УДК 621.391

МОДЕЛЮВАННЯ МЕРЕЖІ SDN З ВИКОРИСТАННЯМ ВІДКРИТОЇ МЕРЕЖЕВОЇ ОПЕРАЦІЙНОЇ СИСТЕМИ ONOS

Романов О.І., Сколець С.С., Марінов А.І.
Навчально-науковий Інститут телекомунікаційних систем КПІ ім. Ігоря Сікорського, Україна
E-mail: a_i_romanov@ukr.net, serskols@gmail.com, anton_marinov@ukr.net

SDN NETWORK SIMULATION USING ONOS OPEN NETWORK OPERATING SYSTEM

This article demonstrates how to modify the parameters of network components and communication channels while describing network topology, as well as test the capabilities of a network that is subordinated to ONOS controller with specified network parameters, such as bandwidth and maximum queue-size.

У роботі досліджені параметри пропускної здібності мережі SDN, на базі використання програмних продуктів Mininet та ONOS, які є у відкритому доступі. Показано як змінювати параметри мережевих компонентів та каналів зв'язку при описі топології мережі. Проаналізовано вплив різних протоколів та розмір черги на пропускну здібність мережі.

Оскільки Mininet являється загальною платформою-емулятором, то при запуску мережі без зазначення мережевих параметрів вона не відповідає дійсній SDN мережі. Тому з початку потрібно зазначити низку параметрів і провести декілька кроків налаштування, щоб мережа відповідала вимогам реальної SDN мережі[1].

В нашому випадку буде використовуватись топологія мережі (рис. 1), яка написана на вищому рівні Mininet API (Python API). Топологія Mininet описана в окремому файлі розширення ".py" з залученням сторонніх бібліотек для більш гнучких налаштувань мережі. В роботі була використана бібліотека TCLink, яка надає змогу задавати пропускну здібність, затримку, джиттер у каналах зв'язку, а також RemoteController для описання ONOS контролера, якій в подальшому буде підключений до мережі в якості централізованого органу управління[2][6].

Конфігураційний файл мережі/топології має наступний вигляд:

```
#!/usr/bin/python
## імпорт бібліотек python
from mininet.topo import Topo
from mininet.net import Mininet
from mininet.link import TCLink
from mininet.node import Controller, OVSSwitch, RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel

def multiControllerNet():
    ##  додання мережевих елементів
    net = Mininet( controller=RemoteController, switch=OVSSwitch, link=TCLink )
    c1 = RemoteController( 'c1', ip='172.17.0.6', port=6653 )
    s10 = net.addSwitch( 's10', dpid='0000000000000010', mac='00:00:00:00:SW:10' )
    ...
    h11 = net.addHost( 'h11', ip='10.0.0.11/24', mac='00:00:00:00:0H:11' )
    ...

    ## додання мережевих зв'язків
    net.addLink( s10, s20, bw=100)
    ...

    ## запуск мережевих компонентів
    net.start()
    c1.start()
    s10.start( [ c1 ] )
    ...
    CLI( net )
if __name__ == '__main__':
    setLogLevel( 'info' )  # for CLI output
    multiControllerNet()
```

Зазначений код описує процес запуску основних елементів мережі. Цей процес складається з наступних етапів:

- імпорт бібліотек python, що надають змогу компілятору python розуміти синтаксис опису саме топології Mininet;
- додання мережевих елементів: комутаторів з підтримкою OpenFlow, хостів, а також задання їм унікальні адреси та ідентифікатори в мережі, описання розташування контролеру та порт на якому він приймає включення;
- додання мережевих зв'язків - підключення комутаторів між собою, а також підключення хостів до комутаторів;
- запуск мережевих компонентів, Mininet консолі, а також підключення кожного комутатора до контролера ONOS.

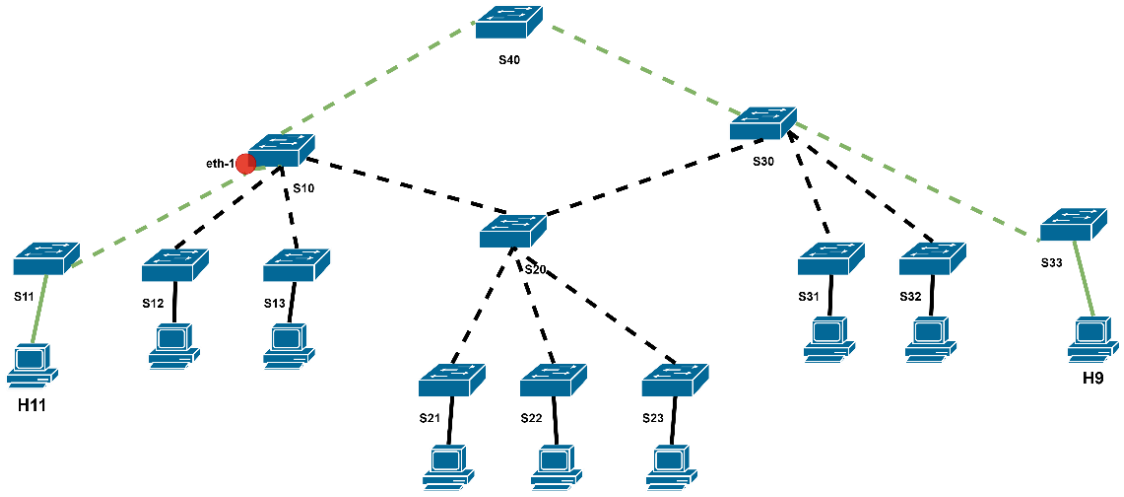

Рис.1. Топологія SDN мережі.

Після запуску мережі необхідно провести тестування параметрів функціонування мережі. В роботі проведено тестування пропускної здатності каналів зв'язку у напрямку від h11 до h9. Для цього, організована передача UDP трафіку при різному навантаженні каналів зв'язку і при різній швидкості передачі. При цьому були використані наступні вихідні данні:

- топологія мережі, що зображена на рис.1;
- інформаційний напрямок зв'язку між хостами h11 - h9;
- кількість проміжних вузлів(комутаторів) – 3;
- навантаження в пакетах, що передається – 500 Мбайт;
- швидкість передачі у лініях зв'язку – 100 Мбіт/с.

У ході проведення досліду було знято часові діаграми швидкості передачі UPD трафіку. Крок підвищення навантаження у лініях зв'язку становить: 10%, 50%, 100% від максимально заданої швидкості передачі лінії зв'язку. Тобто 10, 50, 100Мбіт/с, відповідно. Часові діаграми наведені на рис. 2.

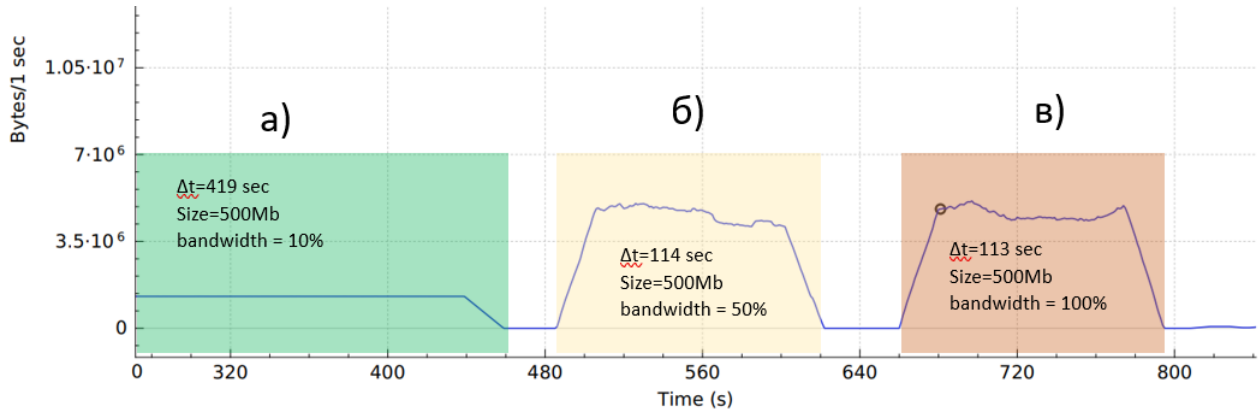

Рис. 2. Швидкість передачі трафіку каналом зв'язку при: а) навантаження 10%; б) навантаження 50%; в) навантаження 100%.

Наступним етапом тестування мережі стала перевірка характеру поведінки мережі шляхом зняття показників функціонування при зміні показника максимального буферу пакетів в черзі (queue size). Відповідне дослідження дало змогу перевірити особливості передачі TCP/UDP трафіку.

Зміна довжини буферу пакетів проводилась на всіх інтерфейсах в напрямку передачі трафіку наступним чином: 1, 5, 10, 25, 50, 75, 100, 200, 500 пакетів.

Результати поведінки мережі відображено на рис. 3

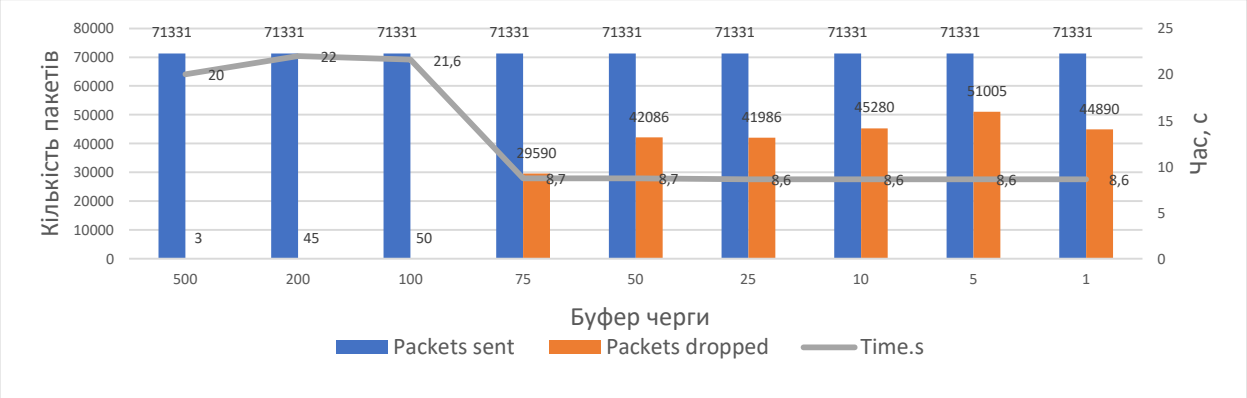

Рис. 3а. Параметри трафіку при передачі за протоколом UDP.

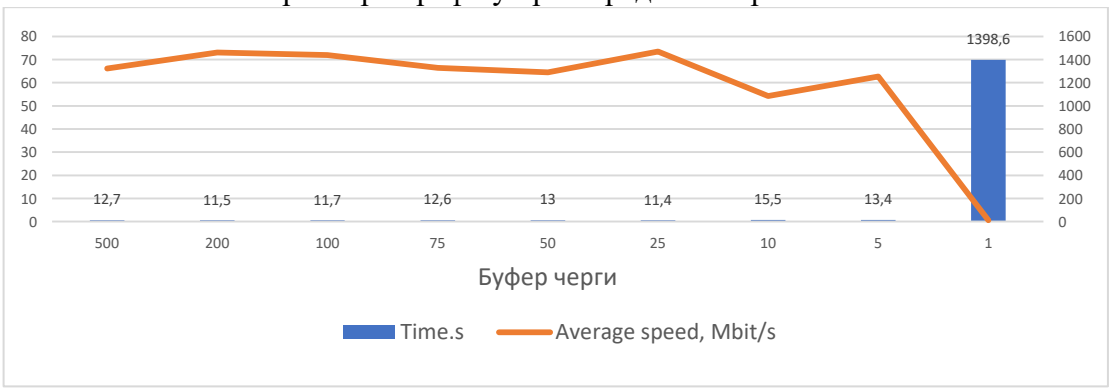

Рис. 3б. Параметри трафіку при передачі за протоколом TCP.

Проведені дослідження дозволяють визначити де які особливості. Побудова мережі SDN довільної топології вимагає додаткового написання програмного коду на мові python. Це дає можливість створювати необхідні зв'язки як між елементами мережі, так і між елементами керування мережею.

В роботі, використовуючи побудовану топологію та перераховані вихідні дані, було знято часові діаграми при навантаженні напрямку зв'язку. Відображені результати показали, що при навантаженні лінії зв'язку до 50%, передача відбувається з постійною швидкістю без суттєвих змін, що не можна сказати при навантаженні більше 50%. В такому випадку, характер швидкості непередбачуваний, хоча в обох випадках, переданий об'єм трафіку доставляється без втрат[4].

Проводячи аналіз досліду, при зміні queue-size, можна зробити висновки: При передачі UDP трафіку, встановивши значення max_queue_size – 100 пакетів і більше, втрати майже непомітні (досягають 0,07% максимум), однак при зменшенні буферу черги від 1 до 100 пакетів, втрати можуть бути суттєвими, тобто 72-41%.

При передачі TCP трафіку, втрати не спостерігаються, хоча швидкість передачі даних зменшується відповідно до зменшення параметру буфера (max_queue_size). Це пояснюється головною особливістю TCP трафіку, а саме наявністю підтверджень (ACK) від отримувача, які повідомляють відправника про успішну доставку кожного пакету[3][5]. Якщо відправник не отримує підтвердження вчасно, він повторно надсилає пакет до отримувача. У UDP така гарантія відсутня, тому пакети можуть бути втрачені в разі перевантаження мережі або інших проблем з передачею даних.

Треба відмітити суттєвий недолік проведених досліджень на базі цих програмних продуктів. При моделюванні ми не змогли зробити балансування трафіку. Тому коли з'являється перевантаження в напрямку зв'язку і при цьому є наявний вільний ресурс в мережі, контролер ONOS не реагує на перевантаження і не змінює шляхи передавання трафіку. Це говорить о том, що ця функція чи поки що відсутня у цьому програмному продукті, чи ми не змогли її підключити.

Література

1. Документація ONOS: https://wiki.onosproject.org/display/ONOS/Documentation
2. S. Rathee, S. Rathee, K. Haribabu. GlobeSnap: An Efficient Globally Consistent Statistics Collection for Software-Defined Networks/Journal of Network and Systems Management 2021, 29(3)/DOI:10.1007/s10922-021-09601-z
3. Romanov, O., Nesterenko, M., Mankivskyi, V., Zhuk, O. Principles of Building Modular Control Plane in Software-Defined Network//Lecture Notes in Networks and Systems, 2023, 548, pp. 333–355, DOI: 10.1007/978-3-031-16368-5_17
https://link.springer.com/chapter/10.1007/978-3-031-16368-5_17
4. Romanov, O., Korniienko N., Svyd I., Romanov A. Optical Network Management by ONOS-Based SDN Controller, September 2022, Radiotekhnika, DOI:10.30837/rt.2022.3.210.16
5. Pedro Sousa, Gonçalo Pereira, Jose Silva. Comparative Study of Software-Defined Networking (SDN) Traffic Controllers, Jun 2019, 2019 14th Iberian Conference on Information Systems and Technologies (CISTI),DOI: 10.23919/CISTI.2019.8760997
6. Romanov, O., TT Dong., Nesterenko, M. The possibilities for deployment eco-friendly indoor wireless networks based on LiFi technology/(2020) Proceedings of International Conference on Applied Innovation in IT, 8 (1), pp. 41-48.