НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут телекомунікаційних систем
(повна назва інституту/факультету)

«Кафедра Телекомунікаційних систем»
(повна назва кафедри)

«На правах рукопису»

УДК _____

«До захисту допущено»

Завідувач кафедри
_____ Л.О.Уривський
(підпис) (ініціали, прізвище)

“___”_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 172 Телекомунікації та радіотехніка

на тему: «Дослідження архітектури побудови центрів обробки даних на базі технології SDN»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ТС-61м
Роговий Віталій Петрович _____

Керівник:
ст. викладач кафедри ТС, к.т.н. Лісковський І.О. _____

Рецензент:

_____

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

Київ – 2018 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем

Кафедра Телекомунікаційних систем

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність (спеціалізація) – 172 «Телекомунікації та радіотехніка» (172.3620.1 «Телекомунікаційні системи та мережі»)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Л.О. Уривський

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Роговому Віталію Петровичу

1. Тема дисертації «Дослідження архітектури побудови центрів обробки даних на базі технології SDN», Лісковський Ігор Олегович, ст. викладач кафедри ТС, к.т.н., затверджені наказом по університету від «___»_____20__р. №____

2. Термін подання студентом дисертації «___»_____ 20__ р.
3. Об'єкт дослідження програмно-визначені мережі
4. Предмет дослідження використання програмно-визначених мереж в центрах обробки даних
5. Перелік завдань, які потрібно розробити:

1) Дослідити референсну архітектуру програмно-визначених мереж та її складових;

2) Дослідити референсну архітектуру побудови центрів обробки даних;

3) Дослідити архітектури побудови центрів обробки даних на базі технології програмно визначених мереж;

4) Запропонувати програмну реалізацію переадресації (маршрутизації) трафіку в мережі на базі технології SDN;

5. Перелік ілюстративного матеріалу

1) Тема, мета, актуальність;

2) Архітектури SDN та додатку;

3) Вимоги при побудові ЦОД та до мережі ЦОД;

4) SDN в центрі обробки даних;

5) Використання SDN в ЦОД: «Big Data» та міграція ВМ;

6) Використання SDN в ЦОД: накладена та транспортна мережі, централізація площини управління;

7) Програмна реалізація SDN;

8) Висновки.

7. Перелік публікацій

1) Роговий В.П. «Загрози безпеці у SDN середовищах», Науково-технічна конференція "Проблеми телекомунікацій": Матеріали конференції. К.: НТУУ "КПІ", 2018. – с. 146-148;

8. Дата видачі завдання_____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування актуальності дослідження архітектури побудови ЦОД на базі технології SDN	1 вересня 2016р. – 31 жовтня 2016р.	
2	Дослідження референсної архітектури програмно-визначених мереж та її складових	1 листопада 2016р. – 31 грудня 2016р.	
3	Дослідження референсної архітектури побудови центрів обробки даних	1 січня 2017р. – 31 травня 2017р.	
4	Дослідження використання технології SDN в центрах обробки даних	1 червня 2017р. – 30 листопада 2017р.	
5	Розробка програмної реалізації переадресації трафіку на базі SDN	1 грудня 2017р. – 30 квітня 2018р.	
7	Написання висновків по дипломній роботі, оформлення роботи	1 травня 2018р. – 10 травня 2018р.	

Студент

Науковий керівник дисертації

РЕФЕРАТ

Текстова частина дипломної роботи: 121 с., 37 рис., 23 джерела, 4 додатки.

Актуальність роботи. Потрібність в динамічному розподілі ресурсів, постійне збільшення трафіку, викликане зростанням мобільного доступу, попитом на відеосервіси, використанням хмарних обчислень призводить до того, що сучасні дата-центри повинні комплексно міняти свою ресурсну базу. Дата-центри повинні мати можливість обробляти нестаціонарний обсяг трафіку і здійснювати великий обсяг паралельних транзакцій, вміти глибоко аналізувати дані, бути легко переконфігурованими під мінливі запити. За умови обмеженості ресурсів, ЦОД не можуть нескінченно розростатися.

Рішення більшості цих проблем було знайдено в віртуалізації, а саме у використанні технології програмно-визначених мереж (SDN - Software-Defined Networking), що була розроблена спільно університетами Берклі і Стенфорда у 2005 році.

Мета роботи — провести дослідження архітектури побудови центрів обробки даних на базі технології програмно-визначених мереж (SDN), проаналізувати референсну архітектуру SDN та референсну архітектуру сучасних центрів обробки даних, яку пропонують різні компанії-виробники, запропонувати програмну реалізацію переадресації (маршрутизації) трафіку в мережі на базі технології SDN.

Об'єкт дослідження — програмно-визначені мережі.

Предмет дослідження — використання програмно-визначених мереж в центрах обробки даних.

Методи дослідження. Для досягнення поставлених в дисертаційній роботі задач використано методи теорії інформації, теорії оптимального управління, системного аналізу, елементи теорії ієрархічних багаторівневих систем.

Наукова новизна отриманих результатів. Проведено дослідження архітектури побудови центрів обробки даних на базі технології програмно-визначених мереж (SDN) та проаналізовано референсну архітектуру SDN та референсну архітектуру сучасних центрів обробки даних, яку пропонують різні компанії-виробники. Крім

цього представлена програмна реалізація переадресації трафіку на базі технології SDN.

Публікації. Роговий В.П. «Загрози безпеці у SDN середовищах», Науково-технічна конференція "Проблеми телекомунікацій": Матеріали конференції. К.: НТУУ "КПІ", 2018. – с. 146-148;

ПРОГРАМНО-ВИЗНАЧЕНІ МЕРЕЖІ, ДАТА-ЦЕНТР, ЦЕНТР ОБРОБКИ ДАНИХ, ВІРТУАЛІЗАЦІЯ, ВІРТУАЛЬНІ МАШИНИ, АРХІТЕКТУРА, МЕРЕЖА, МІГРАЦІЯ, ЦЕНТРАЛІЗАЦІЯ, OPENFLOW, VXLAN, NVGRE, SDN, STP, ТРАФІК.

ABSTRACT

The work contains 121 pages, 37 figures, 4 applications. 23 sources have been used.

Work actuality. The need for a dynamic distribution of resources, a constant increase in traffic due to the growth of mobile access, demand for video services, the use of cloud computing leads to the fact that modern datacenters must complexly change its resource base. Data centers should be able to process nonstationary traffic and implement a large amount of parallel transactions, be able to deeply analyze data, be easily reconfigured for changing requests. Also data centers can not infinitely grow due to limited resources.

The solution to most of these problems was found in virtualization, namely the use of Software-Defined Networking (SDN) technology, developed jointly by the Berkeley and Stanford universities in 2005.

The purpose of the work is to study the architecture of the building the data centers based on the technology of software-specific networks (SDN), to analyze the reference SDN architecture and the reference architecture of modern data centers that offers by various companies, to propose the software implementation of the redirection

(forwarding) of traffic in the network based on SDN technology.

Object of research - software-defined networks.

Subject of research - using of software-defined networks in data centers.

Research methods. To achieve the goals set in the dissertation methods of information theory, theory of optimal control, system analysis, elements of the theory of hierarchical multilevel systems were used.

Scientific novelty of the obtained results. The study of the architecture of the building of data centers based on the technology of software-defined networks (SDN) was conducted, reference architecture of SDN and the reference architecture of modern data centers, which are offered by various manufacturers, were analyzed. In addition, the software implementation of traffic forwarding based on SDN technology was presented.

Publications: Роговий В.П. «Загрози безпеці у SDN середовищах», Науково-технічна конференція "Проблеми телекомунікацій": Матеріали конференції. К.: НТУУ "КПІ", 2018. – p. 146-148;

SOFTWARE-DEFINED NETWORKING, DATA CENTER, VIRTUALIZATION, VIRTUAL MACHINES, ARCHITECTURE, NETWORK, MIGRATION, CENTRALIZATION, OPENFLOW, VXLAN, NVGRE, SDN, STP, TRAFFIC.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ARP	Address Resolution Protocol — протокол визначення адрес
BFD	Bidirectional Forwarding Detection - мережевий протокол для виявлення помилок з'єднання між двома маршрутизаторами шляхом передачі посилання з обох сторін
BPDU	Bridge Protocol Data unit - фрейм (одиниця даних) протоколу управління мережевими мостами, IEEE 802.1d
BSD	Berkeley Software Distribution - назва кількох POSIX-сумісних операційних систем сімейства UNIX
COTS	Commercial off-the-shelf - термін, використовуваний для опису покупок продуктів, які є стандартними промисловими товари, а не переробленими або зробленими на замовлення, виробами
CSS	Cascading Style Sheets - каскадні таблиці стилів
DHCP	Dynamic Host Configuration Protocol - протокол динамічної конфігурації вузла
DWDM	Dense Wavelength Division Multiplexing - щільне спектральне мультиплексування
EoR	End of Row – модель комутації, яка передбачає розташування комутатора, умовно, «в кінці ряду стійок» і обслуговування їм трафіку з з усіх серверів з декількох стійок, розташованих в ряд.
EVB	Edge Virtual Bridging – стандарт IEEE, який включає в себе взаємодію між віртуальними середовищами комутації в гіпервізорі і першим рівнем фізичної інфраструктури комутації

GENEVE	*Generic Network Virtualization Encapsulation – універсальний протокол інкапсуляції для віртуалізованих мереж*
GNU	GNU's Not Unix – вільна UNIX-подібна операційна система, що розробляється проектом GNU
GRUB	GRand Unified Bootloader - програма-завантажувач операційних систем
HTML	HyperText Markup Language - мова розмітки гіпертекстових документів
HTTP	HyperText Transfer Protocol - протокол передачі гіпертексту
ICMP	Internet Control Message Protocol — міжмережевий протокол керуючих повідомлень
IEEE	Institute of Electrical and Electronics Engineers - Інститут інженерів з електротехніки та електроніки
IP	Internet Protocol – Інтернет протокол
IRTF	Internet Research Task Force - дослідницька група Інтернет-технологій
IT	Information Technology - інформаційні технології
JS	JavaScript - прототипно-орієнтована сценарна мова програмування
LACP	Link Aggregation Control Protocol - протокол, призначений для об'єднання кількох фізичних каналів в один логічний в мережах Ethernet
LAN	Local Area Network – локальна мережа
LISP	Locator/Identifier Separation Protocol - протокол маршрутизації, який відокремлює простір IP адрес мережевих пристроїв від простору IP адрес кінцевих хостів

LLDP	Link Layer Discovery Protocol - протокол канального рівня, що дозволяє мережевому обладнанню оповіщати обладнання, яке працює в локальній мережі, про своє існування і передавати йому свої характеристики, а також отримувати від нього аналогічні відомості
MAC	Media Access Control - управління доступом до середовища
MPLS	MultiProtocol Label Switching - багатопротокольна комутація по мітках
MSTP	Multiple Spanning Tree Protocol – розширення протоколу RSTP
NE	Networking Element – мережевий елемент
OSCI	Open Source Cloud Initiative – відкрита хмарна ініціатива
OSI	Open Systems Interconnection - взаємодія відкритих систем
OSPF	Open Shortest Path First - протокол динамічної маршрутизації, заснований на технології відстеження стану каналу (link-state technology), що використовує для знаходження найкоротшого шляху
OTN	Optical Transport Network - оптична транспортна мережа
PCI	*Peripheral Component Interconnect* - взаємозв'язок периферійних компонентів
pNIC	Physical Network Interface Card – фізичний мережевий адаптер
PCEP	Path Computation Element Protocol - спеціальний набір правил, який дозволяє Path Computation Client запросити шлях обчислення з Path Computation Element
QEMU	Quick Emulator - вільна програма з відкритим вихідним кодом для емуляції апаратного забезпечення різних платформ
QoS	Quality of Service - якість обслуговування

REST	Representational State Transfer - «передача стану представлення» - архітектурний стиль взаємодії компонентів розподіленого додатка в мережі
RSTP	Rapid Spanning Tree Protocol - швидкий *протокол* сполучного дерева
SAN	Storage Area Network – мережа зберігання даних
SNMP	Simple Network Management Protocol - простий протокол мережевого управління
SOAP	Simple Object Access Protocol - простий протокол доступу до об'єктів
SSH	Secure Shell - «безпечна оболонка» - мережевий протокол рівня застосунків, що дозволяє проводити віддалене управління комп'ютером і тунелювання TCP-з'єднань
STP	Spanning Tree Protocol - *протокол* сполучного дерева
STT	Stateless Transport Tunneling – тунельний протокол
TCAM	Ternary Content Addressable Memory - троїчна асоціативна пам'ять
TCP	Transmission Control Protocol - протокол управління передачею
TLS	Transport Layer Security - безпека транспортного рівня
ToR	Top-of-Rack - модель комутації, коли в кожній стійці стоїть комутатор, який обробляє трафік з серверів в цій стійці, і з'єднаний з комутатором ядра або з рівнем агрегації
UDP	User Datagram Protocol - протокол датаграм користувача
URI	Universal Resource Identifier - уніфікований ідентифікатор ресурсів - компактний рядок літер, який однозначно ідентифікує окремий абстрактний чи фізичний ресурс
VLAN	Virtual Local Area Network - віртуальна локальна

	комп'ютерна мережа
VM	Virtual Machine – віртуальна машина
VRRP	*Virtual Router Redundancy Protocol - мережевий протокол, призначений для збільшення доступності маршрутизаторів що виконують роль шлюзу за замовчуванням*
VSS	Virtual Switching System – віртуальна система комутації
WMI	*Windows Management Instrumentation - інструментарій управління Windows*
ME	Мережевий елемент
ОС	Операційна система
ПЗ	Програмне забезпечення
СУБД	Система управління базою даних

ВСТУП

Актуальність. Центр обробки даних (ЦОД) є мінливою системою, що складається з тисяч програмних компонент і інфраструктурних складових з гетерогенними компонентами і зв'язками між ними. Реалізація ЦОД може складатися з декількох локацій рознесених географічно. Управління таким ЦОД, в реальному режимі часу, є технічно складним завданням пов'язаним із наявністю великого числа незалежних компонент.

Потрібність в динамічному розподілі ресурсів, постійне збільшення трафіку, викликане зростанням мобільного доступу, попитом на відеосервіси, використанням хмарних обчислень призводить до того, що сучасні дата-центри повинні комплексно міняти свою ресурсну базу. Дата-центри повинні мати можливість обробляти нестаціонарний обсяг трафіку і здійснювати великий обсяг паралельних транзакцій, вміти глибоко аналізувати дані, бути легко перебудовуватися під мінливі запити. За умови обмеженості ресурсів, ЦОД не можуть нескінченно розростатися.

Рішення більшості вищеназваних проблем було знайдено в віртуалізації, а саме у використанні технології програмно-визначених мереж (SDN - Software-Defined Networking), що була розроблена спільно університетами Берклі і Стенфорда у 2005 році.

SDN повністю централізує керування мережею центру обробки даних і дозволяє конфігурувати/контролювати/оптимізувати будь-які мережеві процеси за допомогою окремо встановлених програмно реалізованих додатків за порівняно короткий час. Така автоматизація в центрі обробки даних приводить до того, що повністю змінюється логіка роботи всього дата-центру, а саме апаратне забезпечення переноситься на друге місце – перше місце тепер займає контролююче програмне забезпечення.

Через це весь центр обробки даних можна розглядати з точки зору типової архітектури будь-якого додатку (рис. А), де SDN відповідає за рівні

представлення, сервісів та бізнес-рівень, а мережа центру обробки даних – за рівні доступу к даним та джерела даних.

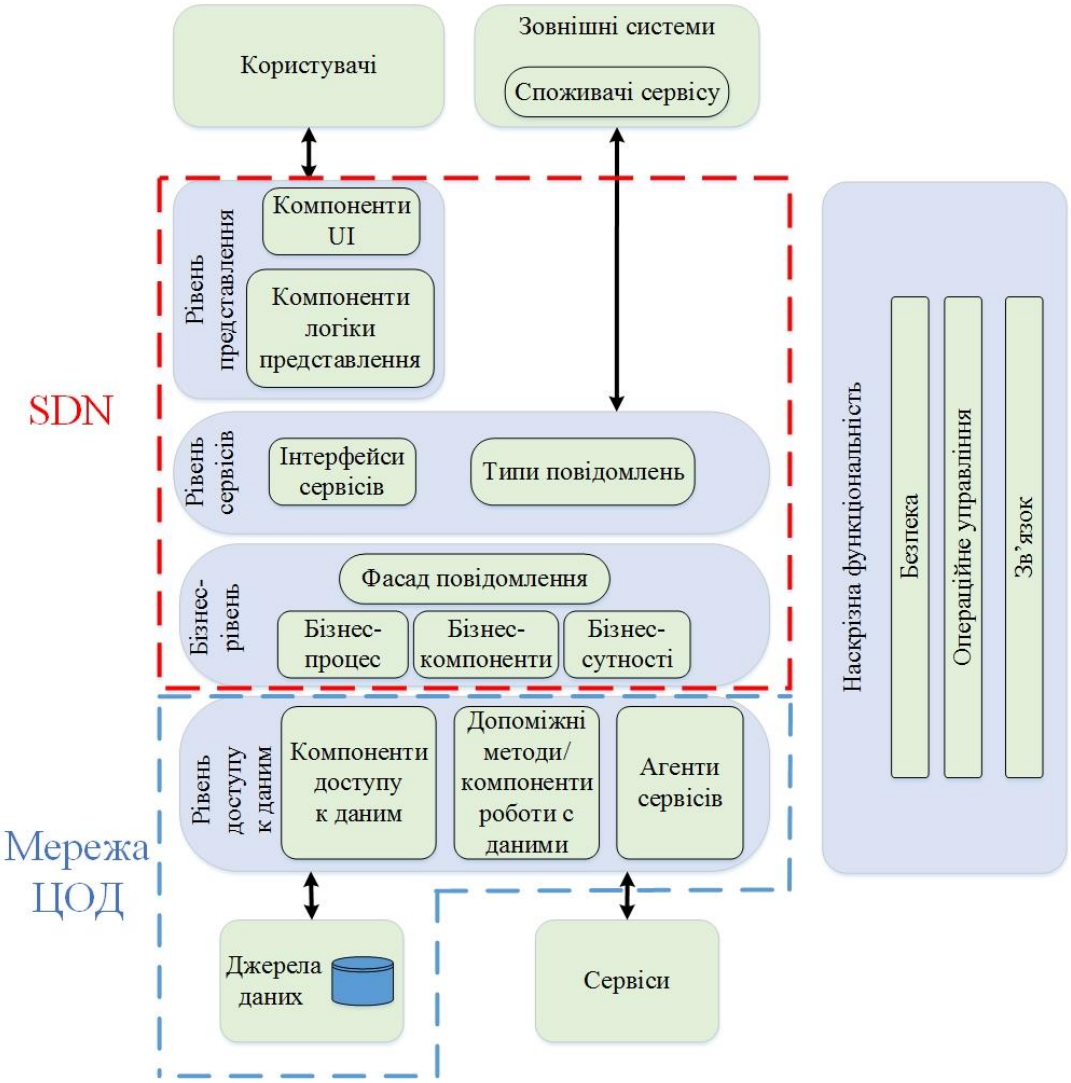

Рис. А Типова архітектура додатку

Таким чином, представлення даного центру обробки даних через таку архітектуру дозволяє здійснити інтеграцію з додатками, що розробляють сторонні виробники, що дає змогу забезпечити наскрізну функціональність, що дозволяє створювати неперервну систему контрою якості, безпеки та управління. Побудова ЦОД на базі такої архітектури є задачею актуальною, так як дозволяє прогнозовано підвищити якість, безпечність, контрольованість функціонування додатків.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ РЕФЕРЕНСНОЇ АРХІТЕКТУРИ ПРОГРАМНО-ВИЗНАЧЕНИХ МЕРЕЖ (SDN) ТА ЇЇ СКЛАДОВИХ

1.1 Причини виникнення нової парадигми

Вибухоподібне зростання і поширення мобільних пристроїв і контенту для них, віртуалізація серверів і поширення хмарних сервісів є основними трендами для переосмислення традиційних мережевих архітектур.

Позначимо фактори більш детально.

1. Зміна патернів трафіку. Усередині центрів обробки даних відбувається суттєва зміна патернів трафіку. На відміну від клієнт-серверної архітектури, коли основний обсяг комунікацій відбувається між клієнтами і сервером, сучасні програми створюють безліч потоків даних між комп'ютерами, а не тільки з серверами. Крім того, користувачі також змінюють патерни трафіку, застосовуючи мобільні пристрої, яким необхідний повноцінний доступ до корпоративних ресурсів та застосунків з будь-якої точки мережі в будь-який час.

2. Використання персональних мобільних пристроїв. Користувачі все частіше використовують на роботі різні персональні мобільні пристрої, такі як планшети, мобільні телефони, комунікатори і ноутбуки для доступу в корпоративну мережу. Архітектура мережі повинна прийняти ці пристрої з можливістю тонкого контролю і з урахуванням захисту корпоративних даних та інтелектуальної власності при збереженні політики використання мережі і даних в ній.

3. Розвиток хмарних сервісів. Підприємства активно використовують публічні та приватні хмарні сервіси. Бізнес-логіка диктує необхідність доступу до додатків, інфраструктури та інших IT ресурсів за запитом і з високим ступенем деталізації. На додаток до цього вкрай важливі питання безпеки при роботі з приватними і публічними хмарами. Вся ця архітектура повинна також враховувати можливість оперативних змін в структурі підприємств, як

внутрішніх, так і різних злиттів і поглинань. Таким чином, слідом за обчислювальними ресурсами мережева інфраструктура повинна також еластично масштабуватися, реорганізовуватися на льоту і бажано керуватися єдиним інструментом разом інфраструктурою зберігання, обчислювальними потужностями і т.д.

4. Різке збільшення обсягів даних. Сучасні програми, наприклад бізнес аналітики, і інші передбачають роботу з величезними обсягами даних і обмін між безліччю серверів, які можуть мати непередбачуваний заздалегідь набір зв'язків один з одним. Збільшення обсягів переданих по мережі даних та непередбачуваність типових шляхів передачі також вимагає від мережі можливості щодо істотного її масштабування, яке неможливо в рамках традиційної мережевої парадигми. Всі ці фактори, по суті нові функціональні вимоги до архітектури мережі, призводять до зміни парадигми, оскільки існуючі архітектурні принципи не здатні задовольнити нові потреби. Мережеві технології на сьогоднішній день представляють собою набір протоколів, розроблених для забезпечення зв’язку хостів по різним типам каналів, топологій і на різних дистанціях. Ці протоколи розроблялися в ізоляції один від одного (власне така ізоляція і лежить в основі багаторівневої моделі сеті виття архітектури, будь то OSI або TCP / IP). Результатом є підвищена складність мереж. Наприклад, щоб додати новий пристрій, необхідно змінити конфігурацію комутаторів, маршрутизаторів, міжмережевих екранів і т.д., оновити списки контролю доступу, інформацію про VLANи, якість обслуговування та інші механізми, використовуючи інструменти адміністрування, специфічні для конкретного обладнання, причому в залежно від виробника обладнання, типу і версій ПЗ. Таким чином, сучасні мережеві архітектури мають складності в оперативній підтримці мінливих вимог додатків і бізнес логіки, мають недостатню масштабованість і складні в налаштуванні і обслуговуванні, залежать від специфіки вендорних рішень.

1.2 Обмеження поточних мережевих технологій

Виконання поточних вимог ринку практично неможливо з традиційними мережевими архітектурами. Зіткнувшись з малими або скороченими бюджетами, корпоративні IT-відділи намагаються вичавити максимум зі своїх мереж за допомогою засобів управління на рівні пристрою і ручних процесів. Компанії стикаються з такими проблемами, як попит на мобільність і збільшення трафіку; прибуток розмивається зростаючими витратами на капітальне обладнання та малим або зниженим доходом. Існуючі мережеві архітектури не були розроблені для задоволення потреб сучасних користувачів, підприємств та носіїв; скоріше проектувальники мереж обмежені обмеженнями існуючих мереж, які включають в себе:

- Складність, що призводить до стазизу: мережеві технології на сьогоднішній день в основному складаються з окремих наборів протоколів, призначених для підключення хостів надійно через довільні відстані, швидкості зв'язку та топології. Для задоволення ділових і технічних потреб протягом останніх декількох десятиліть, промисловість розвивала мережеві протоколи для забезпечення більш високої продуктивності та надійності, більш широкої взаємодії і більш суворої безпеки.

Протоколи, повинні бути визначені в ізоляції, проте, з кожним рішенням конкретної проблеми і без використання будь-яких фундаментальних абстракцій. Це призвело до однієї з основних обмежень сучасних мереж: складність. Наприклад, щоб додати або перемістити будь-який пристрій, потрібно торкнутись кількох комутаторів, маршрутизаторів, міжмережевих екранів, порталів Web-аутентифікації і так далі, і оновити списки контролю доступу, мережі VLAN, якості послуг (QoS), а також інші механізми на основі протоколу за допомогою управління на рівні інструментів керування пристрою. Крім того, мережева топологія, виробник моделі комутатора і версії програмного забезпечення все повинні бути прийняті до уваги. Через цю

складності, сучасні мережі є відносно статичними, оскільки IT прагне звести до мінімуму ризик переривання обслуговування.

Статичний характер мереж різко контрастує з динамічним характером сьогоднішнього серверного середовища, де віртуалізація серверів значно збільшила кількість хостів, що вимагають підключення до мережі і істотно змінила припущення про фізичне розташування хостів. До віртуалізації додатки проживали на одному сервері, і в першу чергу обмінювалисся трафіком з обраними клієнтами. На сьогоднішній день додатки розподілені між кількома віртуальними машинами (ВМ), які обмінюються потоками трафіку одна з одною. Віртуальні машини мігрують з метою оптимізації і відновлення балансу навантажень сервера, викликаючи фізичні кінцеві точки існуючих потоків для зміни (іноді дуже швидко) з плином часу. ВМ міграції використовують багато аспектів традиційних мереж, від схем адресації і просторів імен до базових понять сегментації та маршрутизації.

Також щодо прийняття технології віртуалізації, багато підприємств сьогодні працюють в конвергентній мережі IP для передачі голосу, даних і відео трафіку. У той час як існуючі мережі можуть забезпечити диференційовані рівні QoS для різних додатків, виділення тих ресурсів є ручним. IT необхідно налаштувати обладнання кожного виробника окремо, а також налаштувати параметри, такі як пропускна здатність мережі і QoS для кожного сеансу, для кожної програмної основи. Через свою статичну природу, мережа не може динамічно адаптуватися до мінливих вимог трафіку, додатків і запросів користувачів.

- Непослідовна політика: Для реалізації політики для всієї мережі IT доведеться налаштовувати тисячі пристроїв і механізмів. Наприклад, кожен раз, коли нова віртуальна машина налаштовується, це може зайняти кілька годин, в деяких випадках днів, для того, щоб змінити конфігурацію списків управління доступом по всій мережі. Складність сучасних мереж робить дуже важким для IT налаштування послідовного списку доступу, безпеки, QoS, і інших політик для зростаючої кількості мобільних користувачів, що залишає підприємство

уразливим для порушення безпеки, недотримання правил та інших негативних наслідків.

- Неможливість масштабування: Як вимоги центру обробки даних швидко ростуть, так повинна рости і мережа. Проте, мережа стає значно складнішою із додаванням сотень або тисяч мережевих пристроїв, які повинні бути налаштовані і керовані. ІТ також покладалися на підключення декількох пристроїв в один порт для масштабування мережі, заснованої на передбачуваних моделях трафіку. Однак, в сучасних віртуалізованих центрах обробки даних, моделі трафіку неймовірно динамічні і тому непередбачувані.

Мега-компанії, такі як Google, Yahoo!, Facebook, стикаються з ще більш серйозними проблемами масштабованості. Ці постачальники послуг використовують масштабні алгоритми паралельної обробки і пов'язані з ними набори даних на всіх їх обчислювальних ресурсах. Оскільки сфера застосування кінцевих користувачів додатків збільшується (наприклад, сканування та індексування всієї всесвітньої павутини, щоб миттєво видати результати пошуку користувачам), кількість обчислювальних елементів сильно збільшується і обмін даними між обчислювальними вузлами може досягати петабайт. Ці компанії мають потребують так звані масштабовані мережі, які можуть забезпечити високу продуктивність, недорогий зв'язок через сотні тисяч - потенційно мільйони - фізичних серверів. Таке масштабування не може бути зроблено ручним налаштуванням.

Для того, щоб залишатися конкурентоспроможними, компанії повинні надавати все більші, краще-диференційовані послуги клієнтам. Мультіоренда ще більш ускладнює їх завдання, оскільки мережа повинна обслуговувати групи користувачів з різними додатками і різними потребами в продуктивності. Основні операції, які здаються відносно простими, такі як управління потоками трафіку клієнтів для забезпечення налаштованого контролю продуктивності або доставки за вимогою, є дуже складними для реалізації з існуючими мережами, особливо в масштабах компанії. Вони вимагають спеціалізованих пристроїв на

периферії мережі, тим самим збільшуючи капітальні та експлуатаційні витрати, а також час виходу на ринок для впровадження нових послуг.

- Залежність від виробника: компанії і підприємства прагнуть швидко впроваджувати нові можливості і послуги у відповідь на потреби бізнесу або запити користувачів. Проте, їх здатність реагувати гальмується через життєвий цикл продукції виробників, який може варіюватися від трьох років і більше. Відсутність стандартних, відкритих інтерфейсів обмежує можливості мережевих операторів щодо адаптування мережі до їх потреб.

Ця невідповідність між вимогами ринку і можливостями мережі привела галузь до переломного моменту. У відповідь на це, промисловість створила програмно-визначені мережі (Software-Defined Networking - SDN) із відповідними прикладними стандартами.

1.3 Опис архітектури програмно-визначених мереж

Програмно-визначені мережі - це нова мережева архітектура, де рівень управління мережею відокремлений від пристроїв передачі даних і реалізується програмно.

На рисунку 1.1 зображено логічне представлення архітектури SDN [1]. «Інтелектуальні» функції мережі централізовані в основі програмного SDN контролеру, які підтримує глобальний вид мережі. В результаті, мережа відображається для додатків і систем політик як єдиний логічний комутатор. З SDN, підприємства отримати незалежний від постачальника контроль над всією мережею з єдиної логічної точки, що значно спрощує проектування мережі і її експлуатацію. SDN також значно спрощує мережеві пристрої самі по собі, так як вони більше не повинні розуміти і обробляти тисячі стандартів протоколів, а повинні просто приймати вказівки від контролеру SDN.

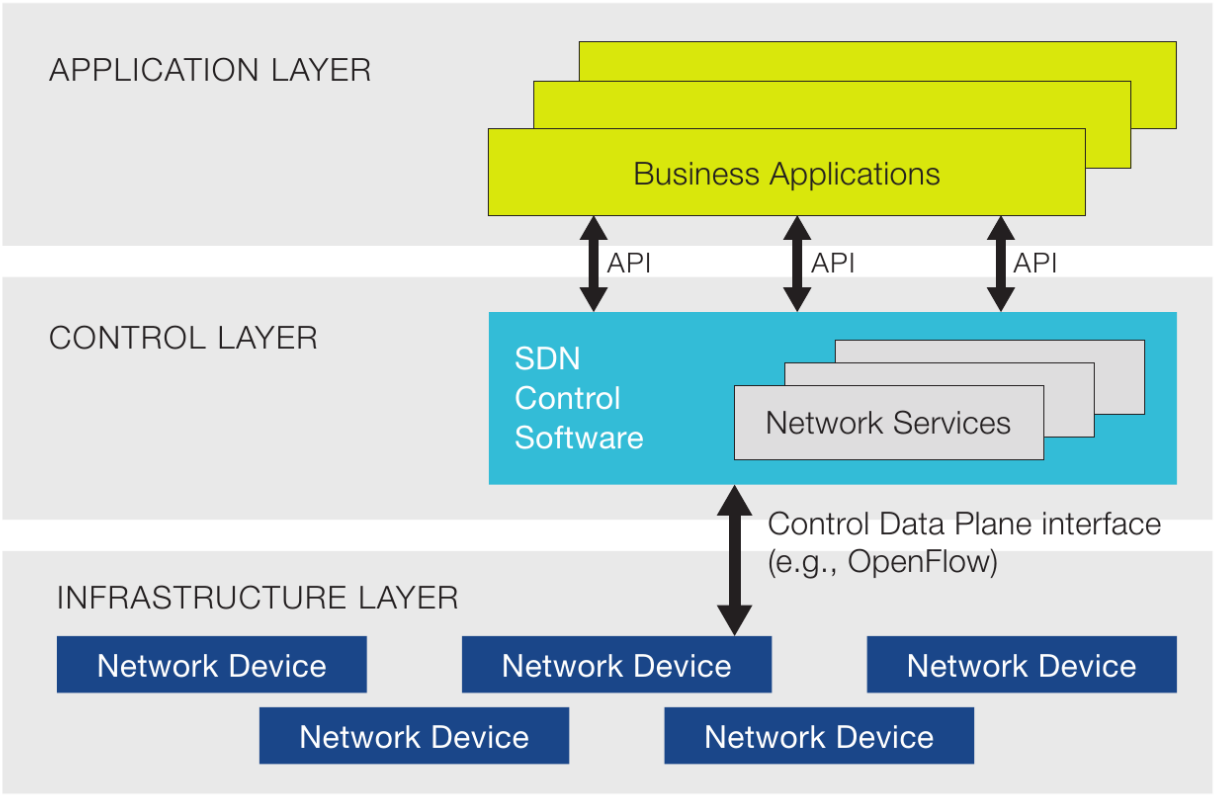

Рис. 1.1 Спрощене відображення архітектури SDN [3]

Найголовніше, що мережеві оператори і адміністратори можуть програмно налаштовувати цю спрощену мережеву абстракцію замість того, щоб вручну прописувати десятки тисяч рядків конфігурації, розкиданих серед тисяч пристроїв. Крім того, використовуючи централізований інтелект контролера SDN можна змінити поведінку мережі в режимі реального часу і розгорнути нові програми та мережеві служби протягом декількох годин або днів, а не тижнів або місяців, необхідних сьогодні. Централізуючи стан мережі в рівні управління, SDN дає мережевим адміністраторам гнучкість при конфігуруванні, управлінні, забезпеченні безпеки і оптимізації мережевих ресурсів за допомогою динамічних автоматизованих програм SDN. Крім того, вони можуть написати ці програми самі, а не чекати, поки можливості будуть вбудовуванні в фірмові проприєтарні і закриті програмні середовища в середині мережі.

На додаток до абстрагування мережі, SDN архітектура підтримує набір API-інтерфейсів, які дозволяють реалізувати спільні мережеві служби, в тому числі маршрутизації, багатоадресної передачі, безпеки, контролю доступу, пропускної здатності, управління, управління трафіком, якості обслуговування, процесорної оптимізації і оптимізації зберігання даних, використання енергії, а також всіх форм управління політиками, написаних за індивідуальним замовленням для задоволення бізнес-цілей. Наприклад, архітектура SDN дозволяє легко визначати і застосовувати послідовну політику як для дротових, так і для бездротових з'єднань в центрі обробки даних. Також SDN дозволяє управляти всією мережею через інтелектуальні системи оркестрації і резервування.

Таким чином, з відкритими API між рівнем SDN керування і рівнем додатків, бізнес-додатки можуть працювати на абстракції мережі, використовуючи мережеві сервіси та можливості, не будучи прив'язаним до деталей їх реалізації. В результаті, обчислення, зберігання даних і мережеві ресурси можуть бути оптимізовані.

1.4 Дослідження референсної архітектури SDN

Метою SDN є забезпечення відкритих інтерфейсів, які включають розробку програмного забезпечення, яке може контролювати з'єднання, що забезпечується великою кількістю мережевих ресурсів і потоком мережевого трафіку через них, разом з можливістю перевірки і модифікації трафіку, котрий може бути представлений в мережі. Ці примітивні функції можуть бути відведені в довільні мережеві служби, деякі з яких не можуть бути в даний час очевидно визначені.

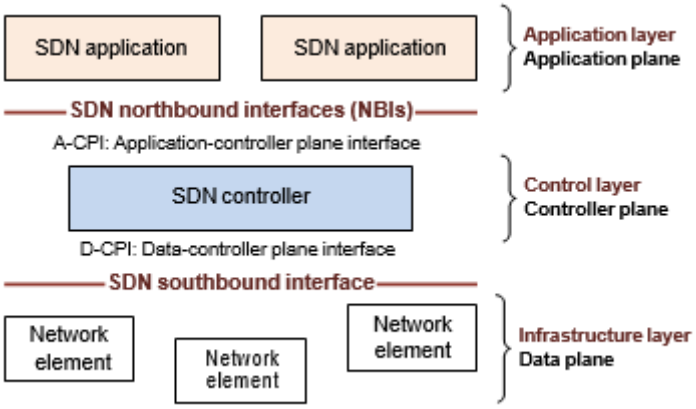

Рис. 1.2 Базові компоненти SDN [1]

Рисунок 1.2 вводить основні компоненти технології: SDN складається з рівнів інфраструктури, управління і додатків (червоний текст), які позначені тут як площини даних, контролера і додатків (чорний текст). Рівень інфраструктури (площина даних) містить мережеві елементи, які надають свої можливості рівню управління (площина контролера) через інтерфейси в південному напрямку від контролера (це називається інтерфейс площини управління-даних). Додатки SDN знаходяться на рівні додатків (площині) і повідомляють про свої мережеві вимоги площині контролера через північні інтерфейси, які часто називають NBIs (northbound interfaces). В середині, контролер SDN переводить вимоги додатків і здійснює низькорівневий контроль над елементами мережі, передаючи при цьому необхідну інформацію до SDN додатків. Контролер SDN може управляти конкуруючими запитами додатків на обмежені мережеві ресурси відповідно до політик.

На рисунок 1.3 додано функцію керування, яку часто опускають з спрощених схем представлень SDN. Хоча багато традиційних функцій керування можуть бути обійдені безпосередньо інтерфейсом площини додаток-контролер (application-controller plane interface - A-CPI), деякі функції керування як і раніше необхідні. У площині даних, керування, по крайній мірі, потрібно для початкового налаштування мережевих елементів, визначення SDN-контрольованих частин і настройка їх SDN контролеру. У площині контролера,

керування необхідно для налаштування політик, що визначають границі контролю, наданих додатку SDN і для моніторингу продуктивності системи. У площині додатку, керування зазвичай налаштовує контракти і угоди про рівень обслуговування (SLA - *Service Level Agreement*). У всіх площинах, керування налаштовує параметри безпеки, які дозволяють розподіленим функціям безпечно взаємодіяти.

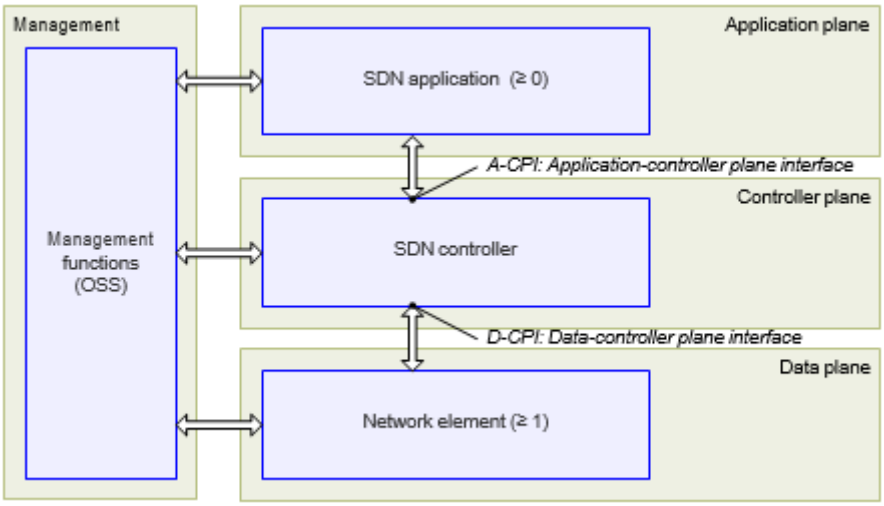

Рис. 1.3 Компоненти SDN з керуванням [1]

Рисунок 1.3 показує визначені площини додатку, контролеру і даних, з інтерфейсами площини контролеру (CPIs - controller plane interfaces), призначеними в якості опорних точок між контролером SDN та площиною додатку (A-CPI), а також між контролером SDN і площиною даних (D-CPI).

1.4.1 Аналіз основних принципів SDN

Основні принципи SDN:

- Розв'язка площин контролера та даних

Цей принцип вимагає розділення площин даних та контролера. Проте, слід розуміти, що контроль повинен обов'язково здійснюватися в системі площини даних. Інтерфейс площини додаток-контроллер між SDN контролером

і мережевим елементом визначається таким чином, що контролер SDN може делегувати значну функціональність до мережевого елементу, залишаючись при цьому в курсі стану мережевого елементу. Далі будуть описані (в площині контролеру) списки критеріїв для прийняття рішення, що делегувати, а що залишити в самому контролері SDN.

- Логічно централізований контроль

У порівнянні з місцевим управлінням, централізований контролер має більшу кількість ресурсів під своїм контролем, і потенційно може приймати більш обґрунтовані рішення щодо їх розгортання. Масштабованість поліпшується як за рахунок розв'язки, так і за рахунок централізації управління, що дає більш глобальне, але менш детальне бачення мережевих ресурсів.

- Вплив абстрактних мережевих ресурсів і стану на зовнішні додатки

Додатки можуть існувати на будь-якому рівні абстракції або деталізації, атрибути часто описуються як різні широти (рівні абстракції), з ідеєю, що «чим далі на північ», тим більший ступінь абстракції. Так як інтерфейс, який надає ресурси і стан можна розглядати як інтерфейс контролеру, відмінність між додатком і контролем не є точною. Той же функціональний інтерфейс можна розглядати з різних сторін різними зацікавленими сторонами. Так само, як контролери, додатки можуть ставитися до інших додатків як однорангові вузли, або як клієнти і сервери.

1.5 Аналіз протоколу OpenFlow в контексті архітектури SDN

1.5.1 Загальний огляд протоколу OpenFlow

В основі концепції SDN лежить відкритий стандарт OpenFlow - стандарт, визначений громадською організацією Open Networking Foundation (ONF, відкритий фонд мережевих технологій) в 2011 році [2]. Даний фонд образовано альянсом компаній Deutsche Telekom, Facebook, Google, Microsoft, Verizon, і Yahoo, і націлено на просування і стандартизацію SDN. Про значимість нового

руху говорить швидко зростаючий склад ONF, в котрий увійшли вже більше 40 найбільших компаній світу, в числі яких Brocade, Cisco, Citrix, Oracle, Dell, Ericsson, HP, IBM, Juniper, Marvell, NEC, Netgear, NTT, Riverbed і ряд інших.

Розвиток OpenFlow пройшов через ряд версій - від версії 1.0 до 1.4. Більшість виробників мережевого устаткування спочатку реалізували підтримку версії 1.0, яка мала безліч обмежень і проблем з масштабністю. Вона показала, що ONF запропонував ринку неповноцінний продукт. Версії 1.1 і 1.2 вважаються перехідними і практично ніхто з виробників не реалізує їх підтримку.

Найбільш перспективною вважається версія 1.3, в яку включена підтримка MPLS міток, per-flow лічильників, Provider Backbone Bridging (PBB) і ще деяких корисних функцій. За результатами тестування великої кількості комутаторів різних виробників (на середину 2015 року), можна сказати, що підтримка версії 1.3 на «залізних» комутаторах поки дуже обмежена.

Може скластися враження, що OpenFlow є повністю «відкритим» протоколом, проте його розробка контролюється закритою групою з вищеназваних компаній. Робота ведеться в прихованому від широкої публіки режимі і її результат видно тільки після публікації нової версії в якості стандарту.

1.5.2 Дослідження особливостей протоколу OpenFlow

Ідея OpenFlow – виніс площини управління на окремий пристрій - OpenFlow-контролер, залишаючи мережевим пристроям лише функції площини передачі даних. Це сильно знижує вимоги до функціоналу обладнання, перетворюючи його в «простий пристрій по перенаправленню пакетів з порту в порт» [3].

Площина передачі даних відповідає за пересилку Protocol Data Unit (PDU) відповідно до певних правил, що зберігаються в таблиці. Для рівня L2 це може бути MAC-таблиця, що складається з MAC-адрес і відповідних вихідних портів.

На вхідний порт приходить PDU, обробляється і відправляється через вихідний порт. Площина передачі даних не відповідає за формування таблиці переадресації.

Площина контролю відповідає за надання необхідної інформації площині даних для забезпечення переадресації і є «мозком», який приймає рішення про те, яким чином перенаправляти PDU. Ця площина не обмежується лише протоколами маршрутизації. Будь-який протокол, який контролює взаємодію між суміжними вузлами, зазвичай є протоколом площини контролю. Прикладом можуть служити BFD, STP і LACP. Ці протоколи не взаємодіють безпосередньо з таблицей переадресації. Наприклад, протокол BFD виявляє фізичні проблеми на каналі зв'язку між пристроями, але самостійно нічого не видаляє з таблиці. Він інформує інші більш високорівневі протоколи про збій, надаючи їм самим прийняти рішення про зміну таблиці переадресації. З іншого боку, протоколи маршрутизації OSPF або IS-IS безпосередньо впливають на таблицю маршрутизації і переадресації;

І нарешті, площина управління надає інтерфейс управління і моніторингу, за допомогою якого проводиться конфігурація пристрою.

Інтерфейс OpenFlow надає доступ і зв'язок між рівнями управління і інфраструктури архітектури SDN як фізичної, так і віртуальної. OpenFlow використовує термін контролер, який ґрунтуючись на загальній конфігурації та інформації у всій мережі, передає по захищеному каналу команди для кожного маршрутизатора (SDN комутатора) окремо для інтерпретації їх в індивідуальних таблиць рішень. Центральний контролер має точну інформацію про структуру і топологію мережі. Це дозволяє оптимізувати просування пакетів даних, і, зокрема, прокладати зв'язки «кожен з кожним» на рівні L2, не вдаючись до IP-маршрутизації (рисунок 1.4).

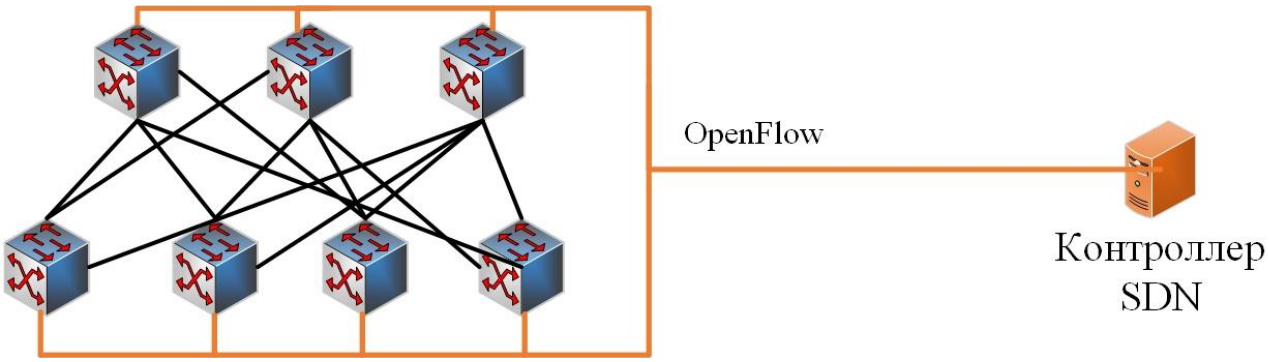

Рис. 1.4 Топологія мережі SDN на основі OpenFlow

1.5.3 Принцип роботи комутатора з підтримкою OpenFlow

В SDN комутаторі з підтримкою OpenFlow реалізований тільки рівень передачі даних. У кожного комутатора своя унікальна таблиця, яку він заповнює тільки на підставі інформації, отриманої від центрального контролера. Така таблиця комутації отримала назву FlowTable (таблиця потоків), так як по SDN - мережі передаються потоки даних, а не окремі пакети (правило в комутаторі встановлюється тільки для першого пакету, а потім всі інші пакети потоку його використовують). За допомогою таких таблиць вхідні пакети класифікуються, ґрунтуючись на порті, MAC-адресі, IP-адресі та інші засобах.

У кожного пакету, що прийшов «вирізується» заголовок (бітова рядок визначеної довжини). Для цього бітового рядка в таблицях потоків, починаючи з першого, шукається правило, у якого поле ознак найближче відповідає (збігається) заголовку пакета. При наявності збігу, над пакетом і його заголовком виконуються перетворення, що визначаються набором інструкцій, зазначених в знайденому правилі. Запис про потік може наказати переслати пакет в певний порт (звичайний фізичний порт або логічний, призначений комутатором, або зарезервований порт, встановлений специфікацією протоколу). Зарезервовані порти визначають спільні дії пересилання: відправка контролеру, широкомовна (лавинна) розсилка, пересилання без OpenFlow.

OpenFlow - звичайний протокол програмування TCAM. Якщо потрібно створити тунель між двома кінцевими точками, то це не можна зробити за допомогою OpenFlow. Подібні завдання можна виконати за допомогою інших протоколів, наприклад, OF-CONFIG, який використовує NETCONF для цього. Порти можуть бути додані або видалені з конфігурації комутатора за допомогою OF-CONFIG, але не за допомогою OpenFlow.

Зміна стану порту не призводить до автоматичного перенаправлення потоку через альтернативний маршрут. Наприклад, якщо порт перейшов в стан Down, то запис про потік, яка використовує цей інтерфейс для відправки, залишиться і всі наступні пакети будуть відкидатися. При зміні стану порту, комутатор спочатку повинен відправити повідомлення на контролер, щоб той вніс необхідні зміни у flow-таблицю комутатора.

Якщо потрібного правила в першій таблиці не виявлено, то пакет інкапсулюється і відправляється контролеру, який формує відповідне правило для пакетів даного типу і встановлює його на комутаторі (або на наборі керованих їм комутаторів), або пакет може бути змінений (наприклад, можна зменшити TTL чи добавити тег VLAN) або скинутий. Інструкції конвеєра обробки дозволяють пересилати пакети в наступні таблиці для подальшої обробки і у виді метаданих передавати інформацію між таблицями. Інструкції також визначають правила модифікації лічильників, які можуть бути використані для збору різної статистики (рисунок 1.5).

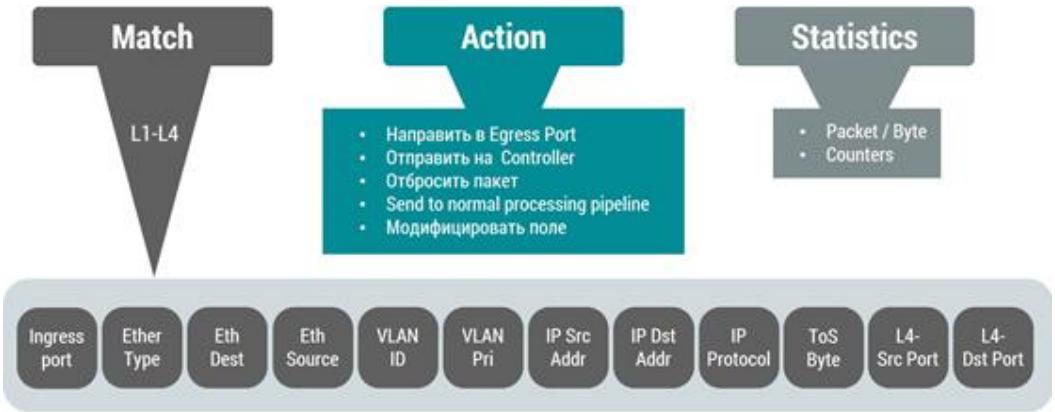

Рис. 1.5 Таблиця потоків в OpenFlow-комутаторі [3]

Висновки з розділу 1

1. Були досліджені причини виникнення нової парадигми програмно-визначених мереж та обмеження поточних мережевих технологій, через які використання програмно-визначених мереж є доцільною.
2. Архітектура програмно-визначених мереж складається з трьох основних рівнів: інфраструктури, управління і додатків. Рівень додатків керує рівнем інфраструктури через рівень управління – додатки SDN повідомляють про свої мережеві вимоги рівень управління, котрий їх переводить і потім здійснює низькорівневий контроль на елементами мережі (рівень інфраструктури).
3. Проаналізовано кожний рівень архітектури, складові кожного рівня, їх взаємодія всередині рівня. Розглянуто основні функції та принципи роботи цих складових. Також була досліджена міжрівнева взаємодія.
4. До основних принципів SDN відносять розв’язку площин контролера та даних, логічно централізований контроль, вплив абстрактних мережевих ресурсів і стану на зовнішні додатки.
5. Розглянуто використання протоколу OpenFlow в контексті архітектури SDN, його основні особливості та принцип роботи комутатора з підтримкою цього протоколу.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ РЕФЕРЕНСНОЇ АРХІТЕКТУРИ ПОБУДОВИ ЦЕНТРІВ ОБРОБКИ ДАНИХ (ЦОД)

2.1 Вступ

Від сучасних архітекторів і менеджерів корпоративних центрів обробки даних (ЦОД) очікують створення мереж, які можуть одночасно консолідувати і географічно розподіляти ресурси, які можуть включати віртуальні та фізичні прикладні сервери, системи зберігання даних, платформи управління і мережеві пристрої. Ці очікування підігріваються потребами бізнесу, щоб бути більш гнучкими (agile), щоб було зроблено більше з меншими витратами, а також для підвищення їх IT ефективності.

Щоб зменшити складність, архітектори ЦОД прийняли архітектури, які використовують мережеву конвергенцію, де системи введення/виведення (I/O) даних і систем зберігання злиті в єдину мережу. Цей конвергентний підхід може усунути фізичні перешкоди і складнощі, в той час роблячи більш ефективне використання мережевих ресурсів. Однак спрощення, у багатьох випадках, це вірно тільки на поверхневому рівні. Мережі ЦОД на сьогоднішній день можуть бути більш складним «нижче поверхневого рівня», ніж будь-коли раніше. Віртуалізація кинула значну кількість викликів, які приводять до нової еволюції мережевої архітектури центрів обробки даних, котра набагато перевищує первісну потребу в конвергентній мережевій інфраструктурі.

Розробники створюють архітектури ЦОД, які дозволяють підприємствам легше узгодити бізнес-ініціативи з вимогами базової мережі. За допомогою сегментації структурних елементів, підприємства можуть вибирати перевірені рішення, які відповідають їхнім потребам, а не бути замкненими в однакових універсальних рішеннях. При використанні стандартних протоколів на кордонах, підприємства можуть забезпечити функціональну сумісність між сегментами мережі і отримати як гнучкість, так масштабованість.

Зазвичай архітектури фокусуються на основних моментах ЦОД, які забезпечують наступні переваги [4], [5]:

- Об'єднання передових, базованих на стандартах платформ і передових мережевих технологій для оптимізації продуктивності і затримки в віртуалізованих серверних середовищах;
- Конвергенція і безпечність мережі центру обробки даних, обчислювальних систем і систем зберігання даних в фізичних і віртуальних світах;
- Зменшення складності;
- Підтримка швидких мережевих змін, зумовлених бізнес потребами;
- Зниження рівня сукупної вартості володіння;
- Об'єднання безлічі протоколів в єдину фабрику, яка є адаптивною для змінного навантаження.

2.2 Основні ділові та технологічні рушії для нової мережевої архітектури ЦОД

Основними рушіями розвитку мережевої архітектури вважають [6], [7]:

2.2.1 Віртуалізація

В даний час основною тенденцією в мережевій архітектурі є віртуалізація. Віртуалізація охоплює здатність керувати декількома серверами одночасно на вершині однієї серверної апаратної платформи, включаючи спільне використання процесора, диска, інтерфейсу і мережевих сервісів. У цьому випадку, кожен віртуальний сервер працює як незалежна сутність на одному фізичному сервері. Ранні розгортання віртуалізованних серверів працювали статично на платформі, на котрій вони були розміщені. Сьогодні

віртуальні сервери є гнучкими для розгортання та можуть переміщатися на інші фізичні серверні платформи.

Крім того, прийняття більш потужних серверів багатоядерних процесорів, інтерфейсів з високою пропускною здатністю, а також блейд-серверів різко збільшує масштаб розгортання центрів обробки даних. Тепер десятки тисяч віртуальних машин, як правило, розміщені в одному центрі обробки даних для консолідації інфраструктури та оптимізації операцій.

Цей тип віртуальної інфраструктури тепер може значно скоротити обладнання, мережеві/експлуатаційні витрати, однак, в той же час ці великомасштабні рішення значно збільшують вимоги до мережевої продуктивності на прикордонному сервері і по всій розширеній мережі. Крім того, віртуалізація та інструменти vMotion/Live Migration для переміщення віртуальних серверів між машинами в одному центрі обробки даних або через територіально розділені ЦОД видають великі потоки міжмашинного трафіку. Це робить негативний вплив на існуючі адміністративні практики, створюючи нову "віртуальну границю", який розмиває традиційні кордони між адмініструванням мережі та серверу.

2.2.2 Консолідація великомасштабних ЦОД

Для багатьох корпоративних клієнтів, центр обробки даних - це бізнес. Критично важливі програми і послуги забезпечують основу для повсякденних операції та надання послуг кінцевим споживачам. Центр обробки даних повинен забезпечити безумовну доступність і відповідати суворим SLA. Експлуатуючи віртуалізацію серверів і низькі витрати на обчислювальні потужності, клієнти впроваджують ще більш складні програми в більшому масштабі. Для того, щоб знизити їх складність і поліпшити операції в цих впровадженнях, клієнти прагнуть консолідувати фрагментовані, розосереджені об'єкти в менших, централізованих місцях.

Сучасних мережі ЦОД повинні бути розроблені так, щоб забезпечити значно вищий рівень продуктивності, масштабованості і доступності, ніж будь-коли раніше, щоб відповідати SLA і підтримувати безперервність операцій. Крім чистої продуктивності, ці мережі повинні швидко відновлюватися після апаратних або програмних збоїв і бути захищеними від серверних вразливостей, вразливостей систем зберігання, мережі, програм для підтримки продуктивності і мінімізації перерв в обслуговуванні.

2.2.3 Розгортання нових додатків і моделі доставки

Традиційні моделі розгортання клієнт-серверного програмного забезпечення та інфраструктури були заміщені новими архітектурними схемами застосування і моделями надання послуг, які змінили обличчя центру обробки даних.

Web 2.0 мешап, рішення сервіс-орієнтованої архітектури, а також інші федеративні схеми в даний час широко використовуються для доставки інтегрованої, контенто-корельованої, контекстно-залежної інформації і послуг кінцевим користувачам в рамках підприємства і за його межами. Ці розгортання приводять до нового, широкосмугового інтенсивного руху потоків в ЦОД і вимагають малу затримку, високопродуктивний зв’язок сервер-сервер і інтра-сервер, а також VM-to-VM.

У той же час, хмарні обчислення і ініціативи XaaS впроваджують жорсткіший рівень обслуговування і вимоги безпеки для більш гнучкої і динамічної інфраструктури. В даний час, співробітники, клієнти і партнери можуть отримувати доступ до своїх програм практично з будь-якого місця - зі штаб-квартири, в університетському містечку, у філії або з будь-якого віддаленого місця, а програми в цей час можуть знаходитися в традиційному або хмарному дата-центрі.

2.2.4 Моделі розгортання центрів обробки даних

Прийняття більш віртуалізованих та динамічних прикладних середовищ впливає на традиційні корпоративні і multitenant (багато-арендні) архітектури центрів обробки даних. Ці методи запускають нові «хмарні» моделі доставки, котрі керують цілим новим набором технологічних вимог на серверах, на системах зберігання і мережевих доменах. Ці всі більш і більш популярні моделі дозволяють корпоративним додаткам бути більш гнучкими в рамках традиційних внутрішніх інфраструктур, а також дають можливість орендованим додаткам і постачальникам послуг (service providers) проводити комерційну діяльність (бізнес), засновану на наданні послуг за допомогою загальнодоступної хмарної моделі. З огляду на діапазон варіантів використання і опцій, клієнти часто розгортують поєднання архітектур для вирішення різноманітних вимог і оптимізації операцій.

2.2.5 Програмно-визначені мережі

Програмно-визначені мережі є новим рішенням, яке переосмислює те, що ми думаємо про мережу в рамках ЦОД, кампусів тощо. Технологія усуває бар'єри на шляху інновацій, надаючи «хмарним» провайдерам і підприємствам повний програмний контроль динамічного, абстрактного відображення мережі. За допомогою технології SDN, IT може стати більш гнучкими, оркеструючи мережеві служби і автоматично управляючи мережею відповідно до високорівневих політик, а не до конфігурації низькорівневих мережевих пристроїв.

Рушійною силою SDN в даний час є «хмарні» провайдери і провайдери послуг, які покладаються на SDN, щоб вирішити деякі з проблем, пов'язаних з розгортанням і управління мережами у великому масштабі.

2.3 Аналіз основних вимог до мережі ЦОД

2.3.1 Віртуалізація масштабування

Віртуалізація в даний час є широко прийнятою і розгорнутою глобально в ЦОД. З її допомогою операційної системи, програми та сервери можуть працювати неспеціалізованим або слабо зв'язаним чином, взаємодіючи між собою в міру необхідності для задоволення потреб підприємства. Віртуалізація забезпечує кілька ключових переваг:

- Підвищення ефективності та зниження капітальних і експлуатаційних витрат: Як багатоядерні/багатопроцесорні системи забезпечують значну кількість обчислювальної продуктивності, так і віртуалізація дозволяє підвищити ефективність використання даного сервера і його ресурсів. Капітальні витрати на апаратне і програмне забезпечення, а також експлуатаційні та допоміжні витрати знижуються.
- Гнучкість центрів обробки даних: менеджери ЦОД повинні підтримувати операції, щоб останні працювали ефективно за рахунок швидкого розгортання ресурсів для зростання бізнесу, так і для періодів стресу. Розробники додатків відповідають за гнучкість процесора і використання пам'яті. Віртуалізація дає можливість робити на льоту міграцію віртуальних серверів або програм між фізичними серверами в центрах обробки даних.
- Відказостійкість: віртуалізація забезпечує можливість перекладення або перемішання програм на підтримку безперервності бізнесу або навіть для планового технічного обслуговування.

Віртуалізація створює потребу в нових методологіях мережевих архітектур центрів обробки даних. Традиційні центри обробки даних з автономними серверами визначаються рековим простором, пропускною здатністю портів, IP-адресацією, а також вимогами до підмереж. Пропускна здатність і переадресація пакетів завжди були ключовими елементами, але

тепер все по іншому – зараз вже відіграє роль менша кількість фізичних серверів і більша кількість процесорних ядер в межах даного центру обробки даних.

З віртуальним середовищем повна корпоративна інформаційна інфраструктурна система може бути розгорнута в меншому обсязі, ніж в минулому. Завдяки віртуалізації кількість серверів, яка потребує певної пропускної здатності в даній стійці, може легко вчетверо збільшити вимогу до пропускної здатності в порівнянні з більш традиційними топологіями центрів обробки даних.

Тепер, більш, ніж будь-коли, архітектура ЦОД вимагає ретельного планування зосередженого на вимогах до пропускної здатності, продуктивності Ethernet комутатора, спільно використовуваних ресурсів систем зберігання даних, широкомовного трафіку, а також географічної надмірності.

Оскільки центри обробки даних стають все більш віртуалізованими, а мережі конвергентними, потреба в архітектурі, що забезпечує продуктивність і ємність стає критичною. Але реальність така, що центри обробки даних не є реалізацією "з нуля". Центри обробки даних були раніше розроблені на базі традиційної мережевої трьохрівневої архітектури, яка працює на базі IP-пірінгу і рекового монтування прикладних серверів тільки одного призначення. Цей статус-кво в даний час перевернувся з голови на ноги зі швидким розгортанням віртуальних серверів. Справа в тому, що дані більше не рухаються між серверами всередині і за межами центру обробки даних, а в даний час рухається по горизонталі в і через віртуальні сервери ЦОД і мережевого периметру. Традиційні трьохрівнева мережева архітектури зосереджена навколо рівня L3 і не дуже добре підходить для підтримки віртуалізованих розгортань.

З 80% трафіку, що проходить від сервера до сервера в центрі обробки даних, підтримка східно-західного трафіку має інші архітектурні вимоги, ніж північно-південний трафік типу хост-клієнт. Таким чином, плоскі мережі рівня L2 з маршрутизацією рівня L3 на рівні ядра або агрегації набули популярності,

оскільки ці мережі є дуже ефективними, особливо щодо мобільності ВМ і динамічної міграції.

Однак у зв'язку з недавнім ухваленням накладених технологій як VXLAN, яка вирішує обмеження нижчележачого шару L2 vMotion (underlay, транспортні мережі), підприємства також в даний час користуються гнучкістю для розгортання кордону L3 від ядра до ToR - Top-of-Rack. Ці L3 середовища здатні використовувати L3 технології вибору маршруту в залежності від його вартості (ECMP - Equal-cost multi-path routing) для розподілу трафіку через «стволові» (магістральні) комутатори, в той же час підтримуючи міграцію vMotion в оверлейних середовищах (накладання).

2.3.2 Технології серверної віртуалізації

Віртуалізація - це розділення ресурсів, додатків або служб з основоположних фізичних компонентів цієї служби. Наприклад, віртуальна пам'ять дозволяє програмам отримувати доступ до більшої кількості пам'яті, ніж фізично встановленої пам’яті. Крім того, віртуалізація серверів може дати програмам видимість того, що вони мають доступ до всього сервера, включаючи пам'ять, процесор і жорсткі диски, в той час як насправді вони можуть ділити фізичне обладнання з іншими операційними системами і програмами. Ключовою перевагою віртуалізації є те, що вона забезпечує можливість запуску декількох екземплярів операційної системи на одному фізичному обладнанні одночасно, а також можливість спільного використання апаратних ресурсів: жорстких дисків, мережі і пам'яті.

За лаштунками, гіпервізор (з підтримкою від апаратного забезпечення) створює те, що виглядає як сервер (так звану віртуальну машину). Операційна система завантажується в ВМ майже так само, якщо була б встановлена на «залізі», і в результаті додатки завантажуються на операційну систему в звичайному режимі.

Адміністратори ЦОД мають можливість використовувати фізичні сервери оптимальним шляхом розміщуючи віртуальні машини через свої фізичні активи системи та інфраструктури.

До основного гіпервізорного програмного забезпечення відносяться:

1. VMware vSphere;
2. Microsoft Hyper-V;
3. Kernel-based Virtual Machine (KVM);
4. Citrix Xen.

2.3.3 Комутація VM-to-VM та прикордонна мережа

При наявності декількох віртуальних машин, які знаходяться на одному фізичному сервері, існують вимоги для ефективної комутації трафіку VM-to-VM на тому ж сервері і можливості примусового застосування політики фізичного комутатора на віртуальних машинах.

Існують дві моделі для забезпечення можливості комутації VM-to-VM:

- Virtual Ethernet Bridge (VEB): VEB є віртуальним комутатором Ethernet. Він може знаходитися на одному фізичному сервері і забезпечувати віртуальну мережу між віртуальними машинами на цьому сервері або він може бути використаний для підключення віртуальних машин до зовнішньої мережі. Більшість VEB базуються на програмному забезпеченні. Проте, з прийняттям стандарту PCI Single Root IO Virtualization (SR-IOV), апаратні віртуальні комутатори можуть бути вбудовані в мережеві адаптери. Апаратні VEB забезпечують кращу продуктивність, ніж програмні VEB.
- Програмні VEB: гіпервізор створює віртуальні мережеві адаптери для кожної віртуальної машини, а потім створює один або кілька vSwitches, які з'єднують віртуальні мережеві адаптери з фізичними мережевими картами. Трафік, отриманий фізичним мережевим адаптером, передається на вірний віртуальним мережевий адаптер на підставі конфігураційної

інформації, що має гіпервізор. Трафік від віртуального мережевого адаптеру може оброблюватися способами описаними нижче.

Якщо місце призначення є зовнішнім до фізичного сервера або до іншого vSwitch, то vSwitch направляє трафік на фізичний мережевий адаптер. Якщо місце призначення знаходиться на тому ж vSwitch і на тому ж фізичному сервері, vSwitch направляє трафік назад у потрібний віртуальний мережевий адаптер (рис. 2.1).

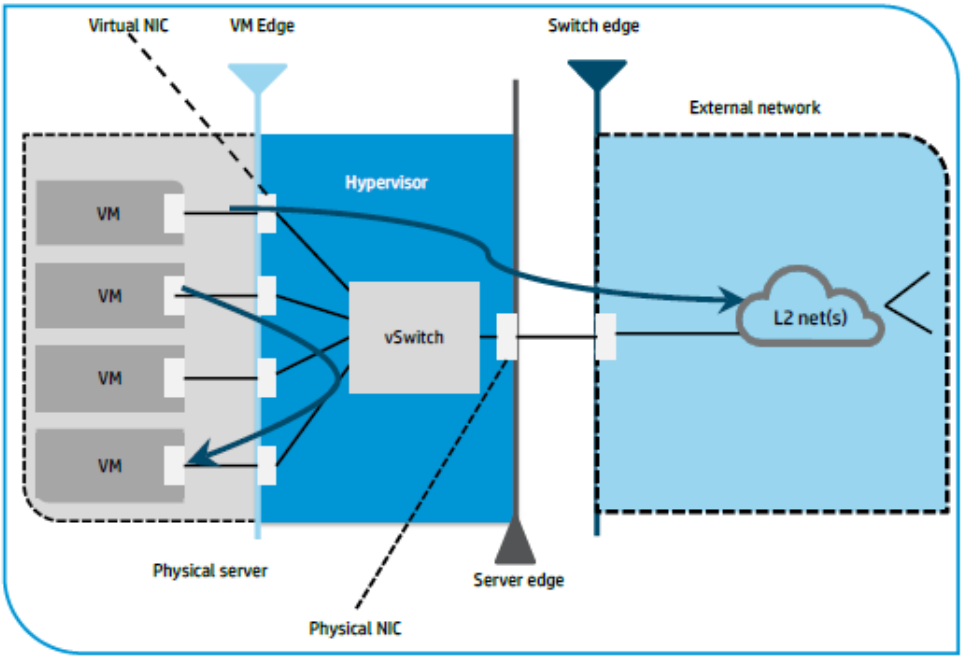

Рис. 2.1 Переадресація vSwitch [7]

2.3.4 Рівень L2 та мобільність віртуальних машин

За допомогою хмарних обчислень і віртуалізації в центрах обробки даних, понад 80 відсотків трафіку в них – це трафік від сервера до сервера. З впровадженням віртуалізації і можливості переміщення віртуальних машин з одного фізичного сервера на інший, використовуючи, наприклад, VMotion, зберігання вихідної IP-адреси і шлюзу за замовчуванням стає критично важливою вимогою. Це може бути досягнуто шляхом використання або фабрик рівня L2, або оверлейних рішень (рис. 2.2).

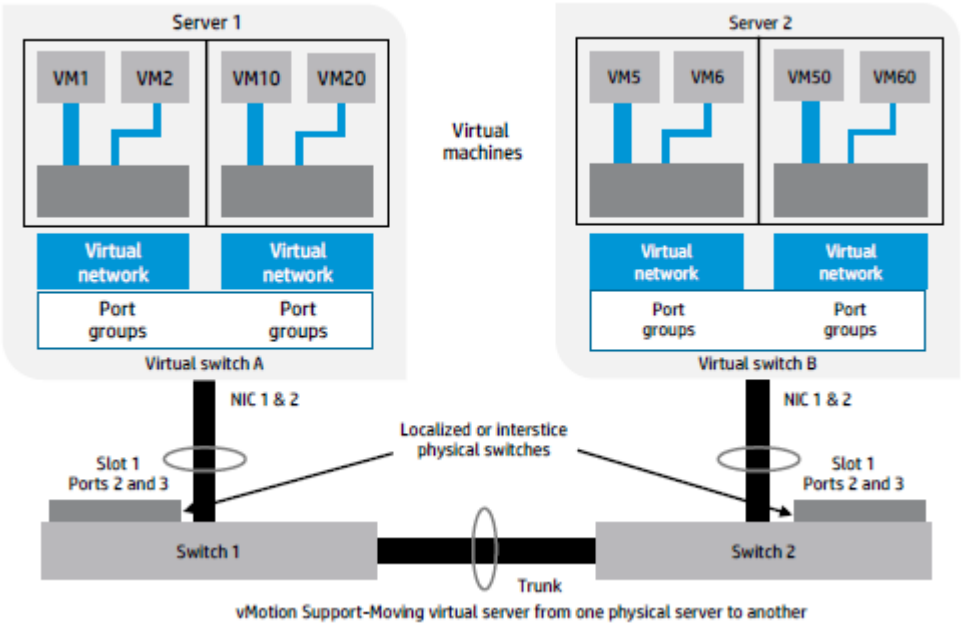

Рис. 2.2 Використання динамічної міграції ВМ на мережах рівня L2 [7]

Переваги рівня L2:

- Віртуальні машини або сервери можуть бути переміщені без необхідності зміни IP-адреси;
- Просте розміщення застарілих додатків/серверів, які вимагають встановлених IP-адрес/ шлюза за замовчуванням;
- Полегшення управління ВМ / серверів через віртуальний центру обробки даних або хостинг в процесі консолідації центрів обробки даних.

Ризики рівня L2:

- Зміна топології може вплинути на пропускну здатність.
- STP за своєю природою обмежений і може працювати при втратах більш ніж 50%. Це зменшує пропускну здатність і, отже, збільшує час завершення при переміщенні віртуальних машин.
- Особлива увага повинна бути сконцентрована на підтримуваному розмірі таблиць MAC на всіх комутаторах в домені 2-го рівня, а також на підтримуваних розмірах таблиць ARP на комутаторах, де буде проводитися маршрутизація (рівні ядра/агрегації).

2.3.5 Продуктивність

Центри обробки даних мають постійно зростаючу потребу в більшій пропускній спроможності. Розгортання віртуалізованих магістральних каналів (бекбон) 2-го рівня дозволяє впровадити кластеризацію комутаторів Ethernet. Завдяки цьому можна досягти набагато більш високої теоретичної пропускної спроможності, ніж у їх попередників, які покладалися на будь-який STP або рівень L3 для управління потоком трафіку. У цій новій інфраструктурі, комутатори можуть бути згруповані в конфігурації «активний/активний» - віртуально виключаючи мережеві посилання на сервери, які знаходяться в латентному стані в традиційних конструкціях. Впровадження технологій, як Shortest Path Bridging (SPB), Transparent Interconnection of Lots of Links (Trill), а також оверлейних рішень, таких як VXLAN, сприятиме подальшому зміцненню парадигми віртуалізованої магістральної мережі.

2.3.6 3-2-1-рівнева мережева архітектура

У традиційній трьохрівневій архітектурі мережі центру обробки даних, мережа складається з комутаторів рівня доступу, рівня агрегації і рівня ядра. Крім того, традиційні клієнт-серверні центри обробки даних складаються з автономних серверів, що підтримують статичні програми. У цьому типі мережевої структури переважав трафік, що йшов «вгору» і з мережі для зв'язку з комп'ютерами і кінцевими точками поза центром обробки даних.

Проте, завдяки віртуалізації і іншим новим тенденціям, більшість мережевого трафіку в сучасних центрах обробки даних - це горизонтальний трафік від сервера до сервера. Це пов'язано із міжхостовим, внутрішньо-ВМ і синхронізованим між-ВМ серверним зв'язком, котрий сполучає всі елементи через цю мережу, а також через територіально-розподілену мережу.

3-рівнева мережа в даний час вже не оптимізована для даного типу моделей трафіку. У сучасному центрі обробки даних горизонтальний потік

даних між серверами обмежується «подорожжю» до рівня агрегації перед поверненням до рівня доступу.

2.3.7 Мережеві протоколи рівня L2

В мережі 2-го рівня існує багато важливих протоколів і деякі з них були створені, щоб виконати різні мінливі потреби всередині і між ЦОД.

Деякі основні протоколи, що використовуються в різних ЦОД:

- Spanning tree protocol (STP);
- Intelligent Resilient Framework (IRF);
- Transparent Interconnect of Lots of Links (TRILL);
- Ethernet Virtual Interconnect (EVI);
- Virtual private LAN services (VPLS);
- Shortest Path Bridging (SPB);
- VxLAN;
- NVGRE.

Spanning tree protocol

Протокол STP – це не новий мережевий протокол, STP є істинним устарівшим протоколом, який забезпечував запобігання створенню петлі всередині мережі, а також широкомовних штормів і мережевих аварій.

STP, як правило, є ядром мережевої 3-рівневої архітектури. Багато з поліпшень, перерахованих в пункті 2.3.7, були розроблені, щоб відійти від обмежень STP, який є неефективним і має тривалий період конвергенції.

STP призначений для забезпечення мостового з'єднання Ethernet без петель (рис. 2.3). Це протокол 2-го рівня, тобто він не маршрутизований, і отримує оновлення від інших комутаторів про стан з'єднань і вартості шляху на основі bridge protocol data units (BPDU).

STP також дозволяє мережі бути створеною із резервними лініями зв'язку, забезпечуючи автоматичне резервування шляхів у випадку обриву зв'язку активних ліній.

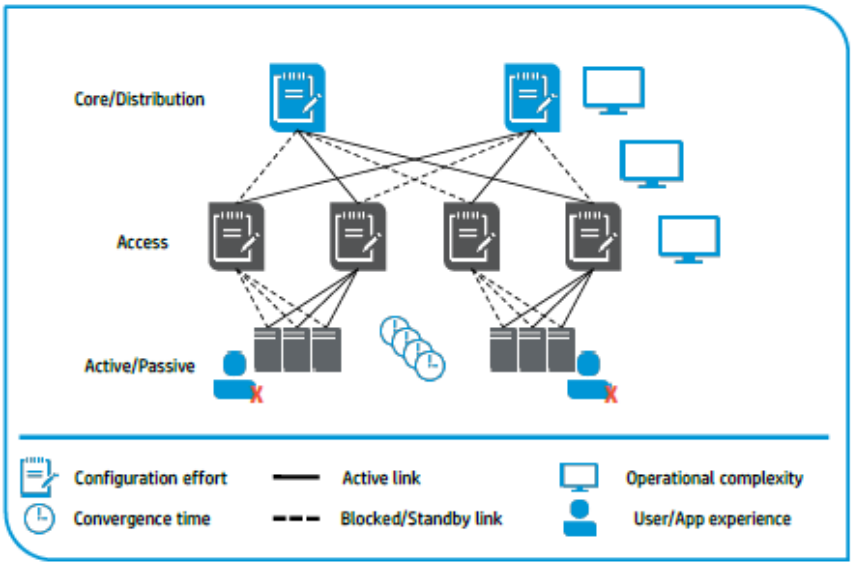

Рис. 2.3 Робота протоколу STP [7]

Intelligent Resilient Framework

Intelligent Resilient Framework (IRF) є пропрієтарною програмною технологією віртуалізації, що розроблена H3C (3Com). Його основна ідея полягає в тому, щоб з'єднати кілька мережевих пристроїв за допомогою фізичних портів IRF і виконати необхідні конфігурації, а потім ці пристрої віртуалізувати в розподілений пристрій. Ця технологія віртуалізації реалізує співпрацю, об'єднане управління і безупинне обслуговування декількох пристроїв. Ця технологія подібна по деяким ключовим особливостям на технологій Cisco VSS і vPC.

Технологія IRF розширює мережеве управління над декількома активними комутаторами. Управління групою комутаторів з підтримкою IRF консолідується навколо одного IP-адресу управління, який значно спрощує конфігурацію і мережеві операції.

Transparent Interconnect of Lots of Links

TRILL (Transparent Interconnection of Lots of Links) - мережева технологія, що дозволяє здійснювати "маршрутизацію" на рівні L2, на відміну від стандартної маршрутизації, яка здійснюється на рівні L3.

TRILL працює за такою ж логікою як і стандартна маршрутизація, приймаючи рішення про вибір маршруту за результатами обчислення найкоротшого шляху, але робить це не для IP, а для MAC-адрес.

TRILL - один з Ethernet-стандартів, який підвищує можливості мережевих технологій в хмарних обчисленнях:

- Збільшення комутаційної ємності;
- Кращий шлях для досягнення призначення;
- Необхідна підтримка мобільних серверів в середовищі віртуалізації;
- Швидша збіжність при скачках напруги і виникнення помилок.

Комутатори, що підтримують технологію TRILL, називають маршрутизуючими мостами, або RBridge (RB).

Shortest Path Bridging

SPB (802.1aq Shortest Path Bridging) є новою технологією, яка значно спрощує створення і конфігурацію транспортної мережі, мережі підприємства або хмарних мереж, підключаючи багатотрактову маршрутизацію.

Shortest Path Bridging є сучасною альтернативою старому сімейству протоколів Spanning Tree (IEEE 802.1D STP, IEEE 802.1w RSTP, IEEE 802.1s MSTP), які вміють використовувати тільки один маршрут пересилання трафіку до кореневого комутатора (root bridge) і блокують будь-які альтернативні шляхи, так як це може призвести до утворення мережевої петлі на 2-му рівні. SPB активно використовує всі наявні маршрути пересилання з

однаковою вартістю (equal cost multipathing), і дозволяє будувати більш масштабні топології на 2-му рівні (до 16 мільйонів сервісів, що набагато більше традиційного обмеження IEEE 802.1Q на 4096 віртуальних мереж VLAN). Він так само має дуже швидкий час збіжності, і збільшує ефективність багатозв'язних (Mesh мережі) топологій шляхом використання більшої смуги пропускання між усіма пристроями і більшою відмовостійкістю, так як трафік використовує і балансується між усіма доступними шляхами пересилання в багатозв'язній Mesh мережі. Для підвищеної надійності рівень доступу SPB може використовувати технології агрегації ліній, такі як стандарт IEEE 802.1AX або пропрієтарні реалізації механізмів MC-LAG.

VXLAN

VXLAN (*Virtual eXtensible Local Area Network*) - це схема інкапсуляції кадрів мережевого протоколу другого рівня в пакетах протоколу третього рівня (тунелювання), з метою створити віртуальну мережу, і обійти деякі обмеження традиційної схеми VLAN. VXLAN працює поверх наявної мережевої інфраструктури, таким чином можна назвати її оверлейною мережею.

VXLAN представляє собою оверлейну технологію, яка інкапсулює MAC-кадри на рівні L2 в UDP заголовок. Зв'язок потім встановлюється між двома кінцевими точками тунелю, котрі називаються віртуальними кінцевими точками тунелю (VTEP - VXLAN Tunnel Endpoint). VTEP здатні інкапсулювати і декапсулювати ВМ трафік в та із заголовка VXLAN (рис. 2.8). З VXLAN мережеві адміністратори можуть створити мережі рівня L2 на базі мережі рівня L3 (оверлейна технологія).

VXLAN має наступні переваги:

- Забезпечує різке збільшення обмеження кількості VLAN в 4096 одиниць;
- Підтримує суміжності мереж L2 понад мережами L3;

- VXLAN VTEP можуть бути реалізовані в віртуальних і фізичних комутаторах, що дозволяє віртуальній мережі зіставляти фізичні ресурси і мережні сервіси;
- Допомагає мережі перейти на архітектуру програмно-визначеного ЦОД.

NVGRE

NVGRE (Network Virtualization using Generic Routing Encapsulation - Мережева Віртуалізація за допомогою Загальної Інкапсуляції Маршрутів) була запропонована Microsoft, Intel, HP і Dell, а також є життєздатним конкурентом VXLAN. Як VXLAN, це ще одна технологія накладення, яка інкапсулює накладення (overlay) віртуальних мереж по всій фізичній мережі. Будучи технологією оверлейної інкапсуляції вона знімає обмеження в 4096 VLAN.

2.3.8 Аналіз взаємодії ЦОД (DCI) та вимог до неї

На рисунку 2.4 показані основні схеми при розгортанні рішень DCI (Data Center Interconnect) [8]:

- Взаємодія рівня L3 (зазвичай на базі існуючого IP ядра підприємства);
- Взаємодія рівня L2;
- Взаємодія SAN.

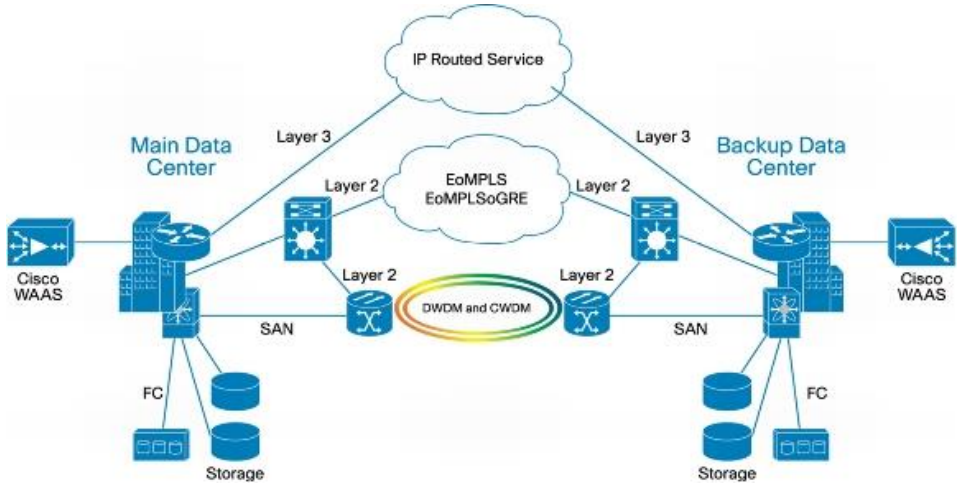

Рис. 2.4 Основні схеми рішень DCI [8]

DCI підтримує два можливі транспортні сценарії:

- Підприємство є власником основної інфраструктури, що з'єднує різні центри обробки даних;
- Провайдер пропонує послуги підключення.

Також можливі наступні види послуг:

- Dark Fiber: це можна розглядати як тип послуг рівня L1. Він користується популярністю серед багатьох клієнтів сьогодні, так як він дозволяє транспортувати різні види трафіку, в тому числі трафік SAN. Це рішення вважається дорогим, особливо тоді, коли кількість сайтів збільшується. Dark Fiber також обмежені в відстані, на якій можлива їх реалізація.
- Послуги рівня L2: У цьому випадку розширення локальної мережі може бути досягнуто за рахунок безпосередньо використанням послуг провайдера. Підприємство направляє власні кадри Ethernet до провайдера, які потім вже будуть доставлені на віддалені об'єкти. З іншого боку, підприємство може використовувати оверлейні VPN рішення рівня L2 на базі мережі провайдера, що дає додаткову експлуатаційну гнучкість.
- Послуги рівня L3: Постачальники послуг надає ці послуги на основі IP або MPLS. В обох сценаріях, підприємство має розгорнути оверлейну технологію для виконання розширення LAN між різними сайтами. Вибір оверлейних рішень, як правило, обмежуються тими, що базуються на IP.

До основних вимог до DCI відносять:

- Висока доступність;
- Ізоляція STP;
- Управління петлями.

2.3.9 Аналіз систем керування мережі ЦОД

В наші дні компанії вимагають високий рівень мобільності від своїх IT-систем. Інформатизація бізнесу - це ключовий конкурентний диференціатор, мережева інфраструктура центрів обробки даних – це хребет компанії, і він повинен бути міцним і гнучким одночасно.

Кожна корпоративна класова мережа повинна включати в себе управління, облік і повідомлення в якості складових частин повної архітектури.

Система керування мережею - це програмне забезпечення наступного покоління для управління, яке забезпечує систему операцій центрів обробки даних комплексною платформою, яка об'єднує мережеві технології і забезпечує функції конфігурації, обліку, продуктивності безпеки і захисту від відмов.

Системи керування мають наступні переваги [9], [10]:

- Більш низькі експлуатаційні витрати і поліпшення сукупної вартості володіння, дякуючи автоматиці та оповіщенням за замовчуванням, а також консолідації інструментів і кореляції інформації;
- Покращена доступність мережі і надійність, що призводить до меншої кількості проблем в користувачів, завдяки автоматизації управління конфігураціями і всеосяжному аудиту;
- Більш швидке розпізнавання проблем і усунення неполадок;
- Покращений кінцевий захист, контроль і видимість;
- Комплексне управління провідними і бездротовими мережами, традиційними і SDN, а також фізичними та віртуальними мережами;
- Чудова гнучкість і масштабованість для мереж будь-якого розміру;
- Підтримка багатьох виробників (6000 пристроїв від більш, ніж 220 виробників – HP IMC).

Нижче представлені деякі системи керування мережами ЦОД від великих вендорів:

- Cisco Data Center Network Manager;

- HP Intelligent Management Center;
- Huawei ManageOne Data Center Management Solution;
- Brocade Network Advisor.

Такі системи керування є комплексним рішенням для управління передовими корпоративними мережами і ідеально підходять для великих IT підприємств і центрів обробки даних. Вони використовують модель сервіс-орієнтованої архітектури, щоб забезпечити повне і розширюване обслуговування і функціональні можливості управління для користувачів. Система керування також забезпечує продуктивність і масштабованість за рахунок розподілених та ієрархічних моделей розгортання і через змінні параметри для операційної системи і підтримки бази даних. Модульна конструкція дозволяє таким системам інтегрувати традиційні окремі засоби управління в єдину уніфіковану платформу.

Базова платформа надає наступні можливості:

- Управління ресурсами, включаючи управління мережевими пристроями по SNMP, Telnet, SOAP, NETCONF, WMI PowerShell, конфігурації SSH, Spanning Tree конфігурацій і управління енергії PoE (*Power over Ethernet*) і багато іншого.
- Конфігурація, аудит та управління змінами параметрів пристроїв і програмних системних файлів для пристроїв. Система ключає в себе зберігання, резервне копіювання, порівняння і розгортання файлів конфігурації та програмного забезпечення за допомогою базової системи.
- Автоматизація в формі автоматичного виявлення і відображення всіх пристроїв в мережі, автоматичне виявлення віртуальних машин і віртуальних комутаторів, а також їх відносин з фізичної мережею.

2.4 Аналіз основних архітектур мережі центрів обробки даних

Архітектури мереж центрів обробки даних традиційно були розгорнуті як багаторівневі середовища, де сервери підключені до комутаторів рівня доступу [7][10]. Потім останні підключаються до комутаторів рівня агрегації або кінцевих/row - комутаторів, котрі вже підключаються до комутаторів рівня ядра. Незважаючи на те, що ці архітектури успішно і широко розгорнуті, вони мають недоліки в тому, що потрібно занадто багато мережевих пристроїв, маршрутизація відбувається тільки в ядрі, і їм потрібні високі рівні subscription ratio.

Ці типи архітектур можуть як і раніше мають своє місце, але сучасні центри обробки даних мають можливість скористатися комбінацією новіших високопродуктивних, масштабованих модульних комутаторів рівня ядра і рівня доступу з багатим набором функцій для вирівнювання і спрощення центру обробки даних.

Основні завдання, які мають вирішувати архітектури мережі ЦОД:

- Перебільшення можливостей мережі через зростання переадресації кадрів та переадресації пакетів;
- Гнучкість vMotion рівня L2;
- Зниження складності управління;
- Нульова необхідність в STP/RSTP;
- Спрощення взаємодії ЦОД;
- Централізована безпека.

2.4.1 Блейд-серверна 1-рівнева архітектура

Дивлячись на сплощеня мережі і забезпечуючи істотну підтримку віртуалізації і потоків трафіку «схід-захід»», блейд-серверна 1-рівнева архітектура надає багато переваг. Блейд-сервери дозволяють суттєво збільшити

обчислювальну щільність на стійку і ряд, а деякі фірми (наприклад, HP) оптимізували портфоліо своїх блейд-серверів, щоб підтримати видимість і реальність віртуалізації. Цей тип блейд-серверної 1-рівневої архітектури оптимізує і поєднує в собі реальність високо-продуктивної мережі з її простотою. Архітектура забезпечує гнучкість в області мережевих технологій, підтримуючи IRF і різноманітні варіанти взаємодії ЦОД.

Типова блейд-серверна 1-рівнева архітектура зазвичай складається з двох-чотирьох комутаторів рівня ядра з використанням IRF, які потім підключаються до блейд-серверів, розміщених в корпусах С-класу. Цей тип архітектури підтримує технології, що описані вище, однак, вона особливо добре підходить для великих розгортань рівня L2, чиї віртуальні локальні мережі проходять через весь центр обробки даних, дозволяючи віртуальним машинам переміщатися від стійки до стійки або, навіть, від центру обробки даних до центру обробки даних.

Цей тип архітектури зазвичай використовує щонайменше 40GbE висхідні лінії зв'язку (4x10GbE агреговані канали) в корпусі С-класу, що при використанні 10GbE LAN на мережевому адаптері материнської плати, забезпечує співвідношення 4:1 oversubscription ratio на корпус. Проте, співвідношення oversubscription ratio може бути зменшене до 2:1, просто додавши ще одну 40GbE висхідну лінію зв'язку в кожному корпусі, або до 1:1 за допомогою чотирьох 40GbE висхідних ліній.

Відображення блейд-серверної блейд-серверної 1-рівневої архітектури представлене на рис. 2.5.

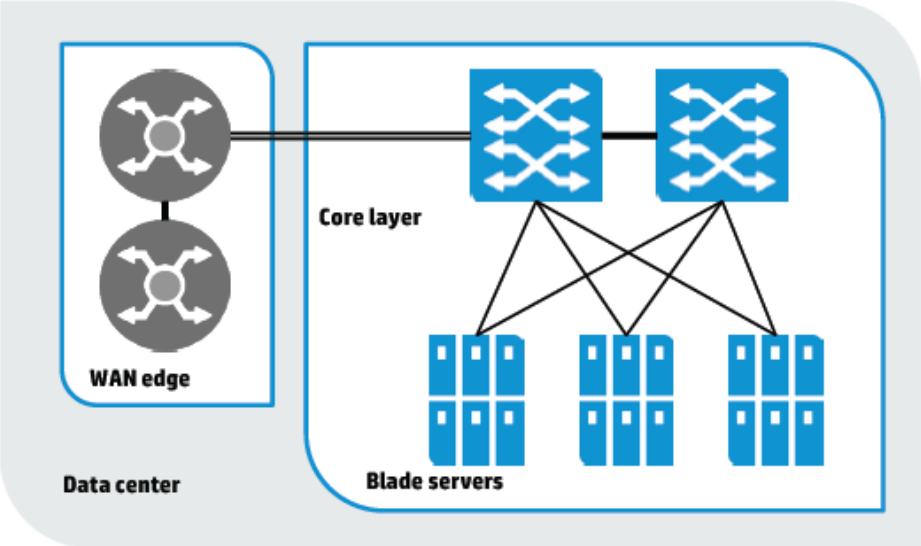

Рис. 2.5 Відображення блейд-серверної 1-рівневої архітектури [7]

2.4.2 Архітектура «spine and leaf»

Архітектура «spine and leaf», яку іноді називають ще 2-рівневою, може забезпечити збалансовану мережу, яка може бути оптимізована для віртуалізації, забезпечуючи при цьому велике масштабування і гнучкість, так що центр обробки даних може адаптуватися до мінливих вимог додатків. Архітектура також використовується, коли фізична кабельна станція не може підтримувати 1-рівневу архітектуру.

Багато розгортань «spine and leaf» використовують від двох до чотирьох великомасштабних комутаторів рівня ядра («spine») з використанням IRF, які потім в свою чергу, підключаються до різних ToR-комутаторів («leaves»). Ці leaf-комутатори, які можуть і не використовувати IRF, потім підключаються до серверів в одній стійці. Ці leaf-комутатори також можуть бути блейд-комутаторами, які встановлюються безпосередньо в корпусах C-класу (HP).

Великомасштабна L2 архітектура «spine and leaf» також може бути побудована з використанням SPB, як технології, що використовується для розширення мереж L2 через центральну магістральну мережу ЦОД. Як і в разі з TRILL, пристрої під управлінням SPB також можуть використовувати IRF.

Великомасштабна андерлейна (основна) L3 архітектура «spine and leaf також можлива. Ці розгортання можуть використовувати IRF, а також можуть бути масштабованими для підтримки десятків тисяч фізичних портів. Розгортання L3 має можливість використовувати протокол OSPF або BGP з ефективною переадресацією трафіку, а також забезпечувати більшу підтримку масштабованості, ніж та, яка може бути досягнута шляхом розгортання L2.

Фізичні з'єднання між spine і leaf в цій архітектурі будуть або групами агрегації каналів (LAG) або маршрутизованим каналом 10GbE, 40GbE і навіть висхідним 100GbE, забезпечуючи дуже високу продуктивність, яка може забезпечити чудову продуктивність лінії між spine і leaf.

Відображення архітектури «spine and leaf» представлене на рис. 2.6.

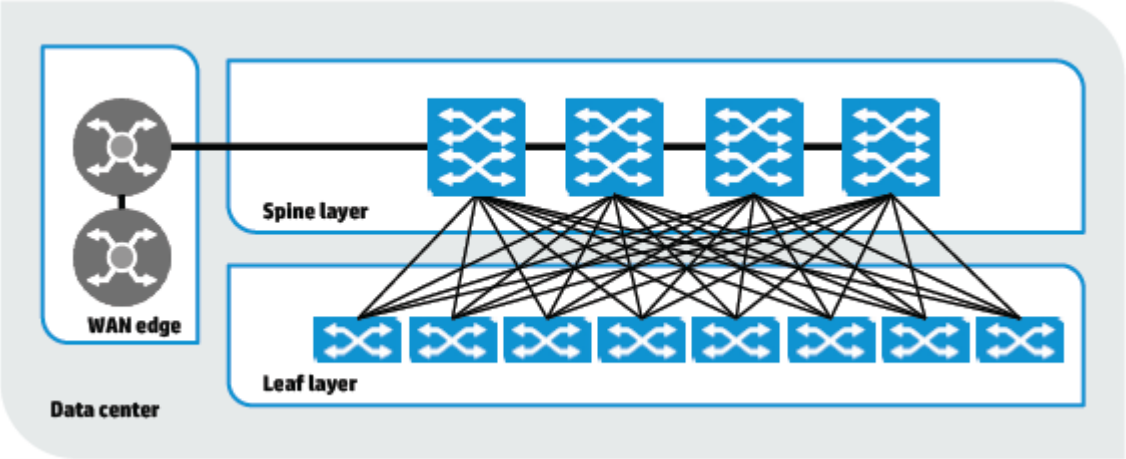

Рис. 2.6 Відображення архітектури «spine and leaf» [7]

2.4.3 3-рівнева архітектура

3-рівнева архітектура традиційно побудована із використанням декількох рівнів, які виглядають як дерево (зовнішня інтерпретація). Цей тип архітектури складається, принаймні, з наступних рівнів:

- Рівень доступу: зазвичай ToR пристрої, які підключаються до серверів;
- Рівень агрегації: забезпечує можливість агрегаційного з'єднання між рівнем доступу і рівнем ядра;

- Рівень ядра: зазвичай забезпечує високорівневі маршрутизаційні послуги всередині і між центрами обробки даних.

Типова 3-рівнева архітектура подібна 2-рівневій архітектурі «spine and leaf», однак, перше рішення має додатковий рівень агрегації, розташований між рівнем ядра та рівнем ToR.. Цей рівень агрегації, як правило, складається з комутаторів на основі шасі і L3 маршрутизації, яка зазвичай відбувається між рівнями агрегації та ядра , а потім в глобальній мережі. Таким чином, домени L2 можуть бути виділені для кожного пристрою рівня агрегації і ToR комутаторів, які підключаються до нього.

Тип і кількість комутаторів, розміщених на кожному рівні в цій архітектурі буде варіюватися в залежності від вимог в межах кожного центру обробки даних. Оскільки ця архітектура агрегує велику кількість пристроїв доступу в комутатори рівня агрегації, які потім підключаються до комутаторів рівня ядра, висхідна лінія зв'язку між комутаторами може представляти собою вузьке місце і з перевищенням кількості підключень. Вкрай важливо, щоб ці розгортання прийняли це до уваги і використовували великі групи агрегації каналів (LAG) і висхідні канали з широкими смугами пропускання між кожним рівнем.

Інша загальна 3-рівнева архітектура може використовувати PODи (ізольовані кластери стійок L2, які вимагають маршрутизації між кластерами) - в цьому випадку кожен POD має важливе значення. Цей тип архітектури виглядає, як 2-рівнева архітектура з третім рівнем, який забезпечує зв'язок між POD, а також за межами дата-центру. Розроблений належним чином, за допомогою цього підходу рівень oversubscription ratio можна зберігати нижчим, так як більша частина трафіку буде залишатися в межах POD.

Відображення 3-рівневої архітектури представлене на рис. 2.7.

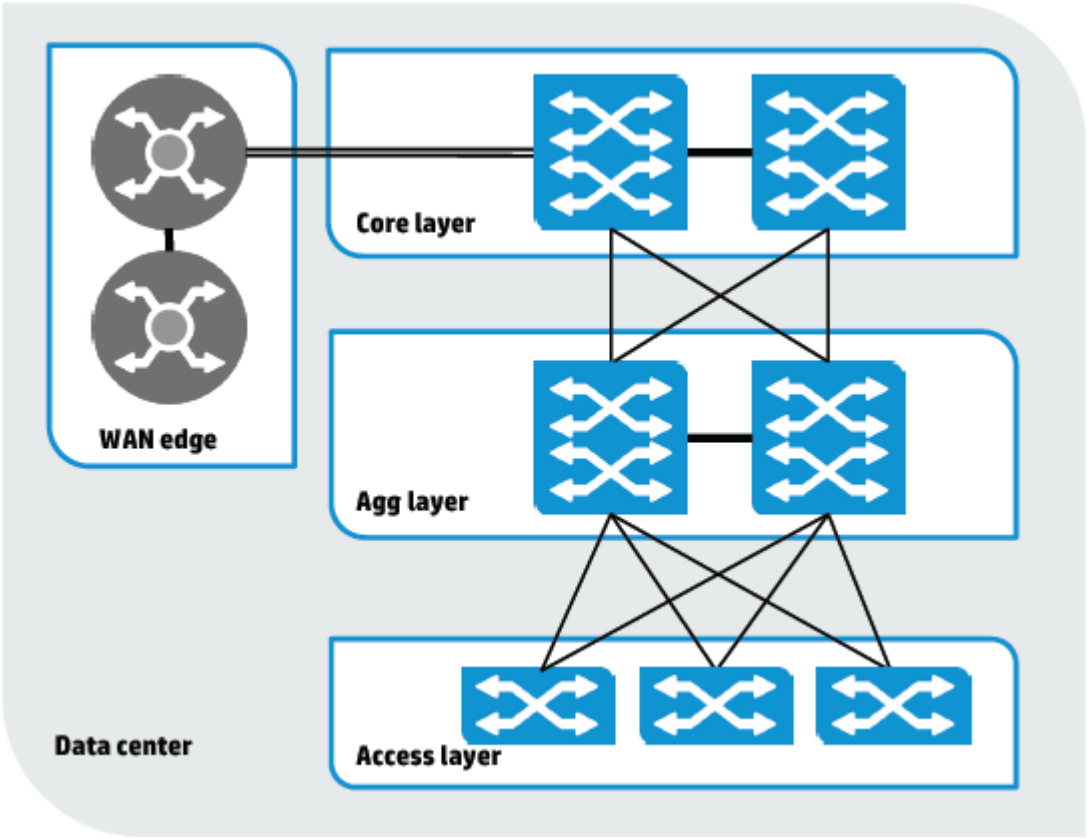

Рис. 2.7 Відображення 3-рівневої архітектури [7]

Висновки з розділу 2

1. Архітектори центрів обробки даних при їх розробці повинні фокусуватися на деяких основних моментах: використання передових платформ і мережевих технологій, конвергенція всієї мережі і інфраструктури ЦОД, підтримка швидкої переконфігурації мережі, гнучкість мережі.
2. Досліджено основні ділові та технологічні рушії для мережевої архітектури центру обробки даних, котрі дозволяють останнім відповідати сучасним потребам користувачів.
3. Проаналізовано основні вимоги до мережі центру обробки даних. Згідно з ними віртуалізація є обов'язковою, так як завдяки ній можна легко підвищити ефективність, гнучкість та відмовостійкість центрів обробки даних. Крім цього, згідно з вимогами, більшість операцій, таких як

міграція або тунелювання, бажано проводити в мережі рівня L2, без використання L3.

4. Через це було досліджено основні протоколи рівня L2, котрі можуть використовуватися в центрах обробки даних. Потрібно зауважити, що деякі з них, наприклад, NVGRE та VXLAN, дещо подібні за принципами роботи, їх використання залежить від обраного виробника мережевого устаткування, котрий підтримує той чи інший протокол. Також були досліджені протоколи, такі як SPB, котрі виступають на заміну старому протоколу STP.
5. Згідно з аналізом вдаємодії центрів обробки даних основними вимогами до DCI вважаються висока доступність, ізоляція STP, управління петлями.
6. Розглянуто основні характеристики і вимоги до систем керування мережі центру обробки даних. Такі системи повинні мати наступні можливості: управління ресурсами мережі, конфігурація, аудит та управління параметрами пристроїв, автоматизація.
7. До основних архітектур мережі центрів обробки даних відносять блейд-серверну 1-рівневу архітектуру, 2-рівневу архітектуру «spine and leaf», а також традиційну 3-рівневу архітектуру. Остання вважається застарілою, проте на даний момент ще використовується в багатьох ЦОД.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ АРХІТЕКТУРИ ПОБУДОВИ ЦЕНТРІВ ОБРОБКИ ДАНИХ НА БАЗІ ТЕХНОЛОГІЇ SDN

3.1 Вступ

З початком використання технології SDN у центрах обробки даних ідеологія і «буденне життя» останніх повністю змінилося. Тепер всі інтелектуальні функції мережі сконцентровані в контролері(ах) SDN, який повністю контролює мережу і адаптивно змінює її під мінливі потреби користувачів. Вже не потрібно бігати по всьому ЦОД і шукати той самий комутатор, а потім власноруч настроювати йому тригери – це все дозволяє робити автоматизовано система управління мережею через контролер SDN. Переконфігурувати один комутатор не дуже важко, але якщо їх сотні чи тисячі, а ще вони й територіально розділені, то це стає вкрай важким завданням - необхідно затратити значні людські ресурси для виконання даного завдання.

Переконфігурування може відбутися через велике навантаження на сервер – контролер, фіксуючи високий рівень навантаження на сервері (більше 80%), може автоматично перевести частину навантаження, такого, як працюючи віртуальні машини, на сусідній сервер. Чи в разі простою частини мережі з нульовим навантаженням, контролер може вимкнути саме цю частину мережі збільшивши енергоефективність центру обробки даних.

У випадку «ЦОД під оренду», ЦОД на SDN для своїх власників стає подібний квартирі, яку здає орендодавець. Орендар, знімаючи квартиру, теоретично може робити в ній, що захоче, але при цьому відповідальність за квартиру лежить на обох – на орендодавці та орендарі, а не повністю передається на останнього. Аналогічне відбувається і в ЦОД: орендар знімає у власне користування частину потужностей дата-центру і майже повністю починає нею володіти і використовувати на свій розсуд. За орендодавцем залишаються тільки різні функції підтримки (апаратне забезпечення ЦОД, енергоживлення, кондиціювання).

На початку оренди на контролер(и) SDN встановлюється додаткове програмне забезпечення чи від орендаря, якщо воно повинне виконувати якісь специфічні дії, чи від орендодавця, для керування мережею та вводом/виводом якихось, наприклад, статистичних, даних. Такий додаток працює паралельно з програмним забезпеченням від власника ЦОД в іншому домені довіри не заважаючи один одному. Програмне забезпечення власника ЦОД в своєму домені довіри, наприклад, встановлює політики функціонування мережі, які були прописані в контракті між власником ЦОД та орендарем– пропускну здатність каналу, QoS, трафік тощо. Також це ПЗ автоматично виставляє рахунок орендарю за користування потужностями ЦОД. Таким чином, орендодавець повністю відмовляється від основного «головного болю» - конфігурування мережі, надаючи таку можливість орендарю з його ПЗ.

У випадку корпоративного ЦОД на базі SDN компанії вже не потрібно мати велику кількість мережевих інженерів, котрі можуть переконфігурувати мережу і обладнання під змінені вимоги. Всі ці дії проводяться автоматично завдяки програмному забезпеченню встановленому на контролері SDN. Такій компанії потрібно мати невеликий штат програмістів, котрі розуміються на написанні програм «під SDN» і будуть оновлювати програмне забезпечення контролеру.

Ще однією перевагою SDN є її можливість реалізації на мережах любих топологій (1-2-3-рівневих і їх похідних) – одним із головних факторів при розгортанні програмно-визначеної мережі в ЦОД є можливість комутаторів працювати по протоколу OpenFlow, тобто докорінно змінювати існуючу інфраструктуру не приходиться. Як приклад – на рисунку 3.1 зображена реалізація SDN в мережі центру обробки даних на основі 2-рівневої архітектури «Spine and Leaf», яка описувалася в пункті 2.4.2.

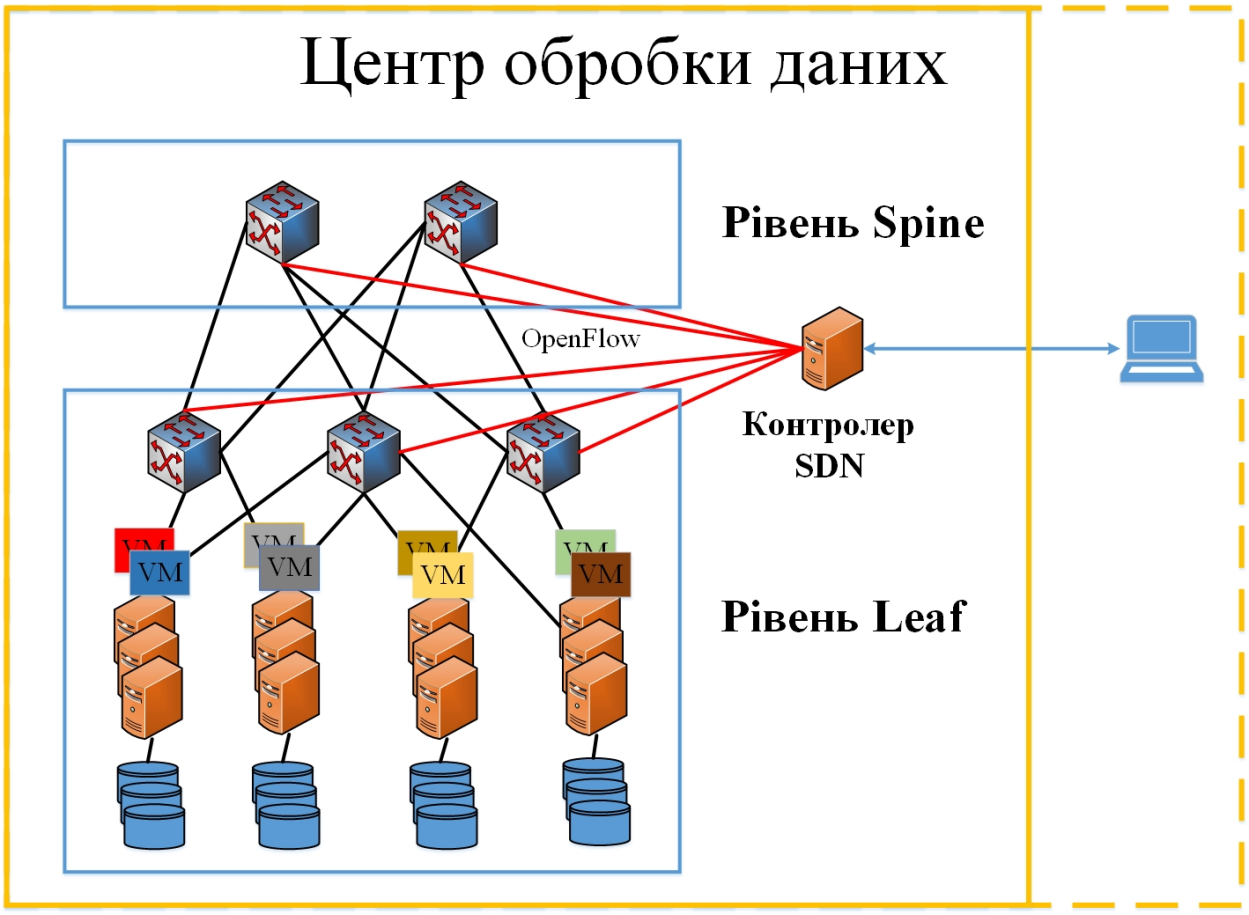

Рис. 3.1 Реалізація SDN в мережі ЦОД на основі 2-рівневої мережевої архітектури

Дійсно, існуючі реалізації центрів обробки даних без SDN від великих компаній (HP, VMware, Brocade, Cisco, Juniper) дуже подібні за своїми технологіями та принципами на ЦОД, виконаний повністю на основі SDN. Ці дата-центри можуть проводити живу міграцію, дякуючи гіпервізорам, можуть частково контролювати чи масштабувати мережу, але все це проводиться мануально чи напівавтоматично і при цьому використовується багато людських ресурсів. В наш час автоматизації і швидкості життя така затримка в просторі інформаційних технологій недопустима.

З початком використання SDN повністю все середовище обробки даних вже не можна розглядати тільки, як «залізо». Тепер в цьому середовищі зайняла перше місце прикладна програма – додаток, який і керує всім апаратним забезпеченням і інфраструктурою центру обробки даних (рис. 3.2). Додаток вже

вирішує як жити в даний момент дата-центру, аналізуючи дані, які поступають на контролер зі всієї мережі.

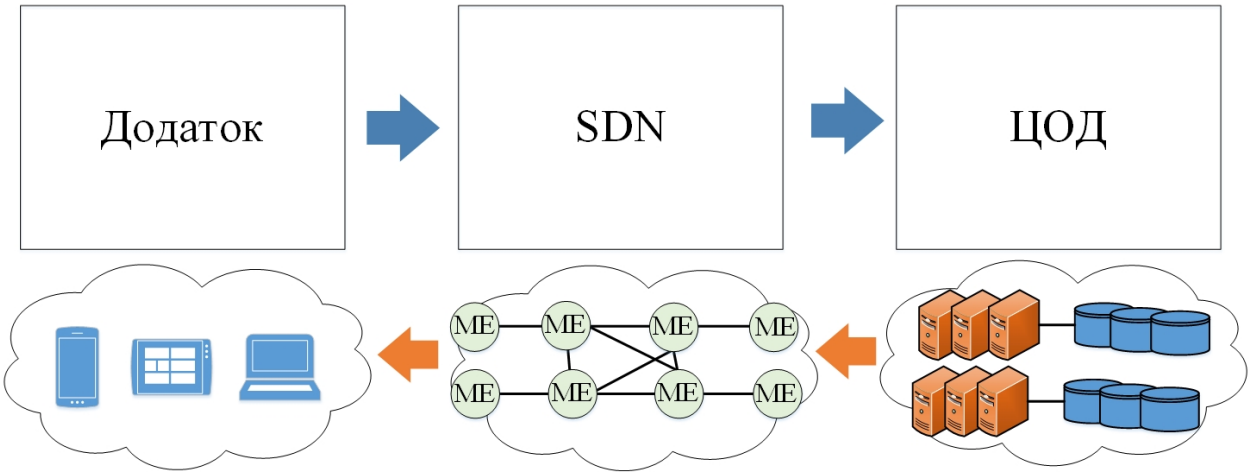

Рис. 3.2 Сьогоденне відображення всього середовища обробки даних

Програмно-визначені мережі фізично майже нічого не змінюють в інфраструктурі центру обробки даних, основні зміни привносяться якраз в невидимому для нас аспекті. Завдяки програмній реалізації технології виносяться на інший рівень основні вимоги до центрів обробки даних:

- Доступність;
- Масштабованість;
- Безпека;
- Продуктивність;
- Керованість;
- Надійність.

Далі будуть проаналізовані основні фактори, завдяки яким використання технології програмно-визначених мереж вважається доцільним в центрах обробки даних.

3.2 Аналіз централізації управління мережею SDN в центрах обробки даних

SDN прагне ефективно програмувати мережі за допомогою програмного забезпечення, що працює на центральному контролері. Сучасні мережеві комутатори і маршрутизатори програмують свої таблиці переадресації на місцевому рівні, що означає, що мережеві пристрої приймають свої власні рішення про те, як направляти трафік. Переадресаційні рішення проінформовані розподіленими протоколами площині управління, таким як STP, OSPF і BGP. Але ці традиційні мережеві протоколи мають обмежену гнучкість. Для того, щоб вони працювали, всі мережеві пристрої, що беруть участь в домені переадресації повинні слідувати тим же правилам, як це визначено стандартом протоколу. Це залишає мало місця для творчості або незвичайних вимог бізнесу.

SDN бере площину управління (як мережевий пристрій буде передавати трафік) і відокремлює її від площини даних (переадресація трафіку мережевого пристрою, заснована на політиці площини управління). З SDN, відокремлена площина управління знаходиться на централізованому контролер, який бачить і знає все про мережі, в тому числі, де вузли підключаються до мережі і те, як виглядає мережева топологія підключень всіх хостів. Всезнаючий центральний контролер дозволяє мережевим інженерам реалізувати унікальні і гнучкі політики переадресації обмежується лише здатністю програмного забезпечення, що працює на ньому.

Контролер виступає в ролі арбітра, абстрагуючи базову топологію фізичної мережі від додатків, які хочуть програмувати її. Типовими прикладами, коли центральний контролер програмує таблиці мережевих переадресацій, є наступні:

- Створення експериментальної мережі, яка працює на тій же фізичній інфраструктурі, як і основна мережі, в той час коли вони логічно ізольовані;

- Транспортні потоки, що динамічно розпізнаються, мають політики, що застосовуються до них, щоб підтримувати певну якість обслуговування (QoS);
- "Маршрутизація для доларів," де переадресаційні шляхи можуть спиратися на фінансові наслідки для ЦОД чи компанії;
- Інтелектуальна мережева безпека, де окремі потоки можуть бути шунтовані, наприклад, до системи виявлення вторгнень для глибокої інспекції пакетів, в той час як інші потоки можуть вільно проходити на основі операційної політики;
- Мереже дзеркалювання трафіку, де потоки трафіку з будь-якої точки в мережі можуть бути скопійовані в будь-яку іншу точку мережі для реєстрації, звітності та аналізу.

Архітектура SDN визначає, що мережева інфраструктура логічно управляється центральною сутністю, відповідальною за управління та застосування політик. Проте, слід чітко вказати, що логічно централізоване управління не обов'язково також має на увазі і фізичну централізацію.

Дійсно, але і існують різні пропозиції фізично централізованих контролерів, таких, як NOX і Maestro. Архітектура фізично централізованого контролю спрощує реалізацію контролера. Всі комутатори знаходяться під контролем фізичної сутності, а це означає, що мережа не підлягає питанням пов`язаним з узгодженням, з усіма додатками бачачи стан тієї ж мережі (яка походить від того ж контролера). Незважаючи на свої переваги, цей підхід страждає від тієї ж слабкості, що і всі централізовані системи, тобто контролер виступає в якості єдиної точки відмови для всієї мережі. Подолати це можна шляхом підключення декількох контролерів до комутатора, дозволяючи резервному контролеру взяти на себе керування в разі виходу з ладу основного. В цьому випадку всі контролери повинні мати узгоджене представлення мережі; в іншому випадку додатки можуть не працювати належним чином.

Крім того, централізований підхід може підняти проблеми масштабованості, так як всі мережеві пристрої повинні управлятися однією і тією ж сутністю.

Один з підходів, який далі узагальнює ідею використання декількох контролерів в мережі ЦОД, - це підтримка логічно централізованої, але фізично децентралізованої площини управління (рис. 3.3). У цьому випадку кожен контролер відповідає за управління тільки однієї частини мережі, але всі контролери спілкуються і підтримують загальне відображення мережі. Таким чином, додатки бачать контролер як єдине сутність, а в дійсності операції управління виконуються за допомогою розподіленої системи. Перевага такого підходу, крім відсутності єдиної точки відмов, це ще й підвищення продуктивності і масштабованості, так як тільки частина мережі управляється окремим контролером.

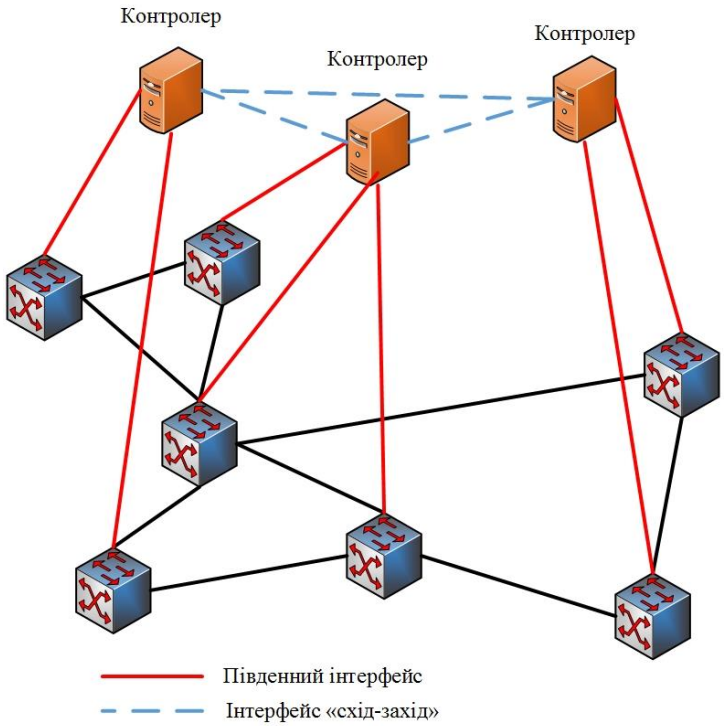

Рис. 3.3 Реалізація логічно централізованої, але фізично децентралізованої площини управління контролерів

3.3 Аналіз системи керування мережі ЦОД на базі технології SDN

Система керування мережі SDN – це програмне забезпечення, що забезпечує єдиний контроль в SDN-мережі, що спрощує управління, виділення ресурсів і оркестрацію [9][10]. Система дає можливість доставки мережевих послуг нового покоління на базі додатків і забезпечує прикладні програмні інтерфейси (API), які дозволяють розробникам створювати інноваційні рішення для зв'язування динамічних бізнес-вимог з інфраструктурою мережі через Java програми або інтерфейси управління загального призначення RESTful. Такі системи в основному є універсальними і призначені для роботи в університетському містечку, центрі обробки даних або середовищ постачальника послуг.

Програмне забезпечення системи керування безпосередньо забезпечує фізичні і віртуальні комутатори контролем за допомогою пропрієтарного OpenFlow протоколу, а також NETCONF і SNMP. Мережеві порти, канали і топології відображаються в системі, що дозволяє вести централізовану політику та ефективний вибір шляху на основі динамічної та глобальної точки зору мережі. Це різко спрощує оркестрацію багатокористувацьких середовищ і спостереження за мережевими політиками, як для мобільних клієнтів, так і для серверів.

3.3.1 Загальний опис системи керування мережі SDN

Незалежно від конкретної сутності контролера, стек програмного забезпечення системи керування буде складатися з двох основних рівнів (рис. 3.4). Верхній рівень Адміністратору буде керувати функціоналом, пов'язаним із поширенням політики, управлінням, особистою взаємодією і зовнішньою взаємодією додатків. Нижній рівень Контролера, з іншого боку, буде керувати забезпеченням політики, взаємодією пристроїв, взаємодією потоків. Інтерфейс (и) між цими двома рівнями забезпечує схему брандмауера і еластичні в тому,

що вони можуть змінюватися разом з сутністю загального пристрою управління. Крім того, вони регулюються правилом, що ніякі примусові синхронні взаємодії не будуть переходити від рівня Контролера до рівня Адміністратора.

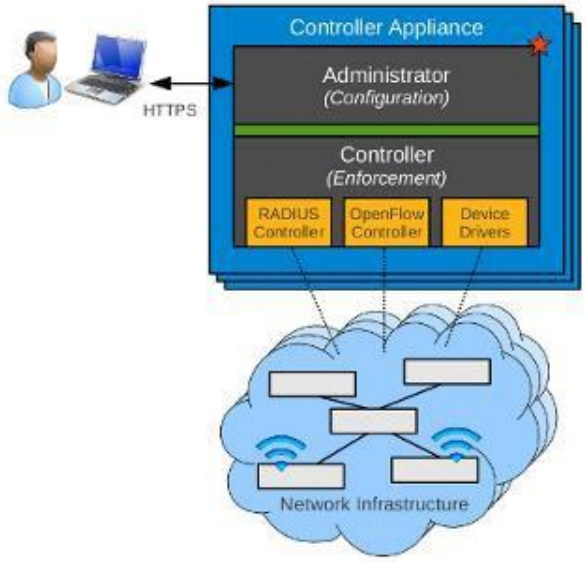

Рис. 3.4 Рівні ПЗ системи керування SDN [11]

Рівень Адміністратору контролера контролює веб-рівень, через який програмні модулі, встановлені на приладі можуть відобразити REST API (або RESTful веб-сервіси) на інші зовнішні об'єкти. Крім того, модулі можуть розширити доступний веб-графічний інтерфейс, що дозволяє мережевим адміністраторам та іншим користувачам взаємодіяти безпосередньо з програмним забезпеченням, що працює на контролері.

Веб-додаток – це додаток, до якого звертаються користувачі через мережу, таку як Інтернет або інтранет. Наприклад, HP VAN SDN Controller працює на веб-сервері, як показано на рисунку 3.5.

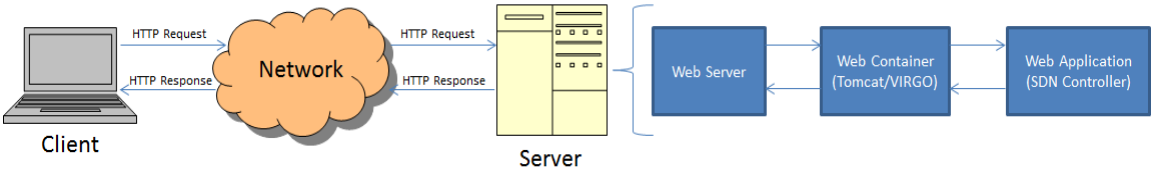

Рис. 3.5 Архітектура веб-додатку [11]

Сервлети – це технологія, що використовується для розширення функціональних можливостей веб-сервера і для доступу до бізнес-систем. Сервлети забезпечують сумісний із різними платформами метод на для створення веб-додатків.

SDN додатки не реалізують сервлети безпосередньо, але замість цього вони реалізують RESTful веб-служби, які основані на сервлетах. Однак веб-служби RESTful також діють як контролери, як зображено на схемі на рисунку 3.6.

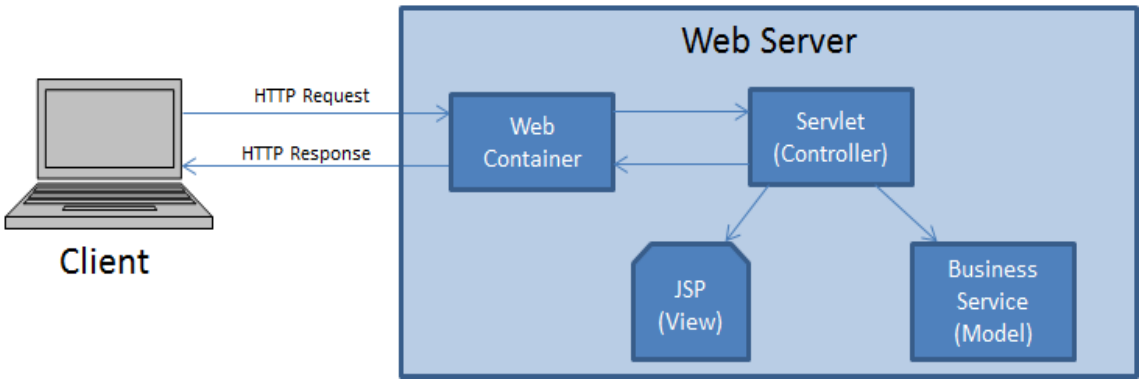

Рис. 3.6 Відображення моделі веб-додатку контроллеру [11]

3.3.2 Архітектура системи керування мережі SDN

Основний програмний стек приладу використовує фреймворк OSGi (Equinox) і контейнер (Virgo) в якості основи для модульного розгортання програмного забезпечення і для забезпечення послуг поділу постачальник/споживач. Програмне забезпечення працює в головному контейнері OSGi і може взаємодіяти з іншими компонентами, запущеними в

інших процесах на приладі. Переважно така IPC взаємодія буде відбуватися з використанням стандартного механізму зберігання, наприклад, RabbitMQ, але вони можуть використовувати будь-які засоби IPC, які найкраще підходять для зовнішнього компонента. Virgo, заснований на Tomcat, являє собою модульний сервер Java додатку, який призначений для запуску корпоративних Java-додатків з високим ступенем гнучкості та надійності. Рисунок 3.7 ілюструє стек програмного забезпечення HP VAN SDN Controller.

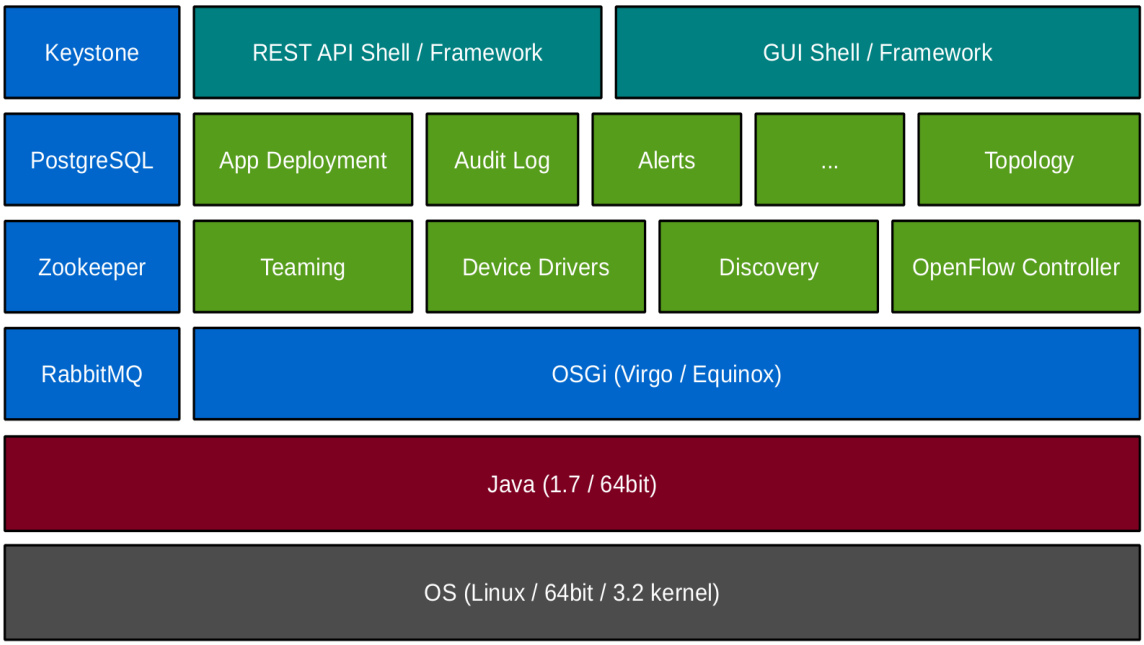

Рис. 3.7 Програмний стек HP VAN SDN Controller [12]

Jersey – це JAX-RS (JSR 311) еталонної реалізації для створення веб-служб RESTful. У Representational State Transfer (REST) архітектурні стиль, дані і функціональні можливості розглядаються, як ресурси, а також доступ до цих ресурсів з використанням єдиних ідентифікаторів ресурсів (URI), як правило, посилань на веб-сайті. Архітектури REST-стилю зазвичай складаються з клієнтів і серверів, і вона призначена для використання протоколу зв'язку без інформації про стан, як правило, HTTP. Клієнти ініціюють запити до серверів; сервери обробляють запити і повертають відповіді. Запити та відповіді будуються навколо передачі представлень ресурсів. Клієнти і сервери

обмінюються представленнями ресурсів з використанням стандартного інтерфейсу і протоколу. Ці принципи заохочують RESTful додатки бути простими, легкими і мати високу продуктивність.

Деякі системи керування також пропонують фреймворки для розробки інтерфейсів веб-користувача. Фреймворк забезпечує основу, на якій розробники можуть створювати веб-браузерні додатки.

Система керування використовує зовнішні сервіси, що забезпечують API-інтерфейси, які дозволяють SDN програмам використовувати їх.

Зовнішня служба Keystone забезпечує аутентифікацію і високорівневу авторизацію послуг. Вона підтримує схему аутентифікації маркери на основі, яка використовується для забезпечення безпеки RESTful веб-служб (або REST API) і веб-інтерфейсу користувача.

Zookeeper - це координаційна служба для розподілених додатків. Це централізована служба для збереження інформації про конфігурацію, імена, що забезпечує розподілену синхронізацію, а також надання групових послуг.

Apache Cassandra - це високопродуктивна, дуже масштабована, відмовостійка (без єдиної точки відмови), розподілена постреляційна реалізація бази даних. Cassandra поєднує в собі всі переваги Google Bigtable і Amazon Dynamo і дає можливість використовувати ті типи управління базами даних, які традиційні постачальники СУБД не можуть підтримати.

В 4 розділі буде представлений графічний користувацький інтерфейс HP VAN SDN Controller.

3.3.3 Опис основних функцій і переваг системи керування мережі SDN

Система керування мережею SDN може представляти із себе, як і повноцінне незалежне програмне забезпечення, так і додаткове програмне забезпечення, яке інтегрується в систему керування звичайною фізичною мережею.

Наприклад, HPE IMC Networks Virtual Application (VAN) SDN Manager Software являє собою HP IMC модуль, який забезпечує комплексне управління, в тому числі помилок, конфігурації, обліку, контролю і безпеки для навколишнього середовища SDN.

IMC VAN SDN Manager Software контролює і управляє всіма трьома рівнями в SDN архітектурі: рівнем інфраструктури, управління, додатків. Це дозволяє візуалізувати SDN, в тому числі розташування комутаторів, які фізично і логічно відносяться до контрольна точка мережі. Транспортний потік через домен SDN контролюється і представляється візуально в IMC VAN SDN Manager Software, що дозволяє проводити швидкий пошук несправностей. Система також підтримує управління життєвим циклом і моніторинг HPE VAN SDN контролера і забезпечує відомості про стан мережевих послуг і OpenFlow-інформацію.

Основні функції і переваги системи керування мережею SDN:

- Проактивна обробка потоку: забезпечує високомасштабовані, централізовано оркестровані мережі SDN;
- Реактивна обробка потоку: забезпечує динамічний моніторинг нових потоків або кінцевих точок, таких як сесій окремого користувача або сервера;
- Графічний інтерфейс користувача (GUI): полегшує адміністрування контролера і API документування;
- Північні API: використовують розширюваний інтерфейс RESTful HTTPS контролера; забезпечують абстрактне представлення базової мережі OpenFlow і дозволяють зовнішнім додаткам, що працюють над контролером, надавати нарадчий бізнес-контроль над мережею;
- Native API: дозволяє Java додаткам працювати в контролері як сукупність OSGi наборів, які забезпечують високопродуктивні події і обробку пакетів;

- Масштабована архітектура: використовує масштабовані, упругі фреймворки баз даних, що дозволяє розширити рамки одного автономного контролеру в рамки високодоступного кластеру;
- Висока доступність: забезпечує "2n+1" модель активної консистентності, яка дозволяє трьом контролерам управляти окремими підмножинами мережі, розділяючи відображення загальної мережі; відмова одного компонента управління генерує швидку відповідь на кластері, щоб забезпечити безперервну роботу мережі;
- Безпека контролера: забезпечує безпеку на різних рівнях; HTTPS використовується для REST API, а аутентифікація користувачів і додатків здійснюється за допомогою служби достовірності Keystone; зв’язок контролер-комутатор забезпечується через протокол шифрування Transport Layer Security (TLS), як зазначено в стандарті OpenFlow;
- Гнучка система обробки пакетів: списки управління доступом можуть бути підготовлені централізовано, наприклад, в той час як L2 або L3 рішення щодо переадресації можуть бути виконані з використанням стандартних мережевих протоколів;
- Інтерфейс управління OpenFlow: використовує узагальнений підхід до південного інтерфейсу контролера для обробки OpenFlow 1.0 і 1.3 повідомлень; це забезпечує ефективний і інтуїтивний механізм для моніторингу і програмування різних мережевих компонентів і обробку нових потокових повідомлень;
- Управління мережею OpenFlow: забезпечує управління OpenFlow ресурсами, політикою потоків, моніторингом трафіку, звітністю, пошуком і усуненням несправностей; можливість візуалізації мережевих потоків трафіку, якості обслуговування і статусу застосування SDN;
- Усунення несправностей: моніторинг помилок на основі топології через несправні лінії зв'язку і позиціонування пристроїв; відображає вражені хости і відповідні набори записів;

- Інформаційний сервіс: забезпечує реальні і історичні статистичні дані в докладних звітах, які можуть бути експортовані в різноманітні формати звітностей;

3.4 Аналіз живої міграції віртуальних машин в мережі SDN центру обробки даних

Жива міграція ВМ вже широко вивчена і популярні гіпервізори (наприклад, Xen, VMware та інші, які були згадані в пункті 2.3.2) тепер пропонують розумні рішення на практиці. Проте, міграція мережевих компонентів і стану мережі отримала мінімальну увагу. Проте, SDN може змінити це шляхом надання API, який дає нам можливість міркувати про міграцію мережі без прив'язки до конкретної реалізації, що дозволяє нам структурувати процес міграції через довільні кроки нашого вибору.

Жива міграція ВМ забезпечує ефективне для центрів обробки даних виконання балансування навантаження за рахунок міграції віртуальних машин з перевантажених серверів на менш сильно завантажені сервери. Адміністратори можуть динамічно перерозподіляти віртуальні машини через механізми живої міграції ВМ без істотних перерв в обслуговуванні. Проте, динамічна міграція ВМ в традиційних мережах як і раніше обмежена через дві основні причини. По-перше, жива міграція ВМ обмежена локальною мережею, оскільки IP не підтримує мобільність без переривання сеансу. По-друге, стан мережі непередбачуваний і важко контролювати його в сучасних мережевих архітектурах [13].

SDN передбачує динамічну міграцію віртуальних машин, так як площина управління є централізованою, що дає глобальну видимість мережі, та незалежна від рівневого стеку IP. Оскільки контролер SDN має інформацію про базові мережеві топології, SDN на основі міграції ВМ зменшило можливість переривання міграції через топологічні складнощі, що існують в традиційних мережах (рис. 3.8). Наприклад, щоб перенести віртуальну машину, нові повні

шляхи переадресації можна легко встановити без переривання обслуговування, просунувши нові правила переадресації в таблицях потоків комутаторів. При зміні існуючих правил потоку в OpenFlow-комутаторах навряд чи буде потрібно тимчасове зберігання існуючих пакетів потоку в порівнянні з перериванням сесії в нинішніх мережевих середовищах.

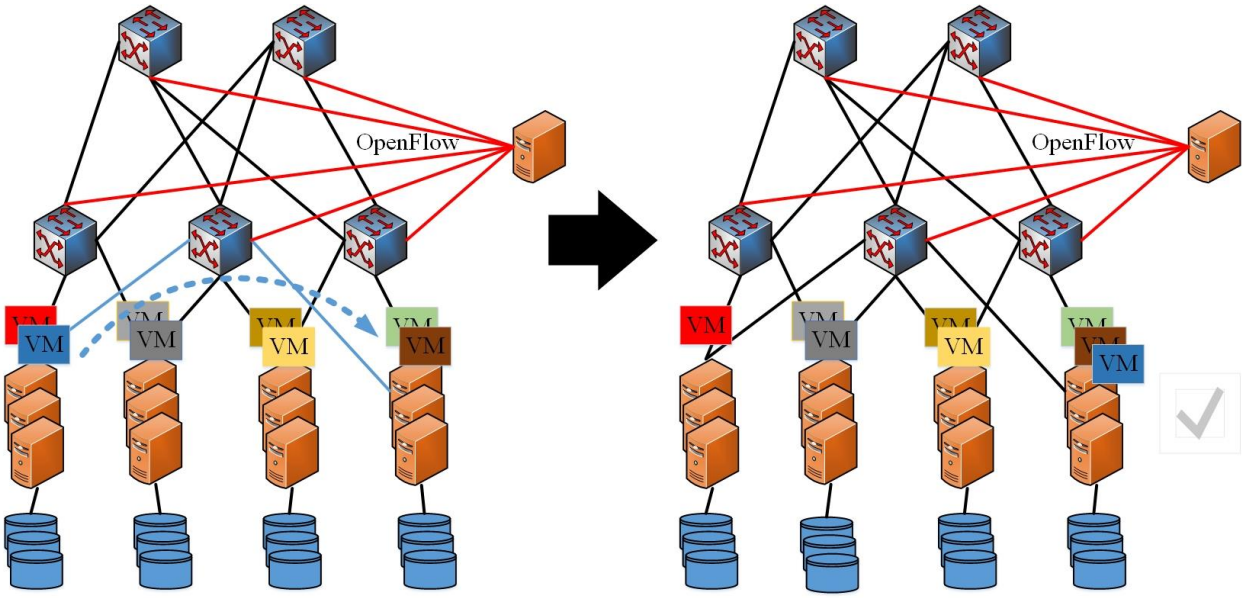

Рис. 3.8 Динамічна міграція ВМ в SDN-мережі центру обробки даних

SDN уможливила міграцію всієї системи, що складається з віртуальних машин, мережі і системи управління з різним набором фізичних ресурсів. Наприклад, проект LIve Migration Ensemble (LIME) використовує логіку поділу площин управління-даних SDN для міграції віртуальних машин, мережі та управління мережею. LIME клонує стан площини даних в новий набір OpenFlow комутаторів, а потім поступово мігрує джерела трафіку [14].

3.4.1 Опис технології LIve Migration Ensemble

LIME є загальним і дієвим рішенням для спільної міграції віртуальних машин і мережі. LIME ґрунтується на технологіях віртуалізації і SDN для дієвої міграції підмножини компонентів одночас.

Як показано на рисунку 3.9, LIME забезпечує головний рівень, що дає центру обробки даних свободу управляти мережею, усуваючи необхідність додатку управління мати справу з цими змінами. Це відповідальність LIME і віртуалізації основного (андерлей) рівня підтримувати абстракцію, що надається кожній віртуальній мережі. LIME дозволяє мережевим операторам (або автоматизованим процесам) ініціювати сумісну (ансамбль) міграцію через API для визначення, які елементи колективно мігрувати і де їх мігрувати. LIME працює нижче в тісному контакті з рівнем мережевої віртуалізації (наприклад, як FlowVisor або Flown), щоб виділити ресурси для міграційного процесу. Важливо відзначити, що LIME маскує ефекти міграції від додатків контролера. Ці додатки представлені з мережевою абстракцією, яка залишається постійною протягом міграції, що дозволяє їм продовжувати працювати в безшовному режимі.

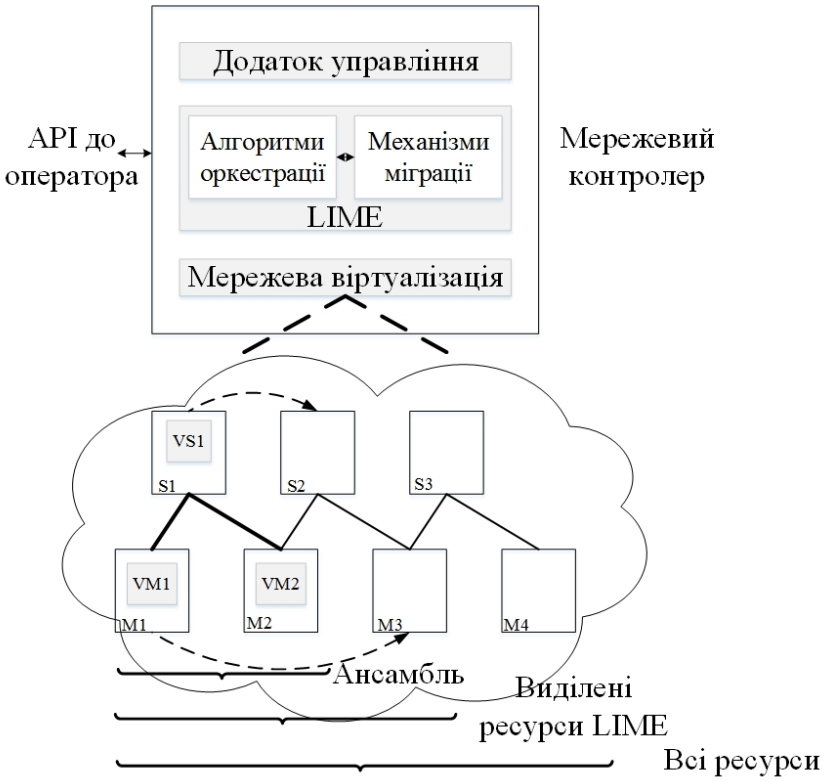

Рис. 3.9 Архітектура LIME

Пояснення до рисунку 3.9: віртуальна мережа складається з двох віртуальних машин і віртуального комутатора, який в даний час зіставлений до фізичних пристроїв в більшій інфраструктурі. Для міграції однієї віртуальної машини і одного віртуального комутатора до різних фізичних пристроїв, LIME працює з рівнем віртуалізації для розширення ресурсів виділених віртуальній мережі під час міграції.

3.5 Оптимізація «big data» в центрах обробки даних з SDN

Під поняттям "big data" розуміють перехід в області високопродуктивних обчислень від спеціально побудованих системи обчислень (наприклад, Sun Microsystems, CRAY і т.д.) до підходу, який дає можливість використовувати переваги економіки апаратних засобів COTS за рахунок використання менших, дешевших пристроїв, які можуть бути згруповані разом. Це досягається за допомогою підходів «розділяй та володарюй», які розділяють обчислювальні завдання на дрібні частини, як з точки зору обчислення даних, так з точки зору і фактичного обчислення, і поширюють їх через це менше, менш потужне, але значно менш дороге обладнання. Слід зазначити, що користувачі методів big data помітили природну спорідненість між топологічним відображенням і центральним управлінням SDN та деякими додатками великих big data [15].

Загалом, big data - зазвичай не виртуалізоване середовище, так як гіпервізор не потрібен. Hadoop є одним з найпопулярніших з класу кластерних обчислювальних архітектур для big data, яка використовує контролер додатків для управління запитами на роботу. Hadoop використовується для класу додатків названих Map/Reduces, що обробляють величезні обсяги даних, розбиваючи задачу (тобто набір даних) на ряд секцій / блоків, розподілених по цілому ряду машин для паралельної обробки. Ця система також використовує переваги розподіленої файлової системи Hadoop – HDFS (*Hadoop Distributed File System*).

Основні накладні витрати в додатках погягають в поширенні секційного файлу, зберіганні і подальшому зборі результатів. Це посилюється стратегією надмірності, що викликає кілька копій одного і того ж блоку, який буде поширений в разі, якщо один з обчислювальних вузлів впаде - реплікація є ієрархічною операцією.

Архітектура Hadoop має три функціональних компоненти: клієнти, провідний (master) і ведений (slave) (рис. 3.10). Клієнт є основним кінцевим користувачем кластера, котрий відправляє запит на роботу з файлом, щоб управляти інструкціями, як управляти/обробляти його, і збірає результати. Провідний вузол в Hadoop несе повну відповідальність за поширення файлів і управління вузлами обробки. Він взаємодіє з двома іншими вузлами: вузлом імен і трекером роботи (також він є залежним від них). Вузол імен несе відповідальність за розподіл/зберігання, а трекер роботи координує обчислення.

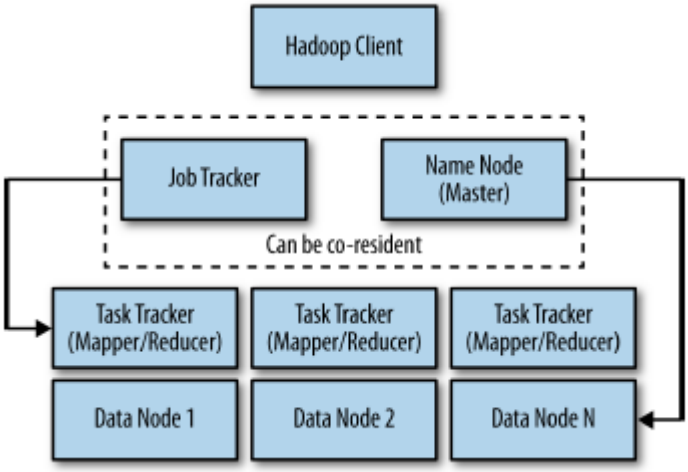

Рис. 3.10 Архітектура Hadoop [16]

Ведені називаються розподільниками (mapper) і беруть блоки, щоб обробляти їх. Перетворювачі (reducer) збирають і узагальнюють результати. Трекер роботи контролює процес трекеру завдань, а також обробляє і координує роботу, представлену клієнтами. Трекер роботи розмовляє з вузлом імен, щоб визначити розташування оброблюваних даних. Трекер роботи також несе відповідальність за представлення роботи вузлам трекера завдання, які

були обрані, щоб зробити роботу. Як засіб резервування і забезпечення високої доступності системи, вузли трекеру завдання повинні пінгувати трекер роботи через певні інтервали часу. Якщо ці сигнали підтвердження не отримано після деякого періоду часу, трекер роботи вирішує повторно відправити роботу в інше місце і заносить в чорний список цей трекер завдань, або просто запам'ятовує експлуатаційні характеристики даного конкретного вузла на майбутнє, тому що відсутність сигнулу підтвердення може бути тільки тимчасовим станом.

Вузол імен зберігає карту зі знаходженням файлів і до яких машин розподілені блоки. Він має деякий рівень розуміння топології сам по собі, тим, що він розуміє відносне положення хостів за допомогою вручну налаштованих номерів стійок, пов'язаних з хостом адміністратором. Вузол імен потім працює з використанням алгоритмів, які намагаються оптимально розподілити дані, щоб скоротити трансфери між стійками, але, як і раніше, підтримуючи поділ копій так, щоб надмірність/реплікація могла працювати (Hadoop – це файлова система рівня L3, тому вона працює в маршрутизованій мережі, що дозволяє архітекторові обмежити розмір мережевих доменів рівня L2 і потенційно використовувати Hadoop на більшій площі; використання глобальної мережі можливе, проте існують практичні межі).

Проблеми з цим обумовлюється ручним характером конфігурації (зокрема, у великому і постійно зростаючому гнучкому центрі обробки даних) і трохи мене динамічним характером алгоритму (є припущення про відносну продуктивність в стійці, що не завжди може бути правдою, особливо при присутності іншого трафіку).

Використовуючи SDN (OpenFlow) контролер та модифіковану версію Hadoop (модифіковані трекери роботи і завдань), альтернативна більш динамічна версія Hadoop може бути реалізована на традиційній комутованій/спільній топології на основі Ethernet (рис. 3.11). Наприклад, недавнє дослідження [15], яке оптимізує фази відтворення у випадковому порядку, де розподільники відправляли результати перетворювачам за

допомогою QoS керованої OpenFlow, так що перемішений трафік може споживати більше пропускної здатності каналу, показало обнадійливі результати.

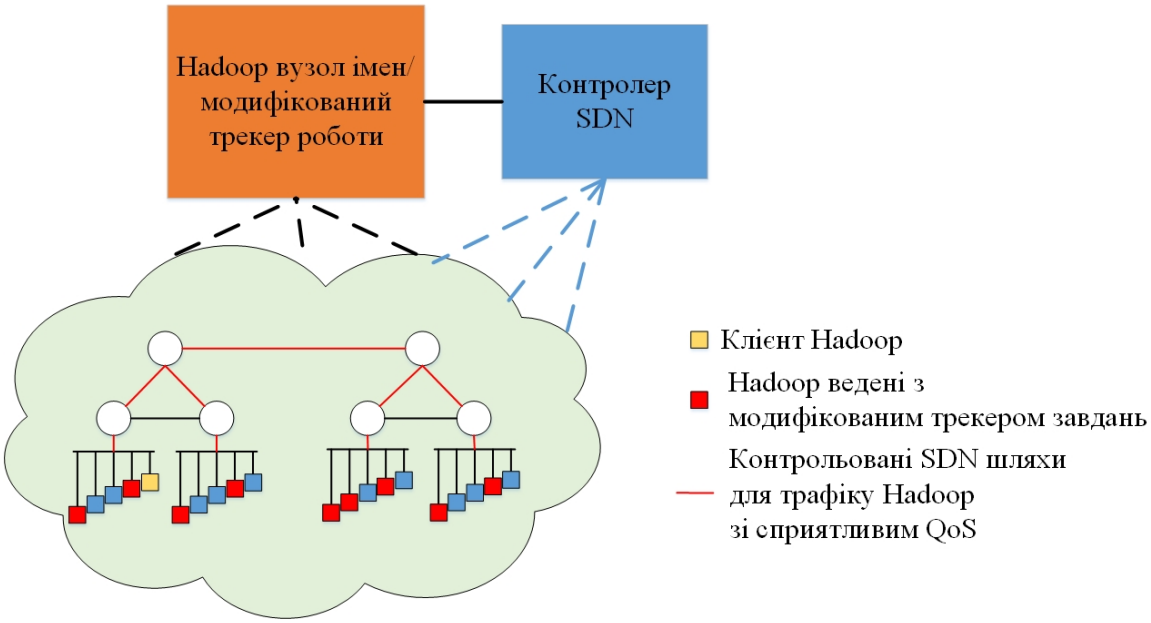

Рис. 3.11 Модифікований Hadoop з керуванням SDN, котре дає Hadoop трафіку сприятливі умови QoS

Це рішення може бути ще більш привабливим, коли інфраструктура комутації базується на програмованій оптиці. У цьому випадку, оптимізовані конфігурації топології можуть бути реалізовані як правила OpenFlow в електрооптичний мережі.

3.6 Взаємодія транспортних та накладних мереж в SDN

Як описано раніше традиційні центри обробки даних складаються зі сховища, обчислювальної частини (наприклад, сервери) і деякої мережевої інфраструктури, що пов'язує ці два елементи воєдино. Відомо, що сервери та додатки (програми) вже давно були віртуалізовані, що ж тепер також потрібно віртуалізувати і мережі. У традиційних мережах, VLAN були першими шагами у віртуалізації мережі, вони віртуалізовували шляхи між серверами,

віртуальними чи bare metal. Як описувалося у розділах 2.3.7 та 2.3.8, існує ряд протоколів, які можуть бути використані, щоб не тільки сформовувати мережеву структуру центрів обробки даних, але й які можна використовувати для віртуалізації мережевих фабрик. Зокрема, далі буде представлене поняття накладених мереж як концепції, яка дозволяє віртуалізувати транспортну (фізичну) мережу.

3.6.1 Транспортна мережа

Транспортна мережа може складатися з низки технологій. Вцілому, їх можна розділити на одно-, дво-, три-рівневі, але у будь-якому випадку – ці всі рішення роблять одне: переносять трафік між серверами в центрах обробки даних. На малюнку 3.12 показано, як накладена мережа створює логічну мережу над фізичною транспортною мережі. Зазвичай накладена мережа створюється за допомогою віртуальних мережевих елементів, таких як віртуальні маршрутизатори, комутатори або логічні тунелі. Транспортна мережа - це типова мережа, така як Ethernet, MPLS або IP-мережа.

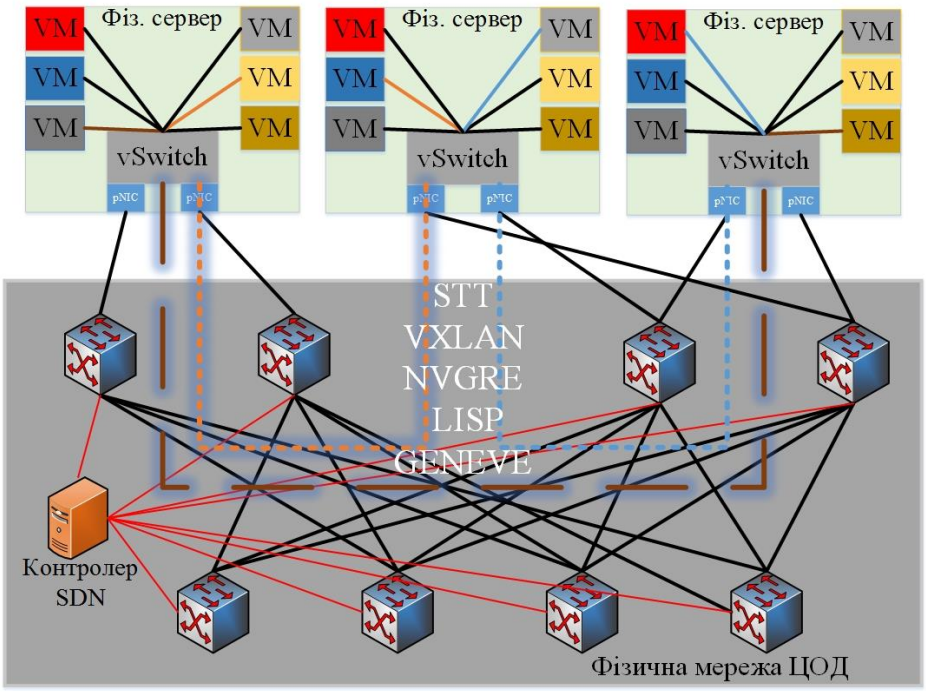

Рис. 3.12 Зіставлення накладеної та транспортної мережі в ЦОД

Чесно кажучи, всі доступні рішення будуть працювати над забезпеченням роботи транспортної мережі, яка, в свою чергу, забезпечує роботу накладеної мережі. Питання полягає в тому, наскільки оптимальним є кожне рішення з точки зору роботи і наскільки воно є добрим для хостінгу конкретної технології накладання. Наприклад, один з варіантів технологій може бути кращим для вивчення MAC-адрес на рівні L2, але і одночасно бути невдалим для руху ВМ або cloud burst (під поняттям розуміють оренду зовнішніх ресурсів для свого ЦОД у хмарних провайдерів, наприклад, для короткочасних великих розрахунків). Інший варіант може гарно адаптовуватися до змін у мережі, але в той же час потребувати більше часу на напів-автоматизоване (або навіть ручне) втручання інженера.

3.6.2 Накладені мережі

Як згадувалося раніше, пока SDN не винайшла поняття логічних накладених мереж, це, очевидно, є одним з речей, які сьогодні рухають і мотивують розробку SDN, особливо в мережах центрів обробки даних. Раніше було представлене поняття накладених мереж. За їх основу використовуються технології мережевої інфраструктури центрами обробки даних та іншими операторами мережі вже сьогодні. За деякими винятками, такими як VxLAN і NGVRE, основні мережеві технології були змінені або доповнені для підтримки додаткової віртуалізації, контекстів користувачів або цілих віртуальних рівнів сама мережа. Як і різноманітні транспортні мережі, існують і різноманітні накладені мережі. Далі будуть описані кожні з них, починаючи з загальних понять про тунелі, шо «заповнюють» проміжок між віртуальними машинами.

3.6.3 Типи накладених мереж

Як згадувалося раніше, існують накладені мережі, які імітують різні логічні рівні мережі. Також існує один підхід, що об’єднує рівні 1 та 2.

Накладені мережі рівня 2

Можна стверджувати, що більшість «накладених» протоколів тунелювання доступні для інкапсуляції мережевого трафіку рівня 2 через мережі рівня 3, і OpenFlow – це очевидний варіант у даному випадку, де звичайним підходом є побудова мережі через сегменти мережі рівня 2, як описано раніше. Мережі рівня 3, як правило, IP, проте, як бачили, можуть бути OpenFlow, GRE або навіть MPLS, що технічно є рівнем 2.5. Точний формат заголовка тунелю змінюється в залежності від типу тунельної інкапсуляції, але основна ідея приблизно однакова для всіх інкапсуляцій.

Фокусування навколо мереж 2-го рівня було історично обумовлено технологіями кластеризації серверів і синхронізації систем зберігання даних, а також необхідністю для мережевих операторів OSS мати більше свободи у прив'язці IP-адрес до віртуальних машин. Однак ці додатки вже не виключно рівня 2: спостерігається все більша міграція до рівня 3.

Накладені мережі 3-го рівня

Інший тип накладених мереж – накладення рівня 3 (тобто, IP). Ці накладання представляють накладення на основі IP замість накладень на рівні 2. Різниця між цими накладеннями і накладаннями рівня 2 полягає в тому, що замість представлення логічної топології рівня 2 між ВМ, це тип представляє мережу рівня 3.

Переваги цього підходу аналогічні існуючим шар 3 VPN. Зокрема, приватна або публічна адресація може бути змішаною розділена легко, тому такі речі, як cloud burst або додавання зовнішніх мереж, легко реалізується.

Міграція також відбувається простіше. У такому підході, якщо термінація тунелю рівня 2 відбувалася у vSwitch, то тунель рівня 3 припиняється у vRouter. Обов'язки vRouter подібні до vSwitch, за винятком того, що він діє як елемент Provider Edge (PE) у VPN рівня 3. Найбільш типовим із запропонованих сьогодні підходів фактично є зміненим MPLS 3 рівня, який інтегрується з vRouter усередині простору гіпервізора.

Висновки з розділу 3

1. Використання SDN в мереж повністю змінило «життя» центрів обробки даних – відбулася автоматизація та прискорення всіх процесів в мережі. Тепер контролер сам автоматично проводить та контролює всі процеси в мережі ЦОД. Сучасні реалізація центрів обробки даних автоматично працювати не можуть – все таки потрібна значна кількість людських ресурсів.
2. SDN можна реалізувати на мережі любої топології, що значно прискорює впровадження технології в центрі обробки даних.
3. Середовище обробки даних з початком використання SDN потрібно розглядати з такої точки зору, що перше місце тепер займає саме прикладна програма, яка і керує повність апаратурою і інфраструктурою ЦОД.
4. Архітектура SDN визначає, що мережева інфраструктура логічно управляється центральною сутністю, відповідальною за управління та застосування політик. Проте, слід чітко вказати, що логічно централізоване управління не обов'язково також має на увазі і фізичну централізацію. Існують реалізації логічно централізованої, але фізично децентралізованої площини управління контролерів.
5. Система керування мережі SDN є універсальними і може застосовуватися як і в університетському містечку, так і в центрі обробки даних.

6. Система керування мережею SDN може представляти із себе, як і повноцінне незалежне програмне забезпечення, так і додаткове програмне забезпечення, яке інтегрується в систему керування звичайною фізичною мережею.
7. Завдяки автоматизованій живій міграції віртуальних машин можно ефективно балансувати навантаження в центрах обробки даних. SDN передбачує динамічну міграцію віртуальних машин, так як площина управління є централізованою, що дає глобальну видимість мережі, та незалежна від рівневого стеку IP. SDN уможливила міграцію всієї системи, що складається з віртуальних машин, мережі і системи управління з різним набором фізичних ресурсів.
8. Завдяки SDN можна оптимізувати «big data» в центрах обробки даних. Для додатків big data, контролер SDN забезпечує інтерфейс, який приймає матриці попиту трафіку з контролерів додатків в стандартному форматі. Додатки big data можуть використовувати мережеву інформацію, представлену контролером SDN, наприклад, топологію, щоб приймати більш обґрунтовані рішення з планування роботи і розміщення.
9. Централізована площина управління SDN забезпечує логічне накладення, яке використовує андерлейну мережу в якості транспортної мережі. Накладена мережа в мережі SDN центру обробки даних може бути реалізована на базі існуючих технологій, наприклад, VXLAN або NVGRE. Завдяки SDN можна отримати єдину накладену мережу для безлічі центрів обробки даних або тільки всередині одного ЦОД використовуючи тільки з’єднання на рівні L2.

РОЗДІЛ4. ПРОГРАМНА РЕАЛІЗАЦІЯ SDN

4.1 Вступ

Програмна реалізація SDN буде виконана на базі 2 віртуальних машин.

Перша ВМ запускає HP SDN VAN controller 2.4.6, який вже згадувався в підрозділі 3.3.2 і який можна завантажити з [18] після ознайомлення з ліцензуванням. В розділі 4.2 буде приведена інструкція щодо встановлення програмного забезпечення HP SDN VAN controller 2.4.6 на базі операційної системи Ubuntu 14.04 LTS.

Друга ВМ може бути запущена на будь-якому дистрибутиві Linux, на який потрібно встановити mininet і apache2 з perl модулем, щоб імітувати віртуальну мережу для контролера і мати можливість використовувати perl / apache для розробки мережі. Для більш швидкого розгортання системи можна, або завантажити вже створену ВМ від mininet.org за посиланням [19], або ВМ від SDNhub.org з програмним забезпеченням MiniNet, Java за посиланням [20]. Проте потрібно зауважити, що ВМ SDNhub використовує Open vSwitch 2.3.9 для експериментальної підтримки OpenFLow 1.4 і ця версія комутатора, навіть якщо вона можете запускати OpenFLow 1.3, має проблеми з контролером HP, тому потрібно понизити версійність ПЗ Open vSwitch, встановивши лише стабільну версію 2.3.1 з openvswitch.org за посиланням [21].

4.2 Встановлення програмного забезпечення

Для роботи з віртуальними машинами буде використовуватися гіпервізор Oracle VirtualBox з налаштованим віртуальним мережевим адаптером у режимі «NAT» з мережею 192.168.125.0/24.

4.2.1 Встановлення програмного забезпечення HP SDN VAN controller 2.4.6 на віртуальну машину

Вважаємо, що вже існує розгорнута віртуальна машина у VirtualBox на базі Ubuntu 14.04 LTS з паролем «toor» для адміністративного користувача «root».

Кроки, які потрібно зробити для встановлення ПЗ HP SDN VAN controller 2.4.6:

1. Додати користувача labuser з паролем labuser:

   sudo useradd labuser

   sudo passwd labuser

2. Налаштування мережі:

   Відкрити конфігурацію мережі:

   sudo vi /etc/network/interfaces

   Додати до цієї конфігурації наступний код:

   iface eth0 inet static

   address 192.168.125.9

   netmask 255.255.255.0

   gateway 192.168.125.1

   dns-nameservers 192.168.125.1 8.8.8.8

   auto eth0

3. Оновити операційну систему:

   sudo apt-get update && apt-get upgrade

4. Встановити додаткове програмне забезпечення до ОС:

   sudo apt-get install joe screen aptitude curl ntp openssh-server

   sudo apt-get install openjdk-7-jre-headless postgresql postgresql-client \ iptables unzip curl

5. Перевірити чи Java 7 є версією Java за замовчуванням:

   sudo update-java-alternatives –l

6. Оновити список репозиторіїв програмного забезпечення:

sudo apt-get install python-software-properties ubuntu-cloud-keyring

sudo add-apt-repository "deb http://ubuntu-cloud.archive. \ canonical.com/ubuntu precise-updates/icehouse main"

7. Встановити KeyStone сервер (сервер аутентифікації та авторизації):

sudo apt-get update && apt-get install keystone

8. Додати порт до списку зарезервованих:

vi /etc/sysctl.conf

У строчці дописати порт:

net.ipv4.ip_local_reserved_ports = 35357

9. Додати користувача sdn з паролем «skyline» до серверу KeyStone:

Код скрипту наведений у додатку А

10. Додати можливість використання UUID для сумісності із HP SDN VAN controller:

Відкрити файл keystone.conf:

vi /etc/keystone/keystone.conf

Додати рядок:

provider=keystone.token.providers.uuid.Provider

11. Встановлення HP SDN VAN controller:

sudo dpkg --unpack hp-sdn-ctl_2.4.6.0627_amd64.deb

sudo apt-get install –f

12. Перевірка інсталяції контролера:

sudo dpkg -l hp-sdn-ctl

Вивід:

Desired=Unknown/Install/Remove/Purge/Hold

| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend |/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)

||/ Name Version Architecture Description

+++-==========-==========-==========

ii hp-sdn-ctl 2.4.6.0627 amd64 HP VAN SDN Controller

13.Перевірка статусу процессу:

sudo service sdnc status

Вивід:

sdnc start/running, process 8511

Наступна команда:

sudo service sdna status

Вивід:

sdnc start/running, process 8512

14.Додаткове встановлення Open vSwitch 2.3.1:

wget http://openvswitch.org/releases/openvswitch-2.3.1.tar.gz

tar -xvzf openvswitch-2.3.1.tar.gz

cd openvswitch-2.3.1.tar.gz

sudo apt-get install build-essential fakeroot

sudo fakeroot debian/rules binary

Після останньої команди згенерується .deb-файл, котрий і потрібно встановити:

sudo dpkg -i xyz.deb

Додатково потрібно встановити 2 пакети:

cd debian

sudo dpkg -i openvswitch-switch openvswitch-common

4.2.2 Встановлення віртуальної машини SDNhub.org

Кроки, які потрібно зробити для встановлення віртуальної машини SDNhub.org [22]:

1. Імпортувати OVA-файл віртуальної машини (наприклад, SDN Hub Tutorial VM) у Virtualbox і завантажити його.
2. В налаштуваннях ВМ видати машині 2 vCPU та 2 Гб пам'яті.
3. Переконатися, що існую підключення до Інтернету з ВМ.

4. Запустити віртуальну машину.
5. Запустити термінал.
6. Налаштувати мережеве з’єднання:

   Відкрити конфігурацію мережі:

   sudo vi /etc/network/interfaces

   Додати до цієї конфігурації наступний код:

   iface eth0 inet static

   address 192.168.125.10

   netmask 255.255.255.0

   gateway 192.168.125.1

   dns-nameservers 192.168.125.1 8.8.8.8

   auto eth0

4.3 Топологія мережі віртуальних машин

Отримана схема мережі віртуальних машин представлена на рис. 4.1.

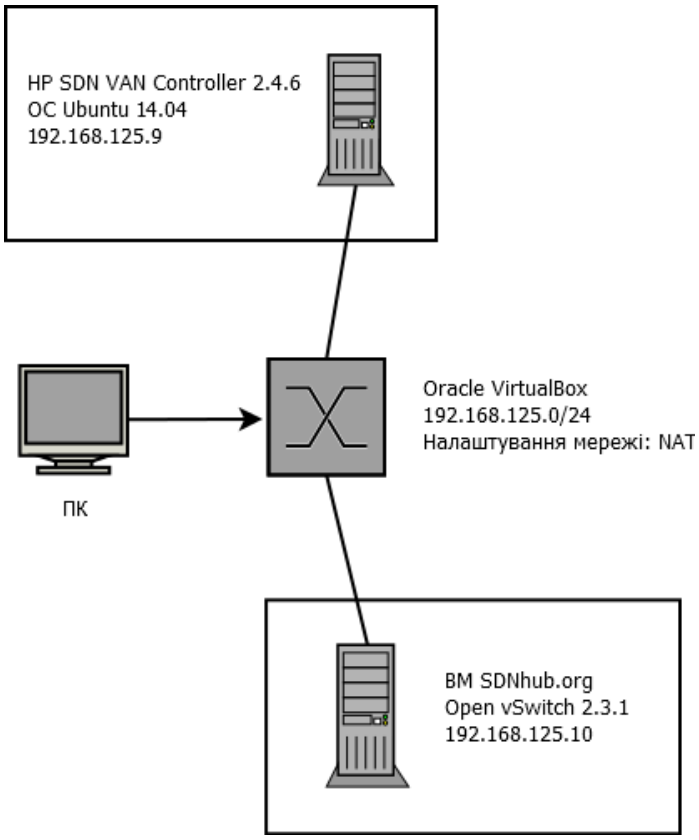

Рис. 4.1. Топологія мережі віртуальних машин

4.4 Побудова віртуальної мережі на базі MiniNet

Далі буде використовуватися топологія, що зображена на рис 4.2.

Скрипт для запуску цієї мережі в MiniNet представлений в додатку Б.

Для того, щоб запустити мережу с цього скрипта потрібно виконати наступні кроки в терміналі:

1. Створити файл network.py:

   touch network.py

2. Відкрити цей файл:

   vi network.py

3. Вставити в нього код скрипта і зберегти файл;
4. Зробити файл виконуваним:

   chmod +x network.py

5. Запустити скрипт:

   python network.py

Скрипт network.py є послідовним та легко читається, тому в нього без проблем за необхідності можна додати ще декілька хостів.

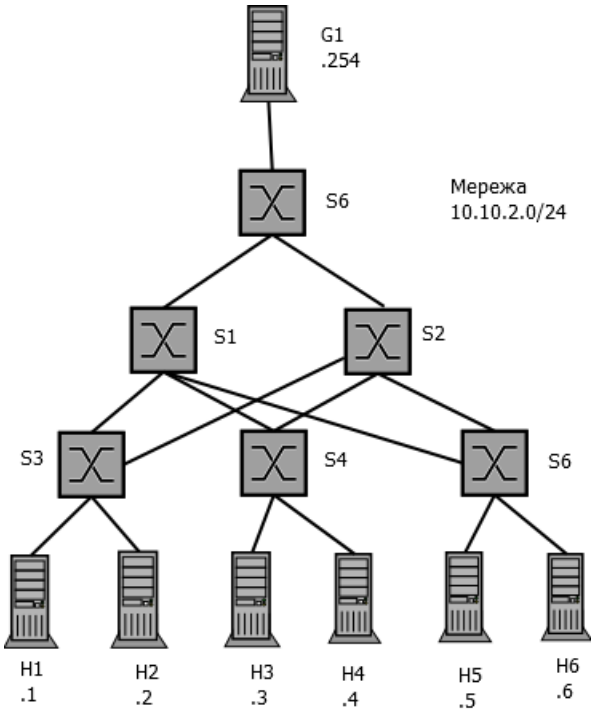

Рис. 4.2. Топологія мережі MiniNet

4.5 Перевірка HP SDN VAN controller

В цьому розділі буде перевірено, чи можна отримати доступ до контролера і побачити топологію MiniNet. Також буде відключена функція контролеру, яка дозволяє працювати в гібридному режимі, для того, щоб створити широкомовний шторм в мережі.

Щоб увійти до свого контролера SDN потрібно просто перейти на сторінку https://192.168.125.9:8443/sdn/ui/ в браузері. Потім, щоб перевірити наявність комутаторів в мережі, потрібно перейти до вкладки «OpenFlow Monitor», як показано на рис 4.3.

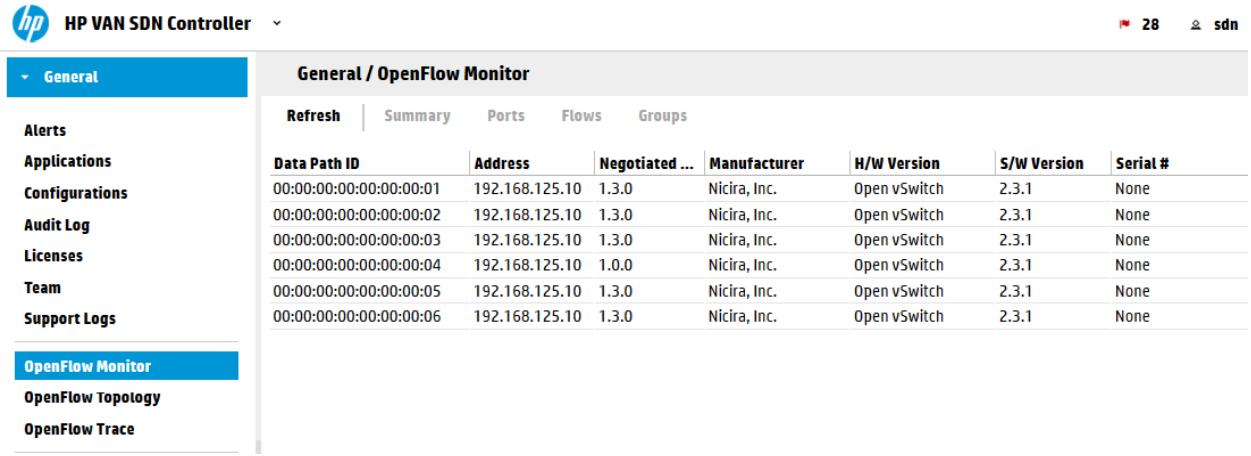

Рис. 4.3. Комутатори, що відображаються в HP VAN SDN Controller

Для перевірки топології, потрібно перейти до вкладки «OpenFlow Topology», як показано на рис 4.4.

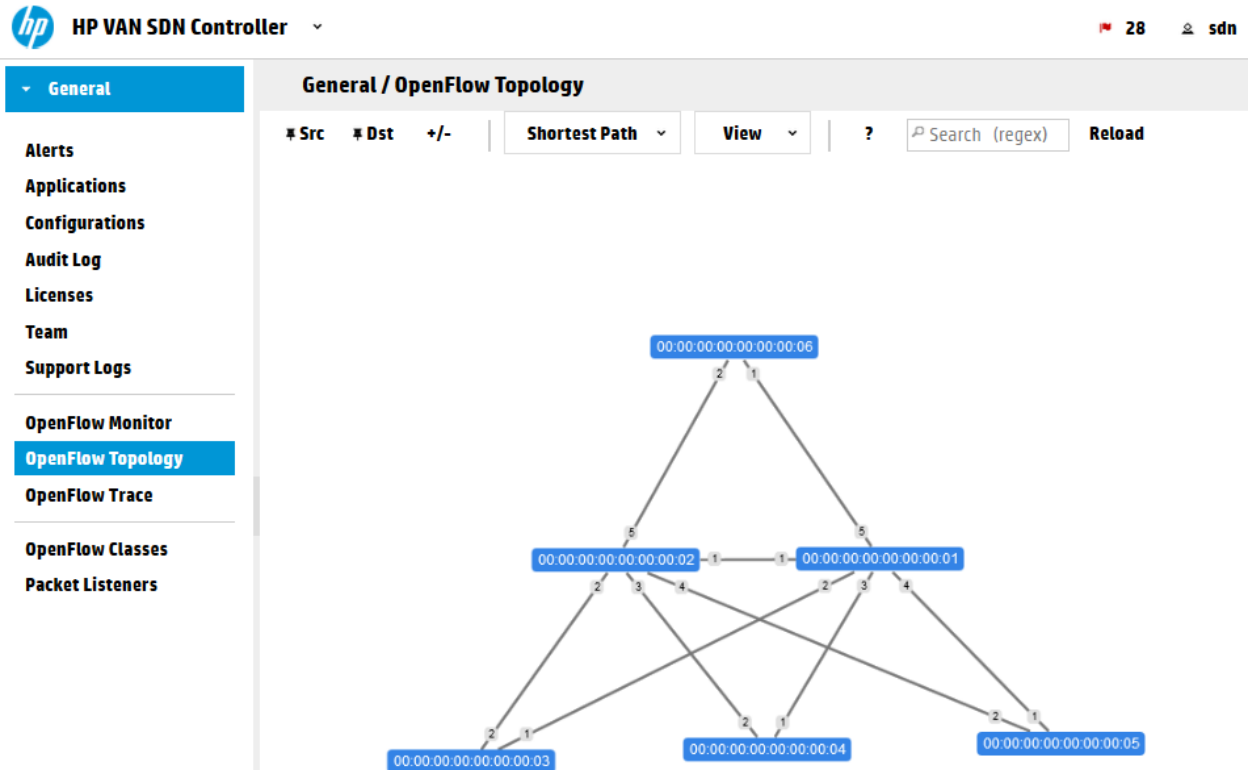

Рис. 4.4. Топологія мережі, що відображається в HP VAN SDN Controller

4.5.1 Відключення гібридного режиму в HP VAN SDN Controller

Гібридний режим - це режим, за допомогою якого контролер залишає логіку переадресації під управлінням комутаторів і керує лише додатковими функціями (наприклад, QoS або безпекою). Ця особливість дає свої переваги, наприклад, якщо ви хочете продовжувати користуватися традиційною маршрутизацією (включаючи Spanning Tree Protocol, який описувався в підпункті 2.3.7.1) і зосередитися лише на нових функціях SDN. Проте у цього є й недолік: у цьому режимі контролер посилає просту команду "Forward ANY: NORMAL" на всі комутатори, яка каже їм, що за замовчуванням весь трафік повинен просто пересилатися традиційним способом. Це суперечить тому, що

на Open vSwitch, який ми запускаємо, не працює протокол spanning-tree і наша топологія буде мати багато циклів. Це і є широкомовний шторм.

Факт полягає в тому, що HP продаючи цю технологію за замовчуванням очікує використання комутаторів HP, які також можуть працювати в гібридному режимі. Це означає, що ви можете використовувати SDN для всіх спеціальних функцій, але все одно буде працювати протокол STP, щоб уникнути циклів. Це насправді дуже вигідно, якщо використовувати разом із технологіями MSTP і IRF, які належать HP – тоді не буде відбуватися блокування якогось порт і це знижає навантаження на контролера SDN у мережі. Про те це не «чиста» переадресація SDN, яку потрібно отримати в цьому розділі.

Тоді ми відключимо гібридний режим і змусимо контролер SDN взяти на себе повну відповідальність за всі рішення щодо переадресації. Можна буде перевірити будь-яку таблицю Flow комутатори до і після виключення гібридного режиму, щоб побачити, що правило за замовчуванням, послане на комутатори, змінюється з "default: NORMAL" за умовчанням на "forward: Controller". Останнє правило означає, що якщо є новий пакет, який не відповідає існуючому правилу, то він відправляється на контролер для аналізу.

Щоб вимкнути цей гібридний режим, потрібно перейти до вкладки Configurations**,** потім ControllerManager і змінити прапорець hybrid.mode на false, як зображено на рис. 4.5. Після того, як буде це застосоване, всі вимикачі перепідключаться до контролеру.

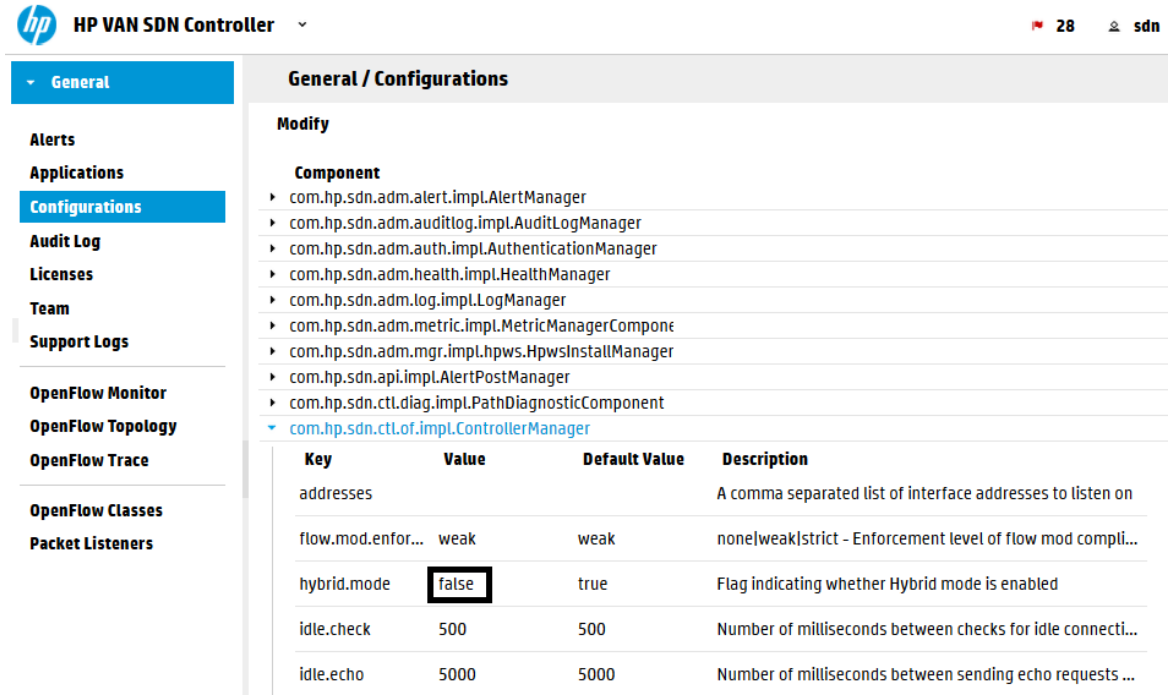

Рис. 4.5 Відключення гібридного режиму на HP VAN SDN Controller

Після відключення гібридного режиму, потрібно перевірити переадресацію, для цього MiniNet має одну команду під назвою "pingall", яка перевіряє зв'язок між усіма хостами, створеними в топології. Для того, щоб отримати вивід команди, потрібно ввести в консолі команду *pingall.*

Вивід команди з успішним результатом:

**** Ping: testing ping reachability*

g1 -> h1 h2 h3 h4 h5 h6

h1 -> g1 h2 h3 h4 h5 h6

h2 -> g1 h1 h3 h4 h5 h6

h3 -> g1 h1 h2 h4 h5 h6

h4 -> g1 h1 h2 h3 h5 h6

h5 -> g1 h1 h2 h3 h4 h6

h6 -> g1 h1 h2 h3 h4 h5

**** Results: 0% dropped (42/42 received)*

На вкладці з топологією з’явилися всі хости в мережі (див. рис. 4.6):

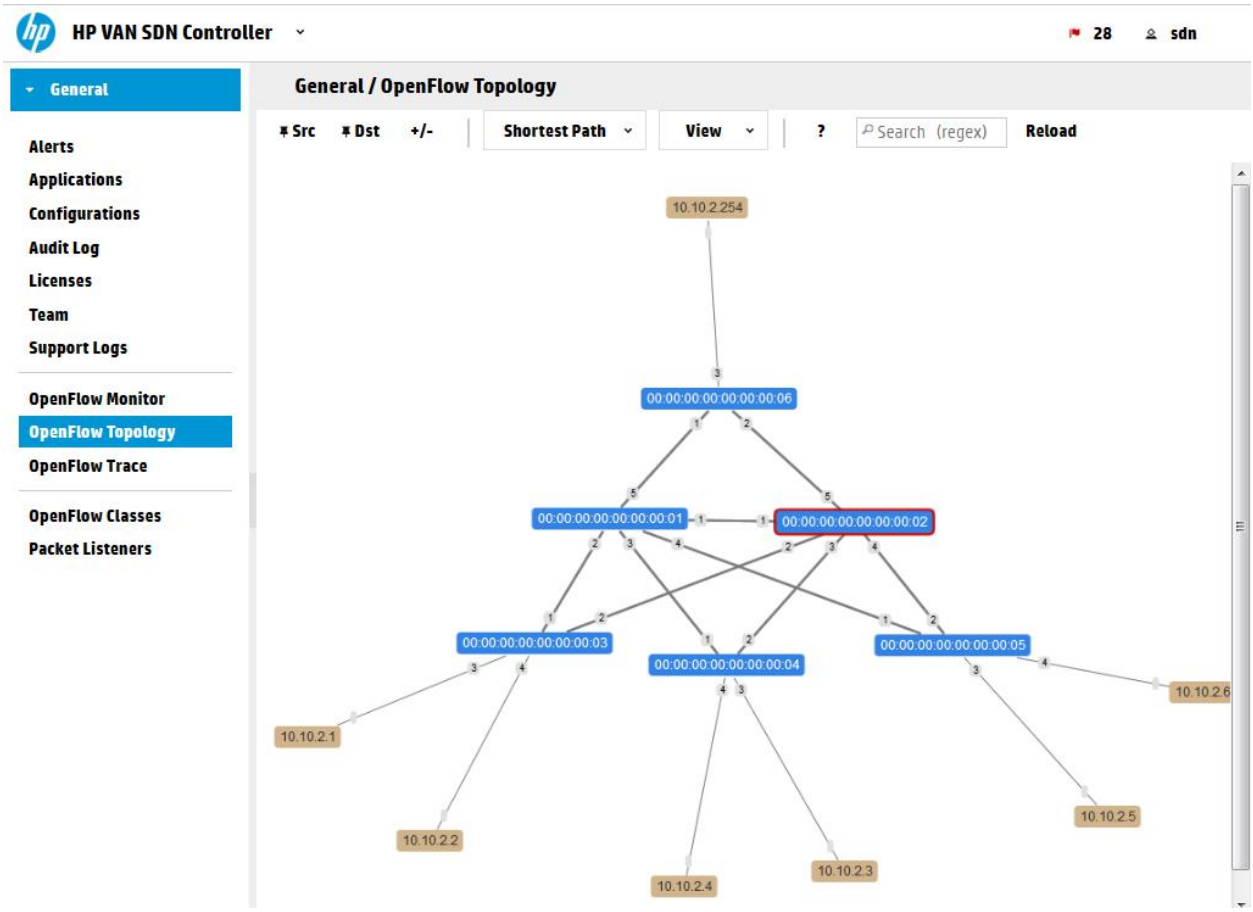

Рис. 4.6. Оновлена топологія після широкомовних пакетів

А також, якщо перейти на вкладку OpenFlow Monitor, як зображено на рис. 4.7, можна обрати комутатор і переглянути його таблиці потоку, там буде багато правил, всі вони відповідають за правила пересилки пакетів ICMP між хостами.

4.6 Відображення потоків на топології

Ще одна особливість SDN контролеру – це можливість показати шлях для певного пакета за топологією OpenFlow. Треба зачекайти кілька секунд, щоб таблиці потоків ICMP обнулилися, перш ніж продовжувати.

На вкладці OpenFlow Topology у верхній частині інтерфейсу потрібно змінити "SPF" на "Follow Flow" у викидному меню. Тепер можна побачити, як у цей момент комутатори передаватимуть трафік з H1 (10.10.2.1) до G1

(10.10.2.254). Тепер потрібно ввести у таблицю "Abstract Packet" IP-адреси джерела та призначення цих вузлів або клікнути на них та обрати їх як SRC / DST, як зображено на рис. 4.8.

Червоні лінії, що з’явилися на топології не показують жодного шляху в даний момент. Це відбувається через те, що ця функція не опитує контролер, а опитує комутатори та перевіряє їхні таблиці потоку, а їхні таблиці потоків вже пусті через тайм-аут.

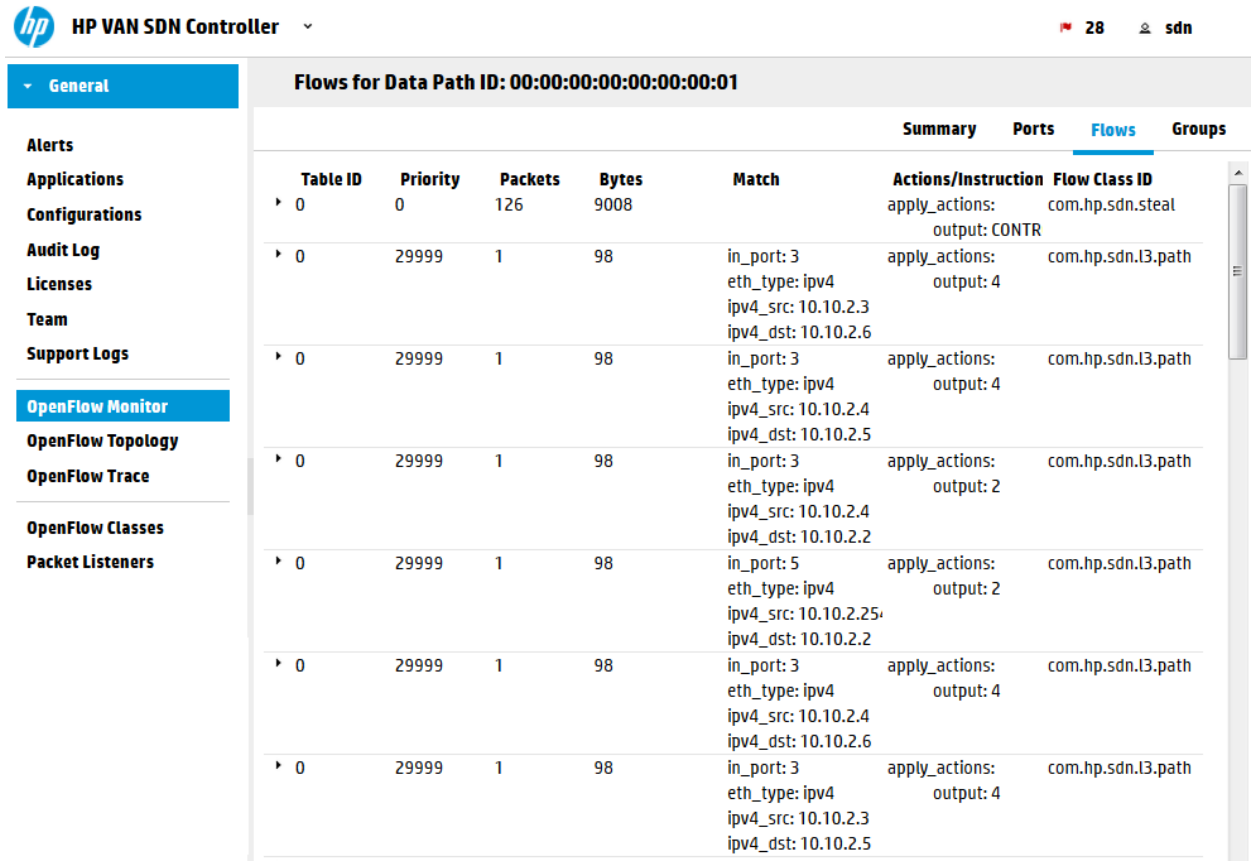

Рис. 4.7. Таблиця потоків комутатора

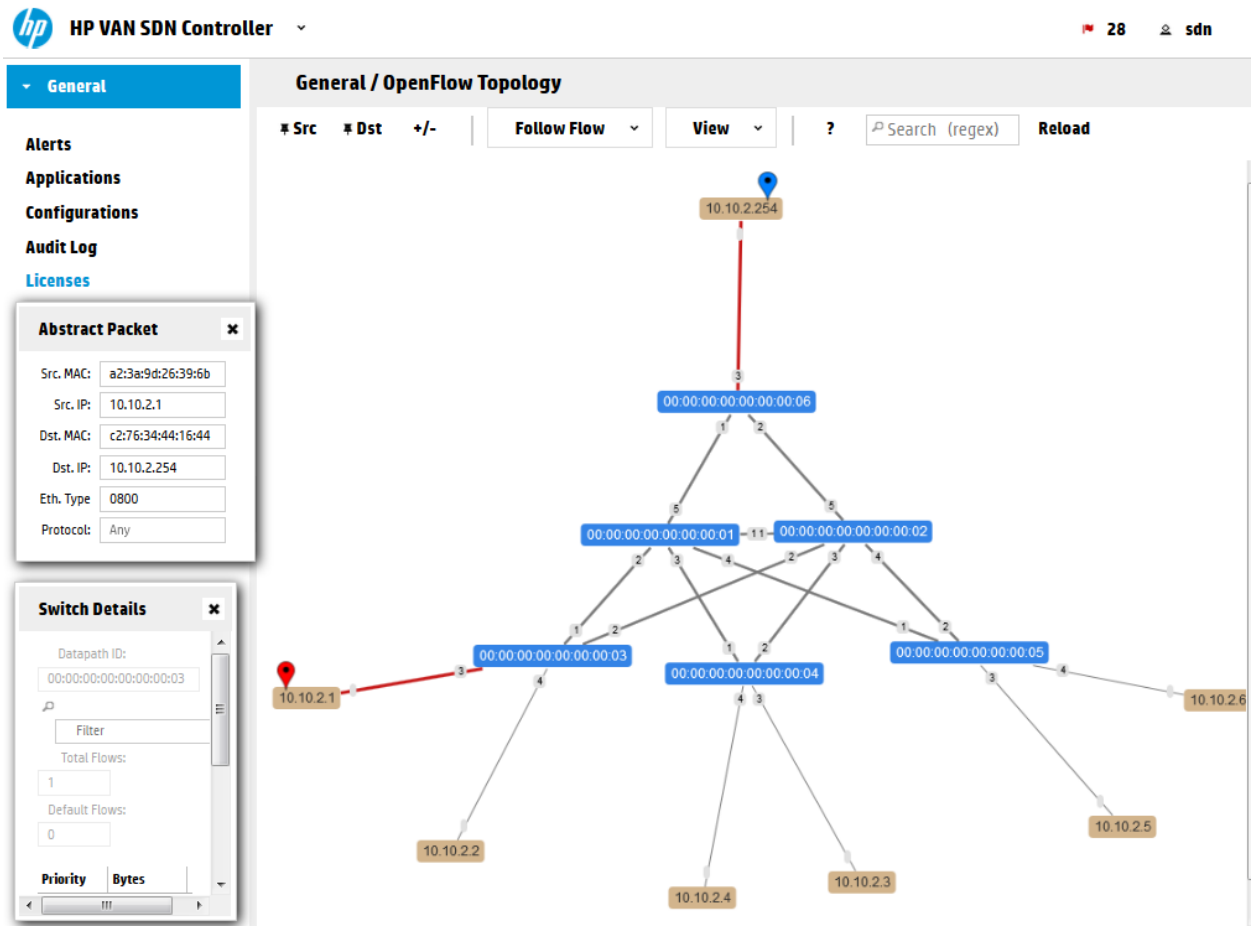

Рис. 4.8. Відсутність шляху в OpenFlow між хостами

Для того, щоб побачити шлях між хостами на топології Mininet можна використати команду для генерації HTTP-серверу на хості. Тому зробимо G1 сервером і H1 буде до G1 звертатися за допомогою HTTP-запиту.

В командному рядку вводимо:

g1 python -m SimpleHTTPServer 80 &

h1 wget 10.10.2.254

Вивід команди:

--2018-04-29 18:18:11-- http://10.10.2.254/

Connecting to 10.10.2.254:80... connected.

HTTP request sent, awaiting response... 200 OK

Length: 942 [text/html]

Saving to: index.html

100%[==========================>] 942 --.-K/s in 0.03s

2018-04-29 18:18:11 (36.0 KB/s) - index.html saved [942/942]

На топології тепер з’явився шлях трафіку між двома хостами:

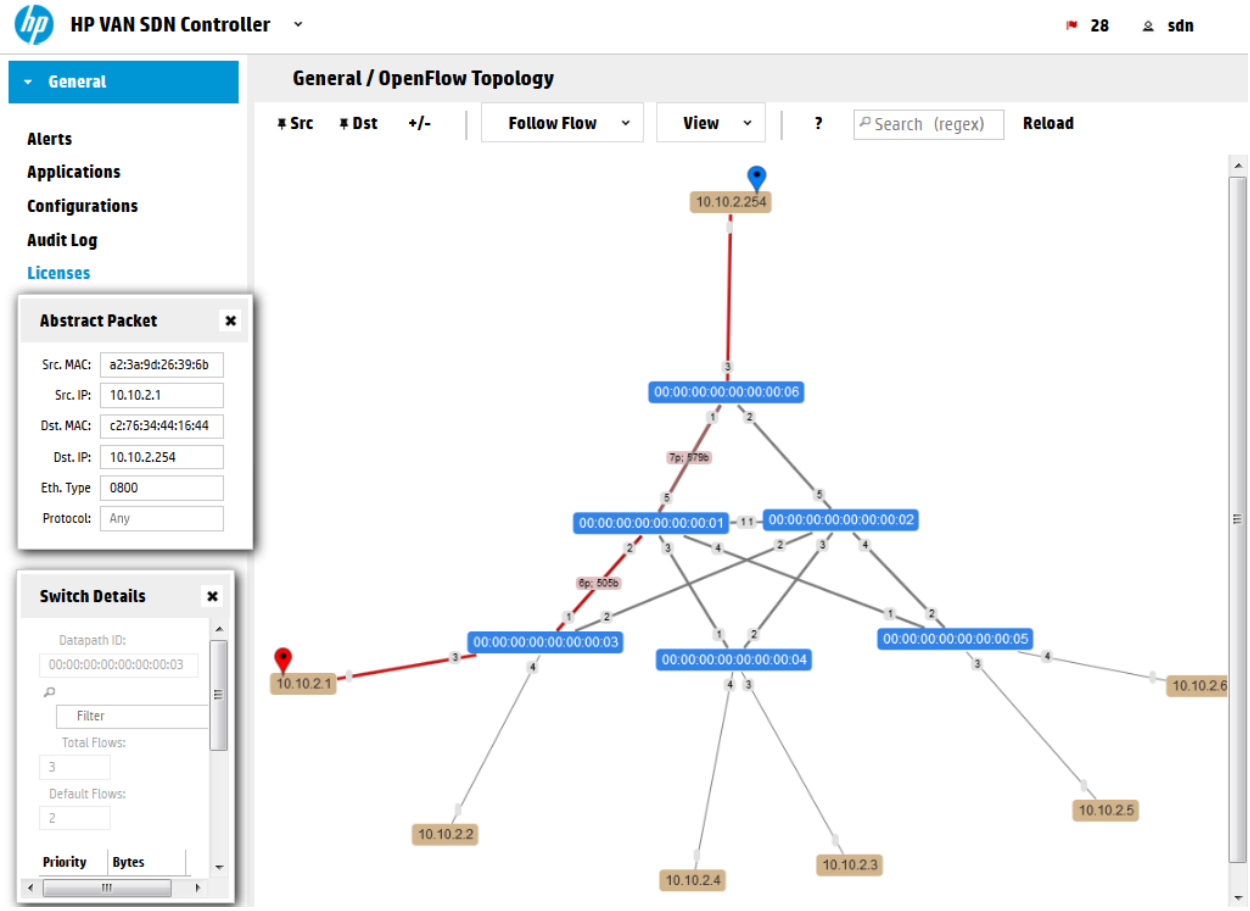

Рис. 4.9. Шлях між двома хостами

4.7 Програмна реалізація керування потоками в контролері SDN

4.7.1 Отримання підготовчих даних

В консолі linux існує утилітна команда "curl", яка буде далі використовуватися [23]. Наприклад, команда для надсилання базової структури JSON з ім'ям користувача та паролем на controller/auth в REST API виглядає наступним чином:

```
curl -sk -H 'Content-Type:application/json' \
    -d'{"login":{"user":"sdn","password":"skyline","domain":"sdn"}}' \
    https://192.168.125.9:8443/sdn/v2.0/auth \
| python -mjson.tool
```

Вивід команди:


```
{
    "record": {
        "domainId": "4ffb24500bcc4122905435c4e6882d6d",
        "domainName": "sdn",
        "expiration": 1432050873000,
        "expirationDate": "2018-04-30 17-54-33 +0200",
        "roles": [
            "sdn-user",
            "sdn-admin"
        ],
        "token": "53c50d119559431e9dd555d1c7cb916c",
        "userId": "8cc58ebf5b0a42b78384a66f577de3a2",
        "userName": "sdn"
    }
}
```


З виводу можна побачити, що отримано "токен", який потрібен для запитів cURL на інші частини REST API. Тепер потрібно командою вивести тільки значення токену та додати його до глобальної змінної в bash.

Перепишемо верхню команду:

```
curl -sk -H 'Content-Type:application/json' \
    -d'{"login":{"user":"sdn","password":"skyline","domain":"sdn"}}' \
    https://192.168.125.9:8443/sdn/v2.0/auth \
| python -mjson.tool \
| grep "token" \
| tr -d [:space:] \
```

*| tr -d "\"," *

| cut -d ":" -f 2

Пояснення до коду:

- grep - команда для фільтрації тільки рядків STDIN, де присутній тільки певна рядок;
- tr - команда truncate, яка замінює всі появи однієї літери з вихідного рядка іншим символом. В цьому випадку команда використовувалася двічі: один раз, щоб видалити пробіли, а другий раз, щоб очистити лапки та знак коми;
- cut - цей інструмент використовується для вирізання певних слів з одного виводу STDIN) з окінченням ":", також вказали, що потрібно вивести другий рядок.

Вивід команди:

2c8bbb5ac2db4162a2f95a064706dcc4

Далі можна буде оптимізувати скрипти, щоб більш зручніше передавати значення токену. Тому потрібно експортувати його в файл у каталог /tmp, щоб всі наступні скрипти могли використовувати це, не запитуючи вивід токену знову.

Команда, яка експортує значення токену в файл SDNTOKEN:

```
token=`curl -sk -H 'Content-Type:application/json' \
    -d'{"login":{"user":"sdn","password":"skyline","domain":"sdn"}}' \
    https://192.168.125.9:8443/sdn/v2.0/auth \
        | python -mjson.tool \
        | grep "token" \
        | tr -d [:space:] \
        | tr -d "\"," \
        | cut -d ":" -f 2`;

echo "This is the token recoeved $token,";
echo "will place it in /temp/SDNTOKEN file for later use.";
echo "$token" > /tmp/SDNTOKEN
```

4.7.2 Створення JSON даних щодо потоків для вставки через REST API

Дивлячись на топологію, ставимо просту мету додати шлях для комутатора S3, щоб той відправляв пакети для хосту G1 / 10.10.2.254 з порту "2" за шляхом: S2->S5->S1->S6->G1. Щоб це реалізувати потрібно:

1. Виклик REST для включення потоків;
2. Потова JSON-структура для введення в цій виклик;
3. Будування виклик cURL, щоб відправити його за допомогою HTTP POST методу.

Виклик REST API

На інтерфейс REST API HP VAN SDN Controller можна перейти за посиланням: https://192.168.125.9:8443/api/

Функція REST API, яку потрібно використовувати - це /of/datapath/(dpid)/flow, але версія POST призначена для вставки (не читання) потоків - коли ви дивитесь на цей метод, немає підказки, як побудувати необхідні "потоки" JSON-полів, як представлено на рис. 4.10.

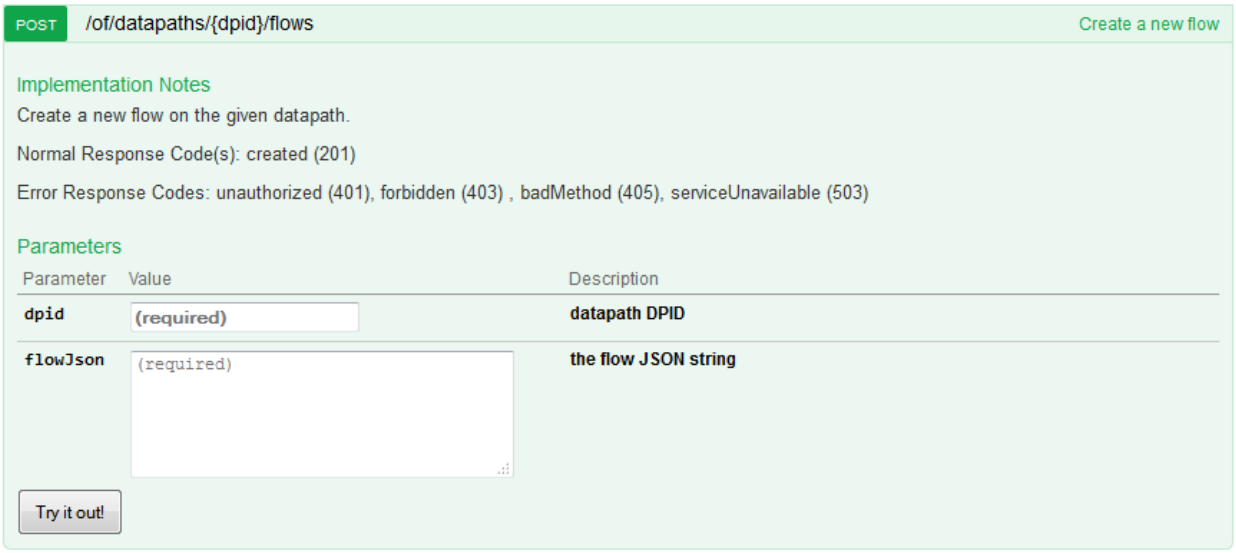

Рис. 4.10. REST API для вставки потоків

JSON структура потоку

API REST контролера має одну дуже корисну функцію, яка дозволяє всі моделі даних JSON переглянути за URL-адресою контролера: https://192.168.125.9:8443/sdn/v2.0/models.

Проте, якщо якщо відкрити це за умовчанням у браузері, ви побачите лише багато неструктурованого тексту, потрібен плагін, щоб відобразити це в читаному форматі. Рекомендовано встановити один із інструментів JSON, які представлені на рис. 4.11 в браузер Chrome.

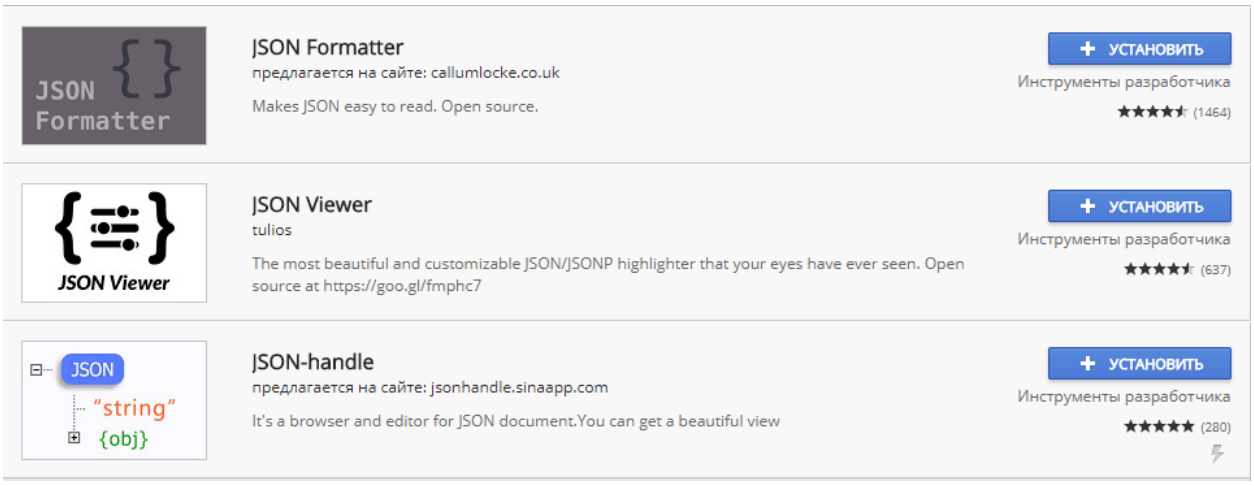

Рис. 4.11. Додаток для просмотру JSON структури

Після того, як перейдемо на веб-сторінку https://192.168.125.9:8443/sdn/v2.0/models з Chrome та ініціалізуємо повнотекстовий пошук щодо того, як визначається структура даних "flow", отримаємо структуру, що представлена в додатку В.

Після отримання структури, можемо побудувати потік JSON для комутатора S3 з dpid 00:00:00:00:00:00:00:03, щоб направити трафік з H1 з IP 10.10.2.1 до G1 з IP 10.10.2.254 з порту "2".

Крім того потрібно задати інший пріоритет за умовчанням, який використовується контролером, тому пріоритет в коді буде дорівнювати 30000, також встановлюється значення часу idle_timeout та hard_timeout для цих правил, щоб це правило автоматично зникало через 5 хвилин.

Частина JSON коду потоку:

```
{
    "flow": {
        "cookie": "0x2031987",
        "table_id": 0,
        "priority": 30000,
        "idle_timeout": 300,
        "hard_timeout": 300,
        "match": [
            {"ipv4_src": "10.10.2.1"},
            {"ipv4_dst": "10.10.2.254"},
            {"eth_type": "ipv4"}
        ],
        "instructions": [{"apply_actions": [{"output": 2}]}]
    }
}
```

Пояснення до коду:

- cookie - це шісткове значення, яке потрібно обрати самому; це не впливає на пересилання; за цим значенням в таблицях потоків можна розрізняти потоки, які додані в систему і що йдуть з інших програм;
- table_id - можемо мати кілька таблиць, але оскільки цього зараз не вимагається, можна використовувати таблицю за замовчуванням з ідентифікатором 0;
- priority - як зазначено вище, щоб перевизначити потоки SPF за умовчанням з пріоритетом 29999, обрано пріоритет 30000;
- idle_timeout - це таймер, якщо він досягне 300 секунд, буде видалений потік, однак його буде оновлено з кожним пакетом, що відповідає цьому правилу, тому, якщо сеанс активний, його не буде видалено
- hard_timeout – це строгий тайм-аут на 300 секунд; правило буде видалено, ігноруючи, чи активний трафік, чи ні; будемо використовувати це зараз в коді, щоб уникнути необхідності ручної очистки потоків;
- match - цей параметр приймає масив як вхідний, який визначається дужками []; всередині них ми визначаються правила для відповідності цього потоку;
- instructions - інший масив, визначений дужками [], в даному випадку буде тільки одна дія, яка рекурсивно оброблює масив, тому що одне правило може зробити декілька дій з пакетом і через "output": 2 ми кажемо, що хочемо, щоб пакет був переадресований з порту 2.

Виклик cURL для створення нового потоку

Маємо JSON запит, який потрібно інтегрувати в команду cURL.

Частина коду скрипту:

```
curl -ski -X POST \
    -H "X-Auth-Token:`cat /tmp/SDNTOKEN`" \
```

```
-H "Content-Type:application/json" \
-d '{
   "flow": {
        "cookie": "0x2031987",
        "table_id": 0,
        "priority": 30000,
        "idle_timeout": 300,
        "hard_timeout": 300,
        "match": [
             {"ipv4_src": "10.10.2.1"},
             {"ipv4_dst": "10.10.2.254"},
             {"eth_type": "ipv4"}
        ],
        "instructions": [{"apply_actions": [{"output": 2}]}]
        }
    }' \

https://192.168.125.9:8443/sdn/v2.0/of/datapaths/00:00:00:00:00:00:00:03/flows
```

Весь код скрипту представлений у додатку Г.

Скрипт назвали network.sh, тому визиватися він буде командою:

./script.sh

Вивід команди:

HTTP/1.1 201 Created

Server: Apache-Coyote/1.1

Location: https://192.168.125.9:8443/sdn/v2.0/of/datapaths/00:00:00:00:00:00:00:03/flows

Cache-Control: no-cache, no-store, no-transform, must-revalidate

Expires: Tue, 30 April 2018 18:05:36 GMT

*Access-Control-Allow-Origin: **

Access-Control-Allow-Methods: GET, POST, PUT, HEAD, PATCH

Access-Control-Allow-Headers: Content-Type, Accept, X-Auth-Token

Content-Type: application/json

Content-Length: 0

Date: Tue, 30 April 2015 18:05:36 GMT

Перевірка пересилання трафіку через вкладку «OpenFlow Topology» контролеру після запуску скрипту представлено на рис 4.12.

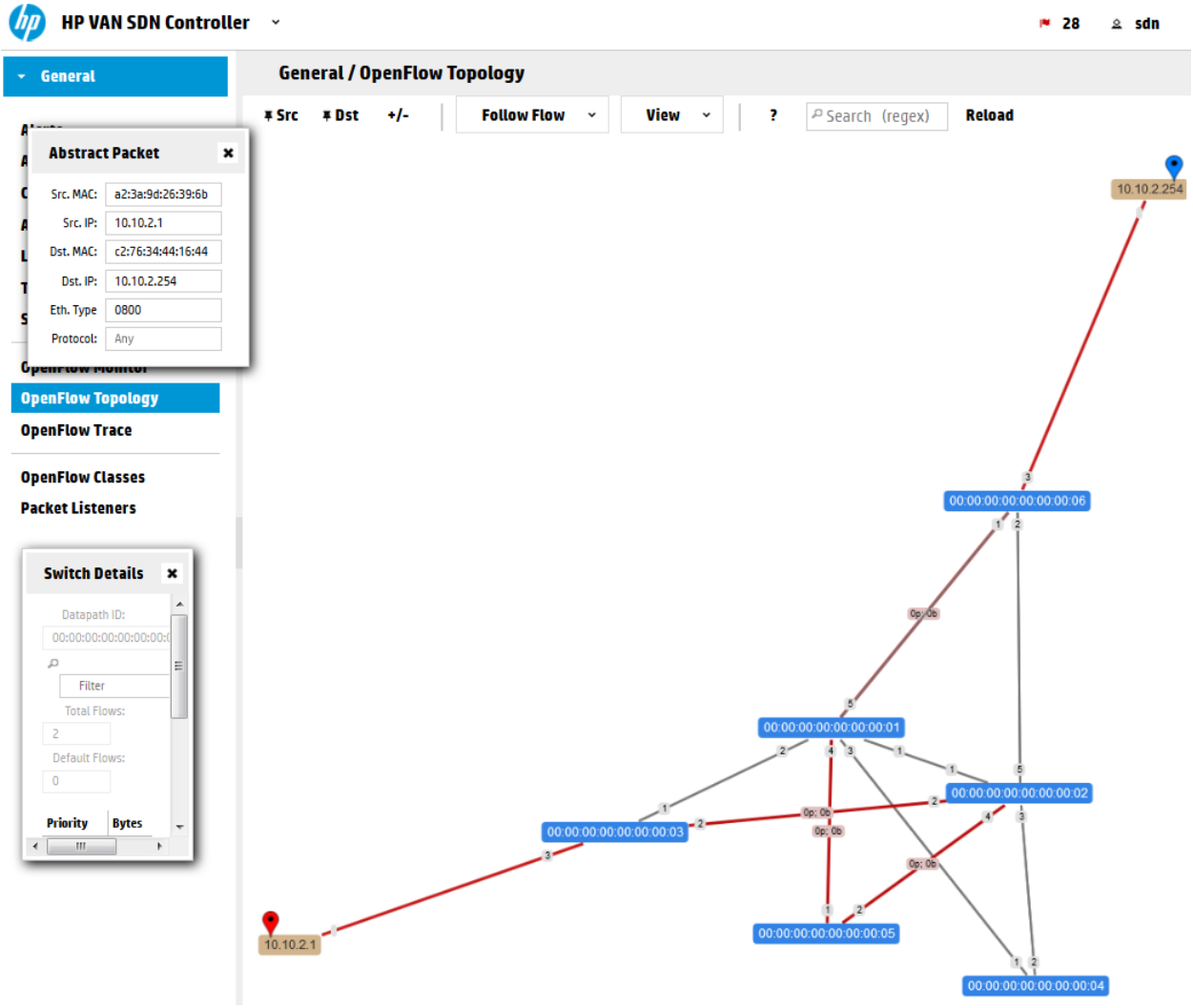

Рис. 4.12 Пересилання трафіку після застосування скрипту

Для перевірки працездатності мережі використаємо команду ping (також слід пам'ятати, що оскільки правила існують заздалегідь, перший ping не має звичайної збільшеної затримки, доки контролер не прийме рішення про маршрутизацію).

Команда для запуску ping з хосту H1 на хост G1:

h1 ping g1

Вивід команди:

PING 10.10.2.1 (10.10.2.1) 56(84) bytes of data.

64 bytes from 10.10.2.1: icmp_seq=1 ttl=64 time=0.026 ms

64 bytes from 10.10.2.1: icmp_seq=2 ttl=64 time=0.032 ms

64 bytes from 10.10.2.1: icmp_seq=3 ttl=64 time=0.041 ms

^C

--- 10.10.2.1 ping statistics ---

3 packets transmitted, 3 received, 0% packet loss, time 1999ms

rtt min/avg/max/mdev = 0.026/0.033/0.041/0.006 ms

Висновки з розділу 4

1. На базі двох віртуальних машин з контролером HP VAN SDN та програмним забезпеченням MiniNet була запрограмована мережа, що складається з 13 хостів;
2. Переведено контролер SDN з гибридного режиму для того, щоб реалізувати переадресацію трафіку на базі SDN, а не STP;
3. Запропоновано скрипт, що генерує шлях між двома хостами, тобто маршрутизує (переадресовує) пакети на визначені кодом комутатори.
4. Перевірено цей скрипт через інтерфейс контролеру та командою ping.

ЗАГАЛЬНІ ВИСНОВКИ

1. Було розглянуто використання протоколу OpenFlow в контексті архітектури SDN. Було визначено, що архітектура складається з трьох основних рівнів: інфраструктури, управління і додатків. Рівень додатків керує рівнем інфраструктури через рівень управління – додатки SDN повідомляють про свої мережеві вимоги рівень управління, котрий їх переводить і потім здійснює низькорівневий контроль на елементами мережі (рівень інфраструктури).
2. Досліджено архітектура побудови центрів обробки даних. Основними концептуальними вимогами повинні бути: конвергенція всієї мережі і інфрастуктури ЦОД, підтримка швидкої переконфігурації мережі, гнучкість мережі.
3. Забезпечення необхідної ефективності, гнучкості та відмовостійкості центру обробки даних досягається за рахунок створення віртуалізації, яка повинна проводитися на мережі рівня L2 для підвищення продуктивності при міграції і тунелюванні.
4. Було досліджено основні протоколи рівня L2, котрі можуть використовуватися в центрах обробки даних. Потрібно зауважити, що деякі з них, наприклад, NVGRE та VXLAN, дещо подібні за принципами роботи, їх використання залежить від обраного виробника мережевого устаткування, котрий підтримує той чи інший протокол. Також були досліджені протоколи, такі як SPB, котрі виступають на заміну старому протоколу STP.
5. Проведено дослідження рівнів архітектури, на яких будується центр обробки даних: блейд-серверна 1-рівнева архітектура, 2-рівнева архітектура «spine and leaf», а також традиційна 3-рівнева архітектура. Остання вважається застарілою, проте на даний момент ще використовується в багатьох ЦОД.

6. Визначено, що архітектура SDN дозволяє управляти центральною сутністю, що відповідає за управління та застосування політик. Також перевагою архітектури SDN є логічно централізоване управління. Проте логічно централізоване управління не обов'язково також має на увазі і фізичну централізацію. Доцільно використовувати логічно централізовану, але фізично децентралізовану площину управління.
7. Завдяки SDN можливо балансувати навантаження центру обробки даних, віртуально переміщуючи віртуальні машини. SDN передбачує динамічну міграцію віртуальних машин, так як площина управління є централізованою, що дає глобальну видимість мережі, та незалежна від рівневого стеку IP.
8. Завдяки SDN можна оптимізувати big data в центрах обробки даних. Для додатків big data, контролер SDN забезпечує інтерфейс, який приймає матриці попиту трафіку з контролерів додатків в стандартному форматі. Додатки big data можуть використовувати мережеву інформацію, представлену контролером SDN, наприклад, топологію, щоб приймати більш обґрунтовані рішення з планування роботи і розміщення.
9. Реалізація площини управління використовує накладену мережу на рівні L2. Накладена мережа в мережі SDN центру обробки даних може бути реалізована на базі існуючих технологій, наприклад, VXLAN або NVGRE. Завдяки SDN можна отримати єдину накладену мережу для безлічі центрів обробки даних або тільки всередині одного ЦОД використовуючи тільки з’єднання на рівні L2.
10. Представлена програмна реалізація переадресації (маршрутизації) пакетів через комутатори, керована HP VAN SDN controller версії 2.4.6.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Malcolm Betts. SDN architecture / Malcolm Betts, Nigel Davis, Rob Dolin [et al.] [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf — Дата доступу: 5.03.2018. — Open Networking Foundation.
2. Anders Nygren. OpenFlow Switch Specification / Anders Nygren, Ben Pfaff, Bob Lantz [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf — Дата доступу: 5.03.2018. — Open Networking Foundation.
3. Коломеец А. Е. Программно-конфигурируемые сети на базе протокола OpenFlow / А. Е. Коломеец, Л. В. Сурков // Инженерный вестник. – 2014. – № 5. – С. 518–525.
4. Software-Defined Networking: The New Norm for Networks [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf — Дата доступу: 5.03.2018. — Open Networking Foundation.
5. The SDN Solutions Showcase [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: https://www.opennetworking.org/images /stories/downloads/sdn-resources/white-papers/ONF_SDN%20Solutions%20Showcase%20WhitePaper_2015.pdf — Дата доступу: 15.04.2018. — Open Networking Foundation
6. HP FlexFabric Reference Architecture: Data Center Trends [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: http://h20195.www2.hp.com/v2/getpdf.aspx/4AA5-6481ENW.pdf ?ver=1.0 — Дата доступу: 16.04.2018.

7. HP FlexFabric Reference Architecture - Architecture guide [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: http://h20195.www2.hp.com/V2/getpdf.aspx/4AA5-6480ENW.pdf — Дата доступу: 20.04.2018.
8. Mauricio Arregoces. Data Center Fundamentals / Mauricio Arregoces, Maurizio Portolani. — USA: Cisco Press, 2003. — P. 5-27.
9. HPE VAN SDN Controller Software [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: http://h20195.www2.hp.com/v2/getpdf.aspx/4AA4-9827ENW.pdf — Дата доступу: 21.04.2018.
10. HPE IMC Virtual Application Networks Software-defined Networking Manager Software [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: http://h20195.www2.hp.com/v2/getpdf.aspx/4AA4-9879ENW.pdf?ver=1.0 — Дата доступу: 21.04.2018.
11. Paul Goransson. Software Defined Networks: A Comprehensive Approach / Paul Goransson, Chuck Black. — USA: Morgan Kaufmann, 2014. — P. 145-166.
12. Madhusanka Liyanage. Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture / Madhusanka Liyanage, Andrei Gurtov, Mika Ylianttila. — India: Wiley, 2015. — P. 32-33, 235.
13. Dushyant Arora. Live Migration of an Entire Software-Defined Network / Dushyant Arora, Diego Perez-Botero [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: http://www.cs.prin ceton.edu/~diegop/data/518_final_project.pdf — Дата доступу: 17.05.2018
14. Eric Keller. Live Migration of an Entire Network (and its Hosts) / Eric Keller, Soudeh Ghorbani, Matt Caesar, Jennifer Rexford [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: https://www.cs.princeton.edu/~jrex/papers/lime-hotnets12.pdf — Дата доступу: 17.05.2018

15. Sandhya Narayan. Hadoop Acceleration in an OpenFlow-Based Cluster / Sandhya Narayan, Stuart Bailey, Anand Daga [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: http://dl.acm.org/citation.cfm?id=2476992 — Дата доступу: 17.05.2018
16. Thomas D. Nadeau. SDN: Software Defined Networks / Thomas D. Nadeau, Ken Grey. — USA: O`Reilly Media, 2013. — P. 250-319.
17. Guohui Wang. Programming Your Network at Run-time for Big Data Applications / Guohui Wang, T. S. Eugene Ng, Anees Shaikh [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: https://www.cs.rice.edu/~eugeneng/papers/HotSDN12.pdf — Дата доступу: 17.05.2018
18. J9863AAE Aruba VAN SDN Ctrl Base w/50-node E-LTU [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: https://h10145.www1.hpe.com/downloads/DownloadSoftware.aspx?SoftwareReleaseUId=11794&ProductNumber=J9863AAE&lang=&cc=&prodSeriesId=&SaidNumber= — Дата доступу: 25.04.2018.
19. Mininet Walkthrough [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: http://mininet.org/walkthrough/ — Дата доступу: 25.04.2018.
20. mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: https://github.com/mininet/mininet/releases/download/2.2.2/mininet-2.2.2-170321-ubuntu-14.04.4-server-amd64.zip — Дата доступу: 25.04.2018.
21. openvswitch-2.3.1.tar.gz [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: http://openvswitch.org/releases/openvswitch-2.3.1.tar.gz — Дата доступу: 25.04.2018.
22. SDN_tutorial_VM_64bit.ova [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: http://yuba.stanford.edu/~srini/tutorial/SDN_tutorial_VM_64bit.ova — Дата доступу: 25.04.2018.

23. HP VAN SDN Controller Programming Guide [Електронний ресурс]. — Електронні текстові дані. — Режим доступу: http://h20628.www2.hp.com/km-ext/kmcsdirect/emr_na-c04003169-3.pdf — Дата доступу: 23.04.2018.

ДОДАТОК А

Створення користувача в сервері KeyStone

```
#!/bin/bash
#------------------------------------------------------------------------
# Setup default keystone environment for the SDN controller
# Create a tenant "sdn"
# Create a user "sdn" with password "skyline"
# Create a role "sdn-admin"
# Create a role "sdn-user"
# Associates the user "sdn" with the create roles in the tenant "sdn"
#------------------------------------------------------------------------
ADMIN_TOKEN="ADMIN"
IP="127.0.0.1"
PORT="35357"
TENANT="sdn"
USER_NAME="sdn"
PASSWORD="skyline"
echo "Configure keystone server at $IP with admin token $ADMIN_TOKEN"
# Create the tenant
tenant_json="{\"tenant\":{\"name\":\"$TENANT\"}}"
tenant=$(curl --header "X-AUTH-Token:$ADMIN_TOKEN" --header "Content-Type:application/json" -ksS --data-binary $tenant_json "http://$IP:$PORT/v2.0/tenants")
tenant_id=$(echo $tenant | sed 's/.*"id": "\([0-9a-z]*\)".*/\1/')
# Create the user
user_json="{\"user\":{\"name\":\"$USER_NAME\",\"enabled\":true,\"password\":\"$PASSWORD\"}}"
user=$(curl --header "X-AUTH-Token:$ADMIN_TOKEN" --header "Content-Type:application/json" -ksS --data-binary $user_json "http://$IP:$PORT/v2.0/users")
user_id=$(echo $user | sed 's/.*"id": "\([0-9a-z]*\)".*/\1/')
# Create the sdn-admin role
admin_role_json="{\"role\":{\"name\":\"sdn-admin\"}}"
admin_role=$(curl --header "X-AUTH-Token:$ADMIN_TOKEN" --header "Content-Type:application/json" -ksS --data-binary $admin_role_json "http://$IP:$PORT/v2.0/OS-KSADM/roles")
admin_role_id=$(echo $admin_role | sed 's/.*"id": "\([0-9a-z]*\)".*/\1/')
# Create the sdn-user role
user_role_json="{\"role\":{\"name\":\"sdn-user\"}}"
user_role=$(curl --header "X-AUTH-Token:$ADMIN_TOKEN" --header "Content-Type:application/json" -ksS --data-binary $user_role_json "http://$IP:$PORT/v2.0/OS-KSADM/roles")
user_role_id=$(echo $user_role | sed 's/.*"id": "\([0-9a-z]*\)".*/\1/')
# Associate user with admin role
```

```
    admin_role_assoc=$(curl --header "X-AUTH-Token:$ADMIN_TOKEN" -ksS -X
PUT "http://$IP:$PORT/v2.0/tenants/$tenant_id/users/$user_id/roles/OS-
KSADM/$admin_role_id")
    # Associate user with user role
    user_role_assoc=$(curl --header "X-AUTH-Token:$ADMIN_TOKEN" -ksS -X
PUT "http://$IP:$PORT/v2.0/tenants/$tenant_id/users/$user_id/roles/OS-
KSADM/$user_role_id")
    echo "Keystone configuration Complete"
```

ДОДАТОК Б

Скрипт мережі MiniNet

```
#!/usr/bin/python

from mininet.topo import Topo
from mininet.net import Mininet
from mininet.node import Node , Controller, RemoteController, OVSSwitch
from mininet.log import setLogLevel, info
from mininet.cli import CLI
from mininet.util import irange
from mininet.link import TCLink

c0=RemoteController( 'c0', ip='192.168.125.9' )

class NetworkTopo( Topo ):
    "A simple topology of a double-star network."

    def build( self ):

        h1 = self.addHost( 'h1', ip='10.10.2.1/24', defaultRoute='via 10.10.2.254' )
        h2 = self.addHost( 'h2', ip='10.10.2.2/24', defaultRoute='via 10.10.2.254' )
        h3 = self.addHost( 'h3', ip='10.10.2.3/24', defaultRoute='via 10.10.2.254' )
        h4 = self.addHost( 'h4', ip='10.10.2.4/24', defaultRoute='via 10.10.2.254' )
        h5 = self.addHost( 'h5', ip='10.10.2.5/24', defaultRoute='via 10.10.2.254' )
        h6 = self.addHost( 'h6', ip='10.10.2.6/24', defaultRoute='via 10.10.2.254' )
        g1 = self.addHost( 'g1', ip='10.10.2.254/24')

        s1 = self.addSwitch( 's1', dpid='0000000000000001',protocols='OpenFlow13' )
        s2 = self.addSwitch( 's2', dpid='0000000000000002',protocols='OpenFlow13' )
        s3 = self.addSwitch( 's3', dpid='0000000000000003',protocols='OpenFlow13' )
        s4 = self.addSwitch( 's4', dpid='0000000000000004',protocols='OpenFlow10' )
        s5 = self.addSwitch( 's5', dpid='0000000000000005',protocols='OpenFlow13' )
        s6 = self.addSwitch( 's6', dpid='0000000000000006',protocols='OpenFlow13' )

        #core
        self.addLink ( s1, s2 )
```

```
        #distribution
        self.addLink ( s1, s3 )
        self.addLink ( s1, s4 )
        self.addLink ( s1, s5 )
        self.addLink ( s1, s6 )

        self.addLink ( s2, s3 )
        self.addLink ( s2, s4 )
        self.addLink ( s2, s5 )
        self.addLink ( s2, s6 )

        #access
        self.addLink( s3, h1 )
        self.addLink( s3, h2 )
        self.addLink( s4, h3 )
        self.addLink( s4, h4 )
        self.addLink( s5, h5 )
        self.addLink( s5, h6 )
        self.addLink( s6, g1)

def run():
    topo = NetworkTopo()
    net = Mininet( topo=topo, controller=c0 )
    net.start()

    CLI( net )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    run()
```

ДОДАТОК В

JSON структура потоку


```
flow: {
    description: "OF flow",
    type: "object",
  - properties: {
      - table_id: {
            description: "Table id (OF 1.1 and above)",
            type: "integer"
        },
      - priority: {
            description: "Flow priority",
            type: "integer"
        },
      - match: {
            description: "Flow match fields",
            $ref: "#/flow_match_fields"
        },
      + duration_sec: {…},
      + duration_nsec: {…},
      - idle_timeout: {
            description: "Idle timeout for this flow",
            type: "integer"
        },
      - hard_timeout: {
            description: "Hard timeout for this flow",
            type: "integer"
        },
      + packet_count: {…},
      + byte_count: {…},
      - cookie: {
            description: "Cookie",
            $ref: "#/hex"
        },
      + cookie_mask: {…},
      + buffer_id: {…},
      - out_port: {
            description: "Flow mod output port number",
          - type: [
                "integer",
                "string"
            ]
        },
      + out_group: {…},
      + flow_mod_cmd: {…},
      + flow_mod_flags: {…},
      - instructions: {
            description: "Flow mod instructions (OF 1.1 and above)",
            $ref: "#/flow_instructions"
        },
      - actions: {
            description: "Flow mod actions (OF 1.0 only)",
            $ref: "#/flow_actions"
        }
    },
  - required: [
        "priority",
        "match"
    ]
},
```

ДОДАТОК Г

Скрипт нового потоку

```
#!/bin/bash
### SCRIPT TO REDIRECT H1-to-G1 via artificial path
### H1->S3->S2->S5->S1-G1

#S3 redirect
curl -ski -X POST \
    -H "X-Auth-Token:`cat /tmp/SDNTOKEN`" \
    -H "Content-Type:application/json" \
    -d '{
       "flow": {
            "cookie": "0x2031987",
            "table_id": 0,
            "priority": 30000,
            "idle_timeout": 300,
            "hard_timeout": 300,
            "match": [
                 {"ipv4_src": "10.10.2.1"},
                 {"ipv4_dst": "10.10.2.254"},
                 {"eth_type": "ipv4"}
            ],
            "instructions": [{"apply_actions": [{"output": 2}]}]
            }
        }' \

https://192.168.125.9:8443/sdn/v2.0/of/datapaths/00:00:00:00:00:00:00:03/flows

#S2 redirect
curl -ski -X POST \
    -H "X-Auth-Token:`cat /tmp/SDNTOKEN`" \
    -H "Content-Type:application/json" \
    -d '{
       "flow": {
            "cookie": "0x2031987",
            "table_id": 0,
            "priority": 30000,
            "idle_timeout": 300,
            "hard_timeout": 300,
```

```
            "match": [
                {"ipv4_src": "10.10.2.1"},
                {"ipv4_dst": "10.10.2.254"},
                {"eth_type": "ipv4"}
            ],
            "instructions": [{"apply_actions": [{"output": 4}]}]
            }
        }' \

https://192.168.125.9:8443/sdn/v2.0/of/datapaths/00:00:00:00:00:00:00:02/flows

    #S5 redirect
    curl -ski -X POST \
        -H "X-Auth-Token:`cat /tmp/SDNTOKEN`" \
        -H "Content-Type:application/json" \
        -d '{
            "flow": {
                "cookie": "0x2031987",
                "table_id": 0,
                "priority": 30000,
                "idle_timeout": 300,
                "hard_timeout": 300,
                "match": [
                    {"ipv4_src": "10.10.2.1"},
                    {"ipv4_dst": "10.10.2.254"},
                    {"eth_type": "ipv4"}
                ],
                "instructions": [{"apply_actions": [{"output": 1}]}]
                }
            }' \

https://192.168.125.9:8443/sdn/v2.0/of/datapaths/00:00:00:00:00:00:00:05/flows

    #S1 redirect
    curl -ski -X POST \
        -H "X-Auth-Token:`cat /tmp/SDNTOKEN`" \
        -H "Content-Type:application/json" \
        -d '{
            "flow": {
                "cookie": "0x2031987",
```

```
            "table_id": 0,
            "priority": 30000,
            "idle_timeout": 300,
            "hard_timeout": 300,
            "match": [
                {"ipv4_src": "10.10.2.1"},
                {"ipv4_dst": "10.10.2.254"},
                {"eth_type": "ipv4"}
            ],
            "instructions": [{"apply_actions": [{"output": 5}]}]
            }
        }' \

https://192.168.125.9:8443/sdn/v2.0/of/datapaths/00:00:00:00:00:00:00:01/flows

    #S6 redirect
    curl -ski -X POST \
        -H "X-Auth-Token:`cat /tmp/SDNTOKEN`" \
        -H "Content-Type:application/json" \
        -d '{
           "flow": {
                "cookie": "0x2031987",
                "table_id": 0,
                "priority": 30000,
                "idle_timeout": 300,
                "hard_timeout": 300,
                "match": [
                    {"ipv4_src": "10.10.2.1"},
                    {"ipv4_dst": "10.10.2.254"},
                    {"eth_type": "ipv4"}
                ],
                "instructions": [{"apply_actions": [{"output": 3}]}]
                }
            }' \

https://192.168.125.9:8443/sdn/v2.0/of/datapaths/00:00:00:00:00:00:00:06/flows
```