УДК 621.391

РОЗГОРТАННЯ ONOS КЛАСТЕРУ ДЛЯ ЕМУЛЯЦІЇ РОБОТИ КОРИСТУВАЦЬКОЇ МЕРЕЖІ SDN

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

ONOS CLUSTER DEPLOYMENT FOR EMULATION OF CUSTOM SDN NETWORK

This article outlines the process of building a custom network based on an ONOS cluster, which involves deploying ONOS controllers and installing them on cluster nodes, creating a virtual user network, configuring and managing network services and traffic on ONOS using a graphical interface or API, collecting and analyzing network data to verify network operation.

У даній статті відображено процес побудови користувацької мережі на базі ONOS кластеру, що передбачає розгортання контролерів ONOS та встановлення їх на вузлах кластеру, створення віртуальної користувацької мережі, налаштування та керування мережевими сервісами та трафіком в ONOS з використанням графічного інтерфейсу або API, збір та аналіз мережевих даних з метою перевірки роботи мережі.

Побудова користувацької мережі на базі ONOS та Mininet має на меті дослідження та тестування різноманітних мережевих сценаріїв. Мінімізуючи витрати на обладнання, можна створити віртуальну мережу, що дозволить вирішити проблеми на різних рівнях мережі. Також даний підхід дозволяє проводити дослідження та тестування різних аспектів мережевих технологій, таких як розподілені протоколи маршрутизації, стратегії керування трафіком, збір статистики про мережу та інші.

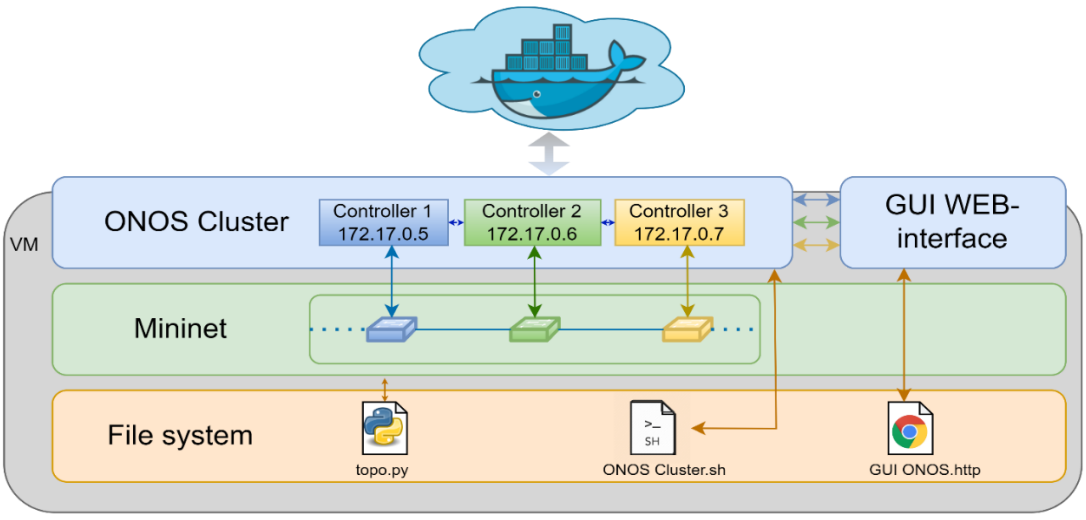

Рис.1. Програмні компонентів мережі на базі ONOS кластеру.

Це дає можливість вирішити проблеми, пов'язані з недостатньою ефективністю та надійністю мережі, виявити проблеми в роботі мережі та знайти їх вирішення, визначити місця заторів та вузькі місця, що може допомогти зробити мережу більш ефективною та надійною. Крім того, це дозволяє відтворити різні сценарії, що можуть статися в реальних мережах, та протестувати різні методи їх вирішення.

Основний процес побудови користувацької мережі на базі ONOS кластеру включає кілька етапів, що відображені на рис. 1.

```
#!/usr/bin/python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.link import TCIntf
from mininet.link import TCLink
from mininet.node import Controller, OVSSwitch, RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel

def multiControllerNet():
#    Create a network from semi-scratch with multiple controllers

    net = Mininet( controller=Controller, switch=OVSSwitch, link=TCLink )

    print '*** Creating switches'
    s1 = net.addSwitch( 's1' )
    s2 = net.addSwitch( 's2' )
    s3 = net.addSwitch( 's3' )
    s4 = net.addSwitch( 's4' )


    print '*** Creating hosts'
    h1 = net.addHost( 'h1' )
    h2 = net.addHost( 'h2' )
    h3 = net.addHost( 'h3' )
    h4 = net.addHost( 'h4' )


    print '*** Creating links'
    net.addLink( s1, s2 )
    net.addLink( s1, s3 )
    net.addLink( s2, s3 )
    net.addLink( s2, s4 )
    net.addLink( s3, s4 )
    net.addLink( s1, h1 )
    net.addLink( s2, h2 )
    net.addLink( s3, h3 )
    net.addLink( s4, h4 )


    print '*** Creating (reference) controllers'
    c1 = RemoteController( 'c1', ip='172.17.0.5', port=6653 )
    c2 = RemoteController( 'c2', ip='172.17.0.6', port=6653 )
    c3 = RemoteController( 'c3', ip='172.17.0.7', port=6653 )


    print '*** Starting network'
    net.start()
    c1.start()
    c2.start()
    c3.start()

    s1.start( [ c1, c2, c3 ] )
    s2.start( [ c1, c2, c3 ] )
    s3.start( [ c1, c2, c3 ] )
    s4.start( [ c1, c2, c3 ] )


    print '*** Running CLI'
    CLI( net )

if __name__ == '__main__':
    setLogLevel( 'info' )  # for CLI output
    multiControllerNet()
```

Рис.2. Опис мережевої топології на API Mininet у вигляді python-файлу.

Етап 1. Починаючи з розгортання контролерів ONOS, першим кроком є встановлення необхідного програмного забезпечення на вузлах кластеру, на яких будуть працювати контролери. Для цього знадобиться комп’ютер із не менше 8 ГБ оперативної пам’яті та не менше 20 ГБ вільного місця на жорсткому диску. Швидший процесор або твердотільний накопичувач пришвидшить час завантаження віртуальної машини. Комп’ютер може працювати під керуванням Windows, Mac OS X або Linux – усі добре працюють із VirtualBox, єдиною програмною вимогою.

Запуск кластера зазвичай відбувається за допомогою інструментів контейнеризації, таких як Docker або Kubernetes. Кожен контролер

запускається як окремий контейнер, і всі контейнери підключаються до мережі, якою вони керують. Зв'язок між віртуальною машиною та контейнерами здійснюється через мережеву взаємодію. Запуск кластеру в нашому випадку виконується за допомогою скрипта, котрий під час запуску звертається до Docker Hub та завантажує контейнери контролерів на нашу віртуальну машину, компілює їх і запускає.

Етап 2. Після встановлення контролерів, необхідно створити віртуальну користувацьку мережу (див. рис. 3), а саме новий файл топології (див. рис. 2) користувацької мережі Mininet з розширенням ".py", наприклад: «topo.py». У якому слід визначити топологію мережі за допомогою API Mininet, вказавши кількість хостів, комутаторів, налаштувавши зв'язки між елементами мережі, а також прописати порядок запуску компонентів мережі. Далі потрібно визначити контролер ONOS, до якого потрібно підключитися, використовуючи також API. Запуск написаної топології, можна виконати шляхом виконання, у терміналі віртуальної машини, команди:

```
>> sudo python topo.py
```

Наповнення файлу «topo.py» виглядає наступним чином:

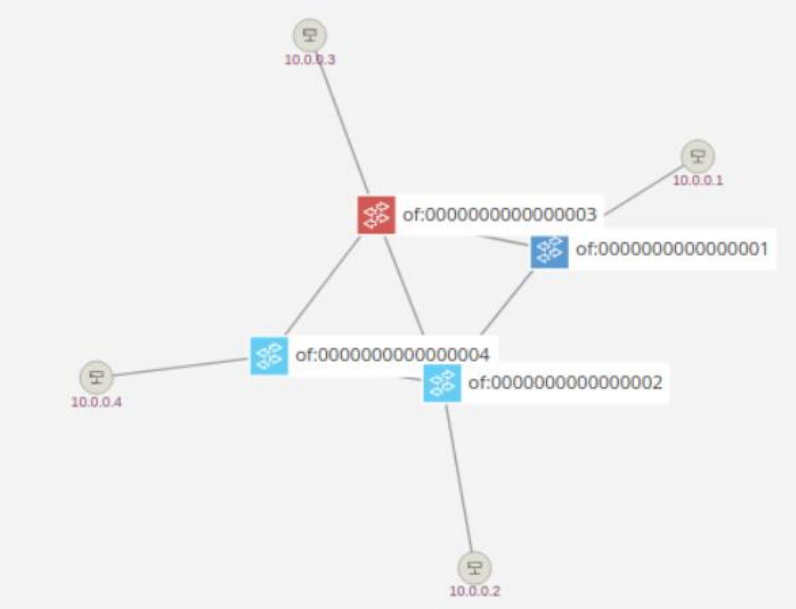

Рис.3. Користувацька топологія та підпорядковування комутаторів різним контролерам в кластері.

Етап 3. Після виконання команди вище, кожен контролер ONOS підключається до мережі і отримує інформацію про стан мережі та мережевий трафік. Інформація обмінюється між контролерами за допомогою протоколу OpenFlow або NETCONF. Таким чином, контролери працюють як розподілені системи, що спільно керують мережею і забезпечують її ефективну роботу.

Етап 4. Після створення мережі, необхідно виконати налаштувати мережевих сервісів та трафіку в ONOS. Для цього можна використовувати графічний інтерфейс або API, який дозволяє отримувати доступ до

різноманітних функцій та сервісів. Наприклад, за допомогою API можна налаштовувати правила маршрутизації, керувати розподіленням навантаження на вузлах мережі, налаштовувати мережеві сервіси тощо.

Графічний інтерфейс ONOS надає можливість керувати різними мережевими додатками, які можуть бути використані для управління мережею та її ресурсами. Основні мережеві додатки, які можна вмикати за допомогою графічного інтерфейсу ONOS, описані нижче:

- Host Location: цей додаток дозволяє відстежувати місцезнаходження хостів у мережі та контролювати їх рух.
- Intent: цей додаток використовується для створення та керування цілями (intents), які можуть бути використані для налаштування мережі та ресурсів.
- Link Connectivity: цей додаток дозволяє відстежувати зв'язок між різними вузлами у мережі та забезпечує стійкість мережі.
- OpenFlow: цей додаток використовується для встановлення зв'язку між контролером та OpenFlow-сумісними комутаторами.

Узагальнюючи результати роботи, було описано побудову користувацької мережі на базі ONOS кластеру з залученням платформи емуляції SDN Mininet. Було розглянуто процес побудови кластеру ONOS, системні вимоги та варіант розгортання на Linux. Було відображено процес описання топології віртуальної мережі за допомогою мови python на одному з рівнів Mininet API, підключення мережі до віддаленого контролера. Описані мережеві сервіси, що можна вмикати через графічний інтерфейс контролера та API. Також було наведено можливі проблеми, які можна вирішити за допомогою побудови мережі на базі ONOS та Mininet.

Література

1. Bhargavi Goswami. Experimenting with ONOS Scalability on Software Defined Network, January 2019, Journal of Advanced Research in Dynamical and Control Systems, ISBN: 1943-023X.
2. James Akerele. PERFORMANCE ANALYSIS OF OPENFLOW-ENABLED NETWORK TOPOLOGIES: OPENFLOW SDN VS TRADITIONAL NETWORK, January 2018, DOI:10.13140/RG.2.2.33523.84008.
3. Luca Boero, M. Cello, Mario Marchese, C. Garibotto. BeaQoS: Load balancing and deadline management of queues in an OpenFlow SDN switch, September 16, ISBN: 1389-1286.
4. Oleksandr Ivanovich Romanov, Ivan Saychenko, Anton Marinov, Serhii Skolets. RESEARCH OF SDN NETWORK PERFORMANCE PARAMETERS USING MININET NETWORK EMULATOR, June 2021, Information and Telecommunication Sciences, DOI:10.20535/2411-2976.12021.24-32.
5. Romanov, O., Korniienko, N., Burlaka, H. Construction of the SDN Transport Network Model using the T-API Interface/2021 IEEE 4th International Conference on Advanced Information and Communication Technologies, AICT 2021 - Proceedings, 2021, стр. 220–224.
6. Romanov, O., Siemens, E., Nesterenko, M., Mankivskyi, V. Mathematical description of control problems in SDN networks/Proceedings of International Conference on Applied Innovation in IT, 2021, 9(1), стр. 33–39.