі контролера SDN для ефективної передачі трафіку по найкоротшим маршрутам та для реконфігурації OpenFlow пристроїв у разі виході з ладу зв'язків між ними. Під час тестування контролеру знадобився дуже малий проміжок часу для внесення коректівів до таблиць маршрутизації, що підтвердило припущення про перевагу даного підходу над традиційним налаштуванням в ручному режимі.

Отримані результати тестування продемонстрували власне можливість та достатню ефективність прикладного використання обраного алгоритму багатошляхової маршрутизації. Програмне забезпечення контролера виявилось сумісним з теоретичними очікуваннями та було здатне виконати поставлену задачу. Це означає, що задача для дослідження сформована правильно, а рішення дало очікувані результати.

Подальші дослідження в цьому напрямку включатимуть в себе: перевірку можливостей контролера по маршрутизації в мережі у разі відключення додатку, призначеного для OpenFlow-маршрутизації; стрес-тест роботи контролера при масштабуванні топології та передачі по пристроям мережі великого об'єму трафіку; порівняння ефективності роботи контролера на основі алгоритму маршрутизації з використанням відомих маршрутів для різних топологій.

Література

1. Yeganeh S.H., Tootoonchian A., Ganjali Y. On scalability of software-defined networking. *IEEE Communications Magazine*. 2013. V. 51, No. 2. P. 136–141.

2. Kreutz D., Ramos F.M.V., Veríssimo P.E., Rothenberg C.E., Azodolmolky S., Uhlig S. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*. 2015. V. 103, No. 1. P. 14–76.

3. Hakiri Akram, Gokhale Aniruddha, Berthou Pascal, Schmidt Douglas, Gayraud Thierry. Software-Defined Networking: Challenges and research opportunities for Future Internet. *Computer Networks*. 2014. V. 75. P. 453–471.

4. Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*. 2008. V. 38, No. 2. P. 69–74.

5. Farooq M.S., Riaz S., Alvi A. Security and Privacy Issues in Software-Defined Networking (SDN): A Systematic Literature Review. *Electronics*. 2023. V. 12, Iss. 14. 3077.

6. Романов О.І., Марінов А.І., Сколець С.С. Дослідження мереж SDN з використанням платформи mininet. *Перспективи телекомунікацій*: матеріали п'ятнадцятої Міжнародної науково-технічної конференції м. Київ, Україна, 12-16 квітня 2021 р. / КПІ ім. Ігоря Сікорського, Київ, Україна, 2021. С. 98–101.

7. Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi

Kobayashi, Toshio Koide, Bob Lantz, Brian O'Connor, Pavlin Radoslavov, William Snow, Guru Parulkar. ONOS: towards an open, distributed SDN OS. *HotSDN '14: Proceedings of the third workshop on Hot topics in software defined networking* / Association for Computing Machinery, New York, USA, 2014. P. 1–6.

8. Коренко Д.В., Череватенко О.В., Русінов В.В., Кулаков Ю.О. Розробка способу багатошляхової маршрутизації з використанням раніше відомих маршрутів у програмно конфігурованих мережах. *Technology Audit and Production Reserves*. 2022. В. 4, № 2(66). С. 19–24.

9. Kulakov Y., Kohan A., Kopychko S., Cherevatenko R. Load Balancing in Software Defined Networks Using Multipath Routing. *Advances in Computer Science for Engineering and Education III. ICCSEEA 2020. Advances in Intelligent Systems and Computing*. 2021. V. 1247. P. 384–395.

Кулаков Ю.О., Череватенко О.В.

МОДЕЛЮВАННЯ МАРШРУТИЗАЦІЇ З ВИКОРИСТАННЯМ ВІДОМИХ МАРШРУТІВ ЗА ДОПОМОГОЮ SDN-КОНТРОЛЕРА ONOS

У даній статті виконується опис моделювання локального мережевого оточення, яке використовує протокол OpenFlow для зв'язку між ланками системи та контролер ONOS, який виконує управління мережею на основі технології мереж, що програмно конфігуруються (SDN).

В ході дослідження побудована віртуальна модель мережі на платформі mininet з довільною топологією. На основі заданої моделі контролер зчитує інформацію про пристрої в оточенні та формує представлення про мережу у своїй пам'яті. Для знаходження шляхів використаний розроблений авторами спосіб багатошляхової маршрутизації з використанням відомих маршрутів (із попередніх досліджень зв'язків на топології, які виконувались контролером). Цей спосіб має одну з найменших часових складностей серед інших методів. Для контролера проведена модифікація програмного алгоритму маршрутизації, якою керує контролер, відповідно до обраного теоретичного методу. Виконано тестування на побудованій моделі та перевірка теоретичних припущень. У якості метрики використовується кількість переходів («хопів») між ланками мережі.

***Ключові слова**: SDN, mininet, багатошляхова маршрутизація, віртуальна мережа, відомі маршрути, ONOS.*

Kulakov Y.O., Cherevatenko O.V.

MODELING OF THE MULTIPATH ROUTING USING KNOWN PATHS WITH ONOS SDN CONTROLLER

This article describes the modeling of the local network environment, which uses the OpenFlow protocol for communication between links in the system and the ONOS controller, which performs network management based on software defined networks (SDN) technology.

In the course of the study, a virtual network model was built using the mininet platform with an arbitrary topology. The controller reads information about devices in the environment based on the given model and forms a representation of the network in its memory. To find paths, method of multipath routing using known paths (from previous studies of connections on the topology performed by the controller) developed by the authors of the article was used. This method has one of the lowest time complexities among other methods. For the controller, a modification of the software routing algorithm, which is administered by the controller, was carried out in accordance with the selected theoretical method. Testing of the model that was built and verification of theoretical assumptions were performed. The number of transitions ("hops") between network links is used as a metric.

***Keywords**: SDN, mininet, multipath routing, virtual network, known paths, ONOS.*