Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання
(повна назва)
Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Аналіз функціонування програмно-конфігурованої мережі на базі протоколу OpenFlow у процесі передачі трафіку
(тема)

Виконав:
студент *2* курсу, групи *ІМІзм-22-1*
Коломоєць М.С.
(прізвище, ініціали)

Спеціальність *172 «Телекомунікації та радіотехніка»*
(код і повна назва спеціальності)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *«Інформаційно-мережна інженерія»*
(повна назва освітньої програми)

Керівник *доц. Колтун Ю.М.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

2024 р.

Не містить відомостей заборонених до відкритого публікування.

Студент / *Коломоєць М.С.* /

Керівник / *Колтун Ю.М.* /

Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність *172 «Телекомунікації та радіотехніка»*
(код і повна назва)

Тип програми *освітньо-професійна*
(освітньо-професійна або освітньо-наукова)

Освітня програма *«Інформаційно-мережна інженерія»*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«*27*» *жовтня* 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Коломойцю Максиму Сергійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Аналіз функціонування програмно-конфігурованої мережі на базі протоколу OpenFlow у процесі передачі трафіку*
затверджена наказом університету від «*27*» *жовтня* 2023 р. № 238 *Стз*

2. Термін подання студентом роботи до екзаменаційної комісії *23 січня 2024 р.*

3. Вихідні дані до роботи *ПЗ, що рекомендується для проведення моделювання: емулятор мережі – Mininet; ПЗ для організації віртуальних машин – VirtualBox; програмна платформа контроллера SDN – OpenDaylight; дослідження процесу обміну трафіком між вузлами мережі – аналізатор пакетів мережі Wireshark. Проаналізувати особливості функціонування та архітектуру SDN, протоколу OpenFlow, SDN-контроллера та OpenFlow-комутатора. Зробити програмне моделювання SDN у середовищі Mininet з 1 комутатором та трьома хостовими вузлами. Дослідити на основі моделі процеси обміну повідомленнями між контроллером та комутатором, обміну пакетами між вузлами мережі та принципи створення записів потоків у таблиці потоків.*

4. Перелік питань, що потрібно опрацювати в роботі
Вступ
1. Загальні принципи організації та функціонування програмно-конфігурованих мереж на базі протоколу OpenFlow.
2. Аналіз принципів функціонування SDN-контроллера та комутатора OpenFlow в процесі передачі трафіку.
3. Дослідження функціонування SDN на базі протоколу OpenFlow.
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) *Слайди у форматі Power Point (назва, мета і актуальність кваліфікаційної роботи, відмінності між традиційними мережами та SDN, концептуальна архітектура SDN, функціонування протоколу OpenFlow, архітектура SDN-контроллера, архітектура та основні компоненти OpenFlow-комутатора, конвеєрна обробка пакетів в OpenFlow-комутаторі, вікно менеджера VirtualBox для управління VM Mininet та ODL, моделювання мережі з OpenFlow-комутатором на VM Mininet та за допомогою графічного інтерфейсу контролера OpenDaylight, повідомлення OpenFlow, обмін пакетами між OpenFlow-комутатором та SDN-контроллером на прикладі повідомлення PACKET_IN, створення записів потоків у таблиці потоків, висновки)*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	27.10 – 30.10.23	*виконано*
2	*Підбір літератури за темою роботи.*	31.10 – 07.11.23	*виконано*
3	*Виконання розділу 1*	08.11 – 21.11.23	*виконано*
4	*Виконання розділу 2*	22.11 – 06.12.23	*виконано*
5	*Виконання розділу 3*	07.12 – 04.01.24	*виконано*
6	*Оформлення пояснювальної записки*	05.01 – 11.01.24	*виконано*
7	*Оформлення презентаційного матеріалу та подання роботи до ЕК*	12.01 – 23.01.24	*виконано*
8	*Підготовка до захисту та захист у ЕК*	24.01 – 30.01.24	*виконано*

Дата видачі завдання *27 жовтня 2023 р.*

Студент _____
(підпис)

Керівник роботи _____ *(доц. Колтун Ю.М.)*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 82 с., 39 рис., 3 табл., 25 джерел, 2 додатки.

ПРОГРАМНО-КОНФІГУРОВАНА МЕРЕЖА, SDN, OPENFLOW, OPENFLOW-КОМУТАТОР, SDN-КОНТРОЛЛЕР, ТАБЛИЦЯ ПОТОКІВ, ПОВІДОМЛЕННЯ OPENFLOW, ЗАПИС ПОТОКУ, МОДЕЛЬ МЕРЕЖІ, MININET, OPENDAYLIGHT, WIRESHARK

Об'єкт дослідження – програмно-конфігурована мережа (SDN), протокол OpenFlow.

Мета роботи – аналіз і моделювання принципів функціонування SDN на базі протоколу OpenFlow у процесі передачі трафіку

Розглянуті загальні принципи організації та функціонування програмно-конфігурованих мереж на базі протоколу OpenFlow. Проаналізовані концептуальна архітектура SDN та протокол OpenFlow. Проведений аналіз принципів функціонування SDN-контроллера і комутатора OpenFlow та їх взаємодія в процесі передачі трафіку. Зроблено дослідження та аналіз принципів функціонування SDN у процесі її програмного моделювання у середовищі мережного емулятора Mininet. Досліджені принципи створення записів потоків у таблиці потоків.

THE ABSTRACT

Explanatory note 82 pages, 39 fig., 3 tab., 25 sources, 2 app.

SOFTWARE DEFINED NETWORK, SDN, OPENFLOW, OPENFLOW-SWITCH, SDN- CONTROLLER, FLOW TABLE, OPENFLOW MESSAGE, FLOW RECORD, NETWORK MODEL, MININET, OPENDAYLIGHT, WIRESHARK

Object of research – software defined network (SDN), protocol OpenFlow.

The purpose of work – analysis and modeling the principles of SDN functioning based on the OpenFlow protocol in the process of traffic transmission.

The general principles the organization and functioning of software-configurable networks based on the OpenFlow protocol are considered. The conceptual architecture of SDN and the OpenFlow protocol are analyzed. The principles of operation of the SDN controller and OpenFlow switch and their interaction in the process of traffic transmission are analyzed. The study and analysis of the principles of SDN functioning in the process of its software modeling in the environment of the network emulator Mininet are carried out. The principles of creating flow records in the flow table are investigated.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ

API (Application Programming Interface) – прикладний програмний інтерфейс;

ARP (Address Resolution Protocol) – протоколу визначення адреси;

DP (Data Plane) – площина передачі даних;

DSCP (Differenciated Services Code Point) – поле коду диференційованих послуг;

CLI (Command Line Interface) – інтерфейс командного рядка;

CP (Control Plane) – площина управління;

ICMP (Internet Control Message Protocol) – протокол керуючих повідомлень Internet;

NMS (Network Management Systems) – система управління мережею;

QoS (Quality of Service) – якості обслуговування;

SDN (Software Defined Network) – програмно-конфігурована мережа;

SNMP (Simple Network Management Protocol) – простий протокол управління мережею;

STA (Spanning Tree) – алгоритм покриваючого дерева;

ToS (Type of Service) – тип обслуговування;

VLAN (Virtual Local Area Network) – віртуальна локальна мережа;

VM (Virtual Machine) – віртуальна машина;

VPN (Virtual Private Network) – віртуальна приватна мережа;

ЕМВВС – еталонна модель взаємодії відкритих систем;

ІОД – інформаційна одиниця даних;

ОС – операційна система;

ПД – передача даних;

ПЗ – програмне забезпечення;

СУ – система управління;

ЦОД – центр обробки даних.

ВСТУП

Підвищення попиту на телекомунікаційні послуги ставить нові вимоги щодо мережних технологій, які мають забезпечити надійний та якісний і, головне, захищений, доступ користувачів до інформаційних ресурсів. Однією із найважливіших проблем сучасних мереж є дуже висока завантаженість каналів і ліній зв'язку, що у свою чергу спричиняє неефективне використання мережної структури в цілому. У цьому аспекті провідні світові компанії, що так чи інакше задіяні на ринку інфокомунікацій, однозначно вважають, що є необхідність у організації гнучкої та надійної системи управління (СУ) телекомунікаційними та інформаційними мережами. Тобто провідним виробникам у сфері мережних технологій, обладнання і програмного забезпечення (ПЗ), стало зрозуміло, що екстенсивний шлях розвитку мереж і мережних технологій, що орієнтований на збільшення кількості спеціалізованого обладнання та обслуговуючого персоналу, є тупиковим. Необхідно шукати нові механізми і засоби до стимулювання розвитку бізнес операторів і сервіс провайдерів. Це призвело до появи нової технологічної концепції побудови та організації роботи мереж передачі даних (ПД), тобто до появи програмно-конфігурованих мереж (Software Defined Network, SDN) [1].

Ключовою особливістю SDN є відокремлення управління мережею від мережного обладнання. Так, у комп'ютерних мережах, у разі організації SDN, функціональність щодо забезпечення управління мережею (що також є відомою як площина управління (Control Plane, CP)) відокремлена від функціональності ПД (що також є відомою як площина ПД (Data Plane, DP)). Крім того, в цих мережах управління розділенням є програмованим. Міграція логіки управління, яка раніше була тісно інтегрована у мережне обладнання (наприклад, маршрутизатори або комутатори) у доступні та логічно централізовані контроллери, дає змогу не прив’язуватися жорстко до базової мережної інфраструктури з точки зору додатків. Тобто таке розділення надає можливості до формування більш економічно ефективної, масштабованої, незалежної від виробника функціонально-програмованої мережної архітектури. Крім того, архітектура SDN включає набір відкритих прикладних програмних інтерфейсів (Application Programming Interface, API), які дозволяють значно спростити організацію і функціонування загальних мережних служб, таких як: контроль

доступу, багатоадресну передачу маршрутизацію, безпеку, управління трафіком, управління пропускною здатністю, забезпечення якості обслуговування (Quality of Service, QoS), енергоефективність, тощо). У підсумку мережні провайдери будуть мати високу програмованість, ефективне управління мережею та її автоматизацію, що у свою чергу дозволить їм створювати гнучкі і високомасштабовані мережі, які можна легко, у разі потреби адаптувати до потреб бізнесу, що постійно змінюються [2].

На цей час одним з найбільш поширеним рішенням для організації управління мережним обладнанням в SDN є протокол OpenFlow, який представляє собою протокол взаємодії між мережним обладнанням і централізованим контроллером. Цей контролер являє собою спеціалізовану мережну операційну систему (ОС), яка розгорнута на виділеному фізичному сервері в SDN. Здійснення такого управління може замінити або доповнити функцію, що функціонує на мережному обладнанні та забезпечую створення маршрутів, побудову таблиці комутації і/або маршрутизації, тощо [3, 4].

Перспективність впровадження SDN пояснюється тенденціями сучасного ринку інформаційно-комунікаційних технологій і послуг. Зокрема вони сприяють переорієнтації бізнес-моделей наявних мережних операторів на хмарну інфраструктуру та цифрові сервіси, що дасть змогу мережним операторам скоротити надлишок обладнання у своїх мережах. Перехід до хмарних технологій, ключовими характеристиками яких є віртуалізація і масштабованість, дасть змогу скоротити операційні та капітальні витрати мережним операторам до 60%, що досягається за рахунок заміни фізичної мережної інфраструктури на основі реального обладнання, новою інформаційно інфраструктурою, основу якої мають складати додатки, які немає потреби прив'язувати до конкретного реального серверного обладнання. При цьому наявне мережне обладнання може розвантажитися у середньому на 25% за рахунок збільшення гнучкості та масштабованості мережі, а терміни впровадження інфокомунікаційних послуг значно скоротяться. Крім того, перехід до SDN дасть змогу забезпечити оперативну модернізацію конфігурації мереж, визначати їх місткість і відстежувати широкий спектр інфокомунікаційних послуг, що надаються мережею. Тобто іншими слова дослідження принципів впровадження і використання SDN на базі протоколу OpenFlow є актуальною задачею [2, 5].

1 ЗАГАЛЬНІ ПРИНЦИПИ ОРГАНІЗАЦІЇ ТА ФУНКЦІОНУВАННЯ ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖ НА БАЗІ ПРОТОКОЛУ OPENFLOW

1.1 Особливості організації та переваги SDN

Класичні принципи побудови мереж, основи яких закладалися ще наприкінці 60-х років минулого століття, у зв'язку зі швидким зростанням різноманіття мережних архітектур і технологій, складності та важливості задач, що ними мають вирішуються, є застарілими і досить часто не здатні ефективно реагувати на нові потреби як користувачів, так і мережних операторів. Повна модернізація такої класичної архітектури до нових потреб ринку та бажань користувачів вимагає чималих фінансових вкладень і часу, що у свою чергу позначається на ефективності мережі в цілому [1].

Існуючі телекомунікаційні та інформаційні мережі, незважаючи на постійний розвиток і широкомасштабне повсюдне використання, схильні до безлічі різноманітних проблем, що виникають через [1]:

- зростання кількості користувачів і трафіку, що може у свою чергу призвести до перевантаження мережного обладнання;

- використання великої кількості протоколів, що вимагає значного часу на перетворення інформаційних потоків під ті чи інші стандарти передачі;

- наявність традиційних мереж (наприклад, таких як традиційні телефонні мережі, мережі ISDN, тощо), ефективність яких практично вже неможливо підвищити через технологічні особливості обладнання і внаслідок цього обмеженість для практично будь-яких нововведень. Зокрема, в таких мережах налаштування мережного обладнання може залежати навіть від моделі пристрою в одного і того ж виробника, тобто потрібні дуже широкопрофільні фахівці в галузі мережних технологій, які повинні вміти аналізувати все це розмаїття мережних апаратних систем і працювати з великою кількістю протоколів;

- використання мережного обладнання різних вендорів, яке досить часто може конфліктувати між собою навіть у сучасних мережах, незважаючи на сталу тенденцію щодо використання універсальних мережних протоколів і принципів обробки інформації;

- складність проведення налаштування мережі з великою кількістю вузлів, що потребує значних часових витрат. Крім того забезпечення управління такими мережами здійснюється шляхом конфігурування їх елементів через спеціалізовані інтерфейси;

- постійну необхідність ознайомлення з мережним обладнанням конкретного виробника, що вимагає постійної можливості здіснювати перепідготовку фахівців, а це, як наслідок, призводить до додаткових витрат.

Таким чином, наявність такої безлічі проблемних факторів призводить до ускладнення і мережного обладнання, і архітектури організації самих мереж, а у цілому – і до підвищення капітальних витрат на розгортання мережі. Одним із можливих шляхів вирішення цієї сукупної проблеми є реалізація правил обробки даних у вигляді програмних модулів, а не робити їх вбудовування в апаратні засоби. Це дозволить мережним адміністраторам забезпечити кращий контроль за мережним трафіком і, отже, у свою чергу, дозволить значно підвищити продуктивність мережі з в аспекті здійснення ефективного використання її ресурсів і швидкості обробки і передачі даних. Саме такий підхід лежить в основі організації мереж SDN, у яких обробка даних відокремлена від апаратних ресурсів, а управління ними реалізовано в програмному модулі, що називається контроллером [2].

SDN надає мережним адміністраторам функціональні інструменти, що дозволяють їм працювати з даними в мережі більш ефективно. Зокрема спираючись на SDN, адміністратори можуть здійснювати контроль потоків даних, а також можуть із центрального вузла здійснювати зміну параметрів комутуючого мережного обладнання (комутаторів, маршрутизаторів, мультиплексорів). При цьому додаток, що здійснює управління, зроблений у вигляді програмного модуля, що виключає необхідність роботи окремо з кожним із цих пристроїв. Наприклад, це дає змогу адміністраторам мережі вносити зміни у таблиці маршрутизації (шляхи маршрутизації) у маршрутизаторах (або робити зміни в таблицях комутації в комутаторах). Також це надає додатковий рівень здійснення контролю над потоками даними, які передаються по мережі, тому що адміністратор може призначити високий або низький пріоритет певним пакетам даних, дозволити або заблокувати певні пакети, що проходять через мережу з різними рівнями управління. У результаті забезпечується ефективний контроль і управління мережним трафіком і, отже, цей принцип можна використовувати в якості механізму забезпечення управління параметрами трафіку, що просувається

по мережі. На рис. 1.1 показані основні відмінності між SDN і звичайними мережами [6].

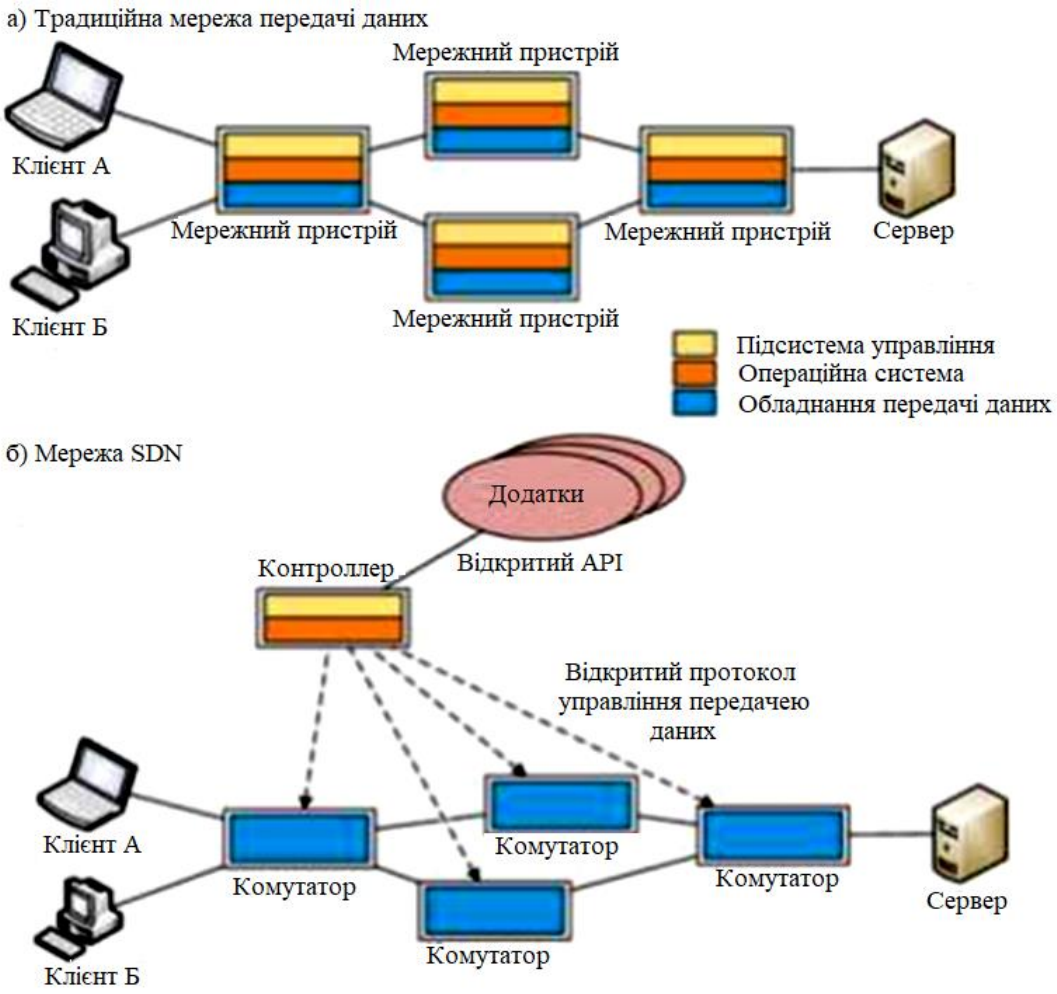

Рисунок 1.1 – Відмінності між традиційними мережами та SDN

В традиційній мережній архітектурі мережне обладнання (наприклад, це може бути комутатор Ethernet у локальній мережі підприємства, або маршрутизатор в операторській мережі) складаються з двох основних компонентів, які містяться у одному фізичному пристрої (рис. 1.1а) [6]:

- площини передачі даних (DP), що описує пристрої ПД;

- площини управління (CP), яка фактично являє собою ПЗ що здійснює управляє передачею даних із входу на вихід мережного пристрою. В якості такого ПЗ може виступати, наприклад, ОС мережного пристрою. або будь-яке інше керуюче ПЗ, що забезпечує обробку і управління мережними ресурсами. Причому в це ПЗ може вносити зміни тільки його розробник (вендор) або виробник мережного обладнання.

Коли інформаційна одиниця даних (ІОД) (наприклад, пакет, кадр або фрейм) прибуває на вхідний порт мережного пристрою, цей порт запитує в ОС, що робити з цією ІОД далі. ОС визначає напрямок передачі ІОД, по таблиці адрес сусідніх мережних пристроїв (для комутаторів – це MAC-адреси їх портів, для маршрутизаторів – IP-адреси портів), які під'єднані до цього мережного пристрою, і визначає вихідний порт, куди цю ІОД потрібно передати. Цей опис передачі даних у традиційній мережі э досить спрощеним, але для подальшого розуміння принципу роботи SDN його цілком достатньо [6].

Склад протоколів ОС є компетенцією її розробника, оскільки тільки він має доступ до вихідного коду. Оператор мережі, що купує мережний пристрій, разом з потрібним йому функціоналом, також вимушений заплатити і за багато непотрібних на даний момент функцій, тільки лише тому, що вони вже є структурно інтегровані в стандартний варіант ОС. І це не єдиний недолік традиційного проектування і функціонування мережі ПД [6].

Мережа, у якій кожен мережний елемент має власні протокол і алгоритм функціонування, з огляду на новітні технології, буде працювати не найкращим чином. У якості порівняльної аналогії, можна навести автомобільний рух у місті без використання хмарного сервісу, який на цей час широко застосовується для навігації з можливістю попередження про затори за маршрутом прямування. У традиційній мережі кожен мережний пристрій буде аналогічним перехрестю, яке проїжджає водій без використання навігатора. Щоразу він сам вирішує, куди повертати, орієнтуючись на візуальну завантаженість наступної ділянки. У разі використання навігатору цих питань у водія не виникає. Він буде витрачати менше часу на прийняття рішення щодо подальших дій на маршруті, що у свою чергу дозволить прибути на місце призначення значно швидше. Це відбувається тому, що навігатор формує оптимальний маршрут, з урахуванням заторів, аварій, дорожніх робіт, тощо. Однак, треба зазначити, що це можливо тільки тоді, коли більшість водіїв (а в ідеалі – всі) використовують хмарну навігацію. Тобто, повертаючись знову до мереж зв’язку, використання оптимізованих маршрутів для просування ІОД, зменшує завантаженість каналів і збільшує не тільки їх пропускну здатність, але й усієї мережі в цілому [6].

Саме такий принцип до реалізації управління маршрутизацією на мережі застосували вчені Стенфордського університету в Каліфорнії. Вони запропонували відокремити функціонал керуючого ПЗ від апаратної частини мережного обладнання, що показано на рис. 1.1б. При цьому традиційна мережа

розділяється на дві фізично окремі площини: CP і DP. Перша площина за цією концепцією відходить до централізованих серверів, а друга – залишається в мережному обладнанні, але яке вже не буде містити надлишкове керуюче ПЗ. При цьому мережа і кожен її мережний пристрій звертаються до площини управління тільки за тими функціями, які в даний момент їм потрібні. В термінах програмно-конфігурованих мереж площина CP називається SDN-контроллером і реалізується на стандартних серверах, які знаходяться в так званих дата-центрах або центрах обробки даних (ЦОД) [6].

Розглянутий підхід щодо організації SDN має на увазі не тільки те, що управління мережними пристроями здійснюється програмно, а також, що вони можуть ефективно і досить швидко перелаштовуватися, а й те, що на одному фізичному пулі мережних пристроїв можна розгорнути безліч мереж, які логічно не будуть залежати одна від одної. Такі логічні мережі можуть здійснювати передачу потоків трафіку різних типів і додатків, не заважаючи при цьому одна одній [6].

Додатки, яким потрібні різні мережні параметри і різні мережні конфігурації, адмініструють SDN-контроллер за допомогою API. У такій архітектурі кожен додаток може через SDN-контроллер зробити під себе конфігурацію логічної мережі із загальних ресурсів фізичної мережі рівня DP. І, як зазначалося, кожна така мережа працюватиме незалежно від інших логічних мереж, у тому ж пулі загальних мережних ресурсів (за умови їх достатності) [6].

Таким чином, можна визначити основні переваги SDN [3]:

- підвищення продуктивності. У зв'язку з тим, що тепер мережному обладнанню не потрібно безпосередньо здійснювати управління лінією або каналом зв’язку та мережним трафіком, мережа SDN дає можливість цьому обладнанню спрямувати всі свої функціональні можливості на прискорення передачі трафіку;

- спрощення процесу адміністрування. На централізованому контроллері SDN системний адміністратор може здійснювати спостереження за всією мережею в цілому, що значно підвищує зручність управління, підвищує рівень безпеки та робить більш ефективним виконання інших задач;

- прискорення реалізації та тестування нових сервісів. Програмні засоби SDN дають змогу мережним адміністраторам додавати нові функції до вже наявної мережної архітектури;

- підвищення безпеки. За рахунок того, що SDN надає можливість мережному адміністратору чітко бачити всі потоки трафіку, тож він зможе легше і швидше помічати можливі несанкціоновані втручання, визначати пріоритети щодо різних типів трафіку, розробляти політики реагування мережі в разі появи перевантажень та проблем з обладнанням;

- застосування хмарних технологій. У хмарній інфраструктурі, за рахунок використання ЦОД на віртуальних серверах, розміщуються дані та додатки. Взаємодія з віртуальними серверами, у разі підключення користувача до мережі, нічим не відрізняється від взаємодії з фізичним обладнанням. Технологія SDN здатна забезпечити управління таким ЦОД і ресурсами що він надає за допомогою відповідних API. Крім того SDN за допомогою API дозволяє забезпечити управління доступом користувачів до ресурсів хмарної інфраструктури, зробивши його більш гнучким, масштабованим і інтелектуальним.

1.2 Концептуальна архітектура SDN

Концептуальна архітектура SDN являє собою трирівневу модель, що складається з рівнів інфраструктури (передачі даних), контролю (управління) і додатків (рис. 1.2) [6, 7].

Якщо детально підійти до аналізу інформаційних потоків в архітектурі SDN, то треба виділити два напрямки обміну інформацією. Перший потік просувається між рівнем додатків і рівнем контролю, другий, відповідно між інфраструктурним рівнем фізичного мережного обладнання (мережних пристроїв) і рівнем контролю. Перший потік отримав назву «північний інтерфейс», а другий – «південний інтерфейс». В якості «північного інтерфейсу» виступає протокол на основі REST API, який надає способи взаємодії додатків із сервером, а в якості «південного інтерфейсу» виступає протокол OpenFlow [7].

Рівень додатків являє собою сукупність додатків, які безпосередньо взаємодіють з SDN-контроллером, здійснюючи запити щодо необхідних ресурсів за допомогою API та вирішуючи високорівневі задачі щодо забезпечення управління мережею. В якості прикладів таких додатків можуть бути зокрема такі, як додатки моніторингу трафіку, безпеки, управління потоками даних та інші [7].

На рівні контролю здійснюється низькорівневе логічно централізоване управління, яке реалізує передачу трафіку на інфраструктурному рівні за

допомогою відкритого інтерфейсу протоколу OpenFlow. Цей рівень постійно здійснює моніторинг всієї мережі, а також може забезпечити управління усім мережним обладнанням. Тут для мережних додатків можна відстежити топологію мережі, і API, що підтримуються [7].

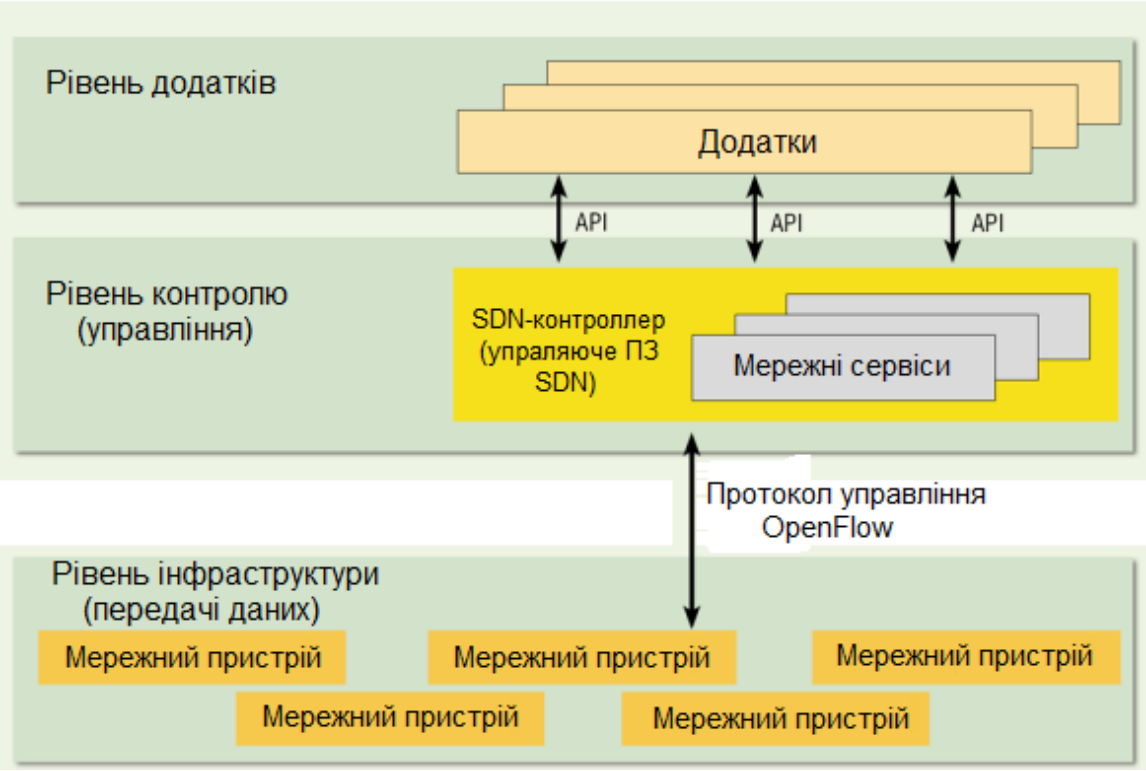

Рисунок 1.2 – Концептуальна архітектура SDN

Рівень інфраструктури формується із середовища ПД і мережного обладнання SDN (наприклад, OpenFlow-комутаторів), які можуть бути як логічними, так і фізичними елементами мережі [7].

Основними мережними компонентами наведеної на рис. 1.2 архітектури SDN є наступні [8]:

- оркестратор – являє собою платформу автоматизації, що подається як апаратний пристрій або ПЗ, та використовується додатками для здійснення конфігурування пристроїв на рівнях управління та інфраструктури (наприклад, здійснює корекцію функціонування декількох контроллерів). Крім того може виконувати конкретні сервісні запити;

- контроллер – являє собою обчислювальний пристрій (наприклад, сервер) на якому працює спеціалізована платформа на основі мережних додатків, які забезпечують процес управління, та мережної ОС. Під мережною ОС мається на

увазі повноцінний фреймворк, у задачі якого входить надання інтерфейсу прикладного програмування для мережних додатків, що здійснюють контроль та управління всією мережею, а також підтримка механізмів управління таблицями одного або декількох комутаторів OpenFlow (зокрема реалізується додавання, видалення чи модифікація правил, збір статистики, тощо). Під мережним додатком SDN мається на увазі програмна реалізація різних мережних сервісів та функцій (наприклад, маршрутизація, балансування навантаження, фільтрація трафіку, мережні екрани, шлюзи, шифрування, DPI, NAT, DHCP, DNS, тощо);

- комутатор OpenFlow – являє собою простий мережний пристрій, який можна програмувати. Він реалізує лише функції комутації даних відповідно із інструкціями контроллера;

- програмний інтерфейс (northbound API) – це варіант реалізації відкритого інтерфейсу програмування, який дозволяє із зовні здійснювати програмування контроллеру. Користувачами цього відкритого API є всі розробники мережних додатків.

Таким чином, можна бачити, що для управління інфраструктурою мережі за посередництвом SDN-контроллера, необхідна наявність ефективного протоколу відкритого управління. Цей протокол, як і концепція управління маршрутизацією, також був розроблений вченими зі Стенфорда і отримав назву OpenFlow. Він забезпечує віддалене управління мережними пристроями маршрутизації, завдяки своїй здатності контролювати таблиці маршрутизації трафіку в мережі. Більш детально протокол OpenFlow аналізується далі, проте слід зазначити, що це не єдиний можливий протокол для такого застосування, а один з найбільш популярних і загальноприйнятих стандартів реалізації SDN [6, 7].

1.3 Особливості та принципи функціонування протоколу OpenFlow

1.3.1 Загальна характеристика та особливості протоколу OpenFlow

Перш за все треба зауважити, що OpenFlow – це один із протоколів взаємодії у процесі управління, а SDN – це реалізація цілої мережної архітектури. Протокол OpenFlow є відкритим, головною задачею якого є об'єднання зусиль щодо здійснення управління мережним обладнанням різних виробників. З моменту своєї появи цей протокол постійно знаходиться у розвитку та

вдосконалюється, тобто додається функціональність, виправляються помилки минулих стандартів, що заважають отримати сповна переваги від використання SDN-мереж [10].

За визначенням OpenFlow – це протокол взаємодії між мережними пристроями, зокрема такими як комутатори та маршрутизатори, і централізованим контроллером, який представляє собою мережну ОС, що встановлена на виділеному фізичному сервері в SDN. Таке управління може замінити або доповнити функцію, що працює на мережному пристрої, яка забезпечує побудову маршрутів, створення таблиці комутації, тощо. OpenFlow-контроллер використовується для управління таблицями потоків комутаторів, на підставі яких відбувається ухвалення рішення про передачу прийнятого пакета на конкретний порт комутатора. Таким чином, у мережі формуються прямі мережні з'єднання з мінімальними затримками передачі даних і необхідними параметрами. На рисунку 1.3 наведена взаємодія SDN-контроллера із вузлами мережі [11].

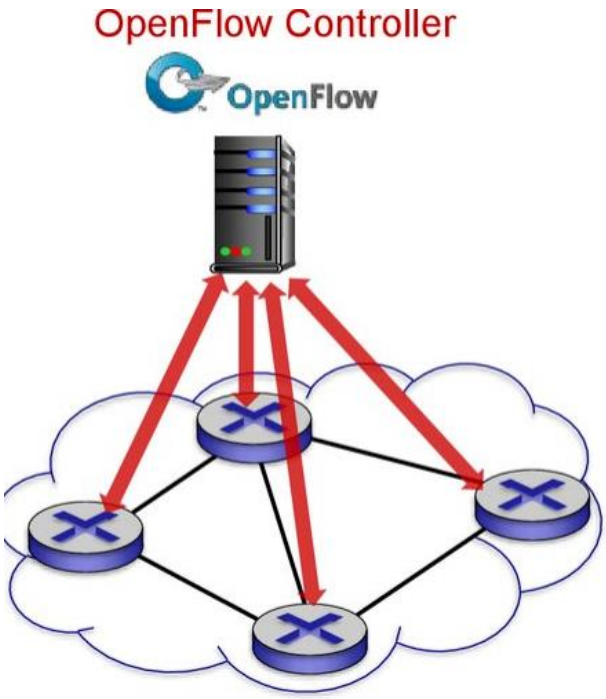

Рисунок 1.3 – Взаємодія SDN-контроллера з вузлами мережі

Первісно в OpenFlow v.1 (2009 р.) потік визначався як множина, що складається з пакетів, які можна описати певним набором полів. Усі записи про потоки знаходилися в таблиці потоків. Спочатку вона складалася з поля заголовка (header fields), лічильників (counter) та дій (actions). У заголовку

розташовувалися поля, за якими можна було призначити потік. OpenFlow v.1 описував 12 параметрів, з яких складався заголовок. Ці параметри наведено в таблиці 1.1 [10].

Коли відбувається збіг полів, то збільшується відповідне значення лічильника. Потоки пакетів, як правило, зачіпають декілька полів у таблиці потоків. Лічильники визначаються як для таблиці маршрутизації (per table), так і для потоку (per flow), порту (per port) або черги (per queue) [10].

Таблиця 1.1 – Параметри OpenFlow v.1

Базові поля в OpenFlow-пакеті	Довжина, біт	Тип
Ingress Port	32	всі пакети
Ethernet Source	48	на активних портах
Ethernet Destination	48	на активних портах
Ether type	16	на активних портах
VLAN id	12	всі пакети с Ethernet type 0x8100
VLAN priority	3	всі пакети с Ethernet type 0x8100
IP Source	32	всі IP и ARP пакети
IP Destination	32	всі IP и ARP пакети
IP Protocol	8	всі IP и ARP пакети
IP ToS, bits	6	всі IP пакеты
TCP/UDP source port	16	все TCP, UDP, ICMP
TCP/UDP dest. port	16	все TCP, UDP. ICMP

Кроме этого существуют поля действий, и когда поля совпадают, то выполняются соответвующие действия. Эти действия (actions) делятся на две группы – обязательные (required) и опционные (optional). Обязательные действия выполняют следующие функции [10]:

- передачи пакетов на физические или виртуальные порты. Если выполняется действие «All», то происходит посылка пакета со всех портов за исключением того, на который пакет пришел; в случае действия «Local» – реализуется посылка пакета на локальный порт; действие «Controller» обеспечивает посылку пакета на контроллер;

- сбрасывания пакета при отсутствии каких-либо применяемых действий.

К опционным действиям относятся: посылка пакетов на виртуальные порты, отправка пакета через очередь на порт для обеспечения обслуживания QoS

(действие «Enqueue»); выполнить модификацию полей (действие «Modify Field»). Отметим, что действие «Modify Field» – позволяет выполнять большое множество действий, которое отображено в таблице 1.2 [10].

Таблиця 1.2 – Дії по модифікації полів у протоколі OpenFlow

Дії	Дані, біт	Опис
Set VLAN ID	12	Додає новий заголовок, що містить VLAN ID і пріоритет «0», або заміщає існуючий заголовок із VLAN ID
Set VLAN priority	3	Додає новий заголовок, що містить значення поля пріоритет і VLAN ID «0» або заміщає в існуючому заголовку поле пріоритету
Strip VLAN header	-	Відкидає заголовок VLAN
Modify Ethernet source MAC address	48	Заміщає існуючу MAC-адресу джерела на задане значення
Modify Ethernet destination MAC address	48	Заміщає існуючу MAC-адресу одержувача на задане значення
Modify IPv4 source address	32	Замінює IPv4-адресу джерела новим значенням і оновлює контрольну суму
Modify IPv4 destination address	32	Замінює IPv4-адресу одержувача новим значенням і оновлює контрольну суму
Modify IPv4 ToS bits	6	Замінює значення у полі ToS (тільки для IPv4)
Modify TCP/UDP source port	16	Замінює TCP/UDP-порт джерела новим значенням і оновлює контрольну суму
Modify TCP/UDP destination port	16	Замінює TCP/UDP -порт одержувача новим значенням і оновлює контрольну суму

Протокол OpenFlow v.1 обладал также и недостатками, которые ограничивали использование возможности SDN. Например, над пакетом можно было реализовать только одно действие, при этом память даже в недорогих коммутаторах была рассчитана на 1 - 2 тысяч записей, что являлось очень сильным ограничением. В последующих версиях попытались эти ограничения свести к минимуму и, кроме того, в каждой

версии расширялся функционал протокола. Основные изменения протокола OpenFlow в процессе его развития показаны в таблице 1.3 [10, 25].

Таблиця 1.3 – Розвиток протоколу OpenFlow відповідно до його версій

Версія протоколу	Зміна функціоналу
OpenFlow 1.0	Одна таблиця містить 12 полів
OpenFlow 1.1	Multi-table, group table, подтримка VLAN, MPLS
OpenFlow 1.2	Підтримка IPv6, збільшення полів match (класифікатори), можливість зміни ролі контроллера в кластері
OpenFlow 1.3	Meters table для забезпечення QoS, збільшення гнучкості обробки пакетів, для яких не знайдено збігів
OpenFlow 1.4	Оптичні порти, моніторинг потоків, bundle-групування змін станів, що між собою пов’язані
OpenFlow 1,5	Scheduled bundle – вводить час виконання (планувальник часу)

Однак у процесі розвитку протоколу OpenFlow залишаються деякі проблеми у разі його впровадження в SDN [11]:

- у разі відсутності зв'язку між контроллером і пристроями мережі комутатори переходять в автономний стан, і рівень передачі даних стає некерованим або зовсім перестає функціонувати;

- можлива помилка в програмуванні контроллера може стати причиною серйозної проблеми на всій мережі, яку він обслуговує;

- контроллер є основною точкою ураження в разі здійснення хакерських атак з боку зловмисників;

- як показано вище, управління в SDN реалізується на окремій площині CP з використанням пропрієтарного програмного забезпечення, і користувач має, куди конкретно звернутися за технічною підтримкою, але якщо з цією метою використовується відкрите програмне забезпечення, то користувачеві вже складніше буде звернутися за підтримкою до будь-якого вендора.

Також слід зазначити, що в кожній із версій протоколу OpenFlow була реалізована можливість надсилання повідомлень «контроллер – комутатор», які починалися на контроллері для управління комутатором, встановлення на ньому потрібної конфігурації, збирання статистики, тощо [10].

1.3.2 Функціонування протоколу OpenFlow

Функціонування протоколу OpenFlow полягає в модифікації змісту таблиці передачі пакетів (Forwarding table) усередині маршрутизатора або таблиці передачі кадрів усередині комутатора. Тому цей протокол не є ні протоколом комутації, ні маршрутизації. За аналогією з концептуальною архітектурою SDN, архітектуру мережних пристроїв також представляють у вигляді трьох площин: адміністрування (management plane), управління і передачі даних. За її допомогою можна наочно показати функціонування протоколу OpenFlow (рис. 1.4) [6, 12].

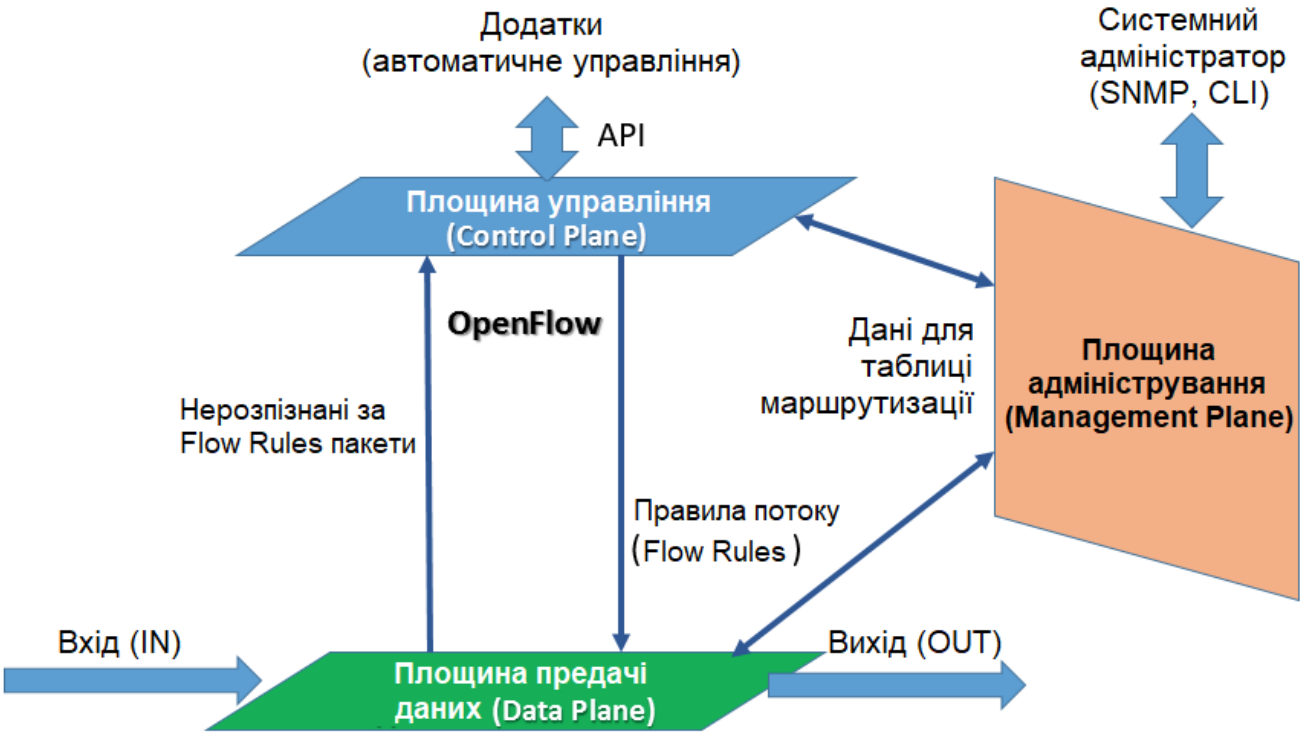

Рисунок 1.4 – Функціонування протоколу OpenFlow

Площина адміністрування регламентує режими роботи пристрою та здійснює модифікацію версій пропрієтарного ПЗ, яке поставляється разом із мережним обладнанням та забезпечує його функціонал; виконує команди простого протоколу управління мережею (Simple Network Management Protocol, SNMP), що надходять від зовнішньої системи управління мережею (Network Management Systems, NMS); виконує команди зовнішньої конфігурації пристрою використовуючи інтерфейс командного рядка (Command Line Interface, CLI). Площина DP передає пакети і кадри даних із входу мережного пристрою (IN) на

його вихід (OUT) у відповідності із таблицею передачі. На площині CP працюють стандартні протоколи маршрутизації та комутації. Вона використовує таблицю маршрутизації для створення таблиці передачі пакетів пакетів, яку потім використовує площина DP [6, 12].

Мережне обладнання здійснює передачу пакетів і фреймів відповідно до стандартних протоколів маршрутизації, однак, додатки не використовують певні пакети для надання послуг. Вони реалізують обмін даними між сервером та клієнтом і організують потоки пакетів від джерела до одержувача. Таким чином, OpenFlow – це фактично механізм управління потоками в мережі. Він визначає стандарт розсилки «правил потоку» (flow rules) у таблицю передачі кожного мережного пристрою таким чином, що площина CP може керувати площиною DP. Ці flow rules містять налаштування для мережного обладнання, зокрема: MAC-адреси вузла джерела передачі і призначення, IP-протокол та TCP-протокол джерела і вузла призначення, дані про віртуальну (логічну) локальну мережу VLAN у загальному домені мережної інфраструктури, мітки QoS і MPLS, та іншу інформацію. Flow rules далі додаються до наявних forwarding table із входу кожного мережного пристрою на вихід [6, 12].

Цінність OpenFlow полягає в тому, що він функціонує автоматично через централізований SDN-контроллер на площині CP. При цьому управління площиною DP стає набагато ефективнішим і швидшим, функціональні можливості пристроїв на мережі значно розширюються, тим самим досягається гнучкість в управлінні потоками трафіку в мережі [12].

Таким чином з вищевикладеного можна бачити, що технологічною основою концепції SDN є здатність контроллера доводити до відома комутаторів інформацію про те, як обробляти і просувати пакети. Далі, у наступному розділі проаналізуємо архітектуру та принципи функціонування комутатора за протоколом OpenFlow.

2 АНАЛІЗ ПРИНЦИПІВ ФУНКЦІОНУВАННЯ SDN-КОНТРОЛЛЕРА ТА КОМУТАТОРА OPENFLOW В ПРОЦЕСІ ПЕРЕДАЧІ ТРАФІКУ

2.1 Особливості управління комутатором від SDN-контроллера

Уже зверталася увага на те, що основна ідея SDN полягає в заміні механізму передачі пакетів у кожному комутаторі на управління від централізованого контроллера, що програмується користувачем стосовно того, як видавати інструкції на передачу пакетів («форвардінг») у кожному комутаторі. Як було показано в першому розділі, контроллер є тим пристроєм, що утворює площину управління технології SDN та являє собою мережну ОС, що встановлена на виділеному фізичному сервері в SDN. Зазначимо, що контроллер може бути як окремим вузлом, так і розподіленим набором вузлів Такий підхід дає змогу одночасно існувати як основному маршруту передачі пакетів, так і резервному [13].

На початку роботи у мережі, а потім періодично у процесі її функціонування, контроллер проводить опитування комутаторів мережі, аби визначити їх стан, а також внутрішні параметри. Далі, на основі цієї інформації, контроллер, використовуючи алгоритм покриваючого дерева (Spanning Tree, STA), будує деревоподібну топологію яка, з одного боку, забезпечує зв’язність всіх сегментів мережі, а з іншого боку – існування єдиного шляху між будь-якими двома комутаторами мереж. Після цього, комутатори будуть передавати пакети тільки за лінками, які були визначені в конфігурації мережі від контроллера. Ланки, які не є частиною створеного дерева STA, також можна використовувати для передачі пакетів за визначеними напрямками, як і в традиційних комутаторах, що підтримують STA. У разі надходження на комутатор пакета з невідомою адресою призначення, він повідомляє про це контроллер, котрий далі приймає рішення про подальші дії. Комутатори зазвичай конфігуруються так, щоб повідомляти нові адреси джерел контролеру, тому контроллер має змогу сповістити всіх інших комутаторів про наявність найкращого маршруту трафіку від цього нового джерела [13, 14].

SDN-контроллер може бути налаштований як стандартний файрвол, який не дозволяє передачу пакетів між певними парами вузлів з міркувань забезпечення безпеки. Наприклад, якщо в ЦОД є клієнт «А» і клієнт «В», і в кожного кілька під'єднаних до ЦОД вузлів, то мережу можна програмно сконфігурувати так, що

жоден пакет із вузла клієнта «А» не потрапить через ЦОД на жоден вузол клієнта «В», і навпаки. Реалізація SDN-контроллера найчастіше здійснюється із застосуванням стандартних програмних модулів. Однак, ПЗ для контроллера можна розробляти і під якусь конкретну мережу, щоб досягти найкращого управління її функціоналом [13].

В якості прикладу реалізації SDN-контроллера наведемо архітектуру від компанії Hewlett-Packard, яка показана на рис. 2.1 [15].

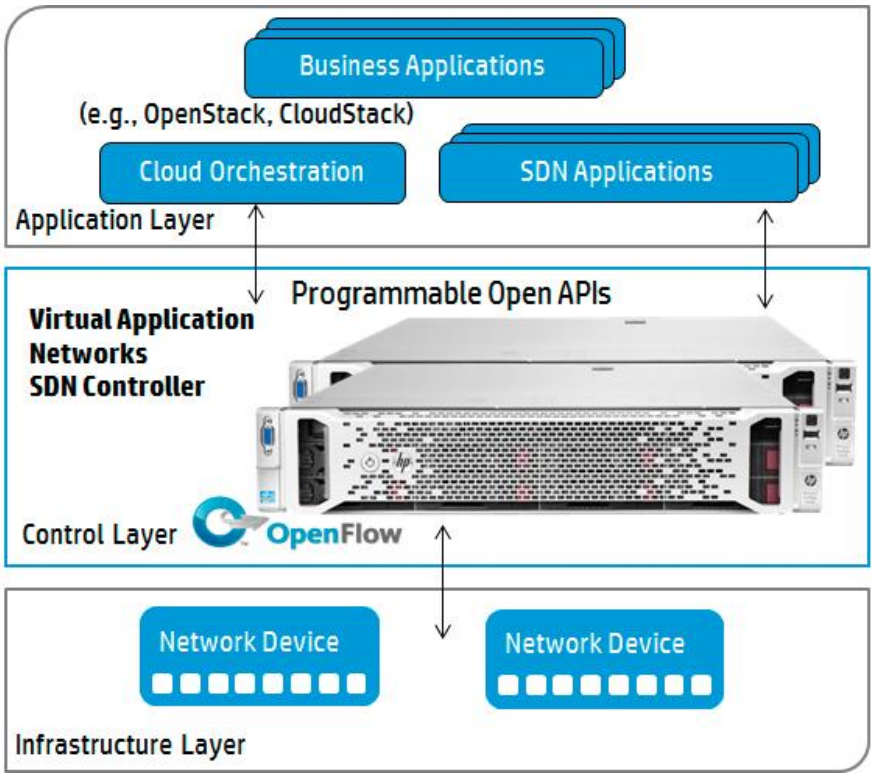

Рисунок 2.1 – Архітектура SDN-контроллера HP Virtual Application Networks SDN Controller

HP Virtual Application Networks SDN Controller був запропонований компанією Hewlett-Packard як готове рішення на базі сервера HP ProLiant. За основу побудови цієї архітектури взята концептуальна архітектура SDN, яка розглянута раніше (див. рис. 1.2). Цей контроллер являє собою ПЗ, яке можна розгорнути на звичайному фізичному (або віртуальному) сервері, або на кластері серверів для вироблення рішення, яке буде більш продуктивним і відмовостійким. Він реалізує стандартну функціональність динамічного конфігурування мережних пристроїв на основі заданих правил завдяки тісної взаємодії з мережними пристроями SDN за протоколом OpenFlow [15].

Цей контроллер підтримує низку вбудованих функцій, зокрема щодо можливостей мережної віртуалізації, гарантування безпеки, управляння трафіком, а також механізмів авторизації та автентифікації для здійснення контролю за доступом інтегрованих у контроллер інструментів і зовнішніх додатків SDN. Для взаємодії з останніми пропонується інтерфейс REST, який можуть використовувати різні системи оркестрації та управління, а також бізнес-додатки [15].

Звідси ще раз зазначимо, шо здатність контроллера сповіщати комутатори про те, як здійснювати передачу пакетів, є основою концепції SDN. Розглянемо далі архітектуру комутатора, що працює з протоколом OpenFlow, та його компоненти.

2.2 Архітектура і компоненти OpenFlow-комутатора

До OpenFlow-комутатора входять одна або декілька таблиць потоків та групова таблиця, в задачу яких входить здійснення пошуку та передачі пакетів. Також він має один або кілька каналів OpenFlow до зовнішнього контроллеру. Комутатор зв'язується з контроллером, а контроллер, як вище було зазначено, здійснює управління комутатором за протоколом OpenFlow (рис 2.2) [4, 16].

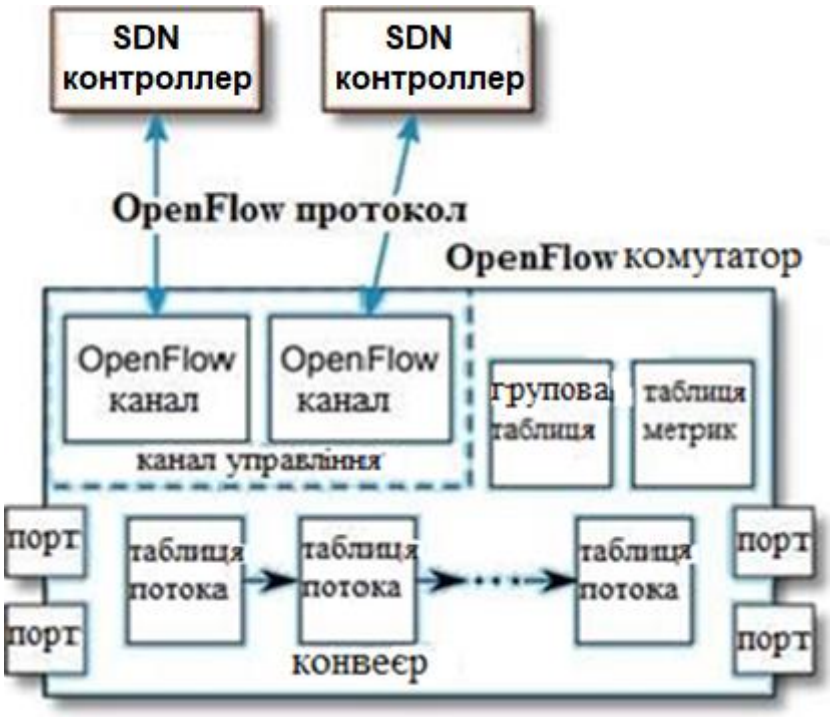

Рисунок 2.2 – Архітектура та основні компоненти OpenFlow-комутатора

За допомогою протоколу OpenFlow, контроллер здійснює додавання, оновлення та видалення записів потоків у таблицях потоків. Кожна така таблиця потоків у комутаторі включає у себе набір записів потоків, кожен з яких містить наступні поля: порівняння, лічильників та набору інструкцій. Ці поля мають застосовуватись до пакетів, що співставляються. Співставлення робиться з першої таблиці потоків і може продовжуватись аж до додаткових таблиць потоків конвеєра. Якщо знайдено відповідний запис, то починають виконуватися інструкції, які відповідають конкретному запису потоку. Якщо в таблиці потоків збігу не знайдено, то результат буде залежати від конфігурації запису потоку таблиць, що були пропущені (наприклад, пакет можна перенаправити на контроллери по каналу OpenFlow, відкинути або можна перейти до наступної таблиці потоків) [16].

Інструкції, що пов'язані з кожним записом потоку, мають дії або роблять зміни щодо конвеєрної обробки. Дії, які містяться в інструкції, описують передачу пакетів, їх модифікацію пакетів та обробку групових таблиць. Інструкції конвеєрної обробки дозволяють здійснювати відправлення пакеті до наступної таблиці для здійснення подальшої обробки, а також дозволяють передавати інформацію у вигляді метаданих між таблицями. Обробка конвеєра таблиці зупиняється, коли набір команд, що відповідає відповідному запису потоку, не вказує наступну таблицю. Зазвичай в цей момент робиться модифікація пакета та його передача [16].

Записи потоку можуть пересилати у порт, який, як правило, є фізичним портом. Цей порт може також бути логічним, що визначається комутатором, або зарезервованим. Зокрема зарезервовані порти можуть надавати загальні дії щодо передачі, наприклад, відправка на контроллер або здійснення передачі з використанням методів не OpenFlow, тобто таких як «звичайна» обробка комутатора, у той час як логічні порти, що визначаються комутатором, можуть вказувати групи агрегації каналів, тунелі або інтерфейси зворотного зв'язку [16].

Дії, які зв'язані із записами потоку, можуть також робити направлення пакетів до групи, яка буде визначати додаткову обробку. Групи надають набори дій, а також складнішу семантику передачі пакетів (наприклад, агрегація каналів, введення або виведення, що орієнтоване багато шляхів). Групи також надають можливості кільком записам потоку здійснювати передачу на один ідентифікатор (наприклад, робиться переадресація за протоколом IP на загальний наступний перехід). Це дозволяє ефективно змінювати загальні вихідні дії в записах потоку [16].

Групова таблиця розміщує записи групи, кожен з яких має список блоків дій з певною семантикою, яка залежить від типу групи [16].

2.3 Порти OpenFlow-комутатора

Порти OpenFlow – це мережні інтерфейси комутатора для передачі пакетів між процесом OpenFlow і рештою мережі. OpenFlow-комутатори логічно з'єднуються між собою через свої порти OpenFlow, причому, пакети від одного комутатора OpenFlow на інший комутатор OpenFlow можуть бути передані тільки через вихідний порт OpenFlow першого комутатора на вхідний порт OpenFlow іншого комутатора. Порти OpenFlow-комутаторів нічим не відрізняються від портів звичайного комутатора рівня другого рівня еталонної моделі взаємодії відкритих систем (EMBBC), тобто кожен порт має вхідний/вихідний буфер та унікальний номер, який виступає у ролі імені порту [17].

Набір портів OpenFlow може не співпадати з набором мережних інтерфейсів, що надаються обладнанням комутатора, деякі мережні інтерфейси можуть бути відключені для OpenFlow, а OpenFlow-комутатор може визначати додаткові порти. OpenFlow-комутатор може підтримувати три типи портів [17, 18]:

- фізичні порти – це порти, що визначені ПЗ OpenFlow-комутатора і відповідають його апаратним інтерфейсам;

- логічні порти – це порти, що визначені ПЗ OpenFlow-комутатора, але які не співвідносяться безпосередньо з його обладнанням. Логічні порти є абстракціями більш вищого рівня, що можуть бути визначені в комутаторі з використанням не-OpenFlow методів (наприклад, груп агрегації каналів, тунелів, інтерфейсів для реалізації зворотного зв'язку). Логічні порти можуть включати інкапсуляцію пакетів та можуть відображатись на різні фізичні порти. Обробка, яка виконується логічним портом, залежить від реалізації і має бути прозорою для обробки OpenFlow, також ці порти можуть взаємодіяти з обробкою OpenFlow-комутатора подібно до його фізичних портів. Єдина відмінність між фізичними та логічними портами є в тому, що пакет, який пов'язаний з логічним портом, може мати додаткове поле конвеєра, що має назву «Tunnel-ID». Значення Tunnel-ID посилається на контроллер вмісті з його номером логічного порту та фізичного порту, що лежить нижче [17];

- зарезервовані порти визначають загальні дії щодо передачі пакетів, такі як передача пакета на контроллер, розсилання на визначені інтерфейси (флуд, flooding) або передача пакетів із застосуванням не-OpenFlow методів, таких як «звичайна» обробка комутатора.

Зарезервовані порти OpenFlow-комутатора бувають обов'язкові і не обов'язкові, причому комутатор не повинен підтримувати всі з них, а тільки ті що відносяться до обов'язкових [18]:

- ALL – відображає всі порти, які комутатор може використовувати для передачі певного пакета. Використовується лише як вихідний порт, тому копія пакета починає вихідну обробку на всіх стандартних портах, крім вхідного порту пакета;

- CONTROLLER – відображає канал управління з SDN-контроллерами. Використовується як вхідний або вихідний порт. У разі використання в якості вихідного порту пакет інкапсулюється в повідомлення «Packet_In» і передається із застосуванням протоколу OpenFlow. Коли використовується як вхідний порт, це ідентифікує пакет, що є вихідним із SDN-контроллера;

- TABLE – відображає початок конвеєра OpenFlow-комутатора. Надсилає пакет у першу таблицю потоків, щоб пакет можна було обробити через конвеєр комутатора;

- IN PORT – відображає вхідний порт пакета. Використовується тільки як вихідний порт, а відправлення пакету здійснює через вхідний порт;

- ANY – спеціальне значення, що використовується в деяких запитах OpenFlow-комутатора, коли порт не вказаний (тобто цей порт відноситься до так званих підстановочних портів). Деякі запити OpenFlow посилаються на певний порт, до якого належить лише запит. Використання номера порту ANY в цих запитах дозволяє їм бути застосованим до всіх портів. Порт ANY не може застосовуватися ні як вхідний, ні як вихідний порт;

До не обов'язкових зарезервовані портів OpenFlow-комутатора відносяться наступні [18]:

- LOCAL – відображає локальний мережний стек комутатора та його стек управління. Використовується як вхідний порт або як вихідний порт. Локальний порт дозволяє віддаленим об'єктам взаємодіяти з комутатором та його мережними службами через мережу SDN на базі протоколу OpenFlow, а не через якусь окрему мережу управління;

- NORMAL – відображає передачу пакетів із використанням традиційного не-OpenFlow конвеєра комутатора. Використовується як вихідний порт. Для

обробки пакетів, застосовує звичайний конвеєр. У випадку, коли комутатор не може передавати пакети з OpenFlow-конвеєра до звичайного конвеєра, він має сповістити, що він не може підтримувати цю дію;

- FLOOD – відображає потокову розсилку з використанням традиційного не-OpenFlow конвеєра комутатора. Використовується лише як вихідний порт. У загальному випадку пакет буде передаватися на всі стандартні порти, але не на вхідний порт або порти, що перебувають у заблокованому стані. Комутатор також може використовувати ідентифікатор пакета VLAN або інші критерії, щоб обирати, які порти використовувати для потокової розсилки.

2.4 Конвеєрна обробка пакетів в OpenFlow-комутаторі

2.4.1 Функціональні принципи конвеєрної обробки пакетів

Комутатори, сумісні з OpenFlow, бувають двох типів: тільки OpenFlow та гібридні. У комутаторах, що підтримують лише операцію OpenFlow, всі пакети обробляються конвеєром OpenFlow і не можуть оброблятись інакше. Гібридні комутатори підтримують роботу як OpenFlow, так і традиційну комутацію або маршрутизацію. Гібридний комутатор може дозволити пакету відправлятись з конвеєра OpenFlow до звичайного конвеєра через зарезервовані порти NORMAL і FLOOD, що були розглянуті вище [19].

Конвеєр OpenFlow кожного логічного OpenFlow-комутатора включає до свого складу одну або декілька таблиць потоків, кожна з яких має по кілька записів потоків. Обробка у конвеєрі OpenFlow-комутатора показує, як пакети можуть взаємодіяти з цими таблицями потоків (рис. 2.3) [11, 16].

Таблиці потоків OpenFlow-комутатора нумеруються в тому порядку, в якому вони проходять пакетами, що надходять на вхідний порт, тобто починаючи з «0». Конвеєрна обробка пакетів робиться у два етапи: вхідна обробка пакетів та вихідна обробка пакетів. Конвеєрна обробка пакетів завжди починається з вхідної обробки. Спочатку пакет має співставитися із записами потоку в (рис. 2.4). Інші таблиці вхідного потоку можуть бути задіяні залежно від результату обробки пакету у «таблиці потоків 0». Якщо результатом вхідної обробки є передача пакета на вихідний порт, OpenFlow-комутатор може зробити вихідну обробку пакета в контексті цього вихідного порту [11, 18].

Рисунок 2.3 – Потік пакетів через конвеєр обробки

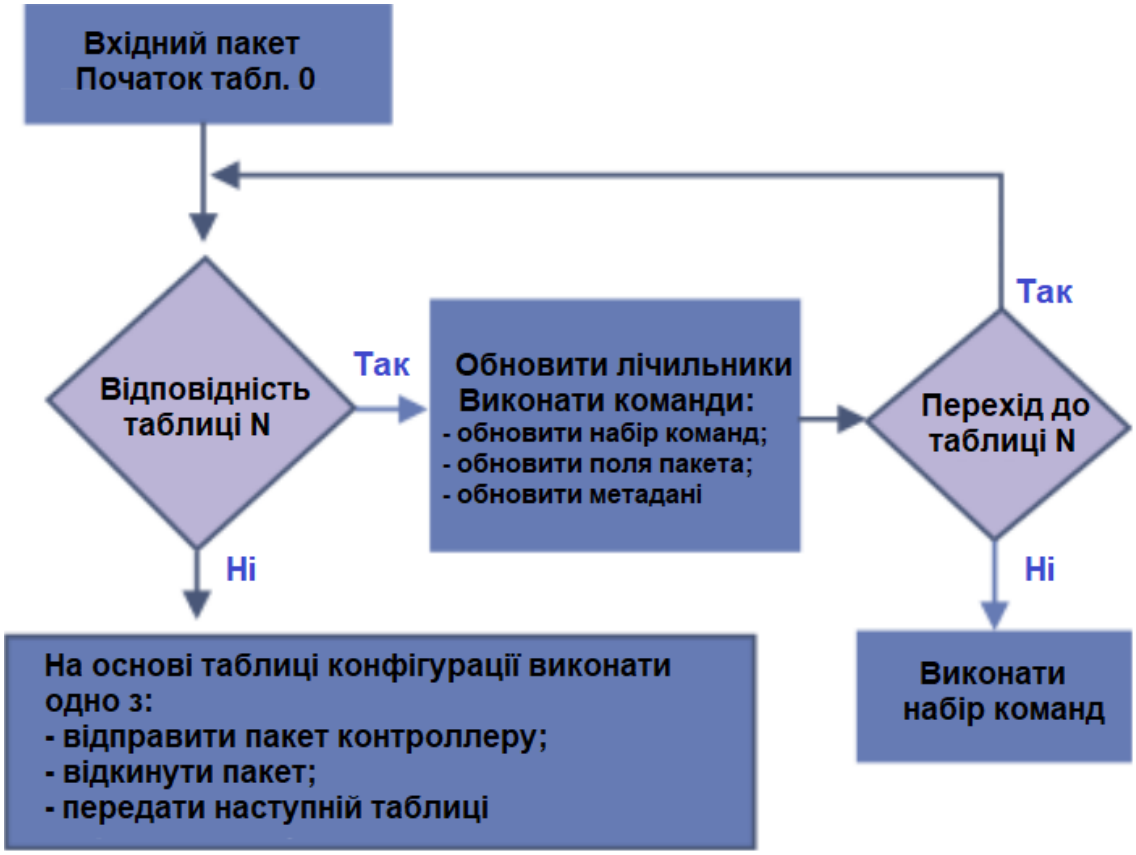

Рисунок 2.4 – Схема спрощеного алгоритму проходження пакета через OpenFlow-комутатор

Вихідна обробка не є обов'язковою, OpenFlow-комутатор може зовсім не підтримувати будь-які вихідні таблиці або просто можуть бути відсутні налаштування щодо їх використання. Якщо жодна із вихідних таблиць не встановлена в якості першої вихідної таблиці, обробка пакета має здійснюватися вихідним портом, і в більшості випадків цей пакет пересилається із комутатора. Якщо вихідна таблиця встановлена як перша вихідна таблиця, пакет має бути співставлений із записами потоку цієї таблиці потоків, і тоді можуть бути використані також і інші таблиці вихідних потоків залежно від результату збігу в цій таблиці потоків [19].

При обробці у таблиці потоків пакет зіставляється із її записами потоків. Якщо запис потоку буде знайдено, то починається виконання набору інструкцій, який включений до цього запису потоку. Ці інструкції можуть робити перенаправлення пакету до іншої таблиці потоків (на рис. 2.4 це «Перехід до таблиці N» (інструкція «Goto_Table»)). У наступній таблиці цей процес повторюється знову. Запис потоку може тільки перенаправити пакет на відповідний номер таблиці потоків, який має бути більшим, ніж номер його власної таблиці потоків, тобто конвеєрна обробка пакетів здійснюється тільки вперед без можливості повернення назад. Якщо відповідний запис потоку не надсилає пакети до наступної таблиці потоків, то обробка пакету конвеєром зупиняється на цій таблиці, де пакет обробляється з відповідним набором дій і зазвичай далі здійснюється його пересилання на вихідний порт [18, 19].

Якщо співпадіння пакета із записом потоку в таблиці потоків немає, то у цьому випадку ця таблиця потоків пропускається. Дії пакету під час пропуску таблиці будуть залежати від наявної конфігурації таблиці. Інструкції, що включені у записи потоків таблиць, що були пропущені, можуть вказувати, як обробляти такі не співставлені пакети. Зокрема опціями таких інструкцій можуть бути видалення пакетів, їх передачу до іншої таблиці або відправлення до SDN-контроллерів по каналу управління [19].

2.4.2 Таблиці потоків OpenFlow-комутатора та їх структура

Кожен OpenFlow-комутатор містить одну або кілька унікальних таблиць, які йому потрібно заповнювати на основі даних, що були отримані з контроллера. З урахуванням того, що по мережі передаються не окремі пакети, а потоки пакетів, то саме тому така таблиця отримала назву «таблиця потоків» (Flow table). Під

потоком маємо на увазі сукупність пакетів (викликів, сеансів зв'язку), що характеризується за певними параметрами та обмежена лише функціоналом, що закладений у самі таблиці (наприклад, сесія за протоколом TCP; вибірка пакетів з певної мережної адреси (IP або MAC); вибірка пакетів, що мають однаковий номер порту OpenFlow-комутатора, тощо) [18].

Формат запису потоку у таблиці потоків наведений на (рис. 2.5) [18, 19].

Поле відповідності	Прорітет	Лічильники	Інструкції	Таймаут	Cookie	Прапори

Рисунок 2.5 – Формат запису потоку у таблиці потоків (Flow table)

Із рис. 2.5 можна бачити, що запису потоку у таблиці потоків складається з наступних компонентів [18, 19]:

- поля відповідності – містяться характеристики пакету, що підлягають аналізу. Сюди входять: заголовки пакетів (field), вхідний порт, метаданні, що були сформовані попередньою таблицею. Запис про потік застосовується до пакета, якщо характеристики поля відповідності, і ті, що прописані у полях заголовку пакета співпали;

- пріоритет – являє собою натуральне число, що визначене із певного діапазону, яке показує ступінь вагомості про потік у таблиці потоків. Використовується для вибіркового застосування запису про потоки до конкретного пакета, що надійшов. Зауважимо, що для обробки пакета застосовується запис з найбільшим пріоритетом);

- лічильники – містять статистичні дані, зокрема інформацію про те, скільки пакетів було оброблено відповідно до цього запису про потік;

- інструкції – це команди, які застосовуються для зміни набору дій або для проведення конвеєрної обробки;

- таймаут – являє собою часові мітки, які показують термін життя запису в таблиці потоків, наприклад, максимальний час перебування на обробці або максимальний термін простою правила;

- cookie – це набір певних даних, які вибрані контроллером, для, наприклад фільтрації записів потоку, на які впливають статистика потоку або зміни потоку чи його видалення. Не використовується для здійснення обробки пакетів;

- прапори – призначені для зміни запису таблиці потоків, наприклад, прапор OFPFF_SEND_FLOW_REM застосовується для видалення запису в таблиці потоків.

Звернемо більш докладну увагу на команди OpenFlow-комутатора, що визначають набір інструкцій, які застосовуються для зміни набору дій чи до пакета, або модифікують процес обробки пакета в конвеєрі. Можна виділити кілька типів інструкцій [16, 18]:

- Apply-Actions – застосовує негайно конкретно визначені дії, без будь-яких змін у наборі дій. Ця інструкція може бути використана, наприклад, для модифікації пакета між двома таблицями або для виконання кількох дій одного та того ж самого типу;

- Clear-Actions – негайно видаляє всі дії зі списку дій;

- Write-Actions – заносить специфіковані дії до списку дій. Якщо дія цього типу є наявною у списку, то вона перезаписується, а якщо немає, то дія буде додана;

- Write-Metadata metadata/mask – записує масковані метадані до відповідного поля метаданих. Маска дає вказівки щодо того, які біти регістру метаданих мають бути модифіковані;

- Goto-Table – вказує наступну таблицю у процесі здійснення конвеєрної обробки пакета. Ідентифікатор наступної таблиці має бути більшим за ідентифікатор поточної таблиці. Ця інструкція підтримується у всіх таблицях потоків, крім останньої. Якщо в OpenFlow-комутаторі є тільки одна таблиця потоків, то ця інструкції не реалізується.

Вище неодноразово зазначалося, що кожному потоку в таблиці потоків відповідає одна або кілька дій, які мають виконуватися, коли потік пакетів є відповідним своєму запису. Під дією мається на увазі відповідна операція, яка вносить зміни у пакет, список дій та/або у процес обробки конвеєром. Відповідно до цього виділяють кілька типів дій [16, 18]:

- Output – дія щодо передачі пакета на вказаний порт (фізичний, логічний, зарезервований);

- Set-Queue – дія яка задає ідентифікатор черги для пакета;

- Drop – дія яка робить відкидання пакета, що належав певному потоку;

- Group – дія, яка відповідає за обробку пакета у зазначеній групі;

- Push-Tag/Pop-Tag – дія щодо здійснення роботи з мітками (наприклад, у мережах VLAN, MPLS);

- Set-Field – дія, що дозволяє змінити значення певного поля заголовка пакета;

- Change-TTL – дія, що дозволяє змінити значення поля, яке визначає час життя пакета в мережі (для IP пакетів це поле TTL (Time to Live)).

В OpenFlow-комутаторі дії можуть застосовуватися не окремо, а у вигляді списку дій. Цей список являє собою набір дій, що пов'язані безпосередньо із пакетом, який зберігається у процесі здійснення його обробки у таблицях потоків Список дій виконується тільки при виході пакета із конвеєра обробки пакетів [18].

2.5 Групова таблиця OpenFlow-комутатора

Група OpenFlow-комутатора – являє собою абстракцію, яка дозволяє спростити складні та спеціалізовані операції з пакетами, які неможливо або складно було б виконати за допомогою запису потоку в таблиці потоків. Кожна група отримує пакети в якості вхідних даних та виконує будь-які дії з цими пакетами. Група не може надсилати пакети до інших таблиць потоків. Крім того, очікується, що пакети були відповідним чином співставлені до входу у групу, тому що групи не підтримують операції співставлення пакетів із записами потоків. Тобто група фактично є просто механізмами виконання додаткових дій чи їх наборів [18].

На рис. 2.6 показана структура групи та її компоненти, з якого можна бачити, що перевага групи є в тому, що вона складається з окремих списків дій. Відповідно кожен окремий список дій називається сегментом. Виходячи з цього можна сказати, що група складається із переліка сегментів, а кожен сегмент або список сегментів може бути застосованим до пакетів (конкретні дії будуть залежати від типу групи) [18, 20].

Ідентифікатор групи (ID) – це 32-бітове поле, яке визначає номер групи. Лічильники (Counters) функціонують аналогічно тим, що застосовуються у звичайних таблицях потоків, і формують статистику групи. Списки або блоки дій (Actions) являють собою набори інструкцій, які відповідають певному сегменту. Тобто це той набір дій, що певним чином змінює заголовок пакета та/або відправляє пакет на вихідний порт [18].

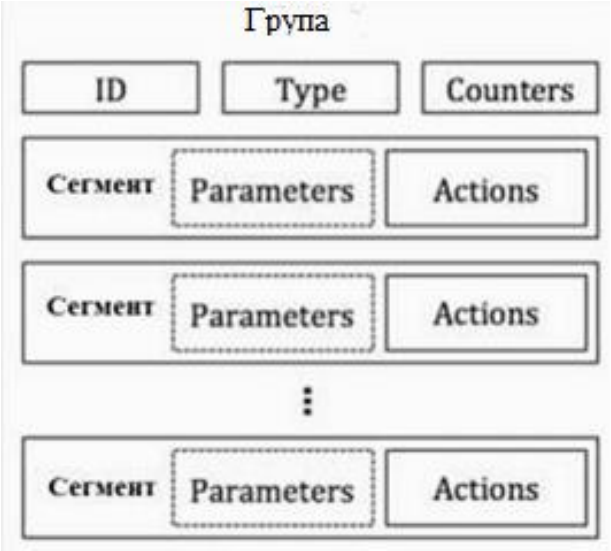

Рисунок 2.6 – Структура групи та її компоненти

Визначається чотири типи груп [18]:

- ALL – приймає будь-який пакет як вхідний і дублює його для того, щоб працювати з ним кожним сегментом, що є у списку сегментів, незалежно. Тобто використання групи ALL полягає у дублюванні та подальшій роботі з окремими копіями пакета, що визначаються діями у кожному сегменті. У кожному сегменті можуть бути різні дії, що дозволяє виконувати різні операції з різними копіями пакета [18];

- SELECT – ця група перш за все призначена для здійснення балансування навантаження. Кожен сегмент групи SELECT має ваговий коефіцієнт, і кожен пакет, що входить до групи, поміщається в один сегмент. Алгоритм вибору сегмента не визначений і залежить від реалізації OpenFlow-комутатора, проте зважений циклічний перебір – є досить очевидним, з одного боку, і досить простим вибором розподілу пакетів по сегментах, з іншого. Ваговий коефіцієнт сегмента є спеціальним параметром для кожного сегмента. Кожен сегмент у групі SELECT – це список дій, тому будь-які дії, що підтримуються протоколом OpenFlow, можна застосовувати в кожному сегменті. Так точно, як і в групі ALL, сегменти не мають обов'язково бути однаковими [18];

- INDIRECT – ця група має лише один сегмент, тобто всі пакети, що були отримані групою, направляються до одного єдиного сегменту. Метою функціонування групи INDIRECT є інкапсулювання загального набору дій, які використовуються багатьма потоками. Наприклад, якщо потоки A, B і C

співпадають із різними заголовками пакетів, але мають загальний набір або підмножину дій, то ці потоки можуть робити відправлення пакетів до однієї групи, вказуючи загальний ідентифікатор. INDIRECT замість того, щоб здійснювати дублювання списку спільних дій для кожного потоку. Ця група використовується для спрощення інсталяції OpenFlow та зменшення обсягів пам'яті, що може займається набором аналогічних потоків [18];

- FAST-FAILOVER (швидке подолання відмови). Ця група спеціально створена для виявлення та подолання збоїв портів. Кожен сегмент групи має порт спостереження та/або групу спостереження у якості спеціального параметру. Такий порт або група спостереження має відстежувати працездатність або статус порту чи групи, за якими ведеться спостереження. Якщо працездатність буде визначена зменшеною, то сегмент не буде використовуватися. Якщо працездатність буде визначена як підвищена, то сегмент можна використовувати. Одночасно можна використовувати лише один сегмент, і цей сегмент не буде змінено, якщо працездатність порту або групи, за якими ведеться спостереження, не зміниться з високого на низький. У випадку настання такої події (працездатність змінилася з високої на низьку), група FAST-FAILOVER швидко здійснює вибір наступного сегменту із списку сегментів з активним портом або групою спостереження [18].

Потрібно зазначити, що мотивація застосування групи FAST-FAILOVER полягає в тому, що це функціонально буде швидшим процесом, ніж звернення до площини управління, у разі необхідності обробити подію щодо виявлення непрацюючого порту чи у разі необхідності вставити новий потік або набір потоків. Застосування групи FAST-FAILOVER для виявлення збоїв каналу та відновлення їх працездатності повністю відбувається на площині даних OpenFlow.

3 ДОСЛІДЖЕННЯ ФУНКЦІОНУВАННЯ SDN НА БАЗІ ПРОТОКОЛУ OPENFLOW

3.1 Обґрунтування та програмні інструменти проведення дослідження

Як зазначалося, потенціал зростання продуктивності мереж на основі традиційних технологій, їх пропускної здатності на цей час є практично вичерпаним. Це зумовлено багатьма факторами: постійним зростанням кількості користувачів та обсягів трафіку, що передається; за рахунок постійного ускладнення структури та архітектури мереж зростають також і труднощі у їх налаштуванні та реалізації управління потоками, та іншими. Ці фактори особливо гостро проявляються, у зв'язку з необхідністю враховувати нові потреби у забезпеченні якості сервісів для високошвидкісних магістралей глобальних мереж та продуктивності мереж ЦОД, де зберігаються та оброблюються великі обсяги різноманітних даних. С появою новітніх технологій, що пов'язані з хмарними обчисленнями, окремо треба виділити зростання потреб у здійсненні віртуалізації мереж, тобто можливість реалізації декількох логічно ізольованих мереж з незалежними політиками обслуговування із застосуванням загального набору фізичних ресурсів мережі. Такі негативні тенденції щодо вичерпання потенціалу мереж на основі традиційних технологій і архітектурних рішень може змінитися у кращій бік як раз через появу мереж SDN. Дотримання підходу SDN щодо організації мереж і здійснення управління ними має забезпечити підвищення продуктивності мережі, зручності у її конфігуруванні, новітніх процесах здійснення віртуалізації, але також потребує проведення додаткових досліджень та розробок. Зокрема, це стосується аспектів організації мережних комутаторів, програмних додатків для управління мережею та платформ для їх розгортання та практичного застосування [1, 4].

Виходячи з вище сказаного, метою дослідження є моделювання і аналіз функціонування SDN на базі протоколу OpenFlow у процесі передачі пакетного трафіку.

Для проведення такого моделювання у цій кваліфікаційній роботі будемо використовувати наступне ПЗ: програмний пакет для розгортання віртуальних машин VirtualBox, емулятор мережі Mininet, контроллер SDN OpenDaylight (ODL), аналізатор мережного трафіку Wireshark. Дамо більш докладну характеристику цьому ПЗ.

VirtualBox являє собою ПЗ для організації віртуальних машин (Virtual Machine, VM) та здійснення процесів віртуалізації, зокрема віртуалізації ОС сімейства Microsoft Windows, Linux, MacOS та інших. Під віртуалізацією розуміється подання набору обчислювальних ресурсів, у вигляді логічного об'єднання, що має низку переваг проти фактичної конфігурації. Найбільш просте для розуміння визначення віртуалізації – це ізоляція програмних і/або апаратних ресурсів один від одного. Віртуалізація ОС дозволяє запускати кілька гостьових операційних систем на одному комп'ютері під управлінням хостової ОС. При цьому кожен екземпляр гостьової ОС функціонує зі своїм набором логічних ресурсів (процесор, оперативна пам'ять, зберігання даних). Ці логічні ресурси надаються із загального пулу ресурсів апаратного обладнання фізичного серверу. Їх виділенням управляє хостова ОС на базі якої розгортається гіпервізор. Гіпервізор або монітор VM – це може бути ПЗ або апаратний модуль, що надає можливість одночасної роботи кількох ОС на одному і тому ж самому хост-комп'ютері. При цьому гостьові ОС за рахунок роботи гіпервізора мають потрібні засоби зв'язку та взаємодії між собою (тобто здійснюється обмін файлами та забезпечуються мережні з'єднання). Крім того, ці ОС «вважають», що вони нібито працюють на різних фізичних комп'ютерах [21, 22].

Мережний емулятор Mininet надає інструментарій швидко розгорнути мережу, що може мати у своєму складі різноманітні віртуальні хости, комутатори, контроллери та канали зв’язку між ними. Ця програмна платформа використовує технології віртуалізації, що є інтегрованими у ядро ОС Linux. Це дозволяє запускати на віртуальних хостах стандартні мережні утиліти ОС Linux. Комутатори, що емулюються за допомогою ПЗ Mininet, мають підтримку протоколу OpenFlow, а це, у свою чергу, дозволяє використовувати це програмне забезпечення для аналізу побудови і функціонування SDN [23].

Контроллер мережі SDN OpenDaylight – це програмна платформа контроллерів SDN з відкритим вихідним кодом для здійснення налаштувань та автоматизації мереж незалежно від їх розміру та масштабу. Проект ODL виник завдяки появі програмно-конфігурованих мереж саме з акцентом на програмованість мереж. Цей проект був розроблений як базова основа комерційних рішень, що призначені для різних варіантів застосування в існуючих мережних інфраструктурах. ODL створений, для того щоб надати і користувачам, і постачальникам послуг зв’язку максимально можливу гнучкість у створенні контролера, який буде відповідати саме їх потребам. Ця платформа дозволяє будь-

якому користувачеві споживати послуги, що створені на інших платформах і у інших мережах, створювати та впроваджувати свої власні рішення та обмінюватись цими рішеннями з іншими. ODL підтримує широкий набір протоколів, що застосовуються у мережах OpenFlow-SDN [24].

Для дослідження процесу обміну трафіком між вузлами мережі будемо використовувати аналізатор пакетів мережі Wireshark. Цей програмний пакет має зручний графічний інтерфейс, а також є функціонал для здійснення сортування та фільтрації інформації. Wireshark може захоплювати весь трафік як у режимі реального часу, так і відкривати логфайли. Зокрема дозволяє здійснювати розпізнавання та моніторинг пакетів протоколу OpenFlow. Також Wireshark може визначати структуру пакетів та дані в них [25].

3.2 Дослідження та аналіз принципів функціонування SDN

Для проведення аналізу функціонування SDN на базі протоколу OpenFlow було створено дві VM у програмі VirtualBox версії 6.1.12. На одній було розгорнуто мережний емулятор Mininet, а на іншій – під управлінням ОС Linux Xubuntu 14.04.4 (далі Ubuntu) було розгорнуто SDN-контроллер ODL (рис. 3.1). Ці VM взаємодіють між собою через мережу хоста.

У процесі розгортання на VM контроллера ODL були підключені наступні його функції (рис. 3.2) [24]:

- `odl-restconf` – можливість використання API `restconf` для здійснення управління ODL;

- `odl-l2switch-switch` – функція підтримки комутатора рівня L2 (другий рівень моделі EMBBC), що необхідна для здійснення комутації пакетів через комутатор Open vSwitch Mininet, тому що Open vSwitch є OpenFlow-комутатором і не може пересилати пакети без контроллера. `l2switch` – це є модуль на SDN-контроллері, який виконує функцію комутатора, що реалізує навчання для Open vSwitch. SDN-контроллер автоматично відправляє потоки до Open vSwitch;

- `odl-mdsal-apidocs` – функція, що дозволяє автоматично створювати документацію API;

- `odl-dluxapps-applications` – функція підтримки графічного інтерфейсу ODL.

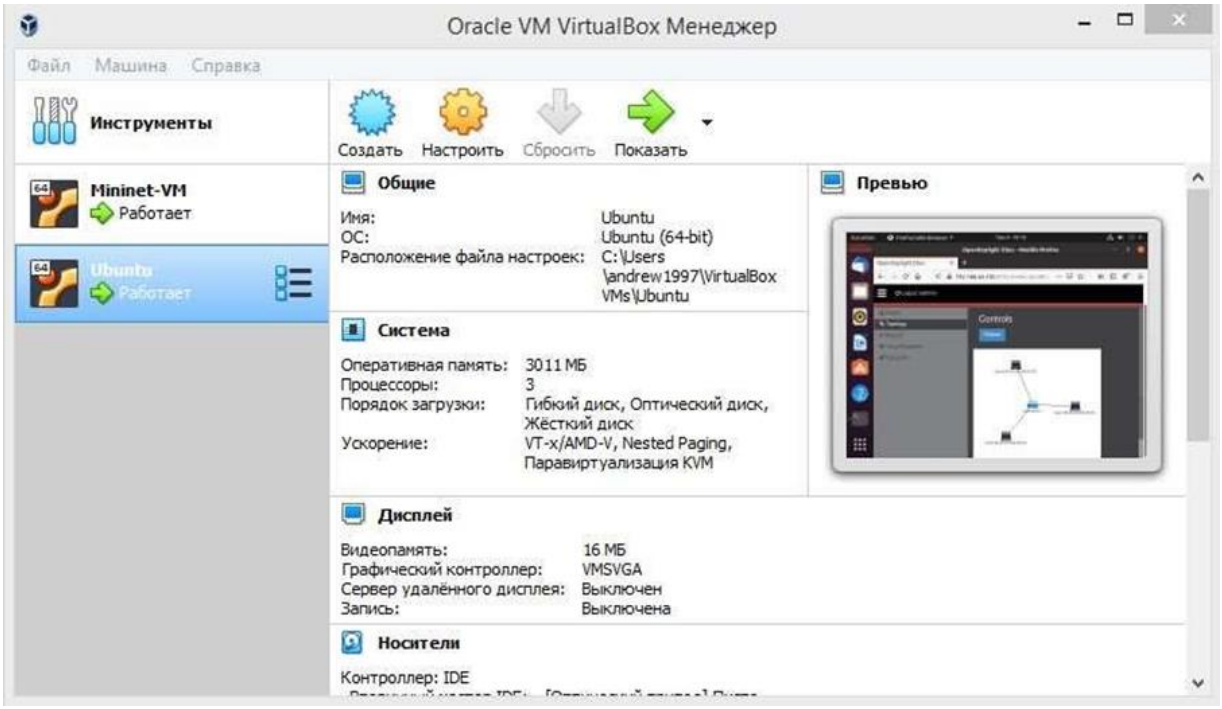

Рисунок 3.1 – Вікно менеджера VirtualBox для управління VM Mininet та ODL

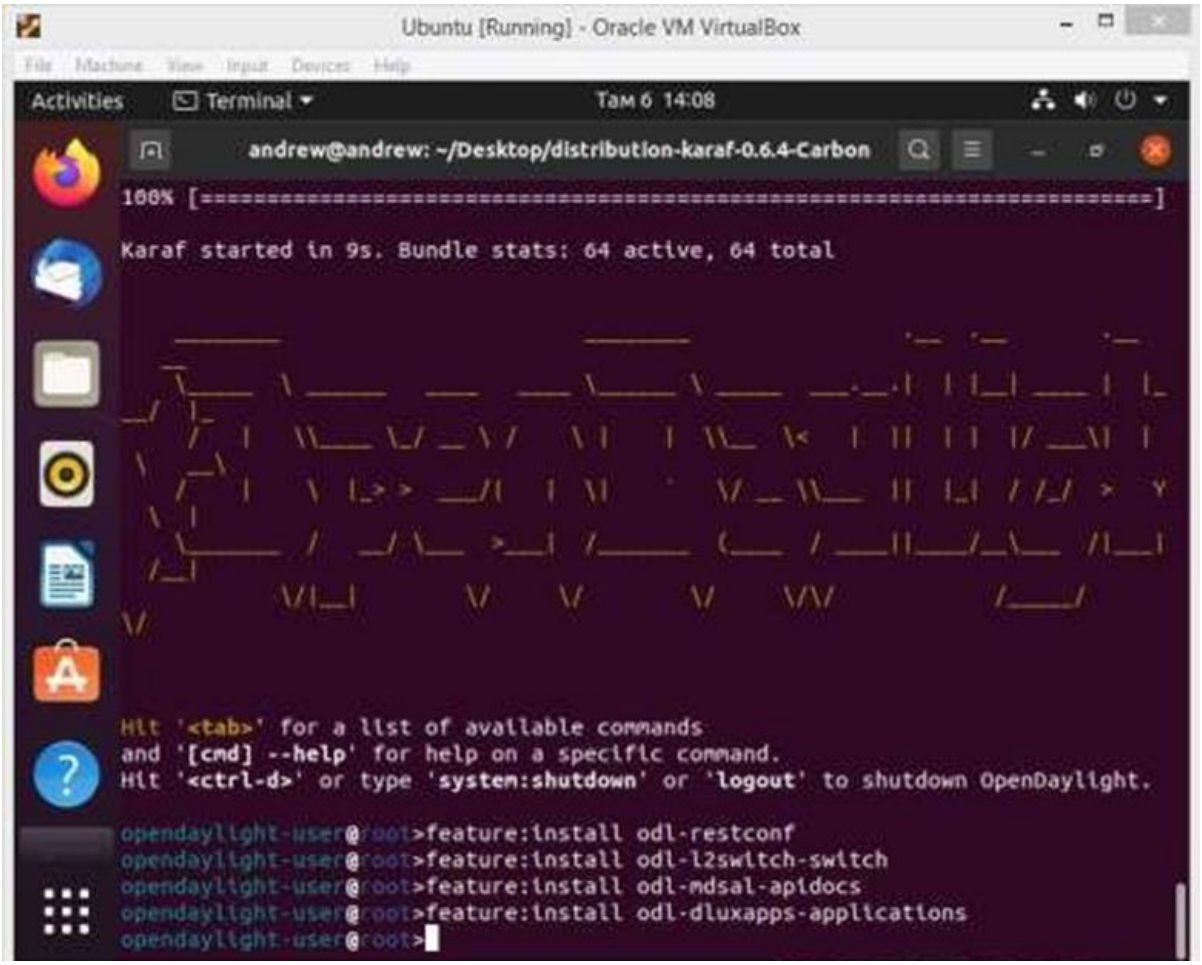

Рисунок 3.2 – Встановлення контролера OpenDaylight

На VM Mininet моделюється мережа з одним OpenFlow-комутатором Open vSwitch (IP-адреса 193.168.56.101) і трьома хостами, що підключені до нього. Управління здійснюється SDN-контроллером ODL (IP-адреса 193.168.56.102). Зазначений процес моделювання мережі описується командою Mininet («`sudo mn -- topo single,3 - Mac -- controller = remote 193.168.56.102, port=6633 - switch ovs.protocols=OpenFlow13`»), що виділена червоним на рис. 3.3. Її опції наступні [23]:

- `topo` – опція, що застосовується для визначення топології мережі. Тут параметр `single,3` показує, що до мережі буде підключений 1 комутатор та 3 хости;

- `mac` – опція, яка каже, що MAC-адреса кожного хосту буде встановлена на просте число;

- `controller` – опція, що застосовується для визначення SDN-контроллера. У нашому випадку SDN-контроллер ODL було розгорнуто на іншій VM, тому застосовуємо опцію `remote` з IP-адресою цієї VM (193.168.56.102);

- `switch` – опція, що застосовується для визначення віртуального OpenFlow-комутатора. Будемо використовувати Open vSwitch Openflow версії 1.3.

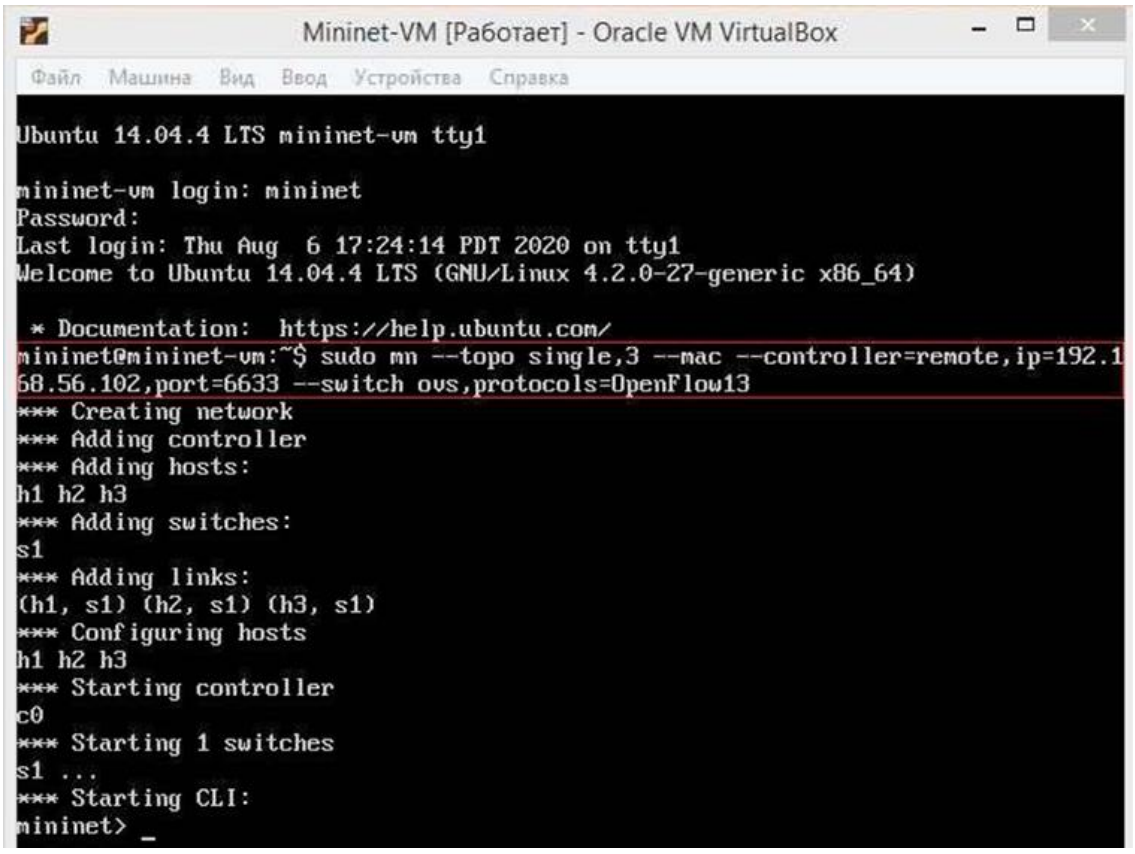

Рисунок 3.3 – Вікно створення мережі на VM Mininet

Для того, щоб подивитися топологію мережі, а саме, щоб зробити співставлення портів комутатора і хостів, застосовується команда «net» (рис. 3.4) [23].

Рисунок 3.4 – Застосування команди «net»

Для того, щоб переглянути інформацію про вузли, комутатори та контроллери в мережі, що моделюється, застосовується команда «dump» (рис. 3.5) [23].

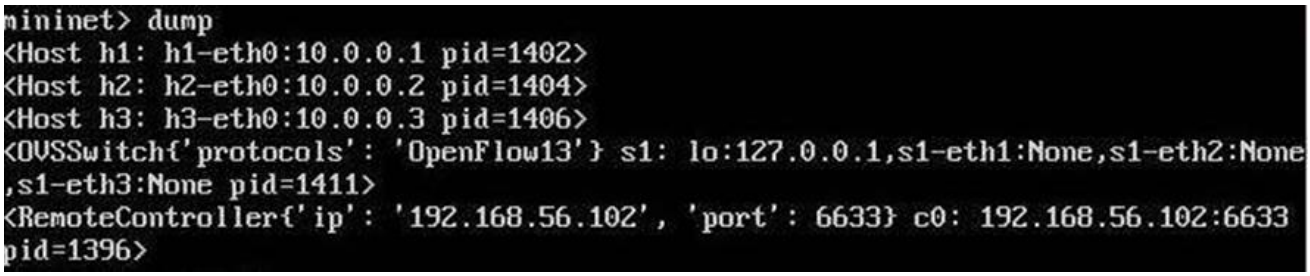

Рисунок 3.5 – Застосування команди «dump»

Для того, щоб здійснити перевірку зв'язку між усіма хостами в мережі застосовується команда «pingall». Ця команда буде робити запуск утиліти ping на кожному хості і тим самим уде здійснена перевірка зв'язку з кожним іншим хостом. Результати будуть надані в інтерфейсі командного рядка Mininet (рис. 3.6) [23].

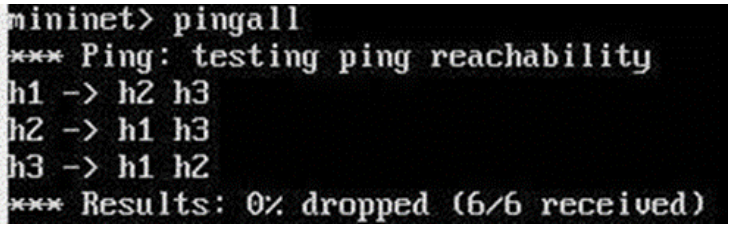

Рисунок 3.6 – Застосування команди «pingall»

Здійснення пінгуваання по всім вузлам необхідне для того, щоб впевнитися у тому, що контроллер ODL функціонує. Далі покажемо створену топологію мережі в графічному інтерфейсі ODL. Він також функціонує на VM і має IP-адресу: 193.168.56.102. Його порт був визначений додатком: 8181 (рис. 3.7) [24].

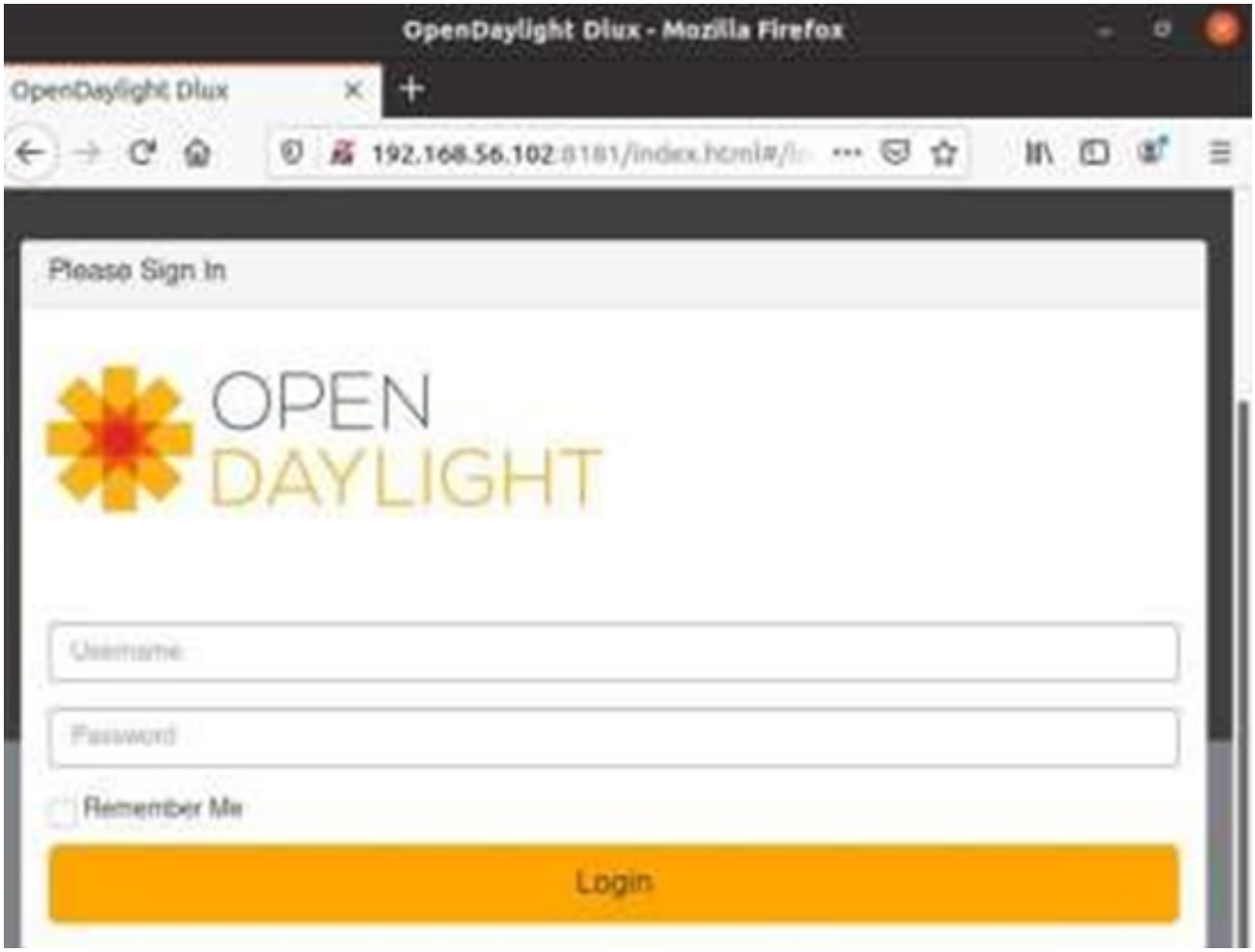

Рисунок 3.7 – Вікно графічного інтерфейсу контроллера ODL

Топологія мережі, що сформована в емуляторі Mininet за допомогою графічного інтерфейсу контролера ODL показана на рис. 3.8.

Тут, у вкладці «Вузли» можна подивитися інформацію про вузли змодельованої мережі (рис. 3.9). Зокрема можна бачити, що хост 1 є підключеним до інтерфейсу `s1-eth1`, хост 2 є підключеним до `s1-eth2`, а, відповідно, хост 3 – до інтерфейсу `s1-eth3`. Треба звернути увагу на те, що існує локальний інтерфейс управління, який є локальною площиною управління комутатора, що застосовується для відправки трафіку безпосередньо на площину CP [24].

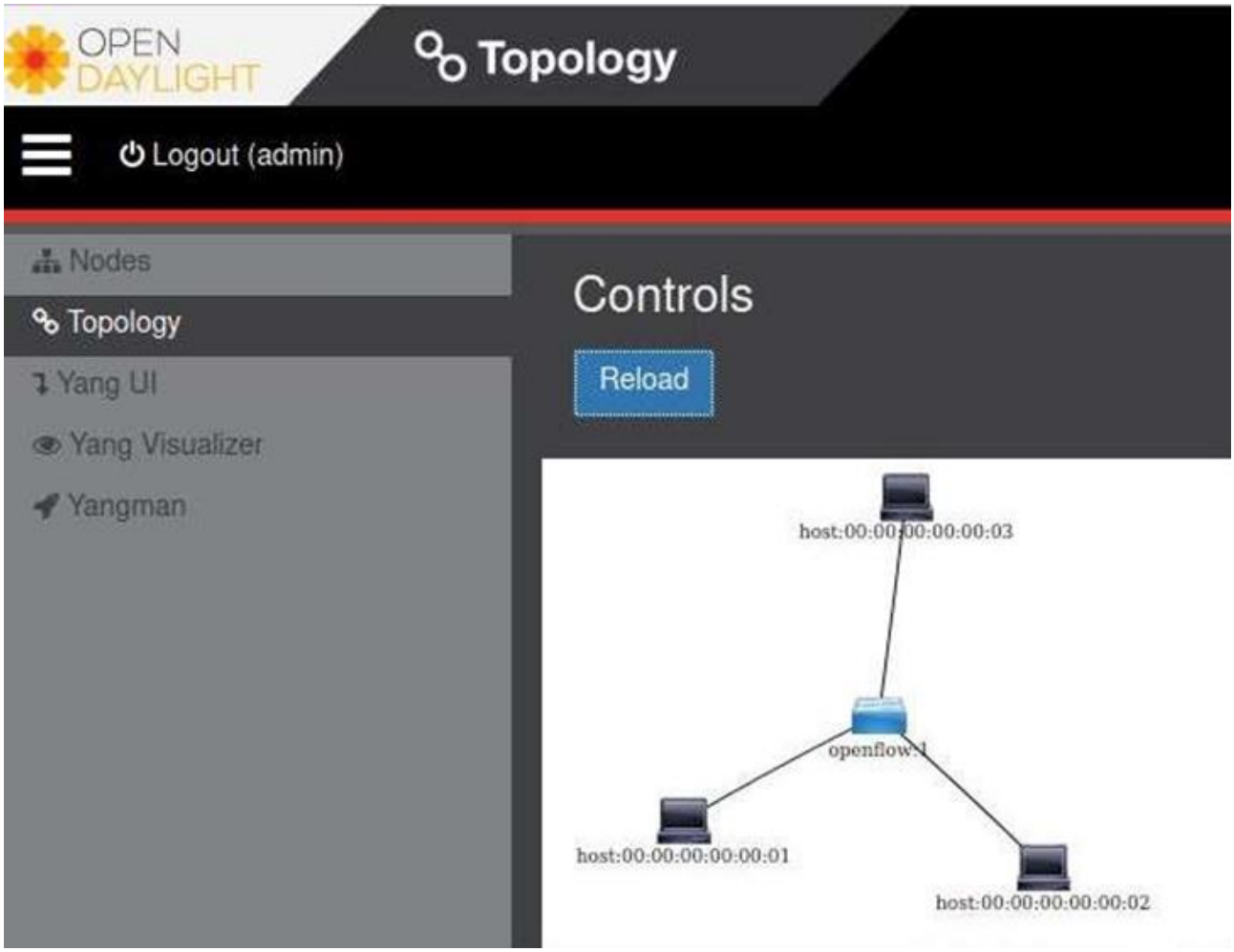

Рисунок 3.8 – Вікно вкладки «Топології» контроллера ODL, у якій графічно відображена топологія мережі, що була змодельована у Mininet

Рисунок 3.9 – Вікно вкладки «Вузли» контроллера ODL

Для перегляду повідомлень протоколу OpenFlow, якими обмінюються контроллер та комутатор у мережі, застосовується аналізатор пакетів Wireshark. Кожне повідомлення протоколу OpenFlow починається з однакової структури заголовка. Ця фіксована структура заголовка виконує три ролі, які не залежать від версії OpenFlow, що використовується. По-перше, поле версії протоколу OpenFlow («version») визначає саме на ту версію, якій належить це повідомлення. По-друге, поле «lenght» показує, де має це повідомлення закінчитися в потоці байтів, починаючи з першого байта заголовка. По-третє, ідентифікатор транзакції «xid», який представляє унікальне значення, що застосовується для співставлення запитів та відповідей. Поле типу («type»), дає уявлення про тип повідомлення та про те, як потрібно інтерпретувати корисне навантаження (рис. 3.10) [18, 20].

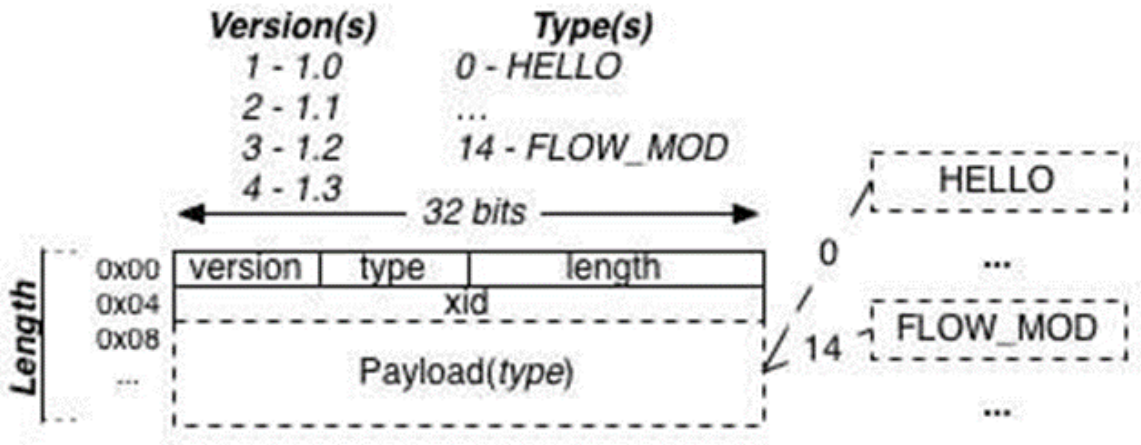

Рисунок 3.10 – Формат заголовка повідомлення OpenFlow

Використовуючи фільтр відображення Wireshark для openflow_v4, що визначає версію як OpenFlow версії 1.3, можна продивитись знайдені пакети (рис. 3.11). Зокрема можна бачити, що перший знайдений пакет – це повідомлення HELLO, яке було відправлене від комутатора (193.168.56.101) до SDN-контроллера (193.168.56.102). Цей пакет використовується для узгодження версії протоколу OpenFlow. Якщо узгодження версії протоколу не вдається зробити, то надсилається повідомлення про помилку з типом «Hello Failed» та кодом «Несумісний». Це є ініціалізацією каналу OpenFlow, що являє собою канал даних для протоколу OpenFlow між контроллером та комутатором [25].

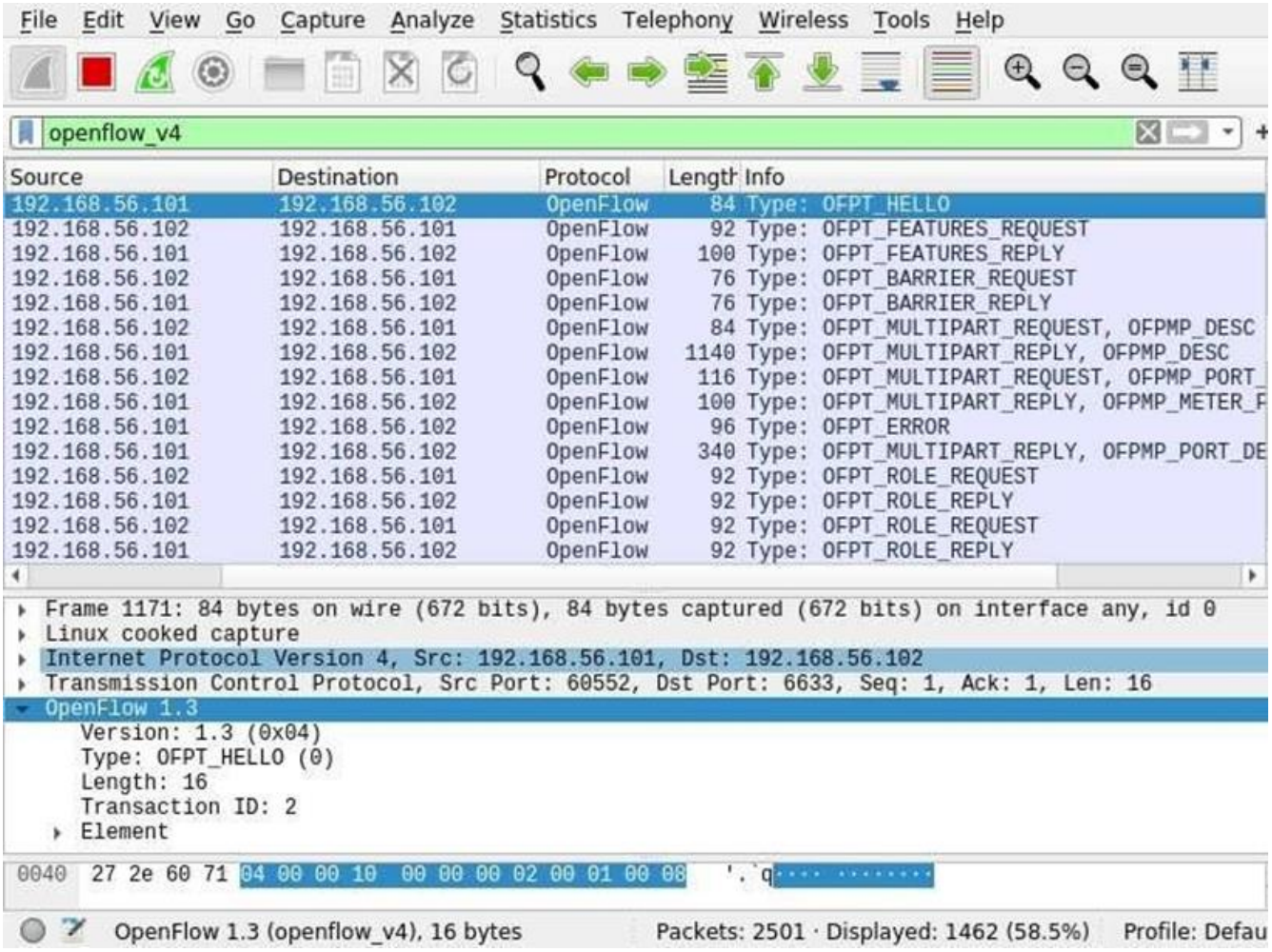

Рисунок 3.11 – Процес здійснення фільтрування пакетів протоколу OpenFlow

Наступний тип повідомлення – це OFPT_FEATURES_REQUEST від SDN-контроллера до OpenFlow-комутатора (рис. 3.12). У разі встановлення транспортного каналу OpenFlow між комутатором та контроллером, першою дією буде визначення функції. Для цього SDN-контроллер відправить на комутатор запит «`Feature Request`» транспортним каналом. Запит функції складається лише з повідомлення заголовка OpenFlow із значенням «`Feature Request`», що встановлено у полі «`type`» [18, 25].

Отримавши запит, комутатор відповідає на нього контроллеру повідомленням OFPT_FEATURES_REPLY, у якому перераховуються його можливості (рис. 3.13). У цьому повідомленні поле «`datapath-id`» – це 64-бітове поле, яке є фактично аналогом MAC-адреси моста Ethernet-комутаторів, його унікальний ідентифікатор для конкретного керованого конвеєра обробки пакетів. Поле «`n_buffers`» показує, скільки пакетів комутатор може поставити

в чергу для дій, що пов’язані із знахватом пакетів і їх пересиланням на контроллер «Packet_In». Кількість таблиць у комутаторі фіксується в полі «n_tables». Поле «Capabilities» показує які можливості комутатора підтримуються встановленим у моделі комутатором [18, 25].

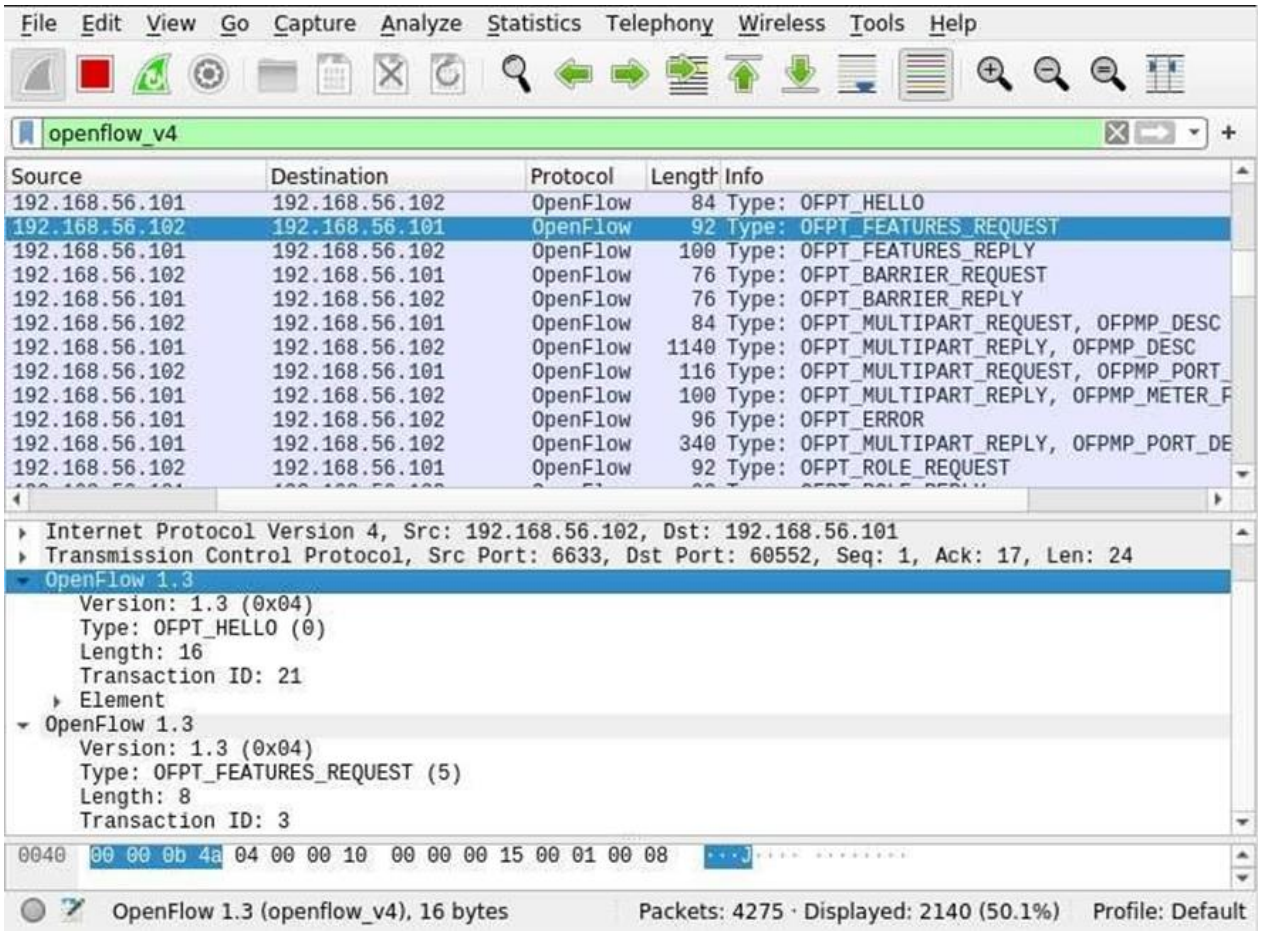

Рисунок 3.12 – Повідомлення OFPT_FEATURES_REQUEST

SDN-контроллер може зробити запит до комутатора про стан. Запит робиться по каналу OpenFlow за допомогою повідомлення OFPT_MULTIPART_REQUEST. Типи повідомлень, що обробляються цим повідомленням, включають різну статистику по потокам, таблицям, портам, чергам і таке інше (FLOW / TABLE / PORT / QUEUE / METER, тощо) або можуть включати різні функції опису: METER_CONFIG, TABLE_FEATURES, PORT_DESC, тощо [18].

У повідомленнях OFPT_MULTIPART_REPLY, OFPMP_PORT_DESC міститься інформація про наявні порти комутатора, що була надіслана на SDN-контроллер. У це повідомлення включено все, від їх типу до статусу та швидкості [18].

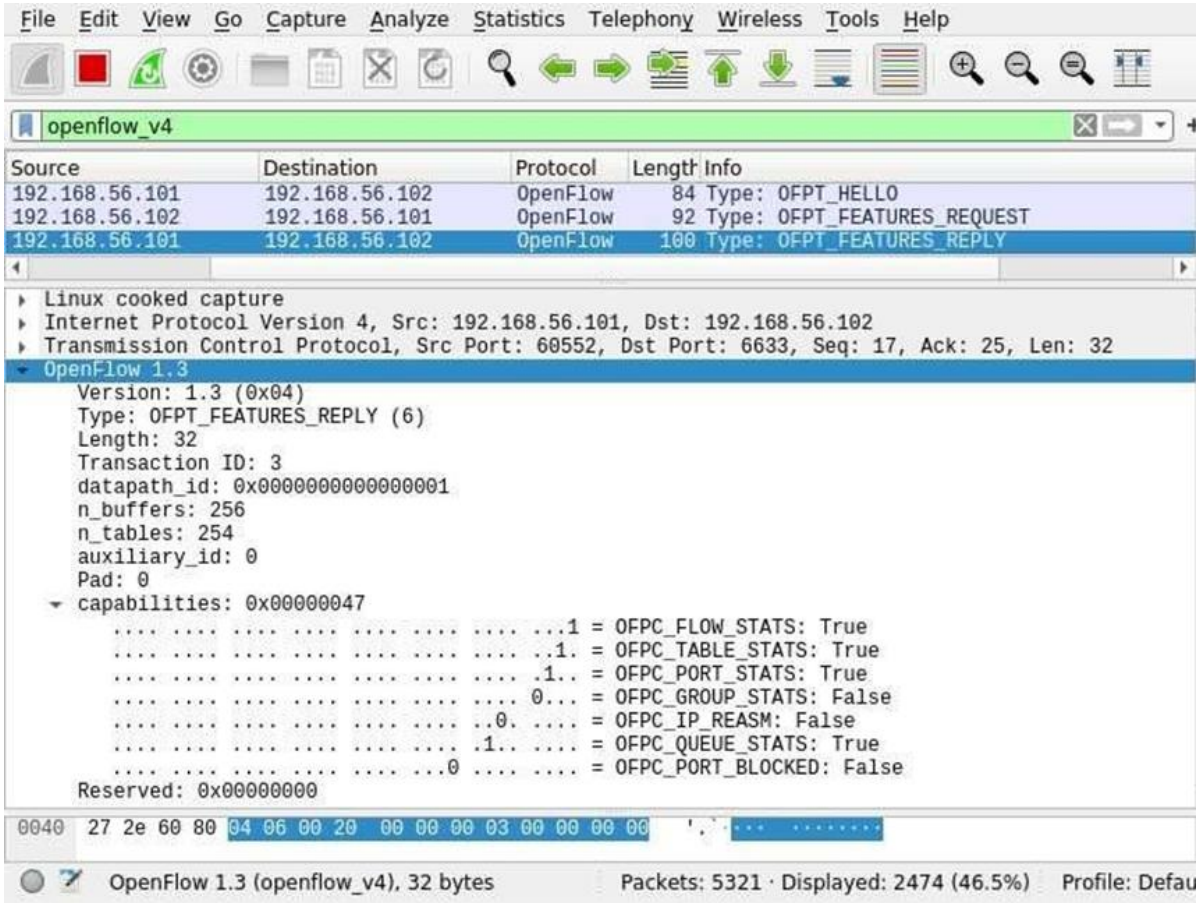

Рисунок 3.13 – Повідомлення OFPT_FEATURES_REPLY

У повідомленні OFPT_MULTIPART_REPLY, OFPMP_DESC, містяться атрибути, що показують хто виробник комутатора Open vSwitch та версії його апаратного і програмного забезпечення (рис. 3.14) [18, 25].

Повідомлення FLOW_MOD дозволяє SDN-контроллеру змінювати стан OpenFlow-комутатора (рис. 3.15). Це повідомлення починається зі стандартного заголовка та супроводжується файлом `cookie`, який являє собою непрозоре поле, що встановлене SDN-контроллером, його маскою, id-таблиці та командою, яка визначає тип модифікації таблиці потоків. Крім цих полів, повідомлення FLOW_MOD має наступні поля [16]:

- `idle_timeout` – поле, яке показує кількість секунд, після яких запис потоку видаляється з таблиці потоків, оскільки жоден з пакетів не відповідає їй;

- `hard_timeout` – поле, яке показує кількість секунд, після яких запис потоку видаляється з таблиці потоків незалежно від того, чи співпадають пакети з нею чи ні;

- пріоритет – поле, у якому показується порядок співпадінь, які перекриваються з вищими числами, що представляють вищий пріоритет;

`Buffer_id` – це поле надає ідентифікатор буферизованого пакета.

Далі у повідомленні FLOW_MOD йдуть out_port, out_group і прапори. В кінці повідомлення містяться доповнення («`pad`»), де вказуються відповідність («`match`») та інструкції [16].

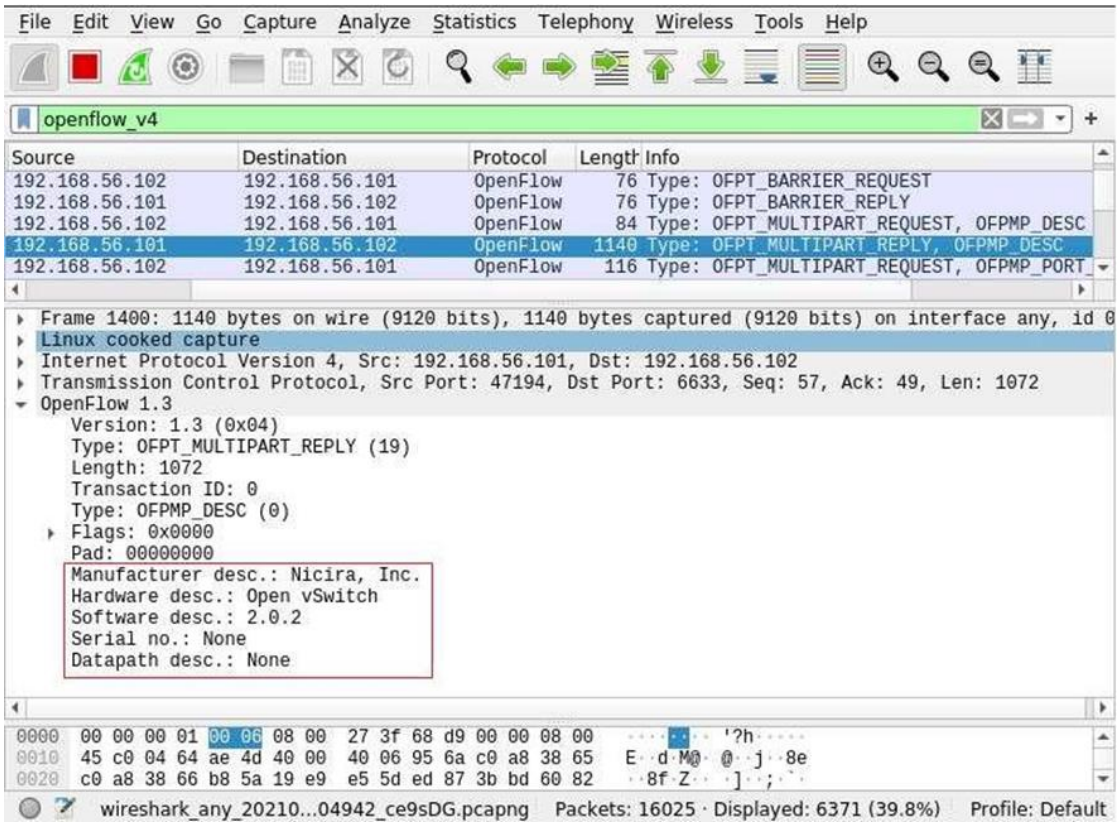

Рисунок 3.14 – Повідомлення OFPT_MULTIPART_REPLY, OFPMP_DESC

Рисунок 3.15 – Формат повідомлення FLOW_MOD

Повідомлення PACKET_IN надає спосіб для OpenFlow-комутатора відправити захоплений пакет SDN-контроллеру. Це може бути необхідним через явну дію у результаті збігу, що запитує таку поведінку, або через пропуск таблиці потоків. Як і будь яке інше повідомлення, це повідомлення має заголовок, за яким йде ідентифікатор буфера (`Buffer_id`), який надає унікальне значення ID для відстеження буферизованого пакета. Довжина знайденого пакета вказується у полі «`total_len`». У полі причини («`reason`») вказується, чому пакет був знайдений та відправлений. У полі «`tbl_id`» вказується ID таблиці, що переглядалась. Поле «`cookie`» застосовується для здійснення ідентифікації визначеного потоку, який буде відповідати цьому пакету. Нарешті захоплена частина пакету починається відразу з доповнень «`pad`» (рис. 3.16) [16].

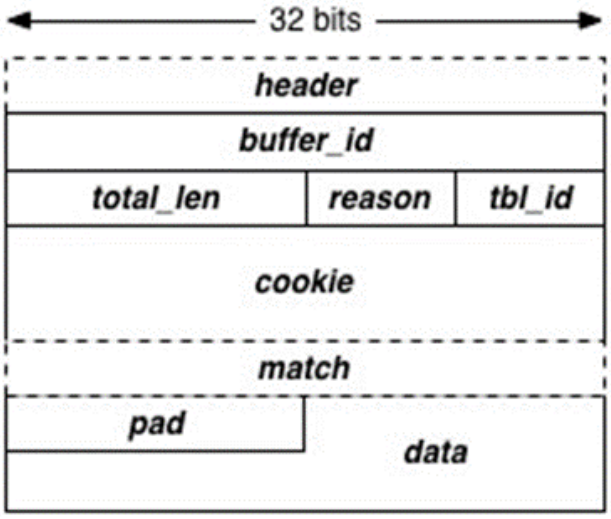

Рисунок 3.16 – Формат повідомлення PACKET_IN

Застосовуючи мережний емулятор Mininet та утиліти `ping` здійснюємо перевірку зв'язку з вузла h1 на h2 (команда «`h1 ping h2`») (рис. 3.17) [16].

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.489 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.062 ms
^C
--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.062/0.275/0.489/0.214 ms
```

Рисунок 3.17 – Пінгування з вузла h1 до вузла h2

На рис. 3.18 демонструється виведення повідомлення PACKET_IN від OpenFlow-комутатора до SDN-контроллера, оскільки для вхідного пакета на комутаторі не знайдено співпадінь у наборі записів і вузол h1 ще не знає, як дістатися до вузла h2. Це робиться у повідомленні протоколу визначення адреси (Address Resolution Protocol, ARP), яке інкапсульоване протоколом OpenFlow через протокол TCP. SDN-контроллер за допомогою OpenFlow-повідомлення типу «контроллер - комутатор» FLOW_MOD дає команду комутатору запам'ятати розташування вузла h1. Далі SDN-контроллер розсилає ARP-запит, що є інкапсульованим у повідомлення PACKET_IN, тільки на ті порти OpenFlow комутатора, до яких підключені ці хостові вузли [16].

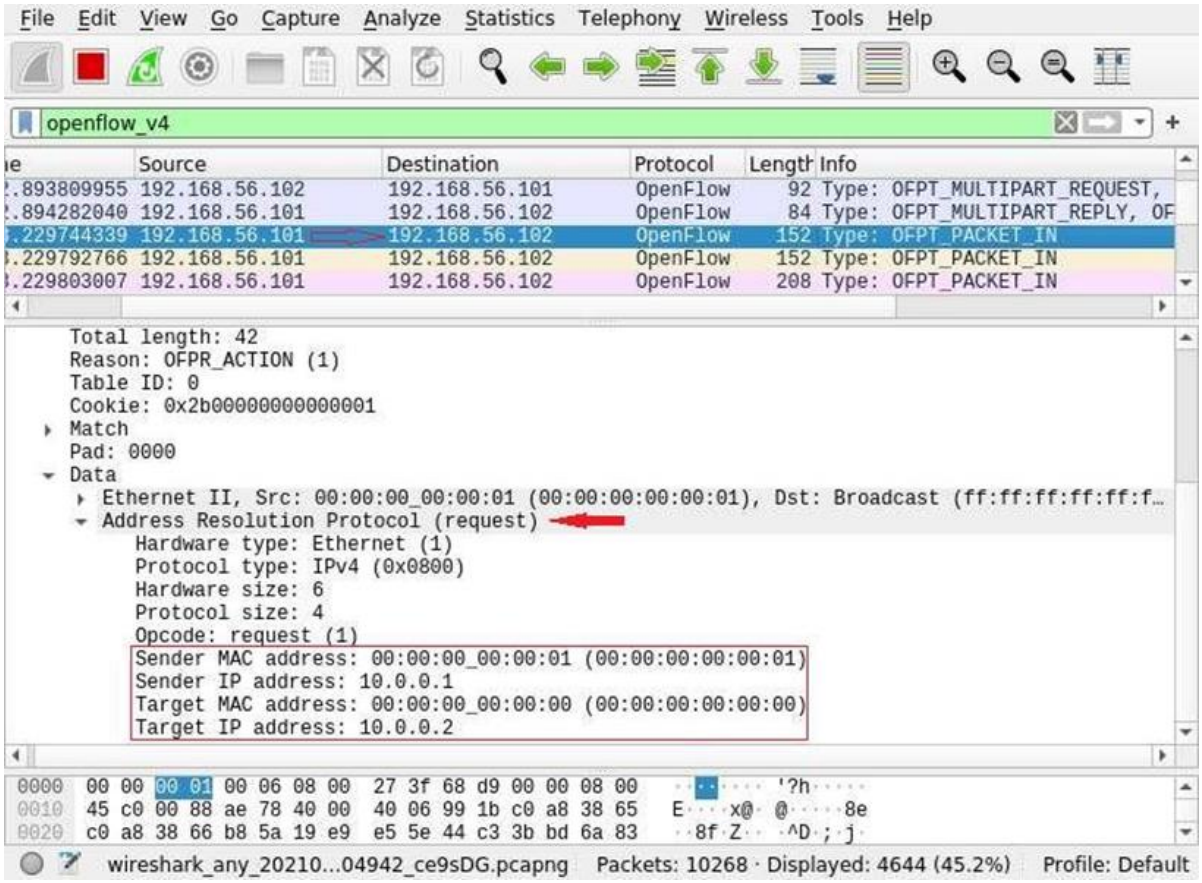

Рисунок 3.18 – Надсилання ARP-запиту від вузла h1 до вузла h2

Вузол h2 у разі виявлення співпадіння запитаної MAC-адреси з власною, відправляє в мережу ARP-відповідь, яка ретранслюється OpenFlow-комутатором на SDN-контроллер. Звідси контроллер тепер буде знати місцезнаходження вузла h2 і встановить відповідне правило у таблиці потоків комутатора. Далі після цього ARP-відповідь передається до вузла h1, що показано на рис. 3.19 [16].

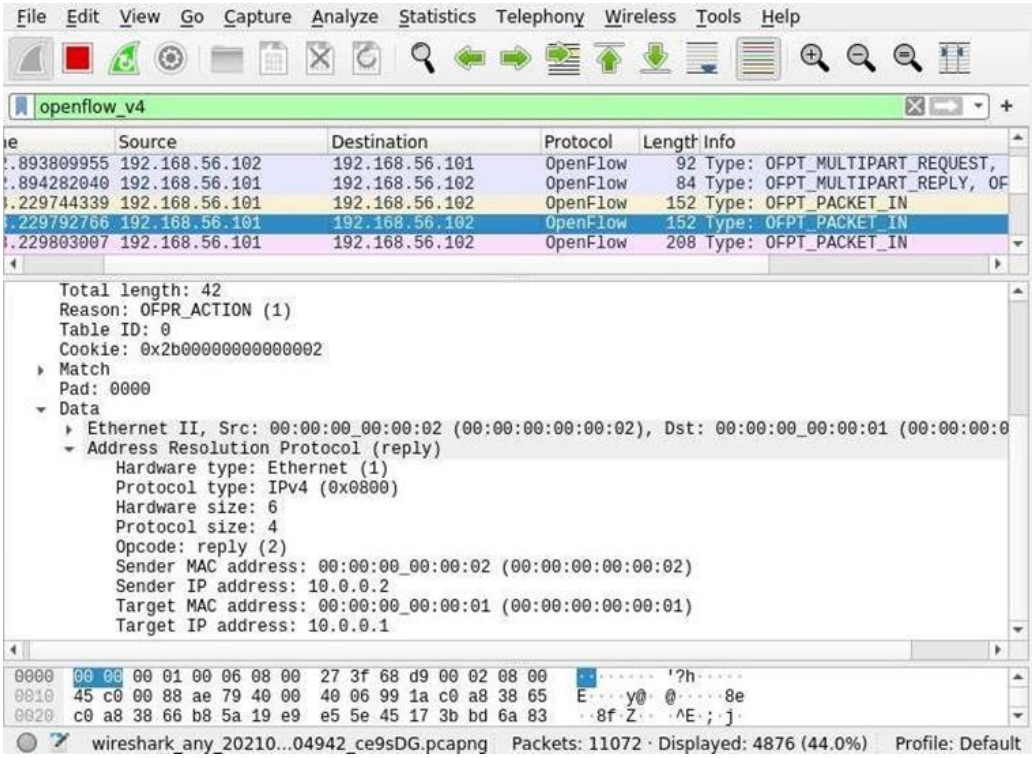

Рисунок 3.19 – ARP-відповідь від вузла h2

Після додавання записів у таблицю потоків, вузол h1 може відправити пакет із запитом до вузла h2, використовуючи протокол керуючих повідомлень Internet (Internet Control Message Protocol, ICMP), для перевірки його доступності (рис. 3.20).

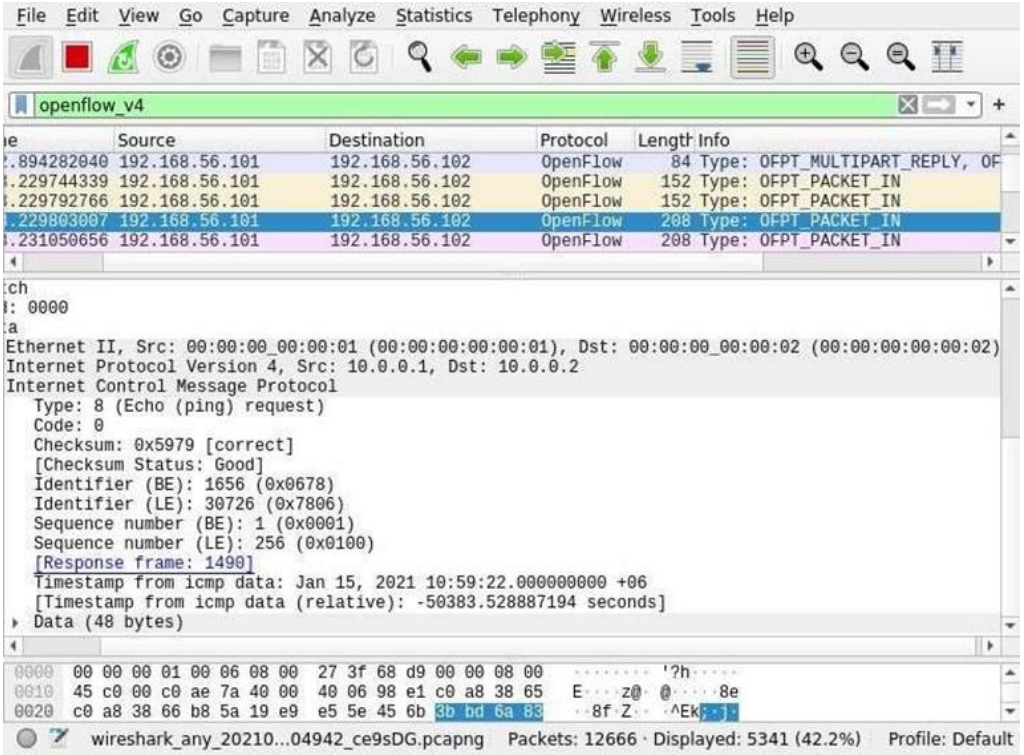

Рисунок 3.20 – Запит ICMP до вузла h2

Отримавши запит, вузол h2 має надіслати відповідний пакет ICMP до вузла h1 з підтвердженням доступності (рис. 3.21).

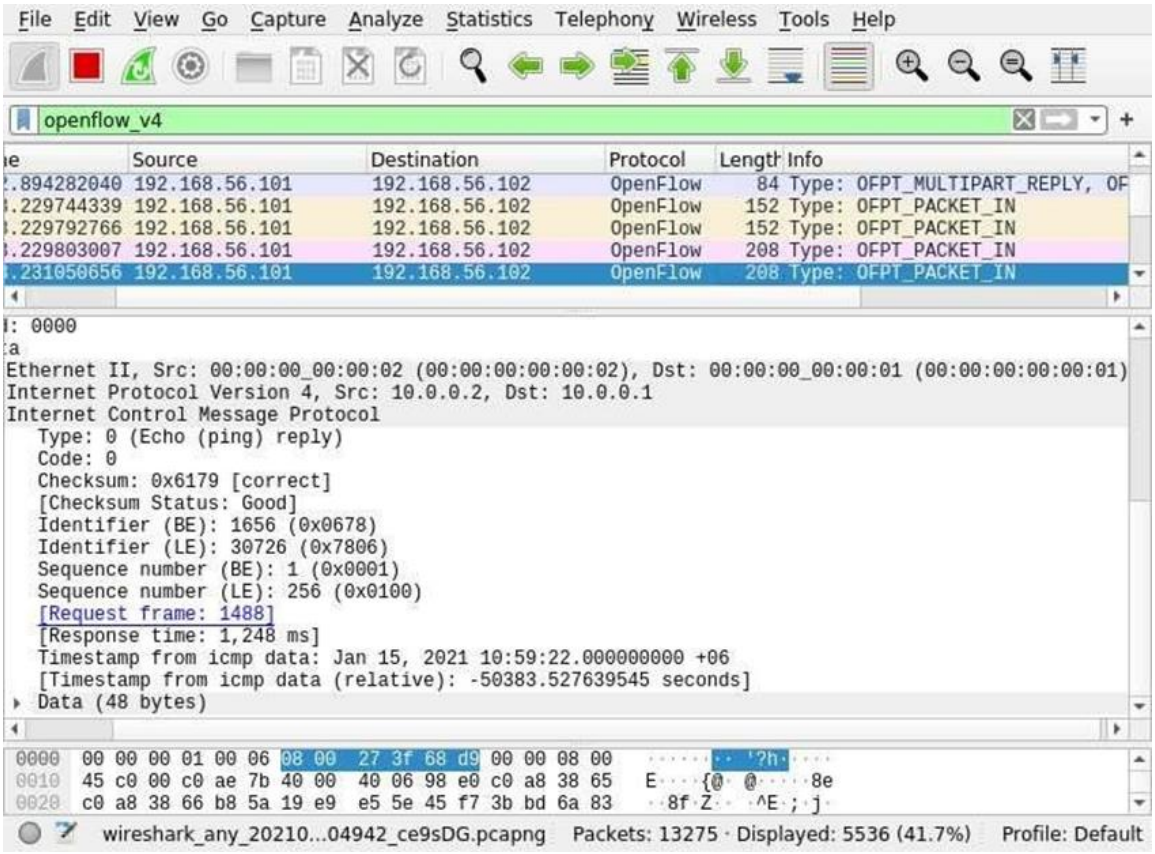

Рисунок 3.21 – Відповідь ICMP до вузла h1

3.3 Створення записів потоків у таблиці потоків

Для проведення дослідження впливу записів потоку на трафік також будемо використовувати створену у VM Mininet топологію мережі з одним комутатором і трьома хостами. Але для проведення даних досліджень команда організації топології мережі трохи зміниться: `sudo mn -- topo=single,3 -- controller=none -- mac`, що наведено на рис. 3.22. В цьому прикладі параметр `-- controller=none` виключає OpenFlow-контроллер з топології мережі. Параметр `topo` визначає мережну топологію, наприклад, якщо змінити значення параметра на «`--topo = 2,4`», то топологія мережі буде мати два OpenFlow-комутатора, а вони, у свою чергу, матимуть по 4 хости. Команда `dump` дозволяє вивести інформацію про всі вузли мережі та хости з параметрами інтерфейсів. Команда `net` показує як з'єднані між собою хости та вузли [11, 23].

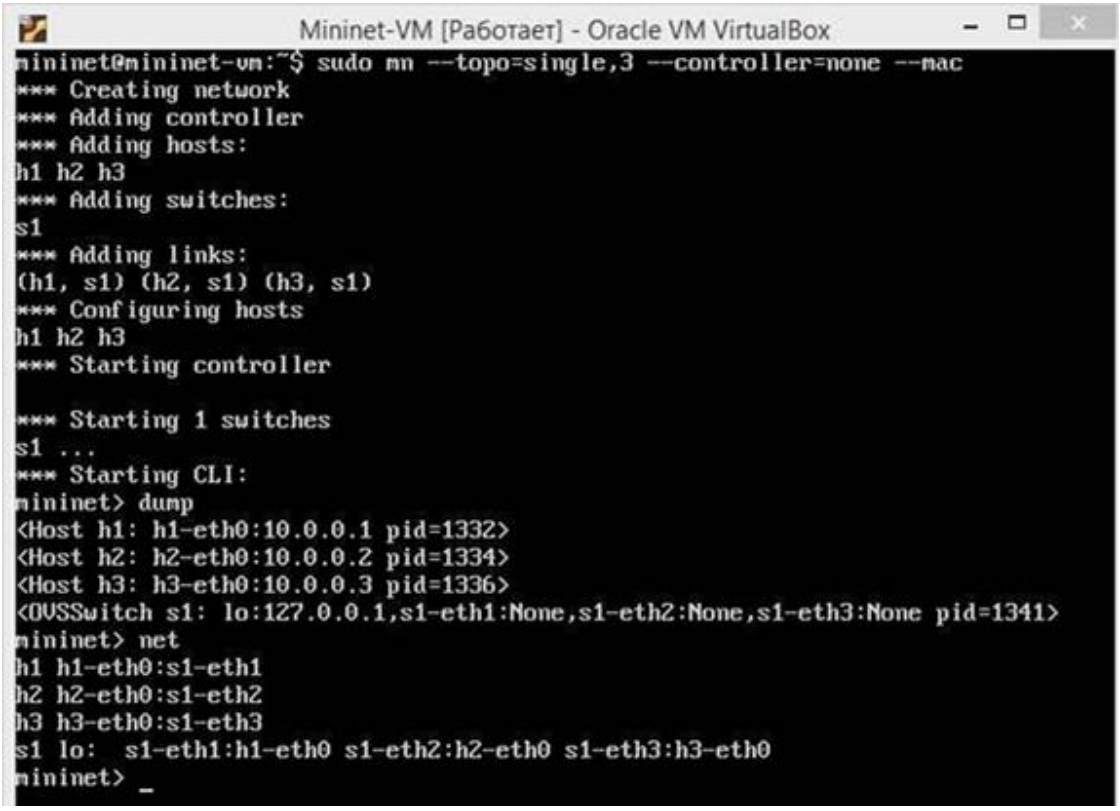

Рисунок 3.22 – Створення мережі із OpenFlow-комутатором на VM Mininet

Подивитись параметри OpenFlow-комутатора можна використовуючи команда `sh ovs-ofctl show s1`, як це показано на рис. 3.23 [11].

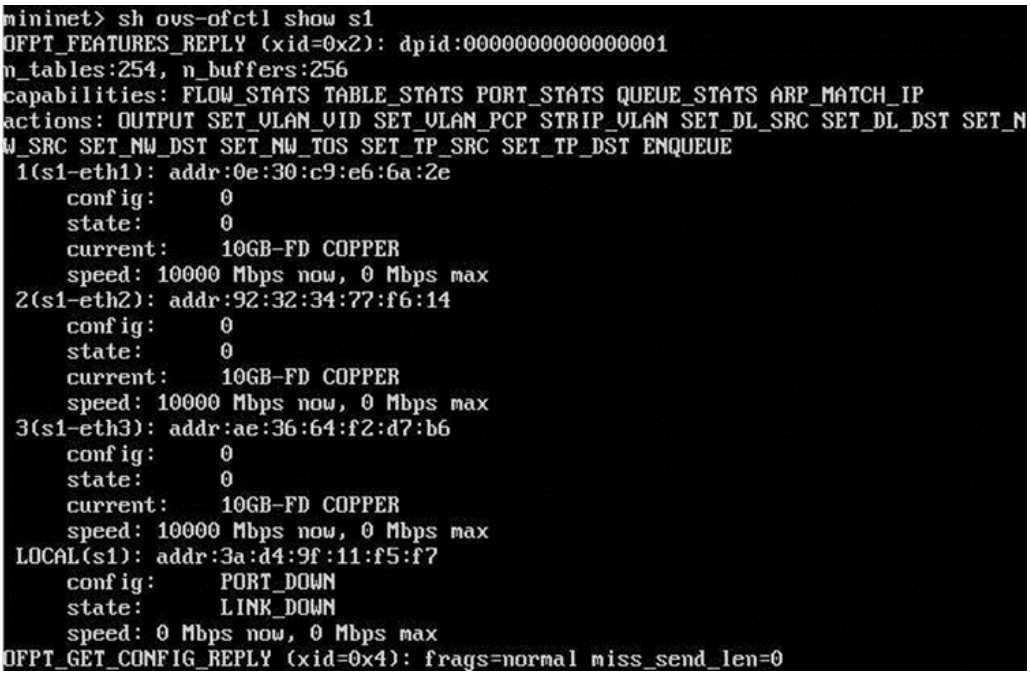

Рисунок 3.23 – Вікно відображення параметрів OpenFlow-комутатора

Для перевірки зв'язку між хостами необхідно додати записи OpenFlow. Треба звернути увагу, що якщо не задати жодного правила на OpenFlow-комутатор, то зв'язку в локальній мережі не буде, що видно на рис. 3.24. Щоб створити стандартний запис OpenFlow, необхідно застосувати команду `sh ovs-ofctl add-flow s1 action=normal` [11].

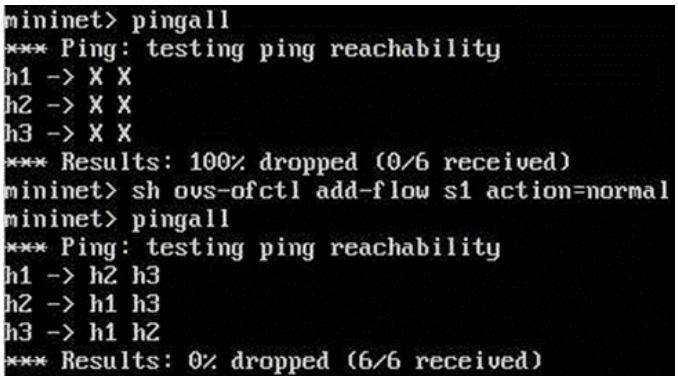

Рисунок 3.24 – Здійснення перевірки зв'язку між хостами

Щоб отримати дані про записи потоку у таблиці потоків застосовується команда `sh ovs-ofctl dump-flows` s1, як це видно із рис. 3.25. З іншого боку, для здійснення видалення всі записи потоків із таблиці потоків, необхідно скористатися командою `sh ovs-ofctl del-flows s1` [11].

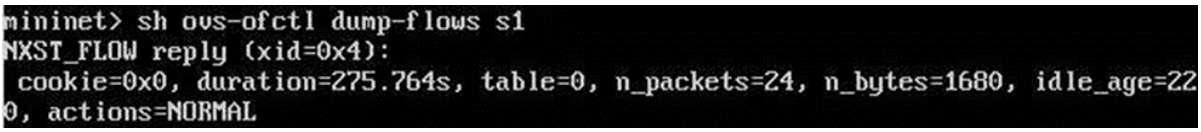

Рисунок 3.25 – Отримання даних про записи із таблиці потоків

Проведемо створення записів потоків у таблиці потоків, які будуть давати вказівки OpenFlow-комутатору, на який із портів він має відправляти трафік, що надійшов із певного порту. Для цього будемо застосовувати команди, що були описані вище (рис. 3.26). Такий запис є першим рівнем відповідності `Layer 1 matching`.

Тепер OpenFlow-комутатор може застосувати правила, що йому були задані у разі передачі трафіку з порту 1 на порт 2 або навпаки. Із рис. 3.26 видно, що немає зв'язку із хостом h3 через відсутність правил для порту, до якого він є підключеним. Після введення команди `sh ovs-ofctl dump-flows s1` можна бачити зміни у таблиці потоків OpenFlow-комутатора [11].

```
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=1,actions=output:2
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=2,actions=output:1
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.278 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.051 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.055 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.051/0.128/0.278/0.106 ms
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=456.581s, table=0, n_packets=4, n_bytes=336, idle_age=135,
 priority=500,in_port=1 actions=output:2
 cookie=0x0, duration=253.48s, table=0, n_packets=4, n_bytes=336, idle_age=135,
priority=500,in_port=2 actions=output:1
mininet> h3 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2010ms
```

Рисунок 3.26 – Таблиця потоків OpenFlow типу `Layer 1 matching`

Здійснювати управління правилами таблиці потоків дозволяє поле пріоритету (`priority`). Проаналізуємо, як буде впливати додавання правила з великим значенням `priority` на передачу трафіку, а потім видалимо його із застосуванням параметра `–strict`, як це показано на рис. 3.27 [11].

```
mininet> sh ovs-ofctl add-flow s1 priority=32768,actions=drop
mininet> h1 ping -c3 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 1999ms

mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=1375.973s, table=0, n_packets=4, n_bytes=336, idle_age=105
4, priority=500,in_port=1 actions=output:2
 cookie=0x0, duration=1172.872s, table=0, n_packets=4, n_bytes=336, idle_age=105
4, priority=500,in_port=2 actions=output:1
 cookie=0x0, duration=162.691s, table=0, n_packets=6, n_bytes=420, idle_age=99,
actions=drop
mininet> sh ovs-ofctl del-flows --strict
ovs-ofctl: 'del-flows' command requires at least 1 arguments
mininet> sh ovs-ofctl del-flows s1 --strict
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=1765.764s, table=0, n_packets=4, n_bytes=336, idle_age=144
4, priority=500,in_port=1 actions=output:2
 cookie=0x0, duration=1562.663s, table=0, n_packets=4, n_bytes=336, idle_age=144
4, priority=500,in_port=2 actions=output:1
```

Рисунок 3.27 – Застосування пріоритету записів у таблиці потоків

Ще одне правило буде порівнювати MAC-адресу джерела трафіку та MAC-адресу призначення, і відправлятиме трафік, відповідно до значення параметра `output` (рис. 3.28). Такий запис є другим рівнем відповідності і називається `Layer 2 matching` [11].

```
mininet> sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00
:02,actions=output:2
mininet> sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00
:01,actions=output:1
mininet> sh ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,actions=flood
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X
h2 -> h1 X
h3 -> X X
*** Results: 66% dropped (2/6 received)
```

Рисунок 3.28 – Таблиця потоків OpenFlow типу `Layer 2 matching`

Для коректної роботи `Layer 2 matching` потрібно встановити правило, що буде контролювати ARP-запити. Перші 2 рядки команди описують передачу трафіку відповіно до фізичних адрес, але для того, щоб OpenFlow-комутатор отримав відомості про те, за яким IP закріплено потрібну MAC-адресу, потрібно описати правило, яке дозволяє пересилати широкомовний трафік. В цьому прикладі дія `flood` дає можливість надсилати запити протоколу ARP (ARP-REQUEST) на всі порти, окрім того, з якого було згенеровано запит [11].

Третій запис у таблиці потоків описує політику передачі трафіку на основі IP адресації та поля IP-пакету, що показує тип обслуговування (Type of Service, ToS). Такий запис є третім рівнем відповідності, тобто називається `Layer 3 matching`. Формується запис `sh ovs-ofctl add-flow s1 priority=500,ip,nw_src=10.0.0.3,actions=mod_nw_tos:184,normal`. У цьому записі визначається тип обслуговування (ToS) за моделлю диференційованих послуг (Differentiated Services, DiffServ), яка дозволяє маркувати трафік спеціальною міткою поля коду диференційованих послуг (Differenciated Services Code Point, DSCP), що визначає його пріоритет. Якщо подивитися уважно на сам запис, то можна побачити, що параметр `priority` відрізняється від DSCP тим, що задає порядок порівняння пакетів трафіку, що був переданий із записами в таблиці потоків. Якщо значення `priority` зробити

більшим у записі для ToS, то спочатку пакет буде співставлений за цим записом, а потім за всіма іншими записами. Приклад створення запису за типом `Layer 3 matching`, наведений на рис. 3.29. Треба зазначити, що така структура таблиць потоків є схожою на роботу списків доступу у процесі маршрутизації трафіку в традиційних мережах [11].

```
mininet> sh ovs-ofctl add-flow s1 priority=500,ip,nw_src=10.0.0.0/24,nw_dst=10.0
.0.0/24,actions=normal
mininet>
mininet> sh ovs-ofctl add-flow s1 priority=500,ip,nw_src=10.0.0.3,actions=mod_nw
_tos:184,normal
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.1,actions=output:1
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.2,actions=output:2
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.3,actions=output:3
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=80.825s, table=0, n_packets=4, n_bytes=168, idle_age=52, a
rp,arp_tpa=10.0.0.3 actions=output:3
 cookie=0x0, duration=181.265s, table=0, n_packets=4, n_bytes=168, idle_age=52,
arp,arp_tpa=10.0.0.2 actions=output:2
 cookie=0x0, duration=259.772s, table=0, n_packets=4, n_bytes=168, idle_age=52,
arp,arp_tpa=10.0.0.1 actions=output:1
 cookie=0x0, duration=376.809s, table=0, n_packets=0, n_bytes=0, idle_age=376, p
riority=500,ip,nw_src=10.0.0.3 actions=mod_nw_tos:184,NORMAL
 cookie=0x0, duration=1306.425s, table=0, n_packets=12, n_bytes=1176, idle_age=5
7, priority=500,ip,nw_src=10.0.0.0/24,nw_dst=10.0.0.0/24 actions=NORMAL
```

Рисунок 3.29 – Таблиця потоків OpenFlow типу Layer 3 matching

Таким чином, проведений аналіз та моделювання функціонування SDN показав, що застосування OpenFlow дає змогу спростити безліч задач, а створення і використання таблиці опрацювання потоків даних забезпечує багаторівневу безпеку. Зокрема показано, як за допомогою SDN-контролера здійснювати управління OpenFlow-комутатором за допомогою внутрішньої мови командного рядка емулятора мереж Mininet. Для більш детальної та практичної демонстрації практичного функціонування SDN цю модель необхідно реалізувати на мережному обладнанні, яке буде підтримувати роботу протоколу OpenFlow. Зокрема у реальній моделі можна буде на практиці показати значимість протоколу OpenFlow в управлінні мережним обладнанням та передачею пакетів.

ВИСНОВКИ

В магістерскій кваліфікаційній роботі проаналізовані концептуальні принципи організації і функціонування програмно-конфігурованих мереж на базі протоколу OpenFlow, який забезпечує віддалене управління мережними пристроями за посередництвом контроллера завдяки своїй здатності контролювати таблиці потоків пакетів, що створюються цими пристроями, зокрема OpenFlow-комутаторами.

У першому розділі кваліфікаційної роботи розглянуті загальні принципи організації та функціонування програмно-конфігурованих мереж на базі протоколу OpenFlow. Зазначено, що в SDN правила обробки даних реалізуються у вигляді програмних модулів, які є відокремленими від апаратних ресурсів, а управління ними реалізується в окремому програмному модулі, що називається контроллером. Зокрема, використовуючи концепцію SDN, адміністратори можуть контролювати потоки даних, а також із центрального вузла змінювати параметри мережного обладнання (комутаторів, маршрутизаторів, тощо). При цьому додаток, що забезпечує управління, також є програмним модулем, що виключає необхідність роботи окремо з кожним із цих пристроїв [2, 6].

Також у першому розділі проаналізована концептуальна архітектура SDN, що складається з трьох рівнів: інфраструктури або передачі даних, управління і додатків. Аналіз архітектури та загальних особливостей SDN показав, що для управління передачею даних у мережі за посередництвом SDN-контроллера, необхідна наявність ефективного протоколу відкритого управління, в якості якого у роботі був досліджений протокол OpenFlow. Протокол OpenFlow є відкритим, головною задачею якого є управління мережним обладнанням різних виробників. Функціонування протоколу OpenFlow полягає в модифікації змісту таблиці передачі пакетів або таблиці потоків, що формуються усередині мережного пристрою, тому він не є ні протоколом комутації, ні маршрутизації. Принцип функціонування OpenFlow за аналогією з архітектурою SDN також представляють у вигляді трьох площин: адміністрування, управління і передачі даних [6, 7]

У другому розділі кваліфікаційної роботи проведений аналіз принципів функціонування SDN-контроллера і комутатора OpenFlow та їх взаємодія в процесі передачі трафіку. Контроллер проводить опитування комутаторів мережі, аби визначити їх стан, а також внутрішні параметри, за протоколом STA будує

деревоподібну топологію, що забезпечує зв’язність всіх сегментів мережі і формує єдиний шляху між будь-якими двома комутаторами мереж. Комутатори у свою чергу конфігуруються так, щоб повідомляти нові адреси джерел контролеру, тому контроллер має змогу сповістити всіх інших комутаторів про наявність найкращого маршруту трафіку від цього нового джерела. В якості прикладу проаналізована архітектура SDN-контроллера HP Virtual Application Networks SDN Controller від компанії Hewlett-Packard (рис. 2.1) [13, 15].

У роботі звернена увага на те, що здатність контроллера сповіщати комутатори про те, як здійснювати передачу пакетів, є основою концепції SDN. Тому у другому розділі кваліфікаційної роботи також значна увага приділена аналізу архітектури, функціонуванню та компонентам OpenFlow комутатора, що працює з протоколом, та його компоненти. Показано, що основними компонентами такого комутатора є одна або декілька таблиць потоків та групова таблиця, в задачу яких входить здійснення пошуку та передачі пакетів. Крім того, він має один або кілька каналів OpenFlow до зовнішнього контроллеру (рис. 2.2). Комутатор зв'язується з контроллером, а контроллер у свою чергу здійснює управління комутатором за протоколом OpenFlow [4, 16].

Протокол OpenFlow надає SDN-контроллеру відповідні інструкції для додавання, оновлення та видалення записів потоків у таблицях потоків. Кожна така таблиця потоків у комутаторі включає у себе набір записів потоків, а кожен запис, що входить у набір створюється кожним пакетом, що надходить на вхідний порт комутатора. Інструкції, що пов'язані з кожним записом потоку – це набір дій, що керують процесом конвеєрної обробки. Ці дії описують передачу пакетів та їх модифікацію пакетів, а також обробку групових таблиць. Інструкції конвеєрної обробки дозволяють передавати пакеті до наступної таблиці для їх подальшої обробки. Дії, які зв'язані із записами потоку окремих пакетів, можуть також робити направлення пакетів до групи, яка буде визначати додаткову обробку. Групи надають набори дій, а також надають можливості кільком записам потоку здійснювати передачу на один ідентифікатор. Ці записи групи розміщуються у групових таблицях. Кожен запис групи має список блоків дій (команд) з певною семантикою, яка залежить від типу групи. Порти OpenFlow являють собою мережні інтерфейси комутатора для передачі пакетів між процесом OpenFlow і рештою мережі [16].

При здійсненні аналізу принципів функціонування OpenFlow-комутатора особлива увага була звернута на конвеєрну обробка пакетів в ньому. Зокрема

показано, що конвеєр кожного OpenFlow-комутатора містить одну або декілька таблиць потоків, кожна з яких має по кілька записів потоків. Обробка у конвеєрі OpenFlow-комутатора показує, як пакети можуть взаємодіяти з цими таблицями потоків. При обробці у таблиці потоків пакет співставляється із її записами потоків. Якщо запис потоку буде знайдено, то починається виконання відповідного набору інструкцій цього запису потоку. Ці інструкції можуть робити перенаправлення пакету до іншої таблиці потоків. У наступній таблиці цей процес повторюється знову. Якщо відповідний запис потоку не надсилає пакети до наступної таблиці потоків, то обробка пакету конвеєром зупиняється на цій таблиці, де пакет обробляється з відповідним набором дій і зазвичай далі здійснюється його пересилання на вихідний порт. Якщо співпадіння пакета із записом потоку в таблиці потоків немає, то у цьому випадку вона пропускається. Дії пакету під час пропуску таблиці будуть залежати від наявної конфігурації таблиці. Також у роботі проаналізовані таблиці потоків і групова таблиця OpenFlow-комутатора та їх структурні складові [18, 19].

У третьому розділі кваліфікаційної роботи зроблено дослідження та аналіз принципів функціонування програмно-конфігурованої мережі у процесі її програмного моделювання у середовищі мережного емулятора Mininet. Для проведення такого аналізу було створено дві віртуальні машини (VM) у ПЗ VirtualBox. На одній було розгорнуто мережний емулятор Mininet, а на іншій – під управлінням ОС Ubuntu було розгорнуто SDN-контроллер OpenDaylight ODL (рис. 3.1). Ці VM взаємодіють між собою через мережу хоста. На VM Mininet моделюється мережа з одним OpenFlow-комутатором Open vSwitch (IP-адреса 193.168.56.101) і трьома хостами, що підключені до нього. Управління здійснюється SDN-контроллером ODL (IP-адреса 193.168.56.102) (рис. 3.3). Далі ця модель топології мережі візуалізується в графічному інтерфейсі ODL. Він також функціонує на VM і має IP-адресу: 193.168.56.102 (рис. 3.8) [21, 23, 24].

Для перегляду повідомлень протоколу OpenFlow, якими обмінюються SDN-контроллер ODL та OpenFlow-комутатор Open vSwitch, у мережі, застосовується аналізатор пакетів Wireshark. Кожне повідомлення протоколу OpenFlow починається з однакової структури заголовка (рис. 3.10). Використовуючи фільтр відображення Wireshark для протоколу OpenFlow версії 1.3, можна продивитись знайдені пакети (рис. 3.11). Зокрема показано, що перший знайдений пакет – це повідомлення HELLO, яке було відправлене від комутатора (193.168.56.101) до SDN-контроллера (193.168.56.102). Цей пакет використовується для узгодження

версії протоколу OpenFlow. Далі були проаналізовані за допомогою пакету Wireshark інші типи повідомлень протоколу OpenFlow, за допомогою яких здійснюється обробка і передача пакетів між компонентами мережі, що створена на VM Mininet (рис. 3.3) та показана у графічному інтерфейсі SDN-контроллера OpenDaylight (рис. 3.8) [18, 25].

Для дослідження безпосередньо самого процесу обміну трафіком між вузлами мережі, що моделюється на VM Mininet, у роботі також було досліджено принципи створення записів потоків у таблиці потоків (рис. 3.22). Для проведення цих досліджень команда організації топології мережі була трохи змінена. У процесі досліджень проаналізовані різні команди протоколу OpenFlow в емуляторі на VM Mininet, що впливають на створення записів потоків пакетів у таблиці потоків у процесі обробки їх у вузлах мережі. Показано принципи формування трьох типів таблиць потоків (рис. 3.26, 3.28 і 3.29) [11].

Проведений аналіз та моделювання функціонування SDN показав, що застосування OpenFlow дає змогу спростити безліч задач, а створення і використання таблиці опрацювання потоків даних забезпечує багаторівневу безпеку. Зокрема показано, як за допомогою SDN-контролера здійснювати управління OpenFlow-комутатором за допомогою внутрішньої мови командного рядка емулятора мереж Mininet. Для більш детальної та практичної демонстрації практичного функціонування SDN цю модель необхідно реалізувати на мережному обладнанні, яке буде підтримувати роботу протоколу OpenFlow. Зокрема у реальній моделі можна буде на практиці показати значимість протоколу OpenFlow в управлінні мережним обладнанням та передачею пакетів [11].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Fei Hu. Network Innovation through OpenFlow and SDN. Principles and Design. CRC Press. 1st edition. February 2014. – 520 p.

2 Thomas D. Nadeau and Ken Gray. SDN Software Defined Networks. OReilly Media. September 7, 2013. – 384 p.

3. Siamak Azodolmolky. Software Defined Networking with OpenFlow. Packt Publishing. October 25, 2013. – 152 p.

4. Коломоєць М.С. Особливості організації управління в програмно-конфігурованих мережах на базі протоколу OpenFlow / М.С. Коломоєць, Ю.М. Колтун // матеріали 11-ої міжнародної науково-технічної конференції «Проблеми інформатизації». Том 3. – Черкаси –Баку – Харків – Бельсько-Бяла. – 16 - 17 листопада, 2023 р. – С. 94.

5. Батура Т.В. Облачные технологии: основные модели, приложения, концепции и тенденции развития / Т.В. Батура, Ф.А. Мурзин, Д.Ф. Семич // Программные продукты и системы. – 2014. – №3. – С. 64 – 72.

6. Алексей Шалагинов Принцип работы SDN [Електронний ресурс] / Telecom & IT. – 2019. – Режим доступу: https://shalaginov.com/2019/10/19/6521/.

7. Логинов С.С. Об уровнях управления в программно-конфигурируемой сети (SDN)/ С.С. Логинов // T-Comm: Телекоммуникации и транспорт. – 2017. – Том 11. – №3. – С. 50-55.

8. Braun W. Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices / Wolfgang Braun, Michael Menth // Future Internet. – #6. – 2014. – P. 302-336.

9. OpenFlow Switch Specification. Version 1.5.1 (Protocol version 0x06) [Електронний ресурс]. – 2015. – Режим доступу: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

10. Kurose J. Computer Networking: A Top-Down Approach. Chapter 4. Network Layer: The Data Plane [Електронний ресурс] / Jim Kurose, Keith Ross. – 2016. – – Режим доступу: https://en.ppt-online.org/348044/.

11. Ильшаев И.А. Исследование принципов работы протокола OpenFlow в программно-конфигурируемых сетях / И.А. Ильшаев, А.В. Красов, И.А. Ушаков // Труды учебных заведений связи. – Т.3.– №2. – 2017. – С. 16 – 27.

12. Алексей Шалагинов OpenFlow: маршрутизация или коммутация? [Електронний ресурс] / Telecom & IT. – 2019. – Режим доступу: https://shalaginov.com/2019/10/21/6534/.

13. Алексей Шалагинов Сетевые технологии (16). Программно-конфигурируемые (Software-Defined) коммутаторы [Електронний ресурс] / Telecom & IT. – 2021. – Режим доступу: https://shalaginov.com/2021/08/19/software-defined-switch/.

14. Олифер В. Г. Новые технологии и оборудование IP-сетей / Виктор Олифер, Наталья Олифер. – СПб. и др.: BHV, 2000. – 512 с.:

15. SDN: альтернатива или дополнение к традиционным сетям? [Електронний ресурс] // Блог компании Hewlett Packard Enterprise. – 2015. – Режим доступу: https://shalaginov.com/2021/08/19/software-defined-switch/.

16. OpenFlow Switch Specification [Електронний ресурс]. – 2015. – P. 18 - 30 – Режим доступу: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switchv1.5.1.pdf.

17. Алексей Шалагинов Сетевые технологии (16-1). Программно-конфигурируемые (Software-Defined) коммутаторы – Порты OpenFlow [Електронний ресурс] / Telecom & IT. – 2021. – Режим доступу: https://shalaginov.com/2021/08/23/openflow-ports/.

18. Семенов Ю.А. Сетевая технология OpenFlow (SDN) [Електронний ресурс] / Ю.А. Семенов. – Доступ здійснено 18.01.2024. – Режим доступу: http://book.itep.ru/4/41/openflow.htm

19. Владыко А.Г. Тестирование контроллеров программно-конфигурируемой сети на базе модельной сети / А.Г. Владыко, Н.А. Матвиенко, М.И. Новиков, Р.В. Киричек // Информационные технологии и телекоммуникации. – Т.4. – №1. – 2016. – С. 17 -28 .

20. Коляденко Ю.Ю. Программно-конфигурируемые сети на базе протокола OpenFlow и их характеристики [Електронний ресурс]/ Ю. Ю. Коляденко, Е. Э. Белоусова // ScienceRise. – № 3(2). – 2016. – С. 11 - 16. – Режим доступу: http://nbuv.gov.ua/UJRN/texc_2016_3%282%29__4.

21. VirtualBox. Welcome to VirtualBox.org! [Електронний ресурс] – Доступ здійснено 05.01.2024. – Режим доступу: https://www.virtualbox.org.

22. Савельев Алексей Решения Microsoft для виртуализации ИТ-инфраструктуры предприятий [Електронний ресурс] / Алексей Савельев // Учебный Internet-курс. – 2012. – Режим доступу до ресурсу: https://intuit.ru/studies/courses/2324/624/info.

23 Mininet. An Instant Virtual Network on your Laptop (or other PC) http://mininet.org/

24. Welcome to OpenDaylight Documentation [Електронний ресурс] / – Доступ здійснено 05.01.2024. – Режим доступу до ресурсу: https://docs.opendaylight.org/en/stable-potassium/.

25. Wireshark. OpenFlow: Software Defined Networking (SDN) Southbound API standard protocol [Електронний ресурс] – Доступ здійснено 05.01.2024. – Режим доступу до ресурсу: https://wiki.wireshark.org/OpenFlow .