НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут телекомунікаційних систем

Кафедра телекомунікацій

До захисту допущено:

Завідувач кафедри

_____ Сергій КРАВЧУК

«___»_____2023 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія та програмування інфокомунікацій»

спеціальності 172 «Телекомунікації та радіотехніка»

на тему: «Аналіз функціонування протоколу OpenFlow в локальних мережах»

Виконала:
студентка IV курсу, групи ТЗ-91
Гірук Діана Олександрівна

_____

Керівник:
Доцент кафедри ТК, к.т.н.
Валуйський Станіслав Вікторович

_____

Рецензент:
Старший викладач кафедри ІКТС НН ІТС, к.т.н.
Новіков Валерій Іванович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут телекомунікаційних систем
Кафедра телекомунікацій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інженерія та програмування інфокомунікацій»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій КРАВЧУК
«___»_____2023 р.

ЗАВДАННЯ
на дипломну роботу студентці
Гірук Діані Олександрівні

1. Тема роботи «Аналіз функціонування протоколу OpenFlow в локальних мережах», керівник роботи к.т.н., доц. каф. Телекомунікацій Валуйський С.В., к.т.н., затверджені наказом по університету від «22» травня 2023 р. № 1884-с.

2. Термін подання студентом роботи 14 червня 2023 р.

3. Вихідні дані до роботи : протокол OpenFlow, програмно-визначена мережа SDN.

4. Зміст роботи :

Аналіз принципів побудови і функціонування програмно-визначених мереж (SDN)

Аналіз функціонування протоколу OpenFlow

Оцінка ефективності застосування протоколу OpenFlow в локальних мережах

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) :

~~Слайд1:~~

~~Слайд2:~~

1) Тема роботи;

2) Мета, актуальність роботи;

3) Аналіз принципів побудови і функціонування програмно-визначених мереж (SDN);

4) Аналіз функціонування протоколу OpenFlow;

5) Аналіз переваг протоколу OpenFlow;

6) Оцінка ефективності застосування протоколу OpenFlow в локальних мережах;

7) Проведення експериментів;

8) Список публікацій;

9) Загальні висновки по роботі.

6. Дата видачі завдання 3 листопада 2022 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Ознайомлення з програмно-визначеною мережею SDN, його архітектурою. Розгляд управління мережею.	15.12.2022 – 10.01.2023	Виконано
2	Ознайомлення з протоколом OpenFlow, аналіз його архітектури	10.01.2023 – 16.02.2023	Виконано
3	Виконання практичної частини: експерименти сумісності та інтеграції протоколу OpenFlow у середовищі локальної мережі	16.02.2023 – 24.03.2023	Виконано
4	Проведення висновків. Аналіз та структуризація дипломної роботи.	24.03.2023 – 30.04.2023	Виконано
5	Оформлення розділів роботи	30.04.2023 – 15.05.2023	Виконано
6	Підготовка доповіді до захисту та оформлення презентації	15.05.2023 – 08.06.2023	Виконано

Студент Діана ГІРУК

Керівник Станіслав ВАЛУЙСЬКИЙ

РЕФЕРАТ

Дипломна робота містить 96 сторінки, 36 рисунки, 11 таблиць . Було використано 50 використаних джерел.

SDN, OpenFlow, VLAN, СУМІСНІСТЬ, ІНТЕГРАЦІЯ, ЛОКАЛЬНА МЕРЕЖА.

Актуальність роботи. Завдяки швидкому розвитку інформаційних технологій та появі нових технологій, таких як хмарні обчислення та великі дані, вимоги до комп'ютерних мереж стають все більш складними. Для забезпечення високої швидкості передачі даних та ефективного управління мережами виникають нові технології з ускладненою інфраструктурою, що потребують вдосконалення інструментів моніторингу і управління. Для забезпечення більш ефективного управління мережами і виконання завдань з'явилася нова технологія - програмно-конфігуровані мережі SDN (Software-Defined Networks). Вона дозволяє вирішувати більше завдань та забезпечувати більш ефективне управління мережами, використовуючи протокол OpenFlow. SDN існує вже понад десять років, але нові реалізації відомих компаній дозволяють використовувати її більш широко і ефективно.

Мета. Проаналізувати переваги протоколу OpenFlow у традиційному середовищі локальної мережі. Інтегрувати протокол OpenFlow в середовище локальної мережі. Проаналізувати і визначити сумісність з деякими популярними протоколами в локальній мережі, наприклад, віртуальними локальними мережами (VLAN). Для аналізу буде детально розглядись програмно-визначена мережа SDN, її архітектура, а також сам протокол OpenFlow.

Об’єкт дослідження - процес фукнціонування програмно-визначених мереж SDN із використанням протоколу OpenFlow.

Предмет дослідження – методи застосування протоколу OpenFlow в локальній мережі.

Мета дослідження - проаналізувати застосування протоколу OpenFlow в традиційному середовищі локальної мережі, його інтеграцію та його сумісність з деякими протоколами локальної мережі, такими як віртуальна локальна мережа (VLAN).

Поставлена мета передбачає розв'язання таких завдань:

1. Ознайомитись з програмно-визначеною мережею SDN.
2. Проаналізувати програмно-визначену мережу SDN, описати її архітектуру та управління мережею.
3. Ознайомитись з протоколом OpenFlow.
4. Здійснити аналіз архітектури протоколу OpenFlow.
5. Провести експерименти сумісності та інтеграції протоколу OpenFlow у традиційному середовищі локальної мережі.
6. Визначити сумісність протоколу OpenFlow з VLAN.
7. Провести аналіз переваг протоколу у традиційному середовищі локальної мережі.

Методи дослідження. Для проведення дослідження протоколу OpenFlow та його інтеграції в середовище локальної мережі використовуються підходи порівняльного аналізу.

Отримані результати. Дослідження показало, що хоча протокол OpenFlow має свої переваги, апаратне забезпечення створює обмеження для його повноцінного використання. Протокол OpenFlow найбільш підходить для центрів обробки даних і магістральних мереж.

Результати досліджень оприлюднені на науково-технічній конференції "Перспективи телекомунікацій" ПТ-2023:

Валуйський С.В., Гірук Д.О. Аналіз функціонування протоколу OpenFlow в локальних мережах. XVII міжнародна науково-технічна конференція "Перспективи телекомунікацій" ПТ-2023: Збірник матеріалів конференції. К.: КПІ ім. Ігоря Сікорського, 2023. стр.328-331.

ABSTRACT

The thesis contains 96 pages, 36 figures, 11 tables. There were 50 references used.

SDN, OpenFlow, VLAN, COMPATIBILITY, INTEGRATION, LOCAL NETWORK.

Relevance of the work. Due to the rapid development of information technology and the emergence of new technologies such as cloud computing and big data, the requirements for computer networks are becoming increasingly complex. To ensure high data transfer speeds and efficient network management, new technologies with complex infrastructure are emerging that require improved monitoring and management tools. To ensure more efficient network management and task performance, a new technology has emerged - Software-Defined Networks (SDN). It allows you to solve more tasks and provide more efficient network management using the OpenFlow protocol. SDN has been around for more than a decade, but new implementations by well-known companies allow it to be used more widely and efficiently.

Objective. To analyze the advantages of the OpenFlow protocol in a traditional local area network environment. Integrate the OpenFlow protocol into a local area network environment. Analyze and determine compatibility with some popular protocols in a local area network, such as virtual local area networks (VLANs). For the analysis, the software-defined network SDN, its architecture, as well as the OpenFlow protocol itself will be considered in detail.

Object of research - the process of functioning of software-defined networks SDN using the OpenFlow protocol.

Subject of research - methods of using the OpenFlow protocol in a local network.

Purpose of the study - analyze the use of the OpenFlow protocol in a traditional local area network environment, its integration and its compatibility

with some local area network protocols, such as a virtual local area network (VLAN).

This goal involves solving the following tasks:

1. To get acquainted with the software-defined network SDN.
2. Analyze the software-defined network SDN, describe its architecture and network management.
3. Get acquainted with the OpenFlow protocol.
4. Analyze the architecture of the OpenFlow protocol.
5. Conduct compatibility and integration experiments of the OpenFlow protocol in a traditional local area network environment.
6. Determine the compatibility of the OpenFlow protocol with VLANs.
7. Analyze the benefits of the protocol in a traditional local area network environment.

Research methods. To study the OpenFlow protocol and its integration into the local network environment, comparative analysis approaches are used.

Results. The study showed that although the OpenFlow protocol has its advantages, the hardware creates limitations for its full use. The OpenFlow protocol is most suitable for data centers and backbone networks.

The results of the research were presented at the scientific and technical conference "Prospects for Telecommunications" PT-2023:

Valuiskyi S.V., Hiruk D.O. Analysis of the functioning of the OpenFlow protocol in local networks. XVII International Scientific and Technical Conference "Prospects of Telecommunications" PT-2023: Collection of conference materials. K.: Igor Sikorsky Kyiv Polytechnic Institute, 2023. pp. 328-331.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API	Application Programming Interface.
BGP	Border Gateway Protocol.
CLI	Command Line Interface
HP	Hewlett-Packard
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
JSON	JavaScript Object Notation
KVM	Kernel-based Virtual Machine
LAN	Local area network
MAC	Media Access Control
MPLS	Multiprotocol Label Switching
OOBM	out-of-band management
OSPF	Open Shortest Path First
QoS	Quality of service
RF	Radio Frequency
RIP	Routing Information Protocol
SDN	software-defined networking
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
TTL	Time to live
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VLAN	Virtual Local Area Network
VM	virtual machine
WAN	Wide Area Network

ВСТУП

У сучасному світі мережеві технології відіграють вирішальну роль у забезпеченні ефективного обміну інформацією між комп'ютерами та іншими пристроями. Розширення мережевих можливостей і висока швидкість передачі даних сприяють появі нових вимог до мережевої інфраструктури. Однак традиційні мережеві архітектури, які базуються на принципі "передачі-зупинці-обробці" (store-and-forward), виявляють свою неефективність і недостатність у забезпеченні гнучкості та управління мережевими потоками даних.

У зв'язку з цим, для вирішення цих проблем було розроблено протокол OpenFlow. OpenFlow є відкритим стандартом, який дозволяє розділити управління мережею від передачі даних. Це означає, що мережеві пристрої, такі як комутатори, можуть бути контрольовані централізованим контролером, який приймає рішення щодо маршрутизації пакетів та управління мережевими політиками.

Метою цієї дипломної роботи є детальний аналіз функціонування протоколу OpenFlow в локальних мережах. Основні завдання включають:

1. Вивчення основних принципів роботи протоколу OpenFlow, його архітектури та компонентів.
2. Аналіз ролі контролера в системі OpenFlow і його взаємодії з комутаторами.
3. Дослідження процесу пересилання пакетів у мережі з використанням протоколу OpenFlow.
4. Оцінка переваг і недоліків використання протоколу OpenFlow у локальних мережах порівняно з традиційними підходами.

Для досягнення поставлених цілей будуть використані науково-дослідницькі методи, такі як літературний аналіз, експериментальне моделювання, аналіз даних та порівняльний аналіз.

В результаті виконання даної дипломної роботи очікується отримання глибокого розуміння протоколу OpenFlow, його можливостей та обмежень у контексті локальних мереж. Результати дослідження можуть бути корисними для мережевих інженерів, адміністраторів мереж і науковців, які працюють у галузі розробки та оптимізації мережевих інфраструктур.

РОЗДІЛ 1.

АНАЛІЗ ПРИНЦИПІВ ПОБУДОВИ І ФУНКЦІОНУВАННЯ ПРОГРАМНО-ВИЗНАЧЕНИХ МЕРЕЖ (SDN)

1.1. Вступ

У мережевих пристроях існує три площини: площина даних, площина керування та площина управління. Площина даних відноситься до апаратної частини, де відбувається пересилання, а площина керування відноситься до програмної частини, де відбувається вся мережева логіка та інтелект. Як правило, у мережевих пристроях рівень керування складається з мікропрограм, розроблених і підтримуваних лише постачальниками [1]. Площина керування, як правило, є частиною площини керування та використовується для моніторингу та керування мережею. У цій дипломній роботі основна увага приділяється площинам даних і керування, і їх можна побачити на Рис.1.1. SDN — це нова мережева архітектура, яка розділяє функціональні можливості даних і площини керування в мережевому пристрої та робить площину керування незалежною, централізованою та програмованою. Міграція площини керування дає змогу абстрагувати мережеву інфраструктуру та розглядати мережу як віртуальну або логічну сутність.

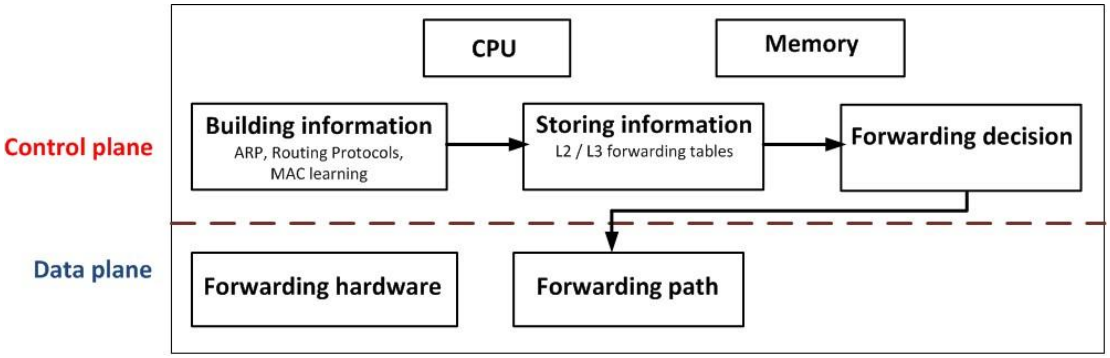

Рис.1.1.Площини даних і управління в мережевому обладнанні

SDN також можна назвати програмованою мережею та розглядати як новий підхід до гнучкості бізнесу шляхом проектування та експлуатації інноваційних мереж, які є гнучкими, автоматизованими та адаптованими до зростаючих бізнес-потреб трафіку. SDN було розроблено для створення абстракцій високого рівня, на основі яких можна побудувати апаратну або програмну інфраструктуру для підтримки нових додатків хмарних обчислень. SDN вирішує основну проблему підтримки топології мережі в зростаючій мережі та допомагає легко вносити необхідні зміни. SDN дозволяє постачальникам послуг розширити свою мережу та послуги за допомогою спільного підходу та набору інструментів, тобто зменшити витрати на обладнання, зберігаючи його контроль та продуктивність. Крім постачальників послуг і центрів обробки даних, SDN також може бути корисним для кампусних і корпоративних мереж.

1.2 Традиційні мережеві технології

Традиційні мережеві технології відносяться до локальної мережі та глобальної мережі (WAN), які складаються з різних мережевих пристроїв, включаючи комутатори, маршрутизатори та брандмауери. Традиційна локальна мережа об’єднує групу вузлів у відносно невеликій географічній зоні, як правило, в межах однієї будівлі, наприклад університету та будинку. Водночас глобальна мережа не прив’язана до жодної географічної області, а може з’єднуватися між значними областями, такими як загальнонаціональна мережа в країні, і з’єднує багато локальних мереж [2]. У цій дисертації сфера застосування традиційних мережевих технологій обмежена лише локальними мережами.

У локальній мережі дані надсилаються у формі пакетів, і для передачі та отримання пакетів можна використовувати різні технології передачі. Ethernet – це той, який використовується найбільш широко, він визначений у

стандарті IEEE 802.3, а його остання версія Gigabit Ethernet підтримує швидкість передачі даних 1 Гбіт/с і набагато вище. Пакет надходить із вузла-джерела та досягає свого вузла призначення, дотримуючись шляху, який визначається мережевими пристроями, доступними в мережі, тобто комутаторами та маршрутизаторами.

1.3 Перемикач (Switch)

Комутатор працює на рівні 2, тобто на рівні каналу даних еталонної моделі OSI, і пересилає дані від одного вузла до іншого вузла в мережі, який може існувати підключеним до того самого комутатора або до іншого комутатора в мережі. Він реєструє адреси керування доступом до середовища (MAC) усіх вузлів або пристроїв, підключених до нього, у свою базу даних, відому як таблиця пересилання. MAC – це апаратна адреса пристрою, яка є унікальною адресою, встановленою виробником пристрою. Коли пакет надходить на комутатор, він переглядає свою таблицю пересилання та пересилає пакет до порту комутатора, до якого підключений вузол призначення [3].

Оскільки комутатор працює на рівні 2, будь-який пакет, який призначається іншій підмережі IP, не може бути оброблений ним і надсилається на маршрутизатор для подальшої обробки. Процес поділу великої мережі на менші сегменти відомий як підмережа, а сформована мережа відома як підмережа. Загальною практикою є використання всіх доступних IP-адрес для мережі, щоб кожному пристрою було призначено унікальну IP-адресу.

1.4 Маршрутизатор (Router)

Маршрутизатор працює на рівні 3, тобто мережевому рівні еталонної моделі OSI, і з’єднує разом кілька мереж у LAN і WAN і виконує

обчислювальні завдання, які включають пошук і спрямування оптимального шляху до пункту призначення від джерела на основі специфікацій протоколу. або спеціальні вимоги, такі як кількість переходів. Він діє як шлюз для пересилання трафіку з однієї підмережі IP до іншої та використовує таблицю маршрутизації для прийняття рішень щодо маршрутизації.

Маршрутизатор також можна назвати комутатором пакетів зберігання та пересилання, який приймає рішення про пересилання на основі IP-адреси призначення на відміну від MAC-адреси, що використовується комутатором. Коли пакет надходить від комутатора до маршрутизатора, маршрутизатор ніколи не пересилає пакет на той самий чи інший комутатор, тобто на MAC-адресу комутатора. Замість цього маршрутизатор пересилає пакет на вузол призначення, якщо вузол призначення лежить у тій самій підмережі IP, що й маршрутизатор, або альтернативно він пересилає пакет на інший маршрутизатор, якщо вузол призначення лежить далі [3].

1.5 Обмеження традиційних мережевих технологій

Зміна моделей трафіку, зростання хмарних сервісів і зростаючий попит на пропускну здатність спонукають операторів послуг шукати інноваційні рішення, оскільки традиційні мережеві технології не в змозі задовольнити ці потреби. Тут перераховані фактори, які обмежують досягнення зростаючого попиту при збереженні прибутку, які обговорюються далі [4]:

- Складність
- Неузгоджена політика
- Неможливість масштабування
- Залежність від постачальника

Мережеві протоколи з часом розвивалися, щоб забезпечити покращену надійність, безпеку, підключення та продуктивність, і вони мають різні специфікації та рівні сумісності. Таким чином, коли плануються зміни для

мережі, усі мережеві пристрої мають бути налаштовані, щоб зміни вступили в силу, що призведе до відносно статичного характеру та додасть рівня складності мережі. Щоб подолати статичну природу, сьогодні використовується віртуалізація серверів, що робить мережі динамічними за своєю природою. Міграція віртуальної машини (VM) створює нові виклики для традиційних мереж, наприклад, схеми адресації, дизайн на основі маршрутизації тощо. Крім того, уся IP-мережа працює для підтримки трафіку голосу, даних і відео та підтримки різного QoS для різних програм для кожного з'єднання або сеанс збільшує складність мережі. Зважаючи на всі ці проблеми, традиційна мережа не здатна динамічно адаптуватися до мінливих програм і вимог користувачів.

Враховуючи надання послуг різного рівня QoS, задовільна політика QoS повинна бути реалізована в мережі. У зв'язку зі збільшенням кількості користувачів мобільного зв'язку, оператор послуг не може застосовувати узгоджену політику до мережі, оскільки це може зробити мережу вразливою до порушень безпеки та інших негативних наслідків.

Мережа має розвиватися відповідно до зростаючих вимог ринку, щоб отримати стійкі та конкурентні ринки, користувачів і прибутки. Аналіз прогнозу мережі був би корисним, але через поточну динамічну природу ринку він не надає великої допомоги для планування масштабованості заздалегідь. Складність і непослідовність політик, що застосовуються до традиційної мережі, обмежують швидшу масштабованість мережі.

Деякі з протоколів, служб і програм, необхідних у мережевому середовищі, залежать від постачальника та несумісні з обладнанням інших постачальників. Коли планується розширення мережі або запровадження нових послуг, існуюча інфраструктура складається з пристроїв від кількох постачальників. Необхідно змінити базову інфраструктуру, і проблема залежності від постачальника також може обмежити її запланований прогрес і функції.

1.6 Архітектура SDN

Логічний вигляд архітектури SDN можна побачити на Рис.1.2. Рівень інфраструктури відноситься до площини даних, де знаходиться все апаратне забезпечення. Рівень керування відноситься до площини керування, де знаходиться інтелект управління SDN, а рівень додатків включає всі інші програми, які обробляються мережею. Рівень інфраструктури та керування з'єднані через інтерфейс площини даних керування, наприклад протокол OpenFlow, тоді як прикладний рівень підключений до рівня керування через інтерфейси прикладного програмування (API).

За допомогою SDN можна отримати незалежне від постачальника керування з єдиної логічної точки, а адміністратор мережі може регулювати трафік із централізованої консолі керування, а не проходити через окремі комутатори. Це також спрощує мережеві пристрої, оскільки пристроям не потрібно розуміти та обробляти численні стандартні протоколи, а лише обробляти інструкції від контролера. В архітектурі SDN API використовуються для реалізації загальних мережевих служб, які налаштовані для задоволення бізнес-вимог, таких як маршрутизація, контроль доступу, керування смугою пропускання, управління енергією, якість обслуговування тощо [4].

Обмеження традиційних мережевих технологій ускладнюють визначення того, де в мережі слід розгорнути пристрої безпеки, наприклад брандмауери. SDN може подолати класичну проблему шляхом впровадження центрального брандмауера в мережі, і таким чином адміністратори мережі можуть направляти весь трафік через центральний брандмауер. Цей підхід полегшує централізоване керування політиками брандмауера безпеки, захоплення та аналіз трафіку в реальному часі для виявлення та запобігання вторгненням. Однак, з іншого боку, центральний брандмауер створює єдину точку збою в мережі [5].

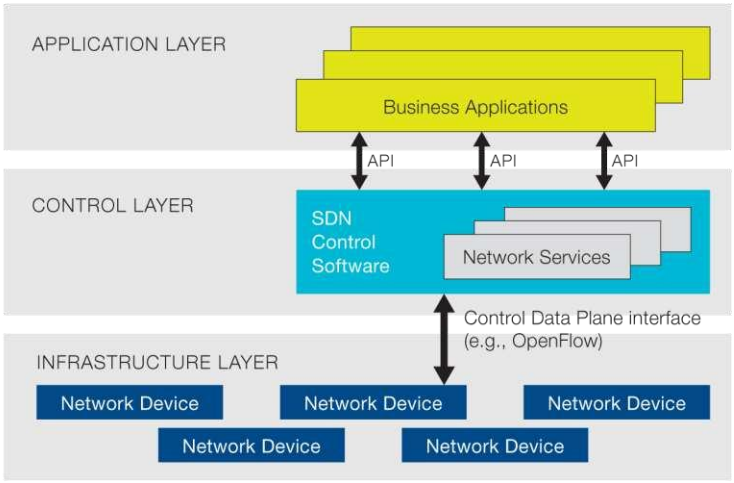

Рис.1.2. Програмно-визначена мережева архітектура [4]

Вузли на рівні керування називаються контролерами, і вони надсилають таку інформацію, як маршрутизація, комутація, пріоритет тощо, до пов'язаних з ними вузлів площини даних. Після отримання інформації від вузла керування мережеві пристрої в площині даних оновлюють свою таблицю пересилання відповідно до інформації, отриманої від площини керування. Вузли керування можна далі архітектурно класифікувати на централізований і розподілений режим [6], які можна побачити на Рис.1.3 і 1.4 відповідно.

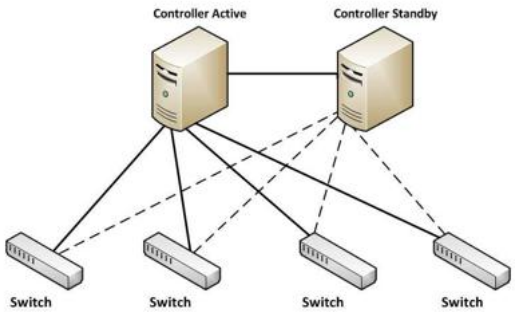

Рис.1.3. Централізована архітектура для SDN

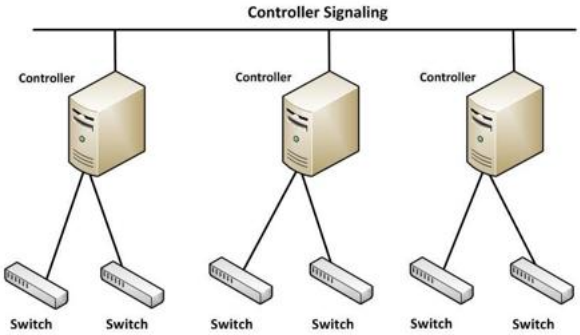

Рис.1.4. Розподілена архітектура для SDN

У централізованому режимі один центральний вузол керування надсилає інформацію про комутацію, маршрутизацію та іншу інформацію всьому мережевому обладнанню в мережі. Тим часом, у розподіленому режимі існує велика кількість вузлів керування, пов’язаних із певним мережевим обладнанням, яке надсилає їм інформацію [7]. Централізований режим має ризик виникнення єдиної точки відмови, тому при розгортанні централізованого підходу часто застосовують механізми балансування навантаження та резервування.

1.7 Моделі розгортання SDN

Для практичного розгортання SDN можна підійти до трьох різних можливих моделей: на основі комутації, оверлейної та гібридної [8]. Згідно з [9], SDN можна класифікувати на вбудовану та накладену SDN, що нагадує моделі на основі комутації та накладання в [8].

Модель на основі комутації означає заміну всієї традиційної мережі мережею SDN і наявність централізованої системи керування для кожного елемента мережі. Він вимагає універсальної підтримки з боку елементів мережі; однак його обмеження не включає вплив на існуюче мережеве обладнання рівня 2 або рівня 3.

У моделі накладання кінцеві вузли SDN є віртуальними пристроями, які є частиною середовища гіпервізора. Ця модель керує віртуальними комутаторами на межі мережі, тобто обчислювальними серверами, які за потреби встановлюють шлях через мережу. Це було б корисно у випадках, коли відповідальність за мережу SDN виконує команда віртуалізації сервера, а її обмеження включають проблеми з налагодженням, голі металеві вузли та витрати на керування інфраструктурою.

Гібридна модель SDN поєднує перші дві моделі та забезпечує плавний перехід до конструкції на основі комутаторів. Пристрої, які не підтримують накладні тунелі, такі як сервери без використання металу, підключаються через шлюзи в цій моделі.

1.8 Мережа Центрів Обробки Даних (ЦОД)

Центр обробки даних — це централізоване сховище, фізичне або віртуальне, у якому використовується багато хост-серверів і мережевих пристроїв, які обробляють запити та підключаються до іншого хосту в мережі або до загальнодоступної мережі Інтернет. Запити, що надходять до центру обробки даних, варіюються від обслуговування веб-контенту, електронної пошти, розподілених обчислень до багатьох хмарних програм. Ієрархічну топологію мережі центру обробки даних можна побачити на Рис.1.5 нижче.

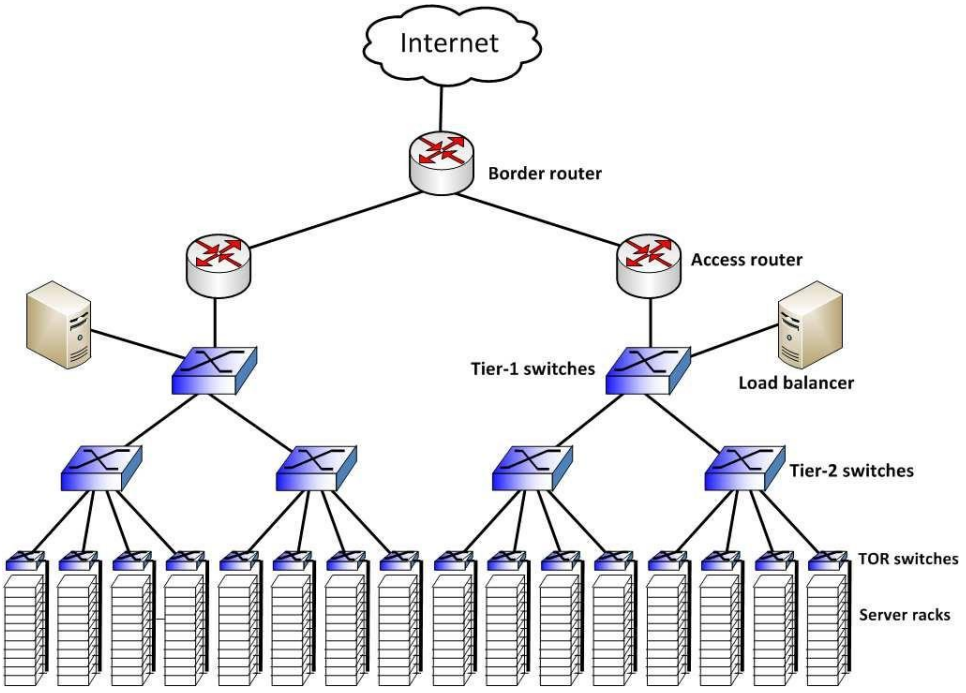

Рис.1.5. Ієрархічна топологія мережі ЦОД [3]

Хост-сервер також відомий як blade, який має центральний процесор, пам’ять і дискове сховище. Кожна серверна стійка містить приблизно від 20 до 40 блейдсерверів, а комутатор під назвою Top of Rack (TOR) розташований у верхній частині кожної серверної стійки, який з’єднується з іншими хостами та іншими комутаторами в мережі. Комутатори рівня 1 пересилають трафік до та з маршрутизатора доступу, і вони контролюють комутатори рівня 2, які керують кількома комутаторами TOR у мережі. Прикордонні маршрутизатори підключають мережу центру обробки даних до Інтернету, обробляють зовнішній трафік і з’єднують зовнішніх клієнтів і внутрішні хости один з одним.

Центр обробки даних надає багато програм одночасно, які пов’язані з загальнодоступною IP-адресою. Балансувальник навантаження діє як ретранслятор і виконує такі функції, як NAT і брандмауер, запобігаючи прямим взаємодіям. Таким чином, усі зовнішні запити спочатку спрямовуються до балансувальника навантаження, який розподіляє відповіді

на пов’язані хост-сервери та балансує навантаження між хост-серверами в мережі. Із зростаючим попитом на трафік традиційну ієрархічну архітектуру, показану на Рис.2.5, можна масштабувати, але це обмежує пропускну здатність хост-хост. Оскільки всі комутатори з’єднані між собою за допомогою каналів Ethernet 1 Гбіт/с або 10 Гбіт/с, із зростанням трафіку загальна пропускна здатність для кожного хоста зменшується. Рішення цього обмеження включає оновлення каналів зв’язку, комутаторів і маршрутизаторів для підтримки вищих ставок, але це збільшує загальну вартість мережі.

Щоб зменшити загальну вартість і підвищити продуктивність затримки та пропускну здатність, ієрархію комутаторів і маршрутизаторів можна замінити повністю підключеною топологією, як показано на Рис.1.6 нижче. У цій архітектурі кожен комутатор рівня 1 підключений до всіх комутаторів рівня 2 у мережі, тим самим зменшуючи навантаження на обробку, а також покращуючи пропускну здатність хост-хост і загальну продуктивність [3].

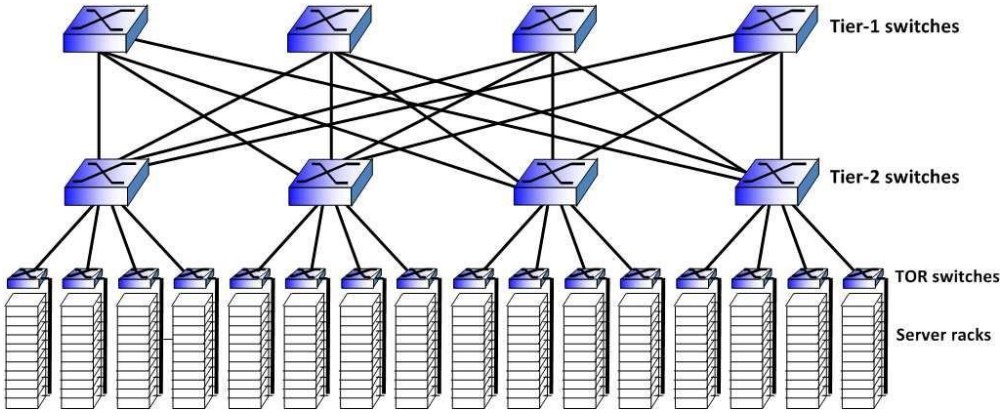

Рис.1.6. Топологія мережі центру обробки даних з високим ступенем взаємозв'язку [3].

Топологія високовзаємозв’язаної мережі центру обробки даних [3] У такій структурі мережі з високим рівнем взаємозв’язку розробка відповідних алгоритмів пересилання для комутаторів була серйозною проблемою. Тут

можна використати підхід SDN, щоб створити незалежну та програмовану переадресацію на основі потоку в мережі, спрощуючи керування мережею та знижуючи витрати на експлуатацію, оскільки мережею можна керувати з однієї точки.

SDN привернула увагу багатьох операторів центрів обробки даних, і Google розгорнула підхід SDN в одній зі своїх магістральних глобальних мереж, відомих як внутрішня (G-scale) мережа, яка передає трафік між центрами обробки даних. Розгорнута мережа SDN працює в Google і пропонує такі переваги, як високе використання ресурсів, швидше усунення збоїв і швидше оновлення. Однак його проблеми включають відмовостійкі контролери, потокове програмування та нарізку елементів мережі для розподіленого керування [10]. Подібним чином NEC також успішно розгорнула підхід SDN у центрі обробки даних та магістральній мережі на своїй власній фабриці програмного забезпечення, Nippon Express Co., Ltd. та лікарні університету Канадзава в Японії [11] [12][13].

1.9 Масштабованість у SDN

SDN має численні переваги, включаючи високу гнучкість, програмовану мережу, незалежність від постачальника, інновації, незалежну площину керування та централізовану мережу.

Централізоване керування в SDN погано масштабується в міру зростання мережі, що викликає занепокоєння щодо продуктивності мережі. Він може не впоратися зі зростаючим трафіком і зберегти той самий рівень QoS, оскільки на один контролер передається більше подій і запитів. NOX є одним із найперших контролерів OpenFlow, розроблених на C++, і тестування контролера NOX показало, що він може обробляти до 30 000 ініціацій потоку на секунду із затримкою 10 мс для кожної інсталяції потоку. Стійкий обсяг потоків може бути достатнім для корпоративних і кампусних

мереж, але він не вписується в середовище центру обробки даних. Занепокоєння щодо централізованого підходу можна усунути шляхом розгортання розподіленої SDN. Основними факторами, які призвели до цих проблем щодо масштабованості, є величина навантаження на контролер, накладні витрати потоку та самовідновлення у випадках відмови, які обговорюються далі [14].

1.10 Масштабованість контролера та зменшення навантаження

У площині керування SDN перенесення традиційних функцій керування на віддалений контролер може збільшити накладні витрати на сигналізацію, що призведе до вузьких місць у продуктивності мережі. Обсяг навантаження на централізований контролер можна зменшити різними способами, які коротко обговорюються тут. Один підхід включає впровадження контролера в паралельних ядрах, цей підхід підвищив продуктивність контролера NOX на порядок порівняно з його реалізацією в одному ядрі. Інший підхід включає класифікацію потоків і подій відповідно до тривалості та пріоритету, короткочасні потоки повинні оброблятися площиною даних, тоді як більш тривалі потоки повинні надсилатися до контролера, що зменшує навантаження на контролер.

Тим часом, у розподіленій ієрархії керування навантаження можна значно зменшити, і цей підхід застосовувався в багатьох програмах, таких як FlowVisor, Hyperflow і Kandoo. Hyperflow синхронізує стан мережі з усіма доступними контролерами, надаючи ілюзорний контроль над усією мережею, таким чином зберігаючи загальну топологію мережі [15]. Kandoo сортує програми за сферою дії: локальні та мережеві; де локальні програми розгортаються поблизу шляху даних, які обробляють там запити та повідомлення, тим самим зменшуючи навантаження на контролер. Програми всієї мережі обробляються контролером, а в розподіленій ієрархії кореневий

контролер піклується про них і оновлює інформацію про них усім іншим контролерам у SDN [16]. Flowvisor розрізає мережу, і кожен зріз обробляється контролером або групою контролерів, зменшуючи навантаження та створюючи ефективний механізм обробки рішень [17].

1.11 Додаткові витрати на потік

У попередньому дизайні SDN пропонувалося, що контролери працюватимуть у режимі реактивної обробки потоку, де пакети для кожного нового потоку, що надходить на комутатор, надсилатимуться контролеру, щоб вирішити, що робити. Після надходження до контролера контролер переглядає свою таблицю потоків, і якщо потік знайдено, він надсилає пакет назад до комутатора разом із своїм потоком, який буде встановлено в комутаторі, інакше він підтверджує комутатор, щоб відкинути цей пакет. Встановлення кожного потоку в базі даних комутатора займає значний час, заповнюється накладні витрати через модифікацію потоку та інші повідомлення, а також може обмежити масштабованість.

Величина затримки може бути приблизно визначена ресурсами контролера, його навантаженням, ресурсами комутатора та їх продуктивністю. Відстань між перемикачем і контролером також впливає на параметр затримки; якщо вони знаходяться в безпосередній близькості від одного перемикача, це приблизно дорівнює 1 мс. Апаратні комутатори здатні підтримувати кілька тисяч установок потоку за секунду з прибл. затримка в 10 мс, а причини такої поганої продуктивності включають слабкі процесори керування, погану підтримку високочастотного зв’язку та неоптимальні реалізації програмного забезпечення, які будуть вирішені в найближчі роки [14].

Враховуючи дизайн реактивного потоку, тобто проект на основі потоку, він не дуже добре масштабується, оскільки пам’ять перемикача

обмежена та фіксована, а накладні витрати потоку та затримка налаштування потоку роблять його менш ефективним. Тому добре підходить проактивний спосіб, коли всі потоки в контролері миттєво встановлюються для перемикання бази даних, і якщо пакети, що надходять, не відповідають жодному потоку, вони надсилаються до контролера, щоб вирішити, яку дію слід виконати над ними. У цій дисертації використовується контролер Floodlight, який слідує підходу проактивного проектування потоку.

1.12 Самовідновлення

У SDN контролер відіграє ключову роль, і його збій призводить до повного або часткового збою мережі в централізованому та розподіленому дизайні відповідно. Тому життєво важливо виявити його збій за допомогою механізмів виявлення та адаптуватися до його відновлення якнайшвидше. Розглянемо ситуацію збою, коли несправний перемикач не вплинув на зв’язок між перемикачем і контролером, як показано на Рис.1.7 [14].

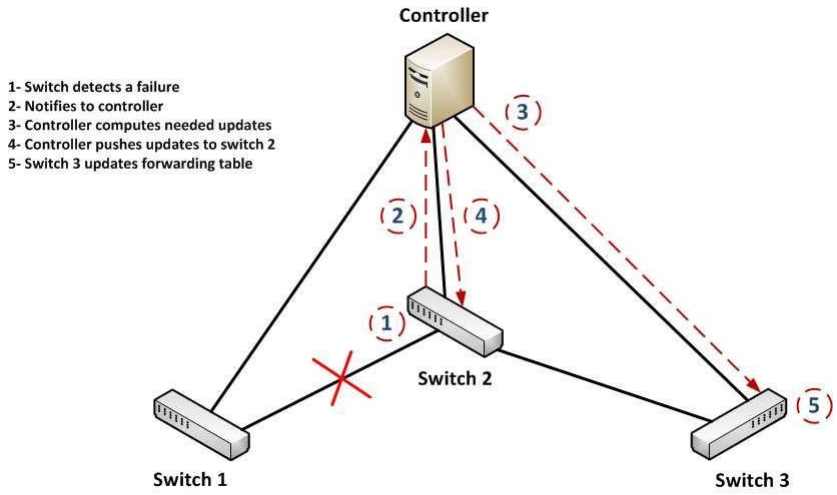

Рис.1.7. Відмова каналу зв'язку в SDN [14]

Перемикач 2 виявляє збій і повідомляє про це контролер. Контролер приймає рішення про ремонтні дії та встановлює оновлення для ураженого

елемента шляху даних і перемикає, у свою чергу, оновлюючи свою таблицю пересилання. Порівняно з традиційними мережами, де всі сповіщення про збій зв'язку переливаються; тут в SDN вони спрямовані тільки на контролер. Однак, беручи до уваги найгірший сценарій, коли контролер виходить з ладу, механізми адаптації повинні бути побудовані в розподіленому підході з використанням таких програм, як FlowVisor, для розподілу навантаження та встановлення таблиці потоків для сусідніх контролерів. FlowVisor описано більш детально в розділі 3.6.2.

Виходячи з вищезазначеного обговорення та проблем масштабованості, у середовищі центру обробки даних можна реалізувати такі рішення, як Kandoo, щоб зменшити навантаження на обробку та зробити SDN масштабованою мережею. Тим часом у виробничій мережі потоки можна агрегувати, щоб зменшити затримку, а програми нарізки мережі, такі як FlowVisor, можуть бути корисними, щоб підтримувати подібну географічну топологію також з точки зору керування.

1.13 Управління мережею SDN

Як обговорювалося раніше, мережеві політики, реалізовані через конфігурацію в традиційному апаратному забезпеченні, є низькорівневими, а мережі є статичними за своєю природою, які не здатні реагувати та адаптуватися до змін стану мережі. Щоб налаштувати мережеві пристрої легше та швидше, мережеві оператори зазвичай використовують зовнішні інструменти автоматизації або сценарії, які зазвичай призводять до деяких неправильних конфігурацій і невеликої кількості усунення несправностей. Крім того, залежність від постачальника обмежується власними інструментами та розробкою додатків, оскільки попит оператора мережі на комплексні політики високого рівня для розподілу трафіку швидко зростає.

Керована подіями структура управління під назвою Procera була реалізована в [18], і її архітектуру можна побачити на Рис.1.8 нижче.

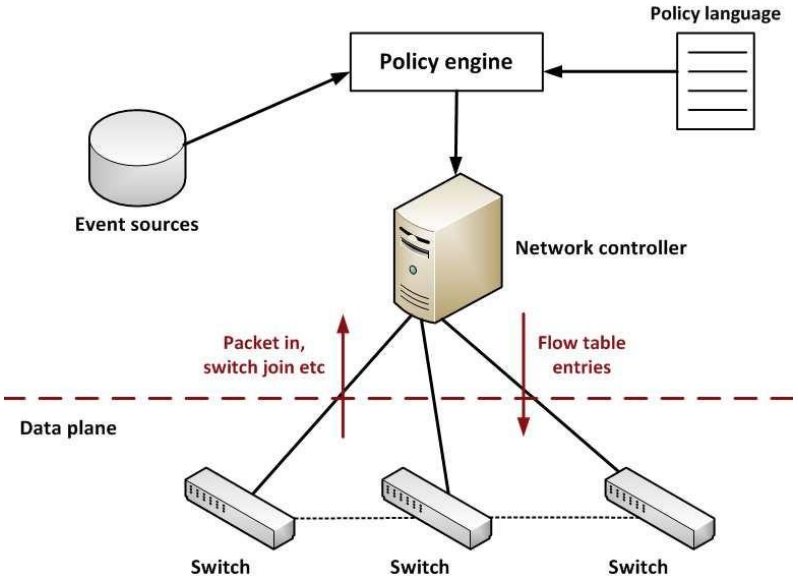

Рис.1.8. Архітектура Procera [18].

Згідно з [18], три основні проблеми керування мережею — це оновлення частих змін стану мережі, підтримка політик високого рівня та забезпечення кращого контролю для діагностики мережі та усунення несправностей. Procera було розроблено з використанням функціонального реактивного програмування (FRP), яке перетворює політики високого рівня в правила пересилання, які встановлюються на основні комутатори. Procera підтримує чотири домени керування, а саме час, використання даних, статус і потік, які є найбільш часто використовуваними параметрами для реалізації політики трафіку в мережі. В архітектурі джерела подій відносяться до мережевих компонентів, здатних надсилати динамічні події до контролера, таких як системи автентифікації, системи моніторингу пропускної здатності, параметри SNMP і системи виявлення вторгнень тощо.

У реалізованому Procera джерела подій періодично надсилали файли, що містять таку інформацію, як споживання смуги пропускання кожним

кінцевим хост-пристроєм із міткою часу. Тим часом механізм політики інтерпретує політики з мови високого рівня, тобто FRP, на контролер і обробляє події, що виникають із джерел подій. Механізм політики оновлюється одночасно, щоб застосувати нові політики або внести необхідні зміни.

Висновок

У цьому розділі було надано огляд архітектури та можливостей Software-Defined Networking (SDN). Архітектура SDN включає три рівні: рівень інфраструктури, рівень керування та рівень додатків. Рівень інфраструктури включає апаратне забезпечення мережі, рівень керування містить інтелект управління SDN, а рівень додатків охоплює програми, які обробляються мережею.

SDN також дозволяє ефективно вирішувати проблему розгортання безпеки в мережі, наприклад, за допомогою центрального брандмауера. Це спрощує централізоване керування політиками безпеки, аналізу трафіку в реальному часі та виявлення вторгнень. Однак, варто враховувати, що центральний брандмауер може стати єдиною точкою відмови в мережі.

Таким чином, SDN надає багато переваг, включаючи централізоване керування, спрощення мережевих пристроїв та ефективне вирішення проблем безпеки.

РОЗДІЛ 2

АНАЛІЗ ФУНКЦІОНУВАННЯ ПРОТОКОЛУ OPENFLOW

2.1 Вступ

OpenFlow — це відкритий стандарт, який пропонує програмне керування мережевим обладнанням. Спочатку він був розроблений для дослідників мереж, щоб перевірити свої експериментальні протоколи на реальних мережевих апаратних пристроях і мережах кампусів, які представляють повсякденне мережеве середовище, наприклад LAN і WAN. Спільнота дослідників мереж стикається з надзвичайно високими перешкодами для експериментування нових ідей або протоколів у виробничому або традиційному мережевому середовищі. Для того, щоб використовувати доступне мережеве обладнання та розгорнуту мережу для тестування експериментальних протоколів і нових ідей, наприкінці 2008 року з’явився OpenFlow, а в грудні 2008 року випустив свої перші специфікації [19].

Дивлячись на прогрес мережевої технології за останні два десятиліття, вона еволюціонувала завдяки широкомасштабним та інноваційним перетворенням, покращуючи її швидкість, всюдисущість, надійність і безпеку. На фізичному рівні мережеві пристрої покращилися з точки зору обчислювальної потужності, і з’явилися різноманітні програми, які пропонують інструменти для легкої перевірки операцій. Але мережева інфраструктура не зазнала значних змін з перших днів. Щоб додати нові послуги, додаються нові компоненти для підтримки подальших додаткових послуг і операцій на вищих рівнях, у той час як вони залишаються незмінними на фізичному та мережевому рівнях (рівні 1-3).

У існуючій інфраструктурі мережеві пристрої приймають рішення на рівні мережі, такі як маршрутизація або доступ до мережі. Існує багато комерційних постачальників мережевих пристроїв, які використовують різні

мікропрограми у своїх пристроях, і мережа зазвичай налаштована у відкритому режимі, а не за пропрієтарним способом для підтримки пристроїв від різних постачальників. Відкритий мод означає розгортання, незалежне від постачальника, а пропрієтарний — розгортання, що залежить від постачальника, і розгортається часто. Залежно від постачальника та його мережевого пристрою було важко перевірити нові дослідницькі ідеї, такі як протоколи маршрутизації, і встановити їх сумісність у реальних мережах. Крім того, спроба будь-яких експериментальних ідей над виробничою мережею з критичним пріоритетом може призвести до збою мережі в якийсь момент, що призвело до статичності та негнучкості мережевої інфраструктури та не привернуло серйозних інновацій у цьому напрямку [20].

Таблиці пошуку в комутаторах і маршрутизаторах Ethernet були основним ключем для впровадження брандмауерів, NAT, QoS або збору статистики продуктивності. OpenFlow враховує загальний набір функцій, що підтримується більшістю постачальників, що допомагає досягти стандартного способу використання таблиць потоків для всіх мережевих пристроїв, незалежно від їх постачальника. Це дозволяє розділити мережу на основі потоку, організовуючи мережевий трафік у різні класи потоків, які можна згрупувати разом або ізольувати для обробки, маршрутизації або контролю в бажаний спосіб. OpenFlow можна широко використовувати в кампусних мережах, де ізоляція дослідницького та виробничого трафіку є однією з найважливіших функцій.

OpenFlow пропонує мережевим адміністраторам програмне керування потоками, щоб визначити шлях, яким потік проходить від джерела до місця призначення, і використовує обробку на основі потоку для пересилання пакетів. Він пропонує спосіб усунути обробку пакетів маршрутизатором для визначення шляху, заощаджуючи енергоспоживання та витрати на керування мережею під час розширення мережі. OpenFlow викликав значний інтерес

серед розробників і виробників мережевих комутаторів, маршрутизаторів і серверів [21].

Термін «пересилання» не стосується комутації рівня 2 у середовищі протоколу OpenFlow, оскільки він також охоплює інформацію рівня 3, але, з іншого боку, він не виконує маршрутизацію рівня 3. Тому можна вважати, що термін переадресація відбувається між комутацією рівня 2 і маршрутизацією рівня 3.

2.2 Архітектура OpenFlow

У мережевих пристроях існує три площини: площина даних, площина керування та площина управління, як обговорювалося в розділі 2.1; однак у цій дипломній роботі зосереджені лише дані та площина керування. Концепція щодо прийняття централізованого контролю над мережею та поділу площини керування та даних обговорювалася дослідниками раніше в [22] та [23]. У SoftRouter була запропонована подібна архітектура, яка підкреслює роз’єднання даних і площини керування, спрямована на забезпечення більш ефективного пересилання пакетів [23]. OpenFlow нагадує ці архітектури в концепції поділу даних і площини керування, але підтверджує концепцію обробки на основі потоку за допомогою таблиць потоку.

Щоб отримати програмований контроль над площиною керування, потрібні комутатори, що підтримують OpenFlow, і контролер, що містить мережеву логіку. OpenFlow базується на комутаційному пристрої з внутрішньою таблицею потоків і забезпечує відкриту програмовану віртуалізовану комутаційну платформу для керування обладнанням комутатора за допомогою програмного забезпечення. Він може реалізовувати функцію комутатора, маршрутизатора або навіть обох, і дозволяє програмно

керувати шляхом керування мережевим пристроєм через протокол OpenFlow, як показано на Рис.2.1 [24].

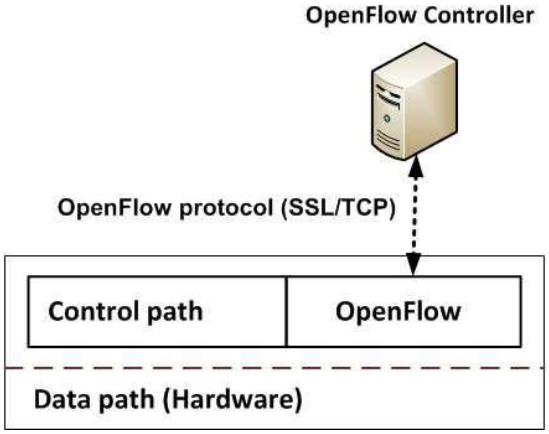

Рис.2.1. Архітектура фізичного рівня OpenFlow [24]

З'єднання контролера OpenFlow захищено здебільшого за допомогою SSL або TLS, але воно може бути вразливим до атак на відмову в обслуговуванні (DoS); тому для запобігання таким атакам необхідно вжити суворих заходів безпеки. В архітектурі OpenFlow перенаправлення потоку даних все ще знаходиться на комутаторі, але рішення про перенаправлення потоку приймаються в окремому контролері OpenFlow або ієрархії контролерів, яка реалізована на сервері (серверах), які спілкуються з комутаторами з підтримкою OpenFlow у мережі. через протокол OpenFlow.

Таким чином, основними компонентами мережі OpenFlow є:

- комутатори з підтримкою OpenFlow
- сервер(и), на яких працює контролер(и).

Компоненти описані далі.

2.2.1 Перемикач із підтримкою OpenFlow

База даних таблиці потоку, подібна до традиційної таблиці пересилання, знаходиться на комутаторі з підтримкою OpenFlow, який містить записи потоку, які допомагають виконувати пошук пакетів і

приймати рішення про пересилання пакетів. Комутатор із підтримкою OpenFlow підключений до контролера через захищений канал, по якому комутатор і контролер обмінюються повідомленнями OpenFlow для виконання завдань налаштування та керування, як показано на Рис.2.2 нижче.

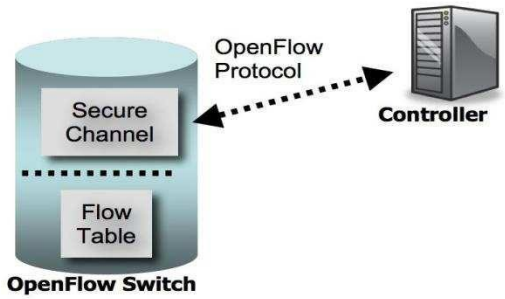

Рис.2.2. Зв'язок між комутатором OpenFlow та контролером [25].

Перемикач із підтримкою OpenFlow містить одну або кілька таблиць потоків і групову таблицю, які виконують пошук і пересилання пакетів. Кожна таблиця потоку в комутаторі містить набір записів потоку, де кожен запис потоку складається з полів відповідності, лічильників і набору інструкцій або дій, які будуть застосовані до відповідних пакетів. Ці поля запису потоку детально описані в розділі 3.5.

Комутатор із підтримкою OpenFlow приймає рішення про переадресацію, переглядаючи записи таблиці потоків і знаходячи точний збіг у конкретних полях вхідних пакетів, таких як номер порту, IPv4-адреса джерела чи призначення, MACадреса джерела чи призначення тощо. Для кожного потоку запис таблиці містить пов’язану дію, яка виконуватиметься над вхідними пакетами. Для кожного вхідного пакета комутатор переглядає свою таблицю потоків, знаходить відповідний запис і пересилає пакети на основі відповідної дії. Якщо записи потоку вхідних пакетів не збігаються з таблицею потоків комутатора, тоді, залежно від конфігурації мережі OpenFlow, комутатор надсилає їх до контролера для прийняття подальшого

рішення або переходить до наступної таблиці потоків, як показано на Рис.2.3 [21] [26].

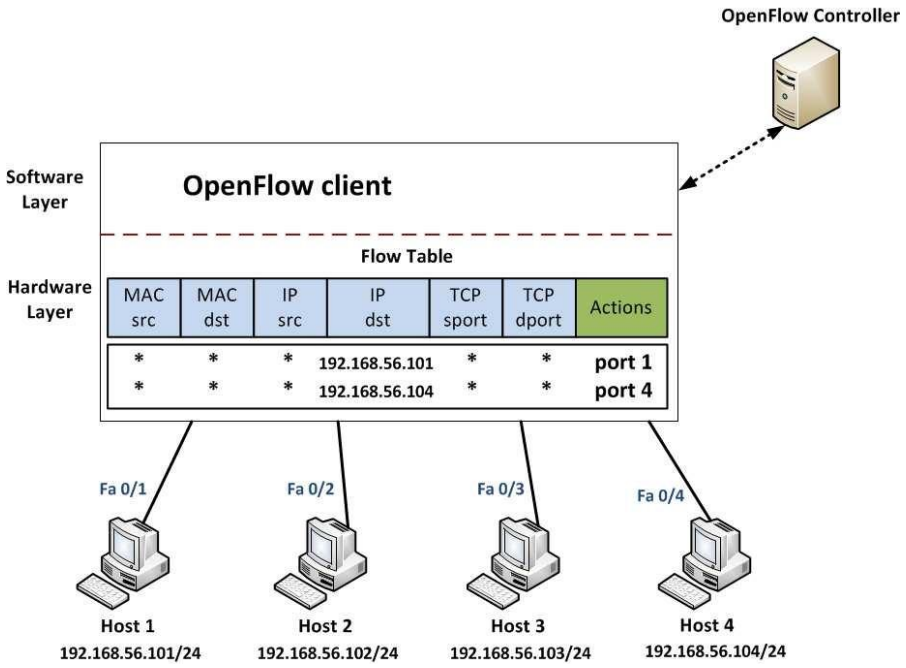

Рис.2.3. Процес пересилання пакетів OpenFlow [24]

Пов’язані дії з кожним записом потоку або містять дії, або змінюють обробку конвеєра. Дії включають пересилання пакетів, модифікацію пакетів і обробку групової таблиці, де, як і в конвеєрній обробці, пакети надсилаються до наступних таблиць потоку для подальшої обробки. Інформація передається з однієї таблиці в іншу у формі метаданих. Обробка конвеєра припиняється, коли набір інструкцій, пов’язаний із записом потоку, не згадує наступну таблицю, і пакет змінюється та пересилається далі.

Записи потоку можуть бути направлені на фізичний порт, логічний порт або зарезервований порт. Логічний порт, визначений комутатором, може визначати групи агрегації каналів, тунелі або інтерфейси петлі; тоді як зарезервований порт, визначений специфікацією, може виконувати загальні дії пересилання, такі як надсилання на контролер, затоплення або пересилання за допомогою методів, відмінних від OpenFlow, тобто

традиційної обробки комутаторів. Асоційовані дії з кожним записом потоку можуть направляти пакети до групи, яка виконує додаткову обробку, таку як лавинна передача, багатошляхове перенаправлення, швидке перенаправлення та агрегація каналів [26].

Групи пропонують спосіб пересилання кількох записів потоку до одного ідентифікатора, наприклад, загального наступного переходу. Таблиця груп містить записи групи, і залежно від типу групи кожен запис групи містить список сегментів дій.

Після надходження пакетів до групи над ними виконуються дії з пов’язаних сегментів.

2.2.2 Контролер

Контролер — це централізована сутність, яка збирає функціональні можливості рівня керування, створює оновлення та видаляє записи потоку в таблицях потоків на комутаторі, де потік відноситься до односпрямованої послідовності пакетів, які спільно використовують набір загальних значень заголовків пакетів. Окрім основної функції, його можна розширити для виконання додаткових критичних завдань, таких як маршрутизація та доступ до мережі.

На даний момент доступно кілька реалізацій контролерів, які є відкритими і базуються на різних мовах програмування, таких як Python, C++ і Java [27]. У цій дипломній роботі для проведення експериментів було обрано контролер Floodlight з відкритим кодом на основі Java. Як правило, контролер працює на сервері, підключеному до мережі, і може обслуговувати один або кілька комутаторів залежно від конструкції мережі. Він може бути розроблений з централізованою ієрархією, де один контролер обробляє та контролює всі комутатори в мережі, або розподіленою ієрархією, де два або більше контролерів обробляють і контролюють дві або більше груп

комутаторів у мережі. У централізованій ієрархії, якщо контролер виходить з ладу, усі мережеві операції перериваються, і це створює єдину точку збою в мережі OpenFlow.

У розподіленій ієрархії всі контролери повинні мати однакову копію подання топології мережі в режимі реального часу, щоб уникнути втрат пакетів. Перегляд топології мережі включає топологію рівня комутатора; розташування користувачів, хостів, середніх блоків та інших мережевих елементів і служб. Крім того, він включає всі прив'язки між іменами та адресами.

2.3 Типи течії

Потоки можна далі розділити на мікропотоки та агреговані потоки відповідно до кількості призначених хостів [24].

Мікропотоки: кожен потік окремо налаштовується контролером і відповідає критерію точної відповідності записів потоку. Таблиця потоків містить один запис на потік і вважається хорошою для контролю дрібної зернистості, політики та моніторингу, наприклад, мережі кампусу. Агрегований: у цьому випадку один запис потоку охоплює великі групи потоків, і дозволені записи потоку зі знаком підстановки. Таблиця потоків містить один запис для кожної категорії потоків і підходить для великої кількості потоків, наприклад, у магістральній мережі. Процес заповнення входів потоку в комутатор можна далі класифікувати на реактивний і проактивний режим [24].

- Реактивний: перший пакет потоку запускає контролер для вставлення записів потоку, а комутатор ефективно використовує таблицю потоку, де кожен потік потребує невеликого додаткового часу налаштування потоку. У разі втрати з’єднання між комутатором і контролером,

комутатор має обмежену корисність, а процес відновлення несправності простий.

- Проактивний: у цьому випадку таблиці потоків у комутаторі попередньо заповнюються контролером, і це не враховує додатковий час налаштування потоку. Зазвичай для цього потрібні агреговані правила (тобто символи підстановки), і в разі втрати з'єднання трафік не порушується.

Агреговані потоки зменшують накладні витрати потоку, а проактивні потоки зменшують навантаження на обробку для контролера, оскільки запит не надсилається до контролера щоразу.

2.4 Методика роботи

Протокол OpenFlow не визначає, як приймаються рішення про пересилання для конкретних полів заголовка (тобто дій). Але насправді ці рішення приймає контролер і просто завантажує або встановлює в таблиці потоків комутаторів. Що стосується перемикачів Open-Flow, таблиці потоків переглядаються, а поля заголовків вхідних пакетів узгоджуються з попередньо обчисленими рішеннями про пересилання, де в разі збігу виконується відповідне рішення. Якщо відповідності не знайдено, пакет пересилається на контролер Open-Flow для подальшої обробки. Залежно від типу потоку, тобто реактивного чи проактивного потоку, контролер переглядає свою базу даних і, знайшовши відповідність, надсилає пакет назад до комутатора OpenFlow і встановлює відповідний потік у таблицю потоків комутатора. Обробку пакетів через потоки в протоколі OpenFlow можна побачити на блок-схемі на Рис.2.4 [26].

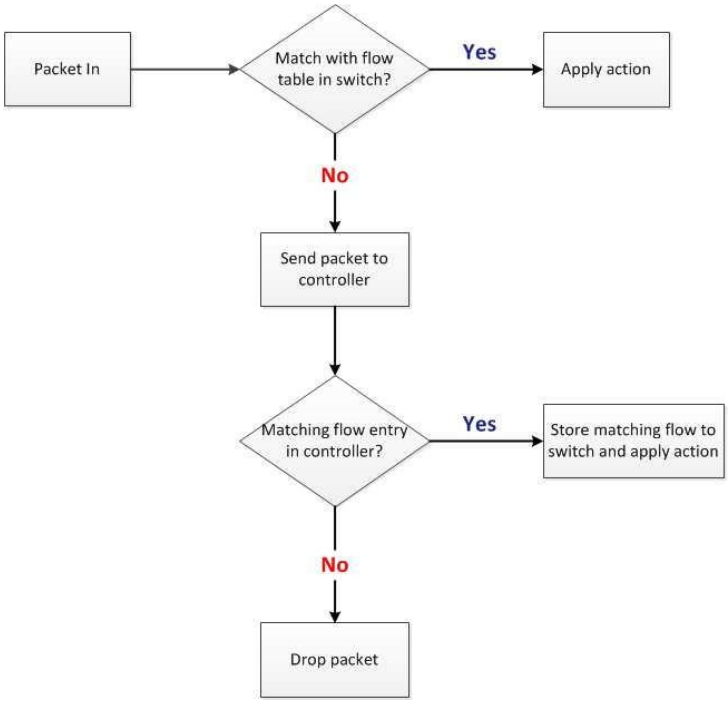

Рис.2.4. Обробка потоку в протоколі OpenFlow [26]

Повідомлення, якими обмінюються під час встановлення з'єднання між комутатором і контролером, і методологія роботи, коли два хости підключаються один до одного в мережі OpenFlow, описані наступний.

2.4.1. Типи повідомлень, якими обмінюються комутатор і контролер

OpenFlow надає протокол для зв'язку між комутаторами OpenFlow і контролером і підтримує три типи повідомлень, якими обмінюються між ними: повідомлення від контролера до комутатора, асинхронні та симетричні повідомлення, які коротко обговорюються тут [26].

Повідомлення від контролера до комутатора ініціюються контролером і не завжди можуть вимагати відповіді від комутатора. Ці повідомлення використовуються для конфігурації комутатора, керування таблицею потоків комутатора та отримання інформації про стан таблиці потоків або

можливості, підтримувані комутатором у будь-який момент часу, наприклад, Features, Config, Modify-State, Read-State та Packet -Out, які описані в наступному розділі встановлення з’єднання.

Асинхронні повідомлення надсилаються без запиту від комутатора до контролера та позначають зміну стану комутатора або мережі, що також називається подією. Однією з найважливіших подій є подія надходження пакета, яка виникає щоразу, коли пакет, який не має відповідного запису потоку, досягає комутатора. У разі виникнення такої події до контролера надсилається повідомлення про вхід пакета, яке містить пакет або частину пакета, щоб його можна було перевірити та прийняти рішення про те, який тип встановлення потоку можна прийняти. Деякі інші події включають закінчення терміну дії потоку, зміну статусу порту або інші події помилки.

Нарешті, третя категорія «симетричних повідомлень» надсилається без запиту в будь-якому напрямку, тобто комутатору чи контролеру. Ці повідомлення використовуються для допомоги або діагностики проблем у з’єднанні комутатора з контролером, а повідомлення Hello та Echo належать до цієї категорії.

2.4.2.Встановлення зв’язку між контролером і комутатором

Коли будь-який комутатор налаштовано в режимі OpenFlow, комутатор починає шукати контролер, надсилаючи повідомлення синхронізації TCP на IP-адресу контролера на TCP-порт за замовчуванням 6633. Отримавши повідомлення підтвердження синхронізації TCP від контролера, комутатор знову надсилає підтвердження до контролер, і відбувається квиткування TCP.

Таким чином, коли будьякий новий комутатор буде додано до мережі OpenFlow, він буде автоматично підключений до контролера. Процес встановлення з’єднання між комутатором і контролером можна побачити на Рис.2.5, де стрілки показують напрямок повідомлень.

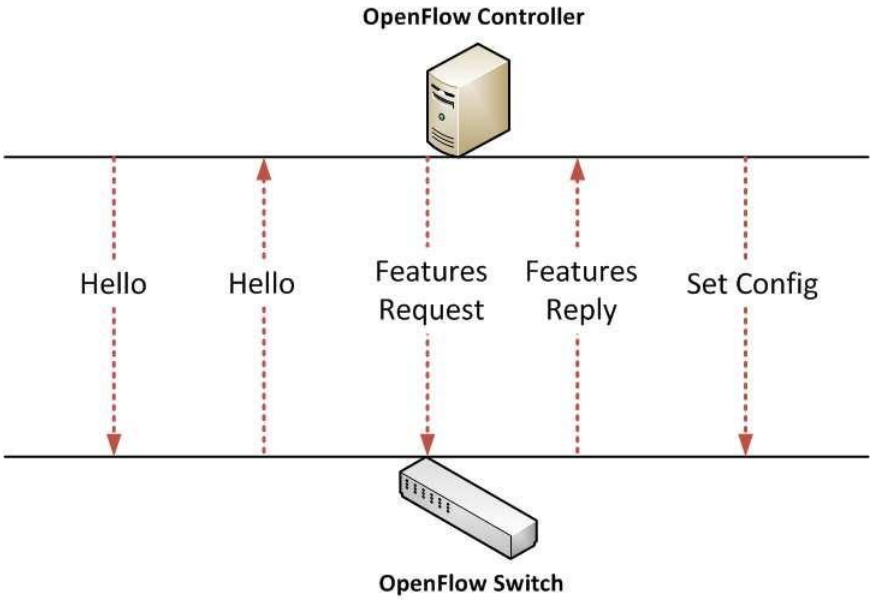

Рис.2.5. Встановлення з'єднання між комутатором і контролером

Після рукостискання TCP процес починається з лівого боку та закінчується налаштуванням конфігурації для перемикання. Повідомлення коротко обговорюються тут [26]:

- **Hello** (контролер → комутатор):Контролер надсилає комутатору номер своєї версії.
- **Hello** (комутатор → контролер): У відповідь комутатор повідомляє номер підтримуваної версії.
- **Features Request**: Контролер запитує, які порти доступні.
- **Features Reply**: Комутатор надає відповідь зі списком портів, швидкостей портів та підтримуваних таблиць і дій.
- **Set Config**: Контролер надсилає запит перемикачу на відправку закінчення потоку.

Інші повідомлення, якими обмінюються, включають Ping Request і Ping Reply.

2.4.3. З’єднання між хостами в мережі OpenFlow

Після встановлення з’єднання між комутатором і контролером відбувається процес зв’язку між двома чи більше хостами через мережу OpenFlow, як показано на Рис.2.6, де стрілки представляють напрямок повідомлень. Повідомлення коротко обговорюються тут [26].

- **Packet-In**: Якщо будь-який вхідний пакет не відповідає жодному запису потоку в таблиці потоків комутатора, він надсилається до контролера.
- **Packet-Out**: контролер надсилає пакети на один або більше портів комутатора.
- **Flow-Mod**: Контролер дає команду комутатору додати певний потік до своєї таблиці потоків.
- **Flow-Expired**: Комутатор інформує контролер про потоки, у яких закінчився таймінг.
- **Port Status**: Комутатор повідомляє контролер про додавання, видалення та модифікацію портів.

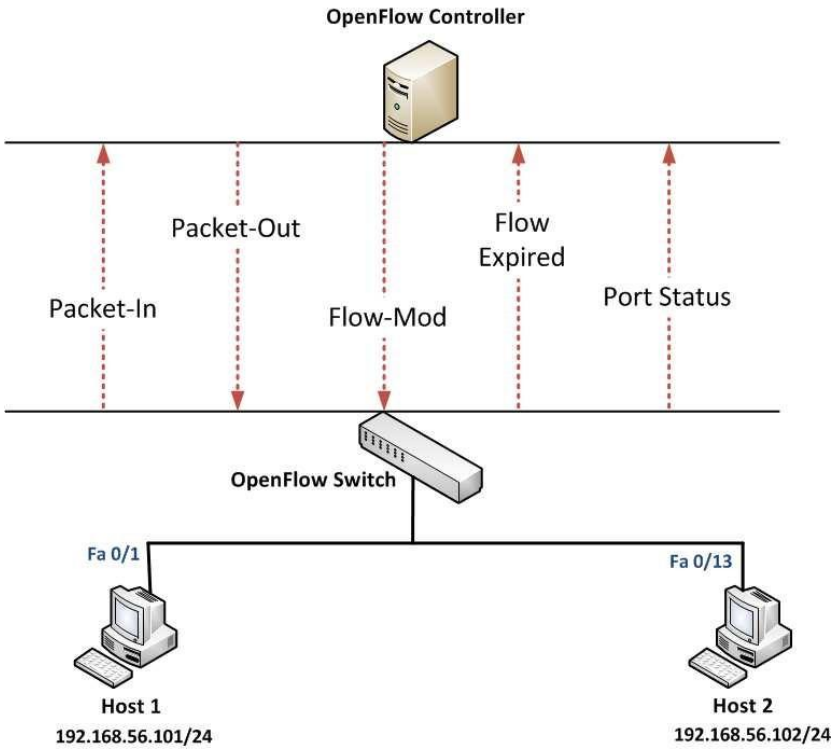

Рис.2.6. З'єднання між хостами в мережі OpenFlow

2.5 Формат пакета

Згідно з [26], протокол OpenFlow складається з трьох основних полів: полів заголовків, лічильників та дій, які містяться в кожному записі потоку в таблиці потоків.

- Поля заголовка або поля відповідності: ці поля ідентифікують потік, зіставляючи пакети з певними полями, як показано в Рис.2.7.

	Ethernet				**IP**				
Ingress port	Src	Dst	Type	VLAN ID	Src	Dst	Proto	Src port	Dst port

Рис.2.7. Поля заголовка для зіставлення з записами потоку [26]

- Лічильники: вони використовуються для статистичних цілей, щоб відстежувати кількість пакетів і байтів для кожного потоку та час, що минув з моменту початку потоку.
- Дії: дія визначає спосіб обробки пакетів потоку. Дія може бути однією з наступних: 1) переслати пакет на вказаний порт або порти, після необов’язкового переписування деяких полів заголовка, 2) відкинути пакет 3) переслати пакет до контролера.

Протокол OpenFlow знаходиться між комутатором і контролером, що дозволяє передавати інформацію про рівень керування. Специфікацію пакетів OpenFlow додано до дисектора Wireshark [28], а пакети протоколу OpenFlow можна захоплювати та аналізувати за допомогою стандартного інструменту захоплення пакетів - Wireshark. Протокол OpenFlow отримав акронім «OFP», і трафік OpenFlow можна фільтрувати в Wireshark, активувавши фільтр «of» для трафіку.

Розглянемо OpenFlow мережу, яку можна побачити на Рис.2.8, де два хости з мережі 192.168.56.0/24 є функціональними вузлами OpenFlow мережі. Контролер OpenFlow з IP-адресою 192.168.58.110 керує комутатором з підтримкою OpenFlow. Комутатор з підтримкою OpenFlow з'єднаний з контролером OpenFlow через порт позасмугового керування (OOBM) з IP-адресою 192.168.58.101.

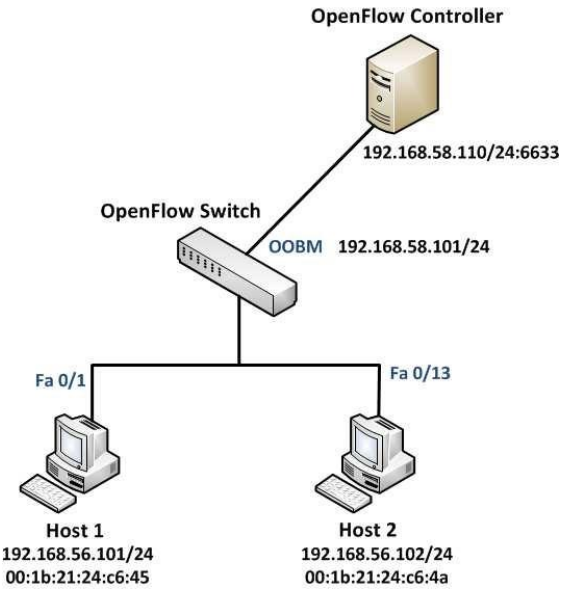

Рис.2.8. Топологія мережі для прикладу OpenFlow мережі

Захоплення пакетів Wireshark з цієї мережі OpenFlow показано на Рис.2.9. На ньому показано поля заголовків, поля відповідності та поля дій, тоді як поля лічильників можна побачити лише в таблиці потоків комутатора і їх не видно в захопленні пакетів. Поля лічильників показано в Розділі 3 на Рис.3.5.

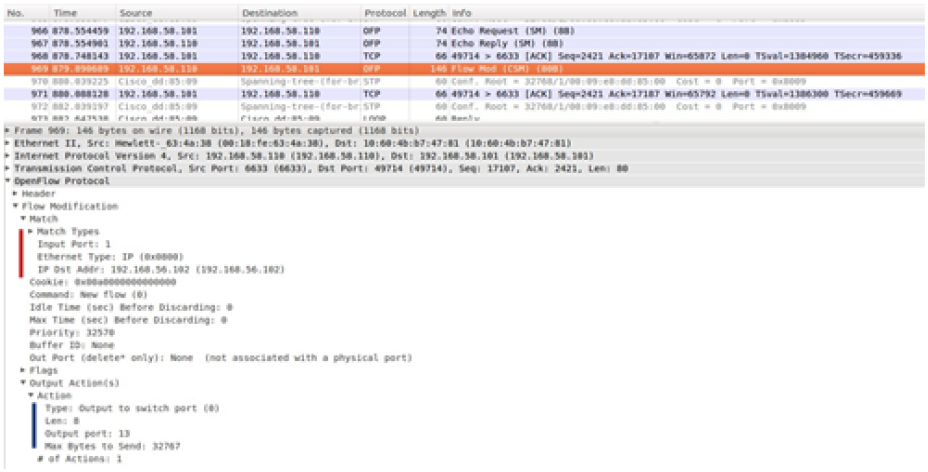

Рис.2.9. Захоплення пакетів OpenFlow у Wireshark

Вибраний пакет на рис.2.9. відноситься до повідомлення про модифікацію потоку (тобто FlowMod), яке викликається, коли контролер додає новий потік. Повідомлення надходить від контролера (192.168.58.110) і призначається комутатору (192.168.58.101), де буде встановлено потік. Приклад потоку від Host1 до Host2 можна побачити в захопленні пакетів, де потік буде виникати з порту 1, де знаходиться Host1, і буде призначено до порту 2, де знаходиться Host2.

Корисне навантаження повідомлення Flow-Mod складається з полів відповідності та пов’язаної дії, яка буде вставлена в таблицю потоку. Із запису пакетів Wireshark стає очевидним, що пакет OpenFlow — це звичайний протокол прикладного рівня, інкапсульований у форматі TCP, IPv4 і Ethernet.

2.6 Проекти OpenFlow

Цей розділ описує деякі проекти та фреймворки OpenFlow, які мають відношення до області, що нас цікавить.

2.6.1. Open vSwitch

Open vSwitch — це багаторівневий програмний комутатор із відкритим кодом, призначений для керування великомасштабними віртуалізованими середовищами. Це мотивовано зростаючими потребами у віртуалізованому середовищі, а для налаштування шляху переадресації комутатора використовується надмножина протоколу OpenFlow. Він використовує підхід централізованого контролера для підключення до комутаторів з підтримкою OpenFlow; однак додаткові інтерфейси керування, такі як SNMP, можна використовувати для конфігурацій.

Він функціонує як віртуальний комутатор, забезпечує підключення між віртуальними машинами (VM) і фізичними інтерфейсами. Він також емулює протокол OpenFlow у віртуалізованому середовищі на основі Linux, включаючи XenServer, віртуальну машину на основі ядра (KVM) і VirtualBox. Він реалізував протокол OpenFlow версії 1.0 і новіших, остання версія версії 1.9.0 також включає підтримку IPv6, яка вказана в OpenFlow версії 1.2 і новіших. На додаток до OpenFlow, він підтримує багато інших традиційних методів комутації, включаючи 802.1Q VLAN, конфігурацію QoS, керування помилками підключення 802.1ag і методи тунелювання, такі як тунелювання загальної інкапсуляції маршрутизації (GRE) [29]. Він також надає корисні інструменти та утиліти для емуляції протоколу OpenFlow, а саме:

- ovs-vsctl - утиліта, яка запитує та оновлює конфігурацію софт-комутатора
- ovs-controller - еталонний OpenFlow контролер
- ovsdbmonitor - графічний інструмент для перегляду баз даних і таблиць потоків Open vSwitch.
- ovs-ofctl - утиліта, яка запитує та керує OpenFlow комутаторами та контролерами.

Архітектуру Open vSwitch показано на Рис.2.10 нижче, де ovsdbserver - це база даних, що містить конфігурацію рівня комутатора, а ovs-vswitchd - це основний компонент Open vSwitch. Ovs-vswitchd підтримує декілька шляхів передачі даних і перевіряє лічильники потоку даних на предмет закінчення терміну дії потоку, а також виконує запити статистики.

Ovs-vswitchd є комунікаційним хабом, який взаємодіє із зовнішнім світом, ovsdb-сервером, модулем ядра і всією системою Open vSwitch. ВМ (віртуальні машини) підключаються до ядра Open vSwitch через віртуальні інтерфейси, а ядро забезпечує підключення до протоколу OpenFlow та фізичних інтерфейсів, що лежать в його основі.

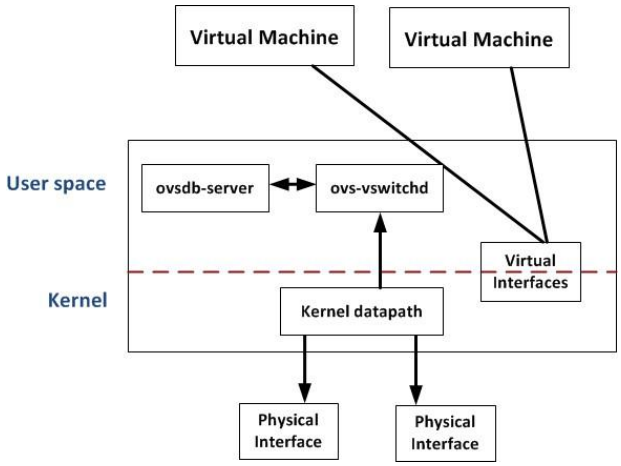

Рис.2.10. Архітектура Open vSwitch

2.6.2. FlowVisor

Фреймворк FlowVisor спрямований на допомогу в розгортанні мережі OpenFlow із підходом розподіленого контролера. Він використовує потокові простори для створення мережевих сегментів, що полегшує та робить незалежним керування обробкою потоку кількома контролерами OpenFlow. Зріз визначається як набір бітів заголовків пакетів, які відповідають підмножині мережевого трафіку OpenFlow, а простір потоку відноситься до області, що представляє підмножину потоків трафіку, які збігаються з бітами

заголовків пакетів. Простіше кажучи, простір потоків можна визначити як контейнер для певної області, де потоки збігаються [17]. Мережу OpenFlow із використанням FlowVisor можна побачити на Рис.2.11 нижче.

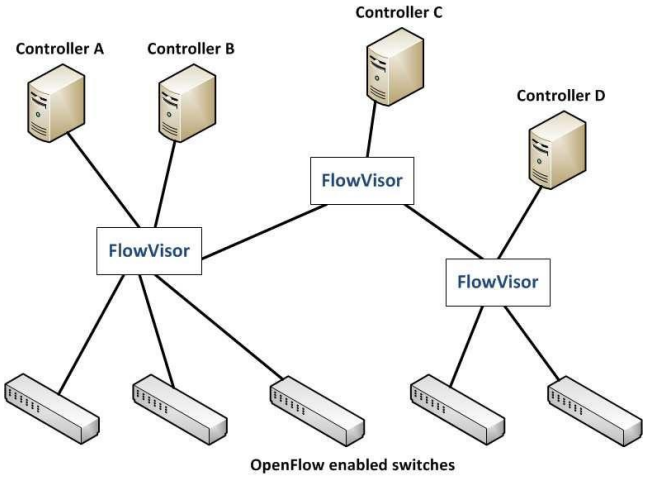

Рис.2.11. Мережа OpenFlow з FlowVisor [30]

Кожен простір потоку можна зіставити з одним або кількома контролерами OpenFlow у FlowVisor. Крім того, кожному контролеру можна надати різні рівні контролю доступу до потокового простору, наприклад потоки запису або модифікації та лише потоки читання. Він також пропонує пріоритетне прийняття рішень і корисний у випадках, коли виникають перекриття потоків.

Повідомлення OpenFlow, що надходять від комутатора, передаються до FlowVisor, де після перевірки вони пересилаються до відповідного контролера на основі правил потокового простору. Тому кожен контролер отримує лише ті пакети та повідомлення, за які він відповідає, тим самим зменшуючи навантаження на обробку від кожного контролера. З іншого боку, механізм контролю доступу допомагає досягти цього, тобто пакети, що надходять від контролера, пересилаються на призначений комутатор, лише якщо контролер має контроль доступу, наданий для цього комутатора або регіону [30].

2.6.3. LegacyFlow

Щоб скористатися перевагами мережі OpenFlow, усі мережеві пристрої (тобто комутатори) мають підтримувати протокол OpenFlow, що збільшує витрати. Запропонована архітектура спрямована на збереження традиційних мережевих комутаторів із використанням мережі Open-Flow. Він перетворює дії OpenFlow у специфічні для постачальника конфігурації для мережевих комутаторів через інтерфейси SNMP або CLI та з'єднує комутатори з підтримкою OpenFlow через традиційні мережеві комутатори за допомогою VLAN на основі каналів.

Гібридна модель була запропонована в [31], яка додає менше комутаторів з підтримкою OpenFlow до вже розгорнутої традиційної мережевої інфраструктури. Він дотримується стратегії SDN, тобто окрема площина керування та даних, як у протоколі OpenFlow, і додає ще один віртуальний шлях даних, який взаємодіє з OpenFlow і традиційними мережами [31] [32]. Мережу OpenFlow, яка використовує архітектуру LegacyFlow, можна побачити на Рис.2.12.

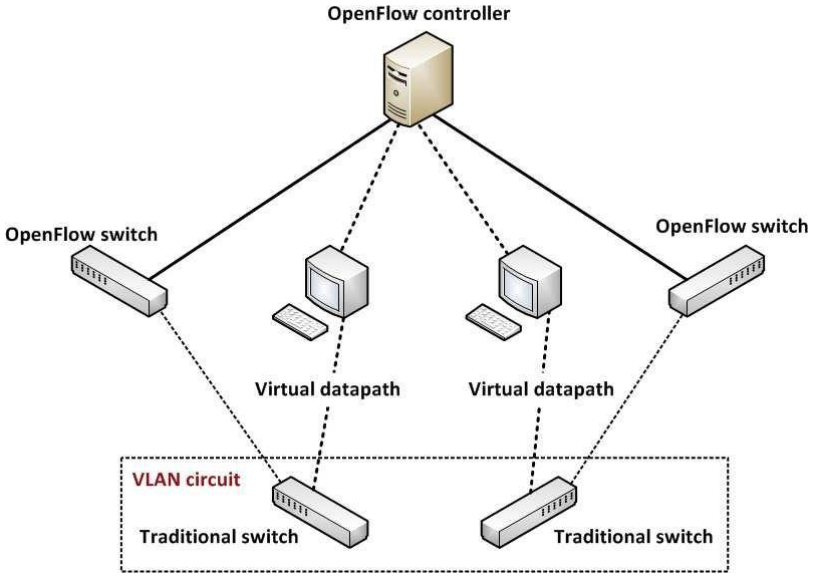

Рис.2.12. Мережа OpenFlow з використанням підходу LegacyFlow [31]

Мережа OpenFlow із використанням підходу LegacyFlow [31] Мережеві інтерфейси є основною необхідною інформацією для ініціювання шляху даних OpenFlow, який передається як параметри. Відповідно до кількості портів, доступних або використовуваних у традиційному комутаторі, відповідна кількість віртуальних мережевих інтерфейсів створюється модулем Linux, що відображає традиційний комутатор. Модуль Linux працює на окремих машинах Linux, як показано на Рис.2.12. Функції традиційного комутатора передаються на віртуальні інтерфейси через SNMP або веб-сервіс.

Віртуальний шлях даних виділяється кожному традиційному комутатору, який працює на реальній або віртуальній гостьовій машині ОС Linux, яка створює два інтерфейси: вхідний і вихідний порт. Він призначений для передачі зовнішнього вигляду шляху даних OpenFlow і надання інформації про функції традиційних комутаторів, таких як швидкість надсилання та отримання пакетів, номери портів тощо. Отримуються повідомлення OpenFlow від контролера OpenFlow; інтерпретується та відповідні дії застосовуються до традиційного комутатора через віртуальний шлях даних, таким чином слугуючи проксі. Дії, що застосовуються до традиційного комутатора з використанням віртуального шляху даних, включають створення ланцюгів, отримання статистичних даних щодо переданих і отриманих пакетів і видалення ланцюгів. Крім того, схеми створюються з функціями: меншою тривалістю, з QoS і без тайм-ауту.

LegacyFlow ініціює віртуальний шлях даних і отримує інформацію про традиційну модель комутатора, а потім ініціює модуль віртуального інтерфейсу та створює віртуальні інтерфейси. Зовнішній виділений позасмуговий канал, тобто порт OOBM, обмінюється даними між традиційним комутатором і віртуальним трактом даних, який отримує повідомлення та застосовує відповідні дії до віртуальних інтерфейсів.

Послідовність повідомлень оновлюється кожні 3 секунди, щоб відстежувати зміни в комутаторі та мережі.

Після фази ініціації відповідні інтерфейси підключаються до віртуального шляху даних, які отримують дії OpenFlow від контролера OpenFlow. Отримавши дію, вони інтерпретуються та перевіряються на сумісність із віртуальним шляхом даних. Якщо він сумісний, потік встановлюється в комутатор OpenFlow. Контролер OpenFlow оновлює свою таблицю потоків і визначає, що пункт призначення може бути досягнутий через інший перемикач OpenFlow, між цими двома елементами встановлюється ланцюг за допомогою традиційних перемикачів.

2.6.4. RouteFlow

RouteFlow — це рішення з відкритим вихідним кодом для надання застарілих служб IP-маршрутизації, таких як OSPF, RIP тощо, через мережі OpenFlow і забезпечує віртуальний шлюз. Архітектуру RouteFlow можна побачити на Рис.2.13. [33]

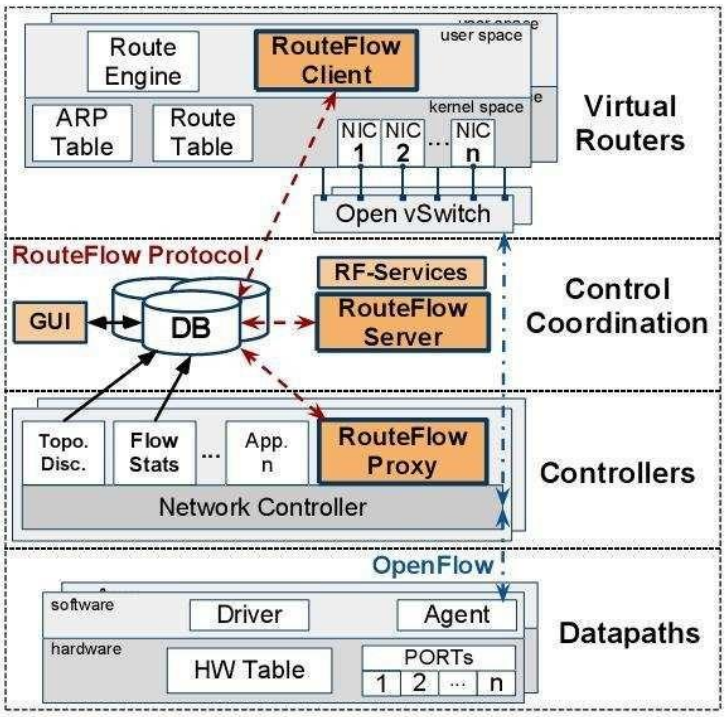

Рис.2.13. Архітектура RouteFlow [33]

Пристрій OpenFlow зіставляється з однією віртуальною машиною (VM) за допомогою Quagga, яка забезпечує площину керування IP, а RouteFlow відстежує зміни таблиці маршрутизації та встановлює відповідні записи потоку в пристрій OpenFlow. Quagga — це пакет програмного забезпечення для маршрутизації під ліцензією GPL, який забезпечує реалізацію OSPFv2, OSPFv3, RIP v1 і v2, а також BGP для платформ Unix [34]. Усі пакети, які надходять на пристрої OpenFlow, надсилаються на відповідну віртуальну машину і навпаки, тоді як повідомлення протоколу маршрутизації, створені Quagga, надсилаються через RF-сервер. Вихідний код RouteFlow за замовчуванням розроблено для роботи у сценарії віртуального середовища, однак можна визначити гнучке відображення між фізичними пристроями OpenFlow і віртуальною площиною керування у віртуальній машині з віртуальними інтерфейсами, зіставленими з фізичними портами в пристрої з підтримкою OpenFlow [33].

RF-клієнт — це «демон», що працює у віртуальних машинах, де виконується Quagga, і призначений для моніторингу змін у таблиці маршрутизації Linux. Він надсилає тестові пакети, діючи як техніка визначення місцезнаходження, яка допомагає зіставляти віртуальні інтерфейси з фізичними інтерфейсами на пристрої OpenFlow. Після виявлення змін у таблиці маршрутизації Linux інформація про маршрут пересилається на RF-сервер.

RF-сервер містить основну логіку RouteFlow, і після отримання повідомлень про зміни в таблиці маршрутизації Linux від RF-клієнта він запускає подію встановлення або модифікації потоку в пристрої OpenFlow. Він також отримує зареєстровані події від модуля контролера RouteFlow (NOX або POX) і вирішує, які дії потрібно виконати для цих подій, наприклад, введення пакетів, приєднання до шляху даних тощо. Остання

функція RF-сервера включає повноваження реєстрації для віртуальних машин. який підтримує синхронізацію з даними.

Дивлячись на архітектуру, показану в [33], від модуля шляху даних, де знаходиться апаратне забезпечення, RouteFlow підключається до модуля контролера через протокол OpenFlow, подібний до всіх інших програм OpenFlow. У модулі контролера RFпроксі піклується про контролер NOX або POX і надсилає керуючу інформацію модулю координації керування. У модулі координації управління RF-сервер виявляє зміни в таблиці маршрутизації Linux і встановлює відповідні потоки через інформацію від RFклієнта. У модулі віртуального маршрутизатора працює RF-клієнт і здійснюється віртуальне відображення портів.

2.6.5. OpenFlow MPLS

Багатопротокольна комутація за мітками (MPLS) — це протокол, який пересилає пакети шляхом збігу міток у заголовку пакета до пункту призначення, і широко використовується в комерційних цілях мережевими операторами. Специфікації OpenFlow v1.0 не підтримують протокол MPLS (специфікація OpenFlow Switch v1.0.0, 2009), а в [35] було запропоновано та реалізовано розширення OpenFlow v1.0 для включення підтримки MPLS.

Заголовки пакетів змінюються трьома діями, а саме надсиланням, видаленням і заміною стека міток MPLS. Ця модифікація додає або видаляє мітку, яка ідентифікує належність до класу еквівалентності пересилання (FEC) для пакетів, і ця мітка вставляється між рівнем 2 і рівнем 3, тобто між IP і MAC. Довжина стека міток MPLS становить 32 біти, з яких 20 біт становлять фактичну мітку, а решта вказує час життя (TTL), параметри QoS та індексацію.

Перш за все, дві мітки MPLS додаються до полів заголовка протоколу OpenFlow, показаного раніше на Рис.3.7, який використовується для

ідентифікації потоку, що робить поля заголовка 12-бітними, і це можна побачити на Рис.2.14 нижче. Одна мітка визначає тип служби, наприклад, VLAN, а інша визначає транспортний тунель, як це видно в більшості комерційних розгортань.

	Ethernet				IP					MPLS	
Ingress port	Src	Dst	Type	VLAN ID	Src	Dst	Proto	Src port	Dst port	Label 1	Label 2

Рис.2.14. Модифіковані OpenFlow поля заголовків для MPLS [35]

Щоб додати дії MPLS, а саме push, pop, swap, decrement TTL у набір дій Open-Flow, було розглянуто механізм абстракції віртуального порту, який може працювати зі складними діями, що вимагають модифікації заголовка пакета. Таким чином, модель площини даних OpenFlow модифіковано для віртуального порту, який підтримує інкапсуляцію та декапсуляцію MPLS. Таблиця віртуальних портів показує групування віртуальних і фізичних портів разом, де кожен запис таблиці містить номер порту, батьківський порт, дії, які потрібно виконати, і статистику, яку можна побачити на Рис.2.15 нижче.

Port No.	Parent port	Actions	Statistics

Рис.2.15. Запис у таблиці віртуальних портів

Крім того, протокол OpenFlow було змінено, щоб додати повідомлення для контролера OpenFlow для впровадження шляхів з комутацією міток (LSP) у комутатори з підтримкою OpenFlow. Ці повідомлення включають vport_mod, vport_table_stats, які додають або видаляють номер віртуального порту, надсилають його дії та повертають статистику для них відповідно. Крім того, повідомлення port_stats (тобто статус порту) було змінено, щоб

отримати статус віртуальних портів, незалежно від того, доступні вони чи ні. Останнє змінене поле містить switch_features_reply, яке повідомляє, чи підтримує комутатор віртуальні порти чи ні.

Після внесення необхідних змін у заголовок і дії протоколу OpenFlow їх було реалізовано на апаратному забезпеченні NetFPGA, а також за допомогою Open vSwitch. NetFPGA діяла як маршрутизатор з комутацією міток (LSR), і потоки MPLS працювали добре. Успішна реалізація цього проекту призвела до додавання MPLS до OpenFlow у Open-Flow v1.1 і новіших версіях.

2.7 Перебіг і поточне розгортання OpenFlow

Комерційні постачальники мережевого обладнання почали включати підтримку протоколу OpenFlow у свої продукти та вітають підхід програмно визначеної мережі (SDN). Більшість постачальників приєдналися до Open Networking Foundation (ONF) — організації, яка стандартизує та підтримує розробку OpenFlow і має різні робочі групи, пов’язані з нею. Деякі з постачальників включають HP, IBM, Netgear, NEC, Brocade Systems і Juniper Networks. На додаток до цих постачальників, Pica8, виробник комутаторів, пропонує програмні комутатори, керовані програмним забезпеченням Xorplus, які підтримують різні протоколи рівня 2 і рівня 3, включаючи OpenFlow [36]. Netgear включила підтримку OpenFlow в один зі своїх повністю керованих комутаторів наступного покоління.

Практичні розгортання протоколу OpenFlow були в багатьох місцях по всьому світу. У США Global Environment for Network Innovations (GENI) — це тестова мережа ресурсів, доступних для дослідників, які розгортаються за допомогою OpenFlow. GENI спонсорується Національним науковим фондом (NSF) і спрямований на співпрацю та дослідження інноваційних відкриттів у глобальних мережах серед академічних кіл, промисловості та громадськості.

Деякі університети, які розгортають GENI OpenFlow, включають Стенфордський університет, Університет Індіани та Прінстонський університет. Більшість комутаторів із підтримкою OpenFlow, які використовуються в розгортанні GENI, походять від HP, NEC і Pronto [38]. Крім того, у США між Університетом Індіани та Стенфордським університетом діє програма Network Development and Deployment Initiative (NDDI), яка спрямована на надання рішення SDN за допомогою OpenFlow. NDDI буде використовуватися для створення кількох віртуальних приватних мереж, і дослідники отримають можливість виконувати експериментальні тести на Інтернет-протоколах і архітектурах одночасно [39].

У Європі FP7 ІКТ-проект Європейського Союзу OpenFlow in Europe: Linking

Infrastructure and Applications (OFELIA) зосереджується на розгортанні OpenFlow у Європі, який спрямований на те, щоб допомогти дослідникам використовувати мережі OFELIA для проведення експериментів із дослідницькими протоколами [40].

Google реалізував і запускає магістральний мережевий трафік у мережі SDN, використовуючи протокол OpenFlow. Google вирішив побудувати програмно визначену глобальну мережу (WAN), яка призвела до вищої продуктивності, відмовостійкості, меншої складності та вартості. Тим часом цікаві параметри: статистика продуктивності мережі, деталі вартості та деталі дизайну мережі не згадуються в [41].

Мережа OpenFlow також була створена в університетській лікарні Канадзава в Японії з використанням пристроїв серії програмованих потоків NEC. Мережа в лікарні стала великою та складною, щоб задовольнити індивідуальні потреби кожного відділення та вмістити інноваційні медичні технології та обладнання. Персоналу лікарні було важко використовувати, керувати та додавати нові пристрої в мережу лікарні, і нарешті був застосований підхід SDN. Розгорнута мережа OpenFlow у лікарні подолала

виклики та запропонувала нову стабільну, гнучку, безпечну та легко керовану мережу [11].

Крім того, NEC впровадила мережу OpenFlow на своїй фабриці програмного забезпечення в Японії, яка підтримує хмарне середовище розробки програмного забезпечення. Розгортання призвело до створення внутрішнього центру обробки даних, переваг SDN і можливості перемикатися між віртуальними серверами в приміщеннях Східної та Західної Японії. Розгортання всієї інфраструктури було досягнуто лише протягом 10 днів, тоді як орієнтовний час розгортання з традиційним обладнанням зайняв би близько двох місяців [12]. Крім того, NEC також розгорнула мережу OpenFlow у компанії Nippon Express Co., Ltd., Японія, пропонуючи їм хмарну мережу для підтримки їх операцій по всьому світу [13].

Висновок

В цьому розділі було визначено основні компоненти мережі OpenFlow, а саме комутатори з підтримкою OpenFlow та сервери, на яких працюють контролери. Також було розглянуто концепцію централізованого контролю над мережею та поділу площини керування та даних. Централізована концепція контролю над мережею та розділ площини керування та даних визначаються в мережевих пристроях. Площина даних і площина керування розділяються для ефективного пересилання пакетів. Протокол OpenFlow не приймає самостійно рішень щодо пересилання пакетів, це завдання виконує контролер. Комутатори OpenFlow переглядають таблиці потоків і виконують встановлені рішення щодо пересилання пакетів на основі збігу полів заголовків. В разі відсутності відповідності пакет пересилається до контролера для подальшої обробки. Контролер, в залежності від типу потоку, переглядає свою базу даних і надсилає відповідні пакети назад до комутаторів, встановлюючи відповідні потоки у таблицях потоків

комутаторів. Таким чином, контролер OpenFlow відповідає за прийняття рішень щодо пересилання пакетів у мережі.

РОЗДІЛ 3.

ОЦІНКА ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ ПРОТОКОЛУ OPENFLOW В ЛОКАЛЬНИХ МЕРЕЖАХ

3.1. Лабораторна установка

Експерименти проводилися в лабораторії мереж і протоколів кафедри інженерії зв'язку. Лабораторія включає різноманітні концентратори, комутатори, маршрутизатори та брандмауери від трьох популярних постачальників, а саме HP, Cisco та Juniper Networks. У лабораторії встановлено чотири стійки з цими пристроями та кілька робочих станцій Linux з операційною системою CENTOS. Для аналізу протоколу OpenFlow використовувалися чотири комутатори серії HP 3800, які підтримують протокол OpenFlow v1.0. Дві основні функції, які специфікації протоколу OpenFlow v1.0 не підтримують, це MPLS і IPv6, які є більш важливими для розгортання комерційної мережі. Робочі станції Linux, доступні в лабораторії, використовувалися як клієнти, а одна з робочих станцій Linux діяла як контролер, на якому запускався контролер Floodlight. Робочі станції Linux оновлено плагіном Wireshark Dissector для протоколу OpenFlow і необхідними утилітами для аналізу.

3.1.1. Комутатори HP

Комутатори HP було обрано, оскільки HP бере участь у розробці протоколу OpenFlow з 2007 року та реалізувала протокол OpenFlow v1.0 у більшій кількості продуктів порівняно з іншими продуктами. постачальники. Іншою причиною використання комутаторів HP було те, що HP пропонувала свої комутатори для експериментальних цілей на короткий період у два

місяці, і тому комутатори HP були природним вибором для проведення експериментів OpenFlow.

У лабораторії були встановлені комутатори HP рівня 3 HP 3800-24G-2XG (інформація про модель J9585A). Прошивку цих комутаторів було оновлено до останньої версії KA.15.10.0003, оскільки протокол OpenFlow підтримується прошивкою KA.15.10 і вище. Оновлене мікропрограмне забезпечення підтримує протокол OpenFlow версії 1.0 у стані бета-версії, деякі функції, зазначені у версії 1.0, включено, а деякі ні. Ці комутатори дозволяють автономне розгортання мережі OpenFlow як а також розгортання в гібридному режимі, тобто OpenFlow і традиційна мережа можуть працювати поруч на одному пристрої.

Мережа OpenFlow може бути розділена на кілька екземплярів (макс. 128) і кілька VLAN (макс. 2048) для цілей ізоляції та ідентифікації [42]. Примірник OpenFlow активується шляхом додавання до нього порту слухання або контролера.

3.1.2. Архітектура

Враховуючи розгортання мережі OpenFlow, комутатори HP архітектурно класифікуються на два режими, а саме режим агрегації та віртуальний режим. VLAN контролера та керування не базуються на OpenFlow і мають подібну структурну позицію в архітектурі. В обох архітектурах можна підключити більше одного контролера для роботи в режимі очікування, балансування навантаження та незалежних операцій.

- **Режим агрегації**

У режимі агрегації всі порти комутатора працюють у режимі OpenFlow, і комутатор не підтримує традиційний мережевий трафік, який не має тегів OpenFlow. Він розглядає всі порти та VLAN, визначені там, в один екземпляр

OpenFlow. Архітектуру режиму агрегації можна побачити на Рис.3.1 нижче. [42]

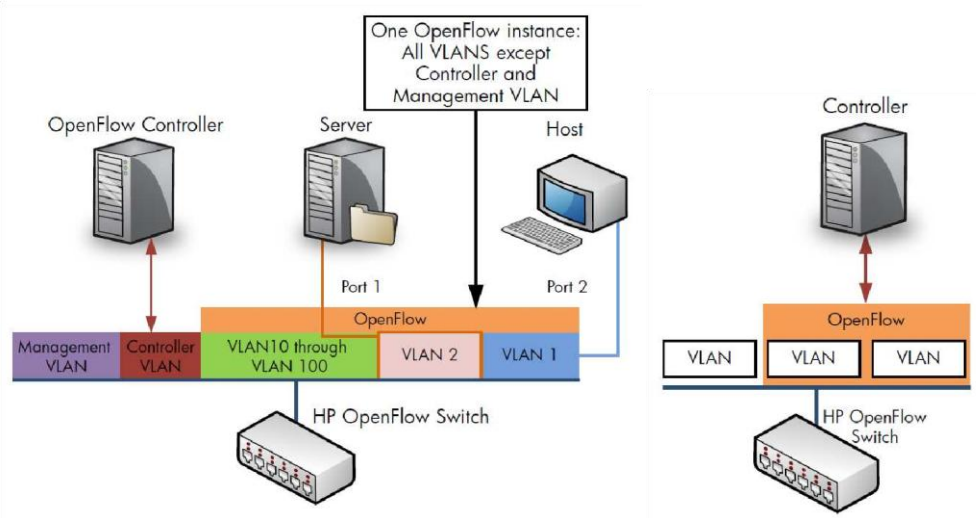

Рис.3.1. Режим агрегації в комутаторах HP [42].

- **Режим віртуалізації**

У режимі віртуалізації працює гібридний режим, виробництво та трафік Open-Flow проходять поруч. Режим віртуалізації підтримує кілька екземплярів для мережі OpenFlow, і членство кожного екземпляра визначається членством у групі VLAN [42]. У конфігурації мережі ті VLAN, які не є членами екземпляра OpenFlow, не є частиною мережі OpenFlow і передають робочий мережевий трафік. Кожен екземпляр є незалежним, має власну конфігурацію OpenFlow і окреме підключення контролера, що забезпечує більш розподілений режим роботи. Архітектуру режиму віртуалізації можна побачити на Рис.3.2.

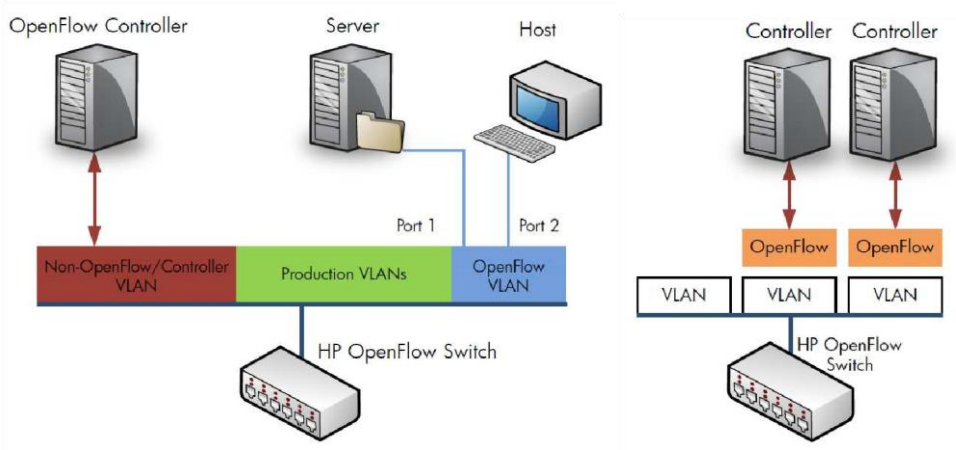

Рис.3.2. Режим віртуалізації в комутаторах HP [42].

Обмеження для режиму віртуалізації включають використання лише однієї VLAN для екземпляра Open-Flow, і та сама VLAN не може підтримувати членство для будь-якого іншого екземпляра. Просто,

кількість екземплярів openflow = кількість VLAN у мережі з відкритим потоком (3.1)

3.1.3. Режими роботи

У цьому розділі описано різні режими роботи екземплярів OpenFlow [42].

- Активний режим: в активному режимі пакети, які не збігаються з жодним потоком у таблиці потоків комутатора, надсилаються до контролера OpenFlow для подальших дій.
- Пасивний режим: на відміну від активного режиму, тут пакети, які не збігаються з жодним потоком, обробляються самим комутатором, а не надсилаються до контролера OpenFlow і чекають рішення звідти.
- Захищений від збоїв режим: у цьому режимі, якщо комутатор втратив з'єднання з контролером OpenFlow, усі пакети та повідомлення OpenFlow, призначені контролеру, видаляються.

- Автономний режим при збої: у цьому режимі, якщо комутатор втратив зв’язок із контролером OpenFlow, нові потоки поводяться подібно до застарілого комутатора чи маршрутизатора, а існуючі потоки екземпляра OpenFlow видаляються. Комутатор більше не переглядає таблицю потоків OpenFlow і працює традиційним способом. Цей режим буде корисним для розробки традиційної резервної мережі та активуватиметься, коли з’єднання контролера буде перервано, і в таких випадках пом’якшить відключення мережі.

В активному режимі комутатор консультується з контролером через повідомлення Packet-In, і контролер відповідає рішенням через повідомлення PacketOut. Зазвичай приймається рішення або про встановлення потоку у разі реактивних потоків, якщо знайдено відповідність потоку, або про трансляцію пакетів. Пакети, що транслюються, в кінцевому підсумку досягають місця призначення, але загальна пропускна здатність знижується, що буде виявлено більш детально в проведених експериментах. Однак у комутаторах HP повідомлення Packet-In та Packet-Out можна зупинити, активувавши пасивний режим, але з іншого боку зв’язок втрачається. Навпаки, в активному режимі мережа знаходиться принаймні в робочому стані зі зниженою пропускною здатністю.

3.1.4. Розташування потоку для OpenFlow

У цьому розділі описано апаратні та програмні потоки для OpenFlow, а також їх операційний пріоритет і виконання. Вони називаються на честь місця, де відбувається виконання потоку.

- Апаратні потоки: ці потоки запрограмовані лише в апаратному забезпеченні, і ті потоки, які мають лише дію для пересилання на порт, є апаратними потоками.

- Потоки програмного забезпечення: ці потоки запрограмовані в програмному забезпеченні, і ті потоки, які мають дії зі змінення, наприклад зміна IPадреси джерела чи призначення тощо, класифікуються як потоки програмного забезпечення.

Потік має бути ретельно запрограмований, щоб увімкнути апаратні чи програмні операції. Якщо потік має кілька дій, включаючи пересилання апаратних дій, і якщо будь-яка з дій підтримується програмним забезпеченням, виконання потоку відбуватиметься лише програмним забезпеченням. Виконання потоку в апаратному забезпеченні відбуватиметься лише тоді, коли параметри та дії відповідності потоку підтримуються в апаратному забезпеченні, і він не має жодних додаткових програмних дій [42][42].

3.1.5. Інструменти та утиліти Linux

У цьому розділі описано різні інструменти та утиліти Linux, які використовувалися для проведення експериментів у лабораторії.

- Curl: це інструмент командного рядка для передачі даних із синтаксисом URL, який підтримує сертифікати SSL, HTTP та інші протоколи [43]. Він використовується для вставлення потоків у API входу статичного потоку для контролера Floodlight, і приклад конфігурації можна побачити в розділі 4.2.
- Avior: це програма на основі Java, що пропонує графічний інтерфейс користувача для адміністрування потоку, логічної панелі патчів і статистики в реальному часі для контролера, комутатора, пристрою та портів [44]. Подібно до curl, він також використовується для вставлення потоків в API запису статичного потоку для контролера Floodlight.

- Iperf: це утиліта командного рядка для вимірювання максимальної пропускної здатності TCP і UDP, а також повідомляє про тремтіння затримки та втрату дейтаграм. Це також дозволяє налаштовувати різні параметри [45].
- Netperf: це інструмент порівняльного аналізу, який використовується для вимірювання різних аспектів продуктивності мережі, включаючи односпрямовану передачу даних і продуктивність запитів або відповідей за допомогою TCP або UDP [46].

3.1.6. Контролер Flooflight

Floodlight — це контролер OpenFlow корпоративного класу з відкритим вихідним кодом на основі Java з ліцензією Apache, який зараз розробляється та підтримується Big Switch Networks [47]. Потоки додаються до контролера за допомогою

REpresentational State Transfer (REST) Application Programming Interface (API) під назвою Static Flow Pusher, який є інтерфейсом JavaScript Object Notation (JSON) для налаштування проактивних потоків у контролері Floodlight [48]. Потоки вставляються, видаляються та перезаписуються в контролер Floodlight за допомогою команди curl, щоб досягти об’єкта Static Flow Entry Pusher. Іншою популярною утилітою для додавання потоків до контролера Floodlight є Avior, яка є програмою графічного інтерфейсу користувача (GUI) на основі Java, яка пропонує адміністрування потоків і статистику в реальному часі, яка часто оновлюється [44]. Floodlight пропонує додавати як проактивні, так і реактивні потоки до комутатора з увімкненим OpenFlow. У проактивному потоці запис потоку надсилається та зберігається в таблиці потоків комутатора з увімкненим OpenFlow до надходження будь-якого трафіку. На відміну від проактивних потоків, у реактивному потокі запис потоку не надсилається миттєво до таблиці потоків комутатора, а

залишається в контролері OpenFlow. Потік буде вставлено в таблицю потоків комутатора лише тоді, коли пакет надходить до комутатора; він не збігається з жодним потоком і надсилається до контролера OpenFlow для подальшого прийняття рішення. Контролер OpenFlow перевіряє свої записи потоку, приймає рішення, надсилає пакет назад комутатору та вставляє запис потоку в таблицю потоків комутатора.

Кожен комутатор OpenFlow, підключений до контролера OpenFlow, ідентифікується унікальним ідентифікатором шляху даних (DPID). DPID має довжину 8 байтів і вказується як десяткове число або як 8 шістнадцяткових (HEX) октетів, наприклад, 00:00:00:23:10:35:ce:a5. DPID ff:ff:ff:ff:ff:ff:ff — це DPID із символом підстановки, який відповідає всім DPID у мережі Open-Flow. Контролер Floodlight пропонує веб-інтерфейс, показаний на Рис.3.3:

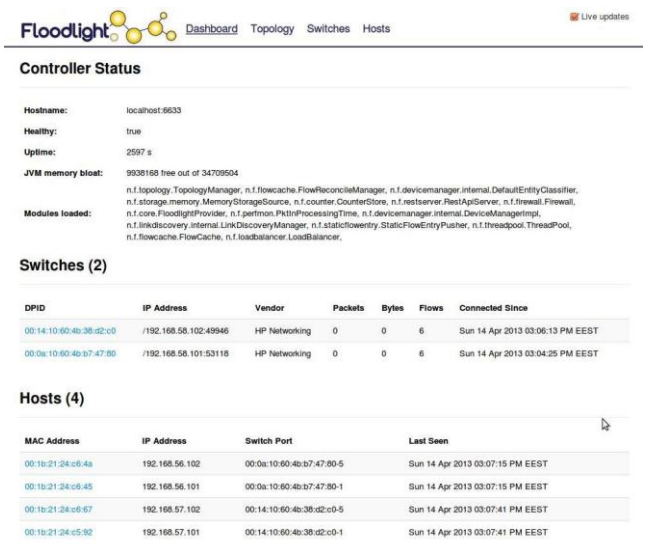

Рис.3.3. Веб-інтерфейс Floodlight

На основі доступних даних про пристрої та хости, підключені до нього, він малює візуальну діаграму, що показує, який хост підключено до якого комутатора та як комутатори підключені до контролера. Веб-інтерфейс

корисний лише для аналізу; тому конфігурацію потоку або додаткову конфігурацію не можна викликати з вебінтерфейсу.

Контролер Floodlight забезпечує проактивні потоки, тим самим зменшуючи навантаження на контролер і час налаштування потоку. Він відповідає на максимальну кількість запитів, що надходять до нього в секунду, але, з іншого боку, він має більшу затримку, ніж контролер Trema. Потоки можна легко встановити за допомогою API штовхача статичного потоку або утиліт третього партнера Curl і Avior, про які йшлося вище. Він заснований на Java, тому його можна розгортати та керувати з кількох платформ, включаючи Android. Він був випущений на початку 2012 року, але розвинувся з контролера Beacon — найпершого контролера OpenFlow, розробленого в Стенфордському університеті на початку 2010 року [49]. Враховуючи проактивні потоки, історію його розробки, активний список розсилки та простіший спосіб встановлення потоку, було обрано контролер Floodlight.

3.2. Експеримент 1: Базове налаштування всередині однієї підмережі

Цей експеримент було проведено для перевірки базової роботи протоколу OpenFlow. Топологію мережі можна побачити на Рис.3.4, де два хости з мережі 192.168.56.0/24 є функціональними вузлами мережі OpenFlow. Комутатор OpenFlow налаштовано в режимі агрегації.

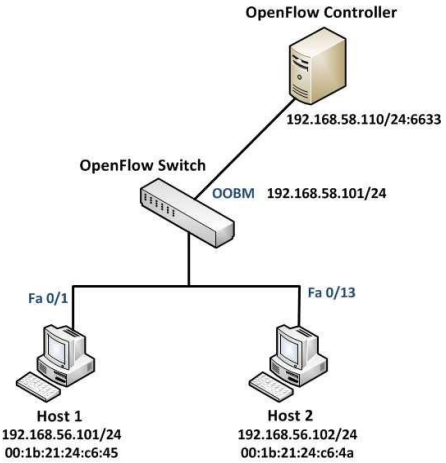

Рис.3.4. Топологія мережі для експерименту 1

3.2.1. Записи потоку

Для цього експерименту контролер OpenFlow Floodlight було налаштовано для створення та встановлення проактивних потоків на комутатор OpenFlow для хостів у мережі. Загалом було налаштовано два потоки: один від Host1 до Host2, а другий – навпаки, що можна побачити в таблиці 3.1 нижче.

Таблиця 3.1.

Потоки для експерименту 1

Flow #	**Matching criterion**	**Actions**
1	in_port 1, dst_ip 192.168.56.102	out_port 13
2	in_port 13, dst_ip 192.168.56.101	out_port 1

У наведеній вище таблиці in_port стосується вхідного порту, dst_ip — IP-адреси призначення, а out_port — вихідного порту. Потоки встановлено на комутатор HP, але спочатку налаштовано на контролер OpenFlow за допомогою утиліти Curl, і приклад конфігурації можна побачити тут. Під час налаштування потоку комутатор ідентифікується своїм унікальним ідентифікатором із восьми шістнадцяткових октетів, відомим тут як DPID.

```
curl -d '{"switch":"10:00:10:60:4b:b7:47:80", "name":"flow1",
"active":"true", "priority":"32570","actions":"output=13","ether-
type":"0x0800","ingress-port":
"1","dst-ip":"192.168.56.102"}'
http://192.168.58.110:8080/wm/staticflowentrypusher/json
```

Контролер Floodlight підтримує проактивні потоки і миттєво встановлює потоки в комутатор HP. Один з потоків з назвою "flow1" з'являється у таблиці потоків комутатора HP, як показано на Рис.3.5.

```
switch2(config)# show openflow instance aggregate flows

 OpenFlow Flow Table

 Flow 1
  Incoming Port  : 1                  Ethernet Type     : IP
  Source MAC     : 000000-000000      Destination MAC   : 000000-000000
  VLAN ID        : 0                  VLAN priority     :
  Source IP      : 0.0.0.0            Destination IP    : 192.168.56.102
  IP Protocol    : 0x00               IP ToS Bits       : 0
  Source Port    : 0                  Destination Port  : 0
  Priority       : 32570
  Duration       : 142 seconds
  Idle Timeout   : 0 seconds          Hard Timeout      : 0 seconds
  Packet Count   : 407920             Byte Count        : 0
  Flow Location  : Hardware
  Actions
    Output                  : 13
```

Header or match fields
Counters
Actions

Рис.3.5. Потік у проточній таблиці в перемикачі HP

3.2.2. Результати

Після налаштування потоків через мережу було надіслано ping-повідомлення ICMP для перевірки з’єднання, і тести виявили працездатність мережі OpenFlow. Запис Wireshark на контролері OpenFlow допоміг у подальшому аналізі кроків протоколу Open-Flow: перемикання на

підключення контролера, потоки, встановлені для комутатора (модифікація комутатора) і моніторинг трафіку OpenFlow.

Оскільки хости знаходяться в одній підмережі, вони можуть зв’язуватися з іншими без будь-яких потоків, налаштованих у мережі OpenFlow. Однак у таких випадках пропускна здатність між хостами або загальна пропускна здатність мережі буде зменшена. Якщо в комутаторі немає ні доступних потоків, ні відповідних потоків для з’єднання в комутаторі, контролер контактів комутатора визначає, які дії слід виконувати з трафіком за допомогою повідомлення Packet-In. Враховуючи запит ping ICMP, повідомлення Packet-In проходить через контролер, а контролер надсилає повідомлення Packet-Out із самим пакетом і пов’язаним із ним рішенням (у цьому випадку переповнення пакетів), і, нарешті, пакет досягає вузла призначення через переповнення.

Повідомлення Packet-In та Packet-Out інкапсулюються в заголовок пакета протоколу OpenFlow і фіксуються як протокол OFP+ICMP у Wireshark. Ці повідомлення та процес у цілому можна побачити на Рис.3.6 і 3.7 відповідно, і зазвичай усі інкапсульовані повідомлення відображаються як повідомлення OFP+ у Wireshark. У таких випадках диссектор Wireshark визначає фактичну IP-адресу джерела та призначення, але фактично пакет переходить від комутатора до контролера для консультації. Розглянувши Рис.3.6, виконується запит ping ICMP від хосту 1 до хосту 2; він іде від порту OOBM комутатора з підтримкою OpenFlow (192.168.58.101) до контролера, і його можна визначити з полів IPv4 пакету.

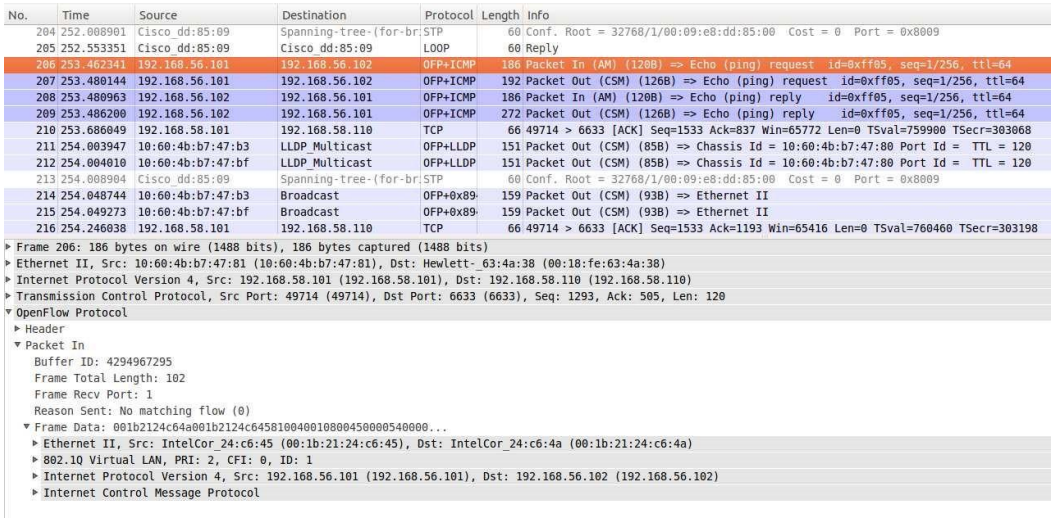

Рис.3.6. Повідомлення Packet-In, коли потоки не визначено

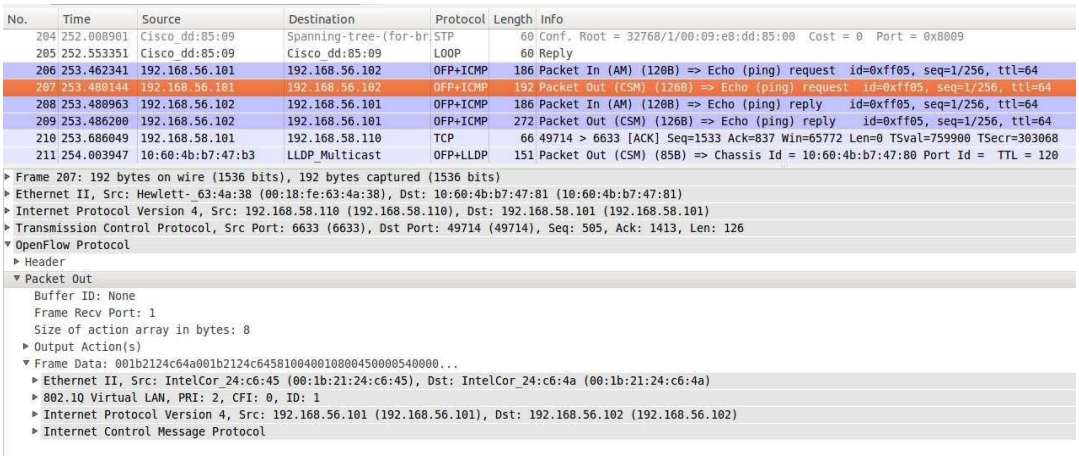

Рис.3.7. Повідомлення Packet-Out, коли потоки не визначено

Він інкапсулює оригінальний пакет ICMP від Host1 до Host2 у звичайний пакет OpenFlow і надсилає повідомлення Packet-In, яке надходить від комутатора, увімкненого OpenFlow. З рисунків видно, що за відсутності потоків контролер виконує флуд, що знижує загальну пропускну здатність мережі OpenFlow. Крім того, очевидно, що навіть незважаючи на те, що потоки визначені, а підключення до мережі зберігається в мережі OpenFlow, продуктивність потоку слід визначати, дивлячись на лічильники таблиці

потоків і пропускну здатність. Лічильники таблиці потоків, такі як кількість пакетів, кількість байтів, можна визначити, переглянувши таблицю потоків комутатора HP, як показано на Рис.3.5. Запит ping ICMP може забезпечувати з'єднання через широкомовні пакети контролером, але лічильники потоку та результати пропускної здатності покажуть реальність. Обидва ці механізми лічильників потоку та аналізу пропускної здатності застосовувалися в усіх експериментах, проведених у лабораторії, для отримання оптимальних результатів. Результати пропускної спроможності були обчислені за допомогою утиліт iperf і netperf для потоків TCP і UDP, однак пропускна здатність TCP і UDP з обома утилітами була однакова. Результат можна побачити в таблиці 3.2 нижче, яка показує різке падіння пропускної здатності мережі, якщо немає жодних потоків або вони не налаштовані належним чином.

Таблиця 3.2.

Пропускна здатність для експерименту 1

Situation	**Network throughput**
Before deploying OpenFlow network	944 Mbits/sec
No flows in OpenFlow network	367 Kbits/sec
Flows in OpenFlow network	944 Mbits/sec

3.3. Експеримент 2 : Перевірка дій модифікацій в потоці

Цей експеримент було проведено, щоб вивчити та перевірити більше про потоки програмного забезпечення. Потоки програмного забезпечення включають дії, відмінні від переадресації на порт, і здійснюються повідомленням про зміни дій (Flow-Mod). Дії зміни включають зміну інформації рівня 2, рівня 3 і рівня 4, наприклад зміну MACадреси або IP-адреси джерела та призначення, зміну порту джерела та призначення TCP. Ці потоки корисні в таких ситуаціях, як NAT, переадресація на основі MAC і

потік передачі від одного комутатора до іншого комутатора тощо. Топологію мережі можна побачити на Рис.3.8, де три хости з мережі 192.168.56.0/24 є функціональними вузлами в мережа OpenFlow. Перемикач

OpenFlow налаштовано в режимі агрегації, і сценарій включає перенаправлення всього трафіку, що надходить на хост 2, на хост 3.

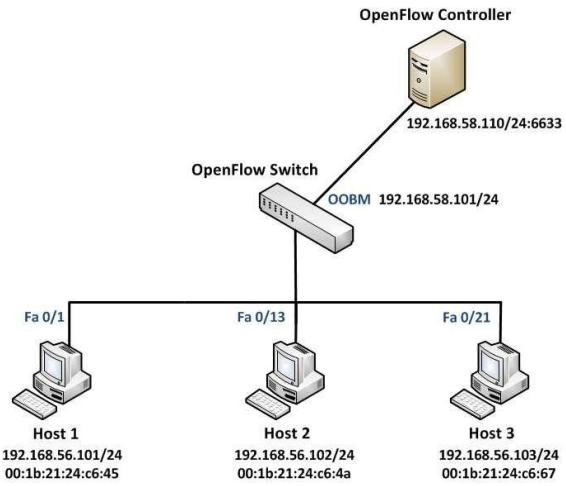

Рис.3.8. Топологія мережі для експерименту 2

3.3.1. Записи потоку

Для цього експерименту було налаштовано загалом три потоки: один від Host1 до Host2, другий навпаки і третій від Host3 до Host1, які можна побачити в таблиці 3.3 нижче.

Таблиця 3.3.

Потоки для експерименту 2

Flow #	**Matching criterion**	**Actions**
1	in_port 1, dst_ip 192.168.56.102	out_port 21, set_dst_ip 192.168.56.103
2	in_port 13, dst_ip 192.168.56.101	out_port 1
3	in_port 21, dst_ip 192.168.56.101	out_port 1

3.3.2. Результати

Результати ping-запиту ICMP, iperf і netperf виявили функціональну мережу OpenFlow, яка перенаправляла весь вхідний трафік на Host3, який був спрямований на досягнення Host2. Результати пропускної здатності подібні до результатів експерименту 1 і їх можна побачити в таблиці 3.4 нижче.

Таблиця 3.4.

Пропускна здатність для експерименту 2

Situation	**Network throughput**
Before deploying OpenFlow network	944 Mbits/sec
No flows in OpenFlow network	336 Kbits/sec
Flows in OpenFlow network	944 Mbits/sec

3.4 Експеримент 3: VLAN у мережі OpenFlow

Цей експеримент було проведено, щоб дослідити інтеграцію та поведінку VLAN у мережі OpenFlow. Розглядаються дві VLAN з мережею 192.168.56.0/24 і 192.168.57.0/24, і кожна VLAN має два хости всередині себе. Кожна VLAN налаштована на окремий комутатор, і комутатори спочатку налаштовані в режимі віртуалізації, а потім також було протестовано режим агрегації. І режими агрегації, і режими віртуалізації дали однакову статистику продуктивності та пропускної здатності. Топологію мережі в режимі агрегації можна побачити на Рис.3.9, тоді як у режимі віртуалізації відбувається додавання ще одного контролера OpenFlow, оскільки для

кожного екземпляра в режимі віртуалізації потрібен окремий контролер, що було виконано далі в розділі 3.5.

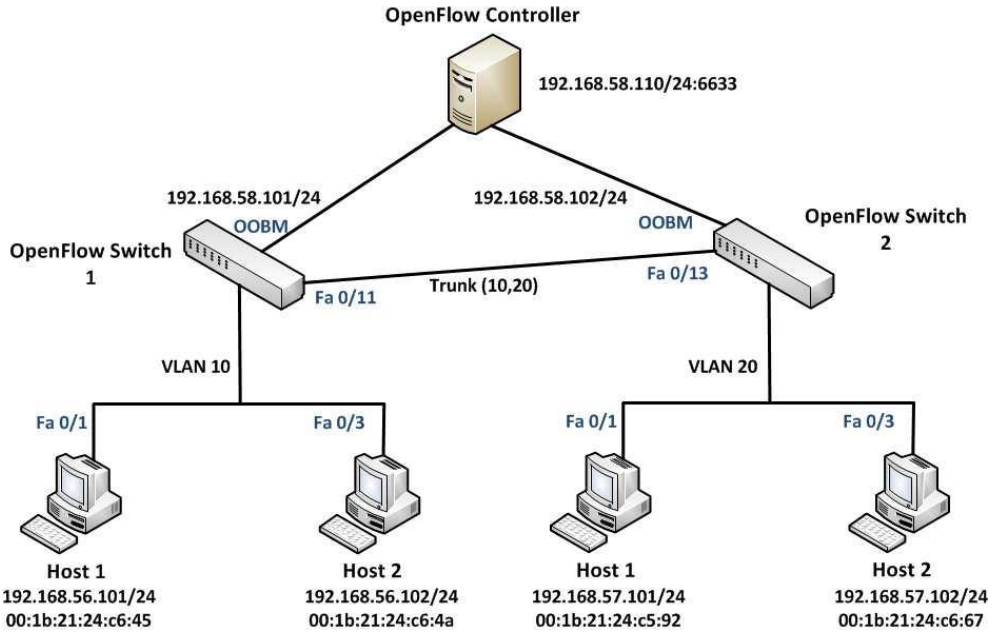

Рис.3.9. Топологія мережі для експерименту 3

3.4.1.Записи потоку

Для цього експерименту було налаштовано загалом шість потоків для кожного комутатора: один від Host1 до Host2 у тій самій VLAN, другий навпаки, один потік для кожного хоста для досягнення хостів в іншій VLAN, а останні два потоки навпаки, тобто для досягнення хостів VLAN 10 з VLAN 20 у комутаторі 1. Потоки для комутатора 1 і комутатора 2 можна побачити в таблиці 3.5 і таблиці 3.6 відповідно.

Таблиця 3.5.

Потоки для перемикача 1 для експерименту 3

Flow #	**Matching criterion**	**Actions**
1	in_port 1, dst_ip 192.168.56.102	out_port 3
2	in_port 3, dst_ip 192.168.56.101	out_port 1
3	in_port 1, dst_ip 192.168.57.0	out_port 11
4	in_port 3, dst_ip 192.168.57.0	out_port 11
5	in_port 11, dst_ip 192.168.56.101	out_port 1

6	in_port 11, dst_ip 192.168.56.102	out_port 3

Таблиця 3.6.

Потоки для перемикача 2 для експерименту 3

Flow #	**Matching criterion**	**Actions**
1	in_port 1, dst_ip 192.168.57.102	out_port 3
2	in_port 3, dst_ip 192.168.57.101	out_port 1
3	in_port 1, dst_ip 192.168.56.0	out_port 13
4	in_port 3, dst_ip 192.168.56.0	out_port 13
5	in_port 13, dst_ip 192.168.57.101	out_port 1
6	in_port 13, dst_ip 192.168.57.102	out_port 3

3.4.2.Результати

Результати запиту ping ICMP, iperf і netperf виявили функціональну VLAN для кожного комутатора в мережі OpenFlow. Однак зв’язок між VLAN не вдалося встановити, коли хост з однієї VLAN намагався зв’язатися один з одним хостом в іншій VLAN. Основна причина включала пошук шлюзу за замовчуванням, який можна побачити на Рис.3.10.

У топології мережі не було жодного маршрутизатора, який міг би вирішити цю проблему, хоча комутатор HP був комутатором рівня 3, а VLAN було налаштовано перед увімкненням OpenFlow. Оскільки OpenFlow переглядає таблиці потоків, а не конфігурацію комутатора, тому конфігурація VLAN у комутаторі HP не може вирішити проблему. Рішення включають підключення маршрутизатора до мережі або використання RouteFlow, які були протестовані в Розділі 3.8 і Розділі 3.9 відповідно.

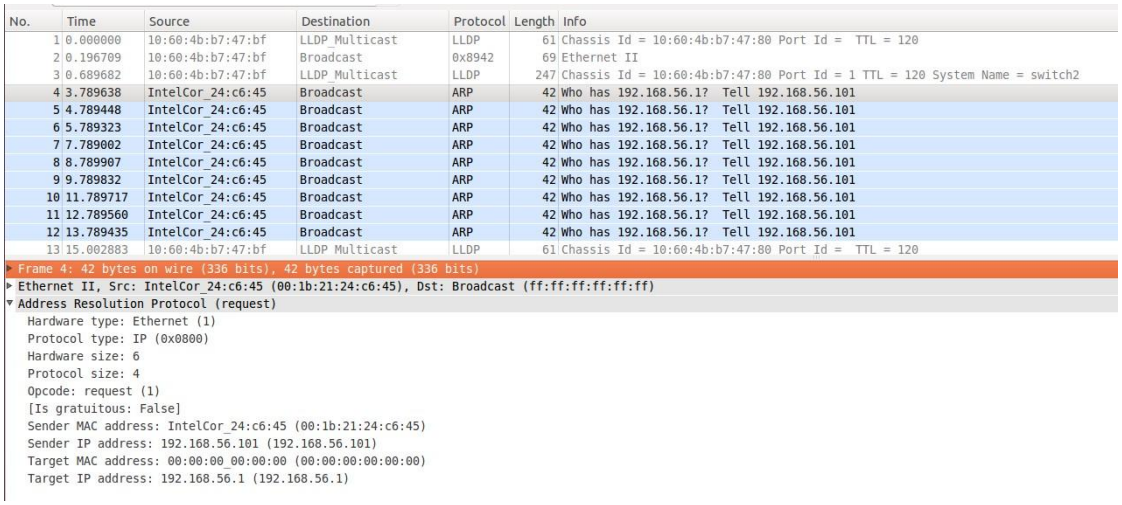

Рис.3.10. Проблема зі шлюзом за замовчуванням в експерименті 3

У межах VLAN потоки працювали належним чином, а результати пропускної здатності підтверджують їх стабільну роботу. Результати пропускної здатності можна побачити в таблиці 3.7 нижче.

Таблиця 3.7.

Пропускна здатність для експерименту 3

Situation	**Network throughput**
Before deploying OpenFlow network	944 Mbits/sec
No flows in OpenFlow network	383 Kbits/sec
Flows in OpenFlow network	944 Mbits/sec

3.5. Експеримент 4:відмовостійкість у разі втрати контролера

Цей експеримент було проведено, щоб дослідити автономний режим відмов для підключення контролера в комутаторах HP OpenFlow.

Автономний режим відмов стане в нагоді у випадках, коли з’єднання контролера порушується, і мережа стикається з єдиною точкою збою. Більш ніж один контролер може бути розгорнутий і виконувати функцію балансування навантаження або розподілу функціональних можливостей, і

такого нарізання та розподілу можна досягти за допомогою FlowVisor [17][50], однак інші рішення включають налаштування комутатора HP у автономному режимі збоїв і уникнення поломка мережі при виході з ладу контролера. Враховуючи топологію мережі для експерименту 4, показану на Рис.3.11, тут додається ще один контролер, і кожен комутатор налаштований у режимі віртуалізації. У кожному комутаторі створюється екземпляр, який має члени VLAN 10 і VLAN 20 відповідно з активованим автономним режимом відмов. Комутатори HP належать до комутаторів рівня 3, вони також можуть керувати IP-маршрутизацією, і в обох з них виконано традиційну конфігурацію VLAN.

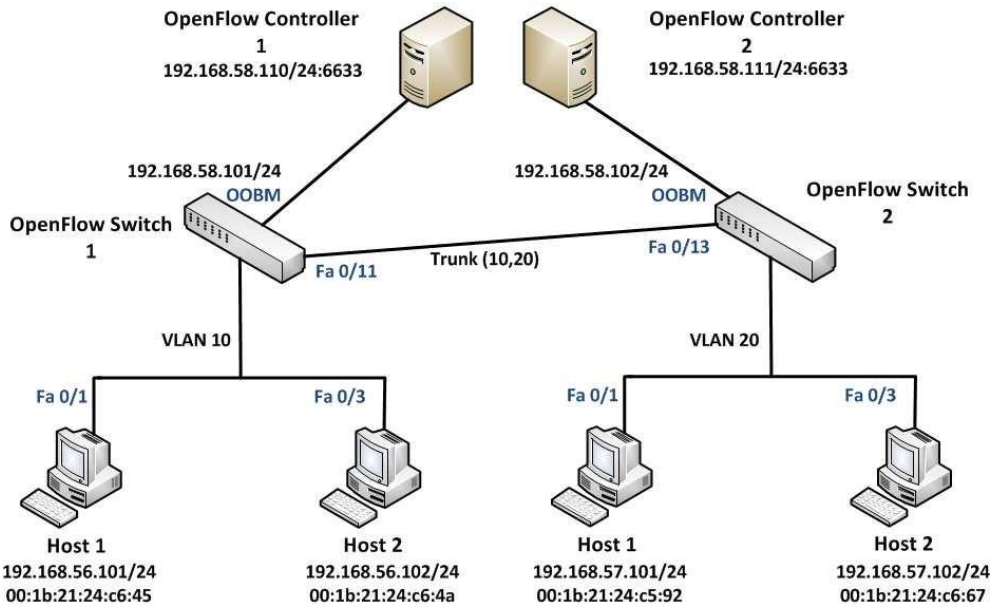

Рис.3.11. Топологія мережі для експерименту 4

3.5.1. Записи потоку

Для цього експерименту записи потоку подібні до тих, що визначені в експерименті 3, і їх можна побачити в таблиці 3.8 і таблиці 3.9 для перемикача 1 і перемикача 2 відповідно.

Таблиця 3.8.

Потоки для перемикача 1 в експерименті 4

Flow #	**Matching criterion**	**Actions**
1	in_port 1, dst_ip 192.168.56.102	out_port 3
2	in_port 3, dst_ip 192.168.56.101	out_port 1
3	in_port 1, dst_ip 192.168.57.0	out_port 11
4	in_port 3, dst_ip 192.168.57.0	out_port 11
5	in_port 11, dst_ip 192.168.56.101	out_port 1
6	in_port 11, dst_ip 192.168.56.102	out_port 3

Таблиця 3.9

Потоки для перемикача 2 в експерименті 4

Flow #	**Matching criterion**	**Actions**
1	in_port 1, dst_ip 192.168.57.102	out_port 3
2	in_port 3, dst_ip 192.168.57.101	out_port 1
3	in_port 1, dst_ip 192.168.56.0	out_port 13
4	in_port 3, dst_ip 192.168.56.0	out_port 13
5	in_port 13, dst_ip 192.168.57.101	out_port 1
6	in_port 13, dst_ip 192.168.57.102	out_port 3

3.5.2. Результати

З режимом віртуалізації та конфігурацією VLAN мережа OpenFlow спочатку була протестована з активним підключенням контролера OpenFlow за допомогою ping, iperf і netperf для кожної VLAN. Однак зв’язок між мережами VLAN не встановлено через проблему зі шлюзом за замовчуванням, описану в розділі 3.4. Пізніше контролер OpenFlow було зупинено, і активовано автономний режим, який припиняє перегляд таблиць потоку Open-Flow. Він починає перевіряти конфігурацію кожного комутатора, тобто традиційну таблицю переадресації в кожному комутаторі, і підключення до мережі відновлюється приблизно через 4 секунди.

Комутатори все ще перебувають у режимі OpenFlow і надсилають повідомлення синхронізації TCP зі свого порту OOBM до контролера, але вони не можуть отримати повідомлення підтвердження синхронізації TCP, які можна побачити на Рис.3.12. Такі ж повідомлення передаються, коли з'єднання з контролером розривається, однак у цьому режимі трафік пересилається з використанням власної конфігурації та таблиці маршрутизації кожного комутатора, а не в очікуванні контролера OpenFlow.

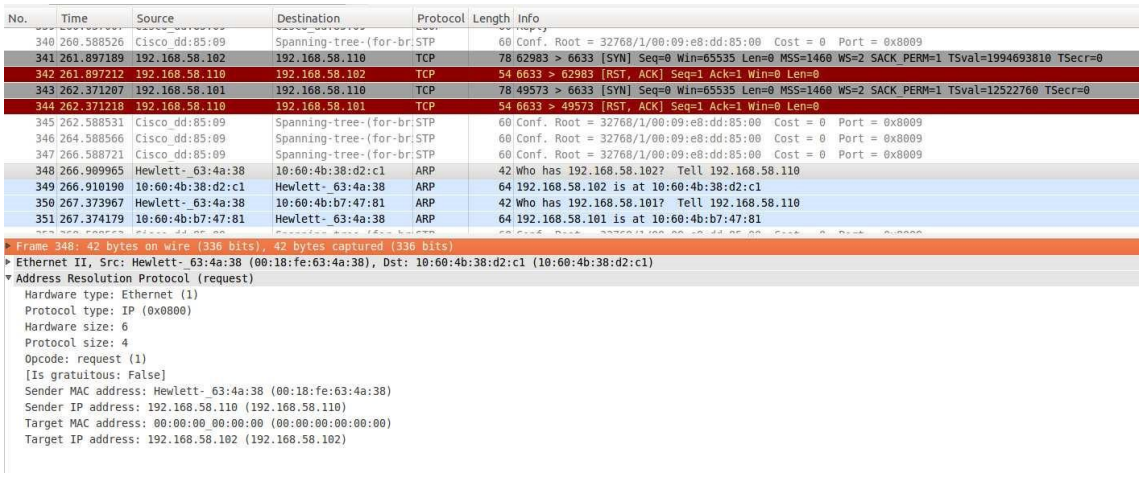

Рис.3.12. Пошук перемикачів контролера в аварійному режимі

Після відновлення підключення до мережі зв'язок між VLAN було встановлено за допомогою традиційної конфігурації VLAN комутатора, а пропускна здатність залишається на рівні 944 Мбіт/с. Автономний режим відмов також може бути ефективним у конфігурації режиму агрегації, якщо необхідна конфігурація знаходиться в комутаторі. Пропускну здатність, досягнуту в експерименті 4, можна побачити в таблиці 3.10.

Таблиця 3.10.

Пропускна здатність для експерименту 4

Situation	**Network throughput**
Before deploying OpenFlow network	944 Mbits/sec
No flows in OpenFlow network	343 Kbits/sec
Flows in OpenFlow network	944 Mbits/sec
Controller connection breaks in OpenFlow network	944 Mbits/sec

Величина затримки (таймаут становить 4 секунди) значно висока в сучасних мережах, але це принаймні кращий варіант, ніж повний розрив мережі.

3.6 Експеримент 5: OpenFlow у гібридному режимі

Цей експеримент було проведено, щоб дослідити роботу OpenFlow у гібридному режимі, тобто мережа OpenFlow працюватиме паралельно з традиційною мережею. Комутатор HP налаштовано в режимі віртуалізації для підтримки гібридного режиму, а мережа включає три різні мережі VLAN: VLAN 10, VLAN 20 і VLAN 30. VLAN 10 містить мережу OpenFlow, тоді як дві інші VLAN призначені для традиційної мережі, що не підтримує OpenFlow. Топологію мережі для експерименту 5 можна побачити на Рис.3.13:

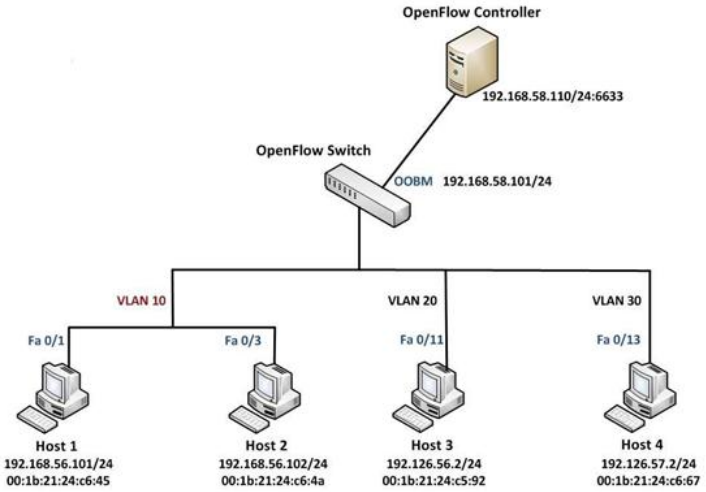

Рис.3.13. Топологія мережі для експерименту 5

3.6.1. Записи потоку

Для цього експерименту є лише два записи потоку для мережі OpenFlow, тобто VLAN 10, яку можна побачити в таблиці 3.11 нижче.

Таблиця 3.11.

Потоки для експерименту 5

Flow #	Matching criterion	Actions
1	in_port 1, dst_ip 192.168.56.102	out_port 3
2	in_port 3, dst_ip 192.168.56.101	out_port 1

3.6.2. Результати

Мережа OpenFlow була протестована за допомогою утиліт ping, iperf та netperf і показала очікувану пропускну здатність 941 Мбіт/с. Традиційні мережі VLAN 20 і VLAN 30 можуть зв'язуватися між собою, але не можуть зв'язатися з мережею OpenFlow. І OpenFlow, і традиційна мережа працюють паралельно, зберігаючи при цьому пропускну здатність мережі. Крім того, ця конфігурація була протестована в автономному режимі, і вона також

показала безперебійну роботу. Цей експеримент продемонстрував інтеграцію OpenFlow з традиційним мережевим середовищем.

3.7 Результати дослідження

Для кожного експерименту було налаштовано потоки, а потім за допомогою ICMP ping-повідомлень визначено встановлення з'єднання. Після цього переглядалися лічильники потоків у таблиці потоків комутатора та проводився аналіз пропускної здатності, щоб визначити, чи правильно налаштовані потоки та чи пересилаються пакети за допомогою потоків. Всі потоки були налаштовані та встановлені на комутатори через контролер OpenFlow. SDN має на меті розробку програмованої мережі, однак встановлення потоків у мережі OpenFlow на даному етапі є ручною роботою. Контролер Floodlight відстежує підключені до нього комутатори та хости, але новий хост не може бути виявлений миттєво, якщо він не досягне IP-адреси контролера Floodlight за допомогою HTTP або ICMP ping-запитів. З іншого боку, комутатори після активації режиму OpenFlow починають надсилати TCP-пакети синхронізації контролеру на порт TCP, а після отримання підтвердження TCP-синхронізації від контролера, комутатор і контролер об'єднуються в пару, і комутатор миттєво приєднуються до мережі OpenFlow.

Якщо потік у таблиці потоків не відповідає новому доданому хосту, пакет надходить на контролер Floodlight через комутатор з підтримкою OpenFlow для подальшої обробки в інкапсульованому пакеті. Контролер Floodlight реєструє хост з його IP-адресою, MAC-адресою, номером порту і комутатором, до якого він підключений, і відправляє пакет назад на комутатор разом з дією, яку потрібно виконати. Зазвичай, рішення, прийняте контролером, включає в себе встановлення відносного потоку в разі реактивних потоків або широкомовну передачу пакетів, якщо відповідного потоку не знайдено. У разі прийняття рішення про широкомовну передачу,

пакети в кінцевому підсумку досягають місця призначення через широкомовну передачу.

У випадках, коли через неправильну конфігурацію потоку мережа OpenFlow не може встановити з'єднання або досягається знижена пропускна здатність, основне усунення несправностей полягає в перегляді лічильників потоку, таких як пакети або байти, що проходять через певний потік. У типовій мережі OpenFlow розміщуватиметься багато комутаторів, і перевірка лічильників потоку на кожному комутаторі буде важким завданням. Однак подібну інформацію також можна перевірити з веб-інтерфейсу контролера Floodlight, де відображається підсумок кожного потоку. Інший підхід включає перегляд захоплення Wireshark для перевірки повідомлень Packet-Out із прийнятим рішенням про затоплення, що призводить до зниження пропускної здатності.

Спочатку кожен щойно підключений вузол у мережі повинен переглядати Інтернет, щоб HTTP-пакети остаточно надсилалися до відповідного контролера OpenFlow, і таким чином контролер реєстрував щойно доданий вузол у своїй базі даних. Крім того, можна розробити структуру або програму автоматизації на Python або іншому сценарії, яка працюватиме на контролері OpenFlow. Платформа автоматизації періодично переглядатиме базу даних контролера Floodlight і виявлятиме, чи внесено будь-які зміни.

Після виявлення будь-яких змін у мережі система автоматизації повинна сформулювати потоки за допомогою DPID комутатора та інших необхідних параметрів і негайно встановити ці потоки в таблицю потоків комутатора. Оскільки мережа масштабується, метод обчислення потоку повинен бути масштабованим, швидким і достатньо інтелектуальним, щоб заповнювати сукупні потоки, а не окремі потоки. Окрім цього, мережу OpenFlow можна розділити за допомогою підходу FlowVisor, щоб зробити структуру автоматизації розумною структурою, де DPID перемикання

відіграє важливу роль. Певну підмережу або область підмережі можна призначити комутатору, і користувач повинен вручну повідомити системі автоматизації, який DPID комутатора відповідає регіону мережі OpenFlow.

Експерименти, проведені в одній підмережі в локальній мережі, були успішними, але коли мережа розподілена на різні підмережі, як у типових локальних мережах, потоки з однієї підмережі не можуть досягти іншої підмережі. Протокол OpenFlow не надає жодних служб рівня 3 до версії 1.3, він просто визначає шлях, яким має пройти вихідний хост, щоб досягти пункту призначення. Коли маршрутизатор використовується для вирішення проблем зі шлюзом, підключення до локальної мережі відновлюється.

Висновок

Протокол OpenFlow ще недостатньо зрілий для розгортання в локальних мережах через його повільнішу інтеграцію в апаратне забезпечення та відсутність підтримки протоколів рівня 3 до версії 1.3. Крім того, модель розгортання не підходить постачальникам послуг, оскільки сценарій гібридного розгортання є недостатньо здійсненним через його сумісність та ізоляцію від існуючих локальних мереж. Крім гібридної моделі, заміна всього мережевого обладнання на обладнання з підтримкою OpenFlow не є гідним рішенням як з економічної, так і з точки зору обслуговування.

Таким чином, враховуючи поточну ситуацію, протокол OpenFlow можна розгорнути як тестовий стенд у менших мережах, таких як мережі університетських міст, для ізоляції трафіку: трафік експериментального протоколу та трафік дослідження. Така ізоляція була протестована в локальній мережі, і це стало однією з основоположних причин для протоколу OpenFlow.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

У цій дипломній роботі було проаналізовано інтеграцію та переваги протоколу OpenFlow у середовищі локальної мережі. На початку було зроблено огляд літератури, щоб вивчити та зрозуміти більше про SDN, протокол OpenFlow, їхні обмеження, переваги та недоліки. Потім було проведено пошук різних постачальників щодо підтримки протоколу OpenFlow у мережевих пристроях, щоб знайти комутатор із підтримкою OpenFlow і кращими функціями. Незважаючи на те, що всі функції OpenFlow v1.0 не включено в мережеве обладнання, на даний момент апаратне забезпечення створює обмеження для використання всіх переваг OpenFlow.

Щоб обробляти зростаючий обсяг даних, традиційна мережева інфраструктура повинна динамічно адаптуватися, але обмеження, накладені складністю, залежністю від постачальника та вимогами до якості обслуговування, призвели до статичної мережі. SDN перекриває ці обмеження та допомагає створити динамічну мережу, де мережевий контроль і керування можуть бути незалежними, програмованими та менш трудомісткими.

Протокол OpenFlow — це звичайний протокол прикладного рівня, інкапсульований у форматі TCP, IPv4 та Ethernet. Коли комутатор працює в режимі OpenFlow, він приймає рішення про переадресацію, переглядаючи свою таблицю потоків, а не переглядаючи традиційну таблицю пересилання комутатора. Таблиця потоків зіставляє вхідні пакети за параметрами, включаючи вхідний порт, MAC-адресу джерела та призначення або IP-адресу, і застосовує до них відповідні дії. У комутаторах HP обмеження для конкретного вхідного порту для кожного потоку, який потрібно зіставити, помножує загальну кількість необхідних потоків, яку можна було б зменшити за допомогою введення вхідного порту під символом підстановки для кожного потоку, тобто сукупних потоків.

У випадках, коли вхідні пакети не збігаються з жодним потоком у таблиці потоків, вони надсилаються до контролера OpenFlow для подальшого прийняття рішення. Залежно від типу потоку контролер вирішує встановити відносний потік або транслювати трафік. У разі прийняття рішення про трансляцію пакети врешті-решт досягають пункту призначення зі значно зменшеною пропускною здатністю. Таке різке зниження пропускної здатності не підходить для виробничих мереж, тому необхідно належним чином налаштувати потоки. Однак у комутаторах HP цей режим консультацій можна вимкнути через ризик втрати підключення до мережі.

Додавання потоків у мережу не допомогло створити динамічну мережу, оскільки процес конфігурації потоку та встановлення на відповідний комутатор через контролер OpenFlow був ручним процесом. Можна розробити структуру автоматизації, щоб справлятися зі зміною стану мережі та відповідно додавати або видаляти потоки. Контролер створює єдину точку збою, і DoS-атаки можуть призвести до розриву мережі, тому до контролера Floodlight неможливо отримати доступ за межами його підмережі. Тому для мережевих адміністраторів необхідно встановити захищений тунель або відповідні брандмауери, щоб установлювати потоки на контролер і віддалено контролювати його стан. Коли відбувається розширення мережі, нещодавно додані комутатори миттєво сполучаються контролером(ами) на відміну від нещодавно доданих вузлів у мережі OpenFlow. У мережі OpenFlow контролери повинні бути налаштовані в розподіленому режимі, а не в централізованому режимі, де один збій впливає на продуктивність усієї мережі. Крім того, до нього також можна застосувати балансування навантаження та механізм очікування.

Концепцію потоку можна інтерпретувати як уникнення маршрутизаторів у мережі, але насправді потоки забезпечують шлях для пакетів, потоки не перекривають функції шлюзу та ті, що пропонуються протоколами маршрутизації, такими як Border Gateway Protocol (BGP). Тому

маршрутизатори будуть присутні в локальних мережах, але концепція потоку пропонує більшу масштабованість і гнучкість мережі у випадках, коли потрібна її модернізація. Цікавий проект під назвою RouteFlow забезпечує віртуальний шлюз між різними підмережами в мережі Open-Flow і змінює таблицю потоків відповідно до рішень про маршрутизацію в популярних протоколах, включаючи RIP і OSPF. Обмеження повторного запуску RouteFlow для кожної зміни конфігурації робить його непридатним для масштабованої мережі та створює серйозний ризик для продуктивності мережі.

Розглядаючи інтеграцію OpenFlow із традиційними локальними мережами, у проведених експериментах спостерігалася успішна інтеграція OpenFlow всередині однієї підмережі в локальних мережах. OpenFlow версії 1.0 і далі до версії 1.3 не підтримують протоколи рівня 3, тому OpenFlow не повністю інтегрований із традиційними локальними мережами на рівні 3. Щодо його розгортання, обидві мережі можуть працювати поруч у гібридному режимі, але потоки лише контролюють Мережа OpenFlow. У гібридному підході контролер OpenFlow не контролює всю гібридну мережу для масштабованості або оновлення, не додаючи більше переваг традиційним локальним мережам. Таким чином, усе мережеве обладнання має бути замінено обладнанням із підтримкою OpenFlow, щоб отримати програмований і незалежний контроль над мережею, що знову ж таки не є оптимальним рішенням як з економічного, так і з точки зору обслуговування. Однак цікавий проект LegacyFlow спрямований на використання кількох пристроїв із підтримкою OpenFlow із традиційною мережевою інфраструктурою та пропонує переваги мережі OpenFlow шляхом перекладу потоків у відповідні команди конфігурації CLI для традиційних комутаторів.

Протокол OpenFlow все ще не на прийнятному рівні для розгортання в локальних мережах через відсутність підтримки протоколів рівня 3 до версії 1.3 та його повільну інтеграцію в апаратне забезпечення. Протокол OpenFlow

більше підходить для центрів обробки даних або магістральних мереж, де потоки можна ефективно використовувати для обробки зростаючих даних, як у розгорнутих мережах Google і NEC. Крім того, протокол OpenFlow також можна розгорнути в тестових стендах, наприклад, у менших мережах, таких як мережі університетських міст, для ізоляції трафіку, такого як дослідницький трафік і тестовий трафік експериментальних протоколів, і така ізоляція була протестована з локальними мережами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ARCHIVED: What is the difference between a LAN, a MAN, and a WAN?: http://kb.iu.edu/data/agki.html
2. Software-Defined Networking: The New Norm for Networks: https://opennetworking.org/product-certification/
3. OpenFlow: enabling innovation in campus networks38(2), ст 69-74.: OpenFlow: enabling innovation in campus networks: ACM SIGCOMM Computer Communication Review: Vol 38, No 2
4. A survey of software-defined networking: Past, present, and future of programmable networks. IEEE Communications Surveys & Tutorials, 16(3), ст 1617-1634.: Error Exponent for Multiple-Access Channels: Lower Bounds | IEEE Journals & Magazine | IEEE Xplore
5. Open Networking Foundation. (2012). OpenFlow Switch Specification Version 1.4.0.: openflow-spec-v1.4.0.pdf (opennetworking.org)
6. A retrospective on evolving SDN, ст. 351-362: Detailed diagnosis in enterprise networks | Proceedings of the ACM SIGCOMM 2009 conference on Data communication
7. Network virtualization and software defined networking for cloud computing, ст 24-31.: Toward a noun morphological analyser of standard Amazigh | IEEE Conference Publication | IEEE Xplore
8. https://www.youtube.com/watch?v=E5bSumTAHZE
9. Software-defined networking: A comprehensive survey. Proceedings of the IEEE, ст 14-76: Fifth power scaling of quality factor in silicon photonic degenerate band edge resonators | IEEE Conference Publication | IEEE Xplore
10. OpenFlow: enabling innovation in campus networks. ст 69-74:

OpenFlow: enabling innovation in campus networks: ACM SIGCOMM Computer Communication Review: Vol 38, No 2

11. https://www.openvswitch.org/
12. OpenFlow and Software Defined Networking ipspace webinar: https://my.ipspace.net/bin/list?id=OpenFlow
13. Software Defined Networking Creates a New Approach to Delivering Business Agility: http://www.gartner.com/newsroom/id/2386215
14. https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf
15. https://opennetworking.org/wp-content/uploads/2013/02/conformance-test-spec-openflow-1.0.1.pdf
16. https://opennetworking.org/wp-content/uploads/2013/07/OpenFlow1.3.4TestSpecification-Basic.pdf
17. https://www.nec.com/en/press/201710/global_20171012_03.html
18. Kanazawa University Hospital: Case Studies | NEC. (2012, June 28). https://www.nec.com/en/press/201403/global_20140304_01.html
19. https://www.nec.com/en/press/201402/global_20140224_01.html
20. https://www.linuxfoundation.org/blog/blog/live-from-linuxcon-the-cloudcast-podcasts-on-sdn-gluster-and-openstack
21. https://www.linuxfoundation.org/press/press-release/onos-project-hummingbird-release-enables-network-services-for-disruptive-and-incremental-sdn
22. Наукові конференції України, XVII Міжнародна науково-технічна конференція "Перспективи телекомунікацій 2023": http://conferenc.its.kpi.ua/2023/paper/view/27799
23. https://www.linuxfoundation.org/blog/blog/the-first-10-years-of-software-defined-networking
24. A theory of routing on power-law graphs, ст 219-234:

Monitoring the initial DNS behavior of malicious domains | Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference

25. OpenFlow: Enabling Innovation in Campus Networks. OpenFlow Report Available from ONF: State of the OpenFlow Union - Open Networking Foundation
26. Apply embedded openflow MPLS technology on wireless Openflow — OpenRoads: Apply embedded openflow MPLS technology on wireless Openflow — OpenRoads | IEEE Conference Publication | IEEE Xplore
27. Open Networking Foundation. (2017). OpenFlow Switch Specification Version 1.5.0: OpenFlow data planes performance evaluation - ScienceDirect
28. Flowlet-level multipath routing based on graph neural network in OpenFlow-based SDN - ScienceDirect
29. https://www.juniper.net/documentation/ru/ru/software/junos/sdn-openflow/index.html
30. https://www.juniper.net/documentation/us/en/software/junos/sdn-openflow/topics/concept/junos-sdn-openflow-virtual-switch-connection-to-controller-overview.html
31. https://www.techtarget.com/whatis/definition/OpenFlow
32. Wireshark Dissector for OpenFlow: https://github.com/noxrepo/openflow/tree/master/utilities/wireshark_dissectors
33. Quagga Routing Suite: https://www.nongnu.org/quagga/
34. OpenFlow – GENI: http://groups.geni.net/geni/wiki/OpenFlow
35. https://www.youtube.com/watch?v=l25Ukkmk6Sk
36. https://www.youtube.com/watch?v=rYW7kQRyUvA

37. https://www.researchgate.net/figure/Architecture-of-OpenFlow-SDN-separates-the-control-plane-from-the-network-equipment-and_fig4_269268917
38. NDDI and Open Science, Scholarship and Services Exchange (OS3E). http://www.internet2.edu/network/ose/
39. cURL: http://curl.haxx.se/
40. Flooflight-controller https://www.sdxcentral.com/networking/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/what-is-sdn-controller/openflow-controller/what-is-floodlight-controller/
41. https://ru.wikipedia.org/wiki/OpenFlow
42. RouteFlow Architecture: https://www.researchgate.net/figure/RouteFlow-architecture-conceptual-design-a-Overview-of-a-RouteFlow-controlled-network_fig5_266970288
43. OpenFlow - Software Defined Networking (SDN): https://ieeexplore.ieee.org/abstract/document/6524222
44. Software Defined Networking: https://docs.broadcom.com/doc/12378912
45. Advanced study of SDN/OpenFlow controllers: https://dl.acm.org/doi/abs/10.1145/2556610.2556621
46. Інтеграція протоколу OpenFlow: http://tk-its.kpi.ua/sites/default/files/2020-02/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC_%D0%9A%D0%BB%D0%B5%D1%86%D1%8C.pdf
47. VLANs in OpenFlow network: https://techhub.hpe.com/eginfolib/networking/docs/switches/5950/5200-4024_openflow_cg/content/499752709.htm
48. OpenFlow in hybrid mode: https://docs.ruckuswireless.com/fastiron/08.0.60/fastiron-08060-sdnguide/GUID-F1C3CD60-18D8-4AB3-9D7A-FB6F97C087DB.html

49. Experiments with OpenFlow: https://www.researchgate.net/publication/311028823_Experiments_with_OpenFlow_and_IEEE80211_Point-to-Point_Links_in_a_WMN
50. Experiments with OpenFlow: file:///C:/Users/dhiruk/Downloads/icwmc_2016_7_10_20037.pdf