Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи аналізу роботи протоколу OpenFlow у програмно-конфігурованих мережах SDN

(тема)

Виконав:
студент II курсу, групи СПм-21–1
Пушкар А.І.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

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

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: доц. Іванісенко І.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп’ютерна інженерія
(код і повна назва)

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

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(Підпис)
"____" _____ 2022 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Пушкар Антону Ігоровичу
(Прізвище, ім'я, по батькові)

1. Тема роботи Методи аналізу роботи протоколу OpenFlow у програмно-конфігурованих мережах SDN..

затверджена наказом по університету від " 07 " листопада 2022 р. № 1454 Ст

2. Термін подання студентом роботи 10 грудня 2022 р.

3. Вхідні дані до роботи
Концепції роботи програмно-конфігурованих мереж SDN.
Залежність величини затримки від кількості комутаторів у мережі.
Затримка налаштування потоку, структура протоколу OpenFlow.
Оцінка впливу оновлення TCAM на величину затримки.

.

4. Перелік питань, що потрібно опрацювати в роботі
Вступ.
Аналіз літератури та особливості побудови мереж спеціального призначення.
Аналіз концепції програмно-конфігурованих мереж.
Розробка моделі для аналізу величини наскрізної затримки.
Дослідження величини затримки трафіку, що передається в мережі SDN.
Висновки. Додаток.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Презентація Powerpoint 15 слайдів.

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначку консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ п./ п.	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз літератури за темою роботи	07.11.22–10.11.22	
2	Постановка мети та задач	31.10.22–02.11.22	
3	Аналіз алгоритму роботи мережі SDN	03.11.22–12.11.22	
4	Моделювання затримки у мережі	13.11.22–18.11.22	
5	Дослідження величини затримки	19.11.22–24.11.22	
6	Експериментальна частина	25.11.22–29.11.22	
7	Розрахункова частина	30.11.22–02.12.22	
8	Підготовка пояснювальної записки	03.12.22–06.12.22	
9	Розробка презентації та доповіді	07.12.22–09.12.22	
10	Подача роботи у ЕК	10.12.22	

Дата видачі завдання 07.11. 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Іванісенко І.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 96 с., 55 рисунків, 8 таблиць, 16 джерел.

ПРОТОКОЛ OPENFLOW, ПРОГРАМНО-КОНФІГУРОВАНІ МЕРЕЖІ SDN, WIRESHARK, OPENFLOW КОМУТАТОР

Мета кваліфікаційної роботи полягає у визначенні характеристик якості роботи програмно-конфігурованих мереж з використанням протоколу OpenFlow.

У ході виконання кваліфікаційної роботи було розглянуто концепцію створення програмно-конфігурованих мереж, показано їх значення у розвитку мережевих технологій. Виконано аналіз роботи протоколу OpenFlow у мережі SDN. Представлено процес управління програмно-конфігурованою мережею, створеною в Mininet, та підключеною до контролера OpenDaylight.

Проаналізовано трафік, що передається в результаті взаємодії контролера та OpenFlow-комутатора. Для дослідження процесу обміну трафіком між вузлами мережі використали аналізатор мережевих пакетів Wireshark. У подальшій роботі треба розробити модель формування затримки трафіку, що передається по мережі SDN та дослідити величину затримки переданого трафіку в мережі SDN.

ABSTRACT

Master's thesis: 96 pages, 55 figures, 8 tables, 16 sources.

OPENFLOW PROTOCOL, SOFTWARE-CONFIGURED NETWORKS SDN, WIRESHARK, OPENFLOW SWITCH

The goal of qualifying work is to determine the performance characteristics of software-configured networks using the OpenFlow protocol

During the qualifying work the concept of creating software-configured networks was considered, and their importance in the development of network technologies was shown. An analysis of the operation of the OpenFlow protocol in the SDN network was performed. The process of managing a software-configured network created in Mininet and connected to the OpenDaylight controller is presented.

The traffic transmitted as a result of the interaction between the controller and the OpenFlow switch was analyzed. The Wireshark network packet analyzer was used to study the process of traffic exchange between network nodes. In further work, it is necessary to develop a model of the formation of the delay of the traffic transmitted through the SDN network and to investigate the amount of delay of the transmitted traffic in the SDN network.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

QoS – якість обслуговування (англ., Quality of Service)

SDN – програмно-конфігуровані мережі

МОС – мережева операційна система

TCAM — Ternary Content-Addressable Memory

TLV – Type-Length-Value

OXM – OpenFlow Extensible Match

ODL – OpenDaylight (платформа контролерів SDN)

API – Application programming interface

ETSI - European Telecommunications Standards Institute

CLI –Command-line interface

KVM – Kernel-based Virtual Machine

SDK – Software development kit

OVS – Open Switch

NBI – Northbound interface

OVSDB – Open Switch Database

ВСТУП

Зростання попиту на телекомунікаційні послуги зумовлює нові вимоги до мережевих технологій, які мають забезпечити захищений, надійний та якісний доступ користувачів до інформаційних ресурсів. Однією з основних проблем сучасних мереж є висока завантаженість каналів зв'язку, що призводить до неефективного використання мережевої структури загалом. У цьому контексті провідні фірми та корпорації світу дійшли однозначного висновку – необхідно створення гнучкої та надійної системи управління телекомунікаційними мережами. Стало ясно, що екстенсивний шлях розвитку з урахуванням спеціалізованого устаткування тупиковим. Необхідні нові підходи до розвитку бізнес-операторів та сервіс-провайдерів.

Відділення керування мережею від мережевих пристроїв є ключовою ідеєю програмно-конфігурованої мережі (SDN). SDN – це зсув парадигми комп'ютерних мережах, де функціональність управління мережею (також відома як площина управління) відокремлена від функціональності пересилання даних (також відомої як площина даних), і, крім того, керування поділом є програмованим. Міграція керуючої логіки, яка раніше була тісно інтегрована в мережеві пристрої (наприклад, комутатори Ethernet) у доступні та логічно централізовані контролери, дозволяє абстрагуватися від базової мережної інфраструктури з точки зору додатків. Такий поділ відкриває шлях до більш гнучкої, програмованої, незалежної від виробника, економічно ефективної та інноваційної мережевої архітектури. Крім абстракції мережі, архітектура SDN надає набір прикладних програмних інтерфейсів (API), які спрощують реалізацію загальних мережевих служб (наприклад, маршрутизація, багатоадресна передача, безпека, контроль доступу, управління пропускною спроможністю, управління трафіком, QoS, енергоефективність та різні форми управління політикою).

У результаті підприємства оператори отримують безпрецедентну про-

грамованість, автоматизацію та управління мережею, що дозволяє їм створювати гнучкі мережі з високим ступенем масштабованості, які легко адаптуються до мінливих потреб бізнесу.

На сьогоднішній день найпоширенішим рішенням для реалізації управління мережевих пристроїв у програмно-конфігурованих мережах є протокол OpenFlow. OpenFlow – це протокол взаємодії між мережними пристроями, такими як комутатори та маршрутизатори, та централізованим контролером, який є мережевою операційною системою, встановленою на виділеному фізичному сервері, у програмно-конфігурованій мережі. Таке управління може замінити або доповнити функцію, що працює на мережному пристрої, здійснює побудову маршрутів, створення таблиці комутації тощо.

Актуальність використання мереж SDN пояснюється ринковими тенденціями. Технологія SDN повинна сприяти переорієнтації операторських бізнес-моделей на хмарні та цифрові послуги. Також оператори сподіваються звільнити мережі від надлишку обладнання.

Телекомунікаційної промисловості віртуалізація здатна скоротити операційні та капітальні витрати операторів на 60%. Такий ефект досягається за рахунок заміни комунікаційної системи фрагментованого типу новою інфраструктурою, в основу якої покладено програми, які не прив'язані до конкретного серверного обладнання. При цьому наявне мережеве обладнання розвантажиться в середньому на 25% за рахунок збільшення гнучкості мережі, а терміни виведення телекомунікаційних послуг на ринок значно скоротяться. У перспективі SDN дозволить оперативно модернізувати конфігурацію мереж, заміряти їх ємність та відстежувати спектр послуг, що реалізуються.

Новизна роботи полягає в комплексній оцінці величини затримки в програмно-конфігурованій мережі.

Практична значущість роботи полягає у можливості використання результатів дослідження у навчальних курсах мереж SDN.

1 ОГЛЯД ЛІТЕРАТУРИ І АНАЛІЗ ПРОБЛЕМИ ДОСЛІДЖЕННЯ

1.1 Концептуальні положення побудови програмно-конфігурованих мереж SDN

У сучасному світі, при різноманітності постачальників телекомунікаційного обладнання, прогрес, модифікація та складність різних технологій прагнуть зростання кількості пристроїв і обсягів переданих даних. Класична архітектура мережі, основи якої закладалися наприкінці 60-х років минулого століття, у зв'язку зі швидким зростанням різноманіття, складності та важливості розв'язуваних завдань, застаріла і не завжди здатна ефективно реагувати на нові потреби ринку. Щоб повністю перетворити історично усталену архітектуру, необхідно витратити чималих коштів, а модернізація окремих елементів мережі неспроможна тривати нескінченно [1].

Сьогодні інфокомунікаційні мережі, незважаючи на повсюдне використання та масштабний розвиток, зустрічають на своєму шляху безліч проблем:

- зростання кількості користувачів та трафіку, що призводить до перевантаження мережевих пристроїв;
- величезна кількість протоколів та його стеків, кількість з кожним днем лише збільшується;
- традиційні мережі, які пропрієтарні, обмежені для досліджень та практично будь-яких змін. Інтерфейс налаштування мережного обладнання іноді залежить навіть від моделі в одного і того ж виробника, отже, фахівці в галузі мережевих технологій повинні вміти аналізувати системи різних вендорів та працювати з великою кількістю протоколів;
- обладнання різних виробників, яке часто може конфліктувати між собою, незважаючи на універсальність мережевих протоколів;
- налаштування мережі з великою кількістю вузлів, що вимагає багато часу, а керування такими мережами здійснюється шляхом конфігурування їх

елементів через спеціалізовані інтерфейси;

- освоєння системи обладнання конкретного виробника, що потребує перепідготовки спеціалістів.

Таким чином, велика кількість проблемних факторів веде до ускладнення мережного обладнання та структури самих інформаційних мереж та, як наслідок, подорожчання мережного обладнання. Можливим вирішенням цієї проблеми є реалізація правил обробки даних у вигляді програмних модулів, а не вбудовування в апаратні засоби. Це дозволяє мережевим адміністраторам краще контролювати мережевий трафік і, отже, має великий потенціал для значного покращення продуктивності мережі з точки зору ефективного використання ресурсів та швидкості. Такий підхід визначено у програмно-конфігурованої мережі (SDN). У SDN обробка даних ізольована від апаратного забезпечення, а її керування реалізовано в програмному модулі, званому контролером [2].

SDN дозволяє мережним адміністраторам працювати з даними в мережі більш ефективним та інноваційним способом. Використовуючи SDN, адміністратори мають можливість контролювати потік даних, а також змінювати характеристики комутуючих пристроїв (пристроїв маршрутизації) в мережі з центрального місця розташування, при цьому програма управління реалізована у вигляді програмного модуля, без необхідності працювати з кожним з них.

Пристрій індивідуальний. Це дозволяє мережним адміністраторам змінювати таблиці маршрутизації (шляхи маршрутизації) у пристроях мережевої маршрутизації. Це також дає додатковий рівень контролю над даними мережі, адже адміністратор може призначити високий/низький пріоритет певним пакетам даних або дозволити/заблокувати певні пакети, що проходять через мережу з різними рівнями управління. В результаті мережевий трафік може ефективно контролюватись і, отже, може використовуватися як механізм керування навантаженням трафіку в мережах. Використовується кілька стандартів для впровадження SDN. OpenFlow - це один із найпопулярніших і зага-

льноприйнятих стандартів для реалізації SDN. OpenFlow забезпечує віддалене керування мережними пристроями маршрутизації завдяки своїй здатності контролювати таблиці маршрутизації трафіку в мережі. Рисунок 1.1 ілюструє основну різницю між SDN і звичайними мережами.

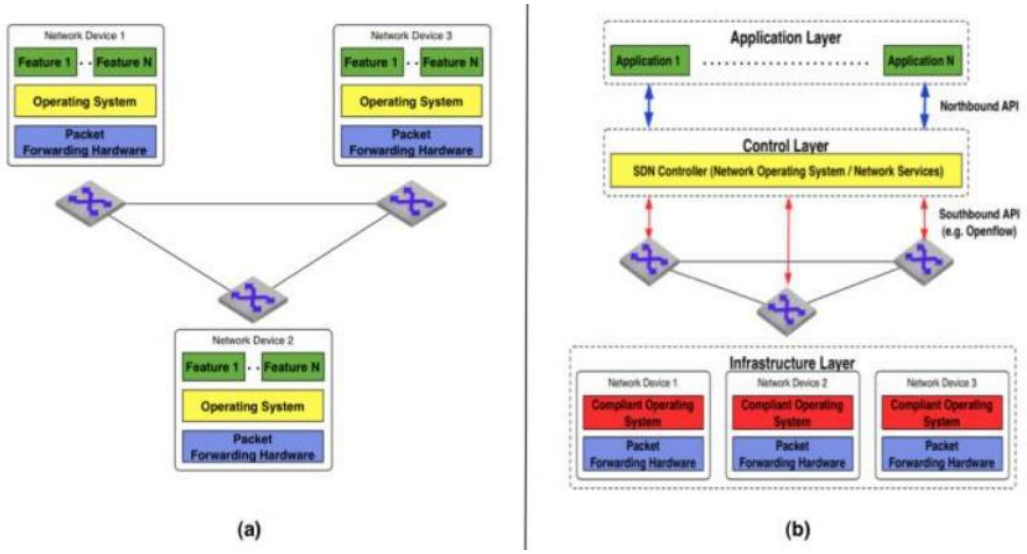
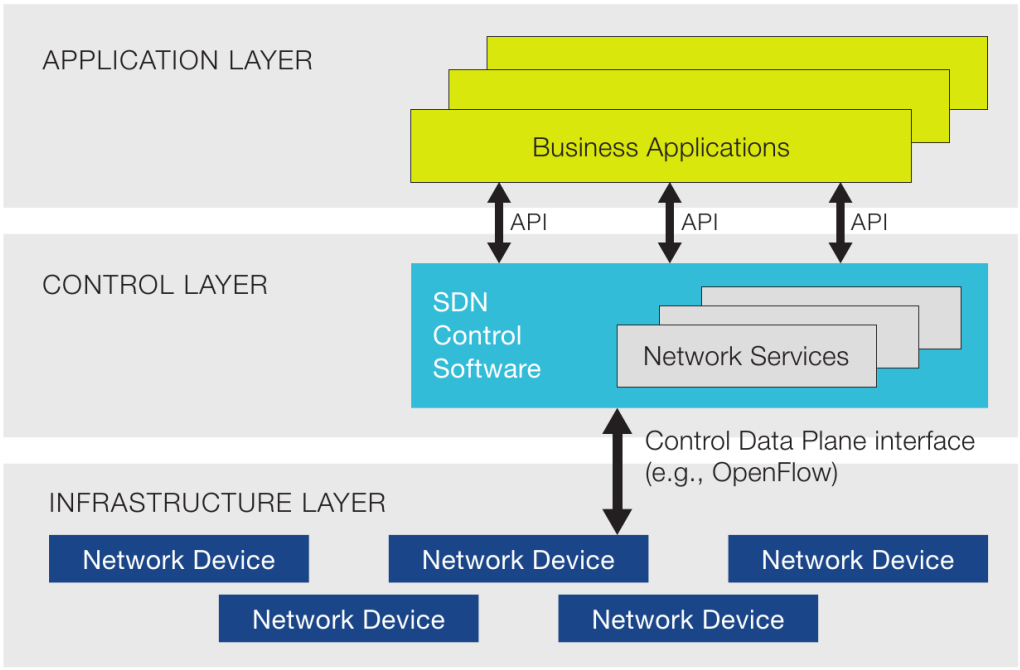


Рисунок 1.1 - Традиційна архітектура (a) і архітектура SDN (b)

На рисунку 1.2 зображено логічне представлення архітектури SDN [1]. «Інтелектуальні» функції мережі централізовані в основі програмного SDN контролеру, які підтримує глобальний вид мережі. В результаті, мережа відображається для додатків і систем політик як єдиний логічний комутатор. З SDN, підприємства отримати незалежний від постачальника контроль над всією мережею з єдиної логічної точки, що значно спрощує проектування мережі і її експлуатацію. SDN також значно спрощує мережеві пристрої самі по собі, так як вони більше не повинні розуміти і обробляти тисячі стандартів протоколів, а повинні просто приймати вказівки від контролеру SDN.



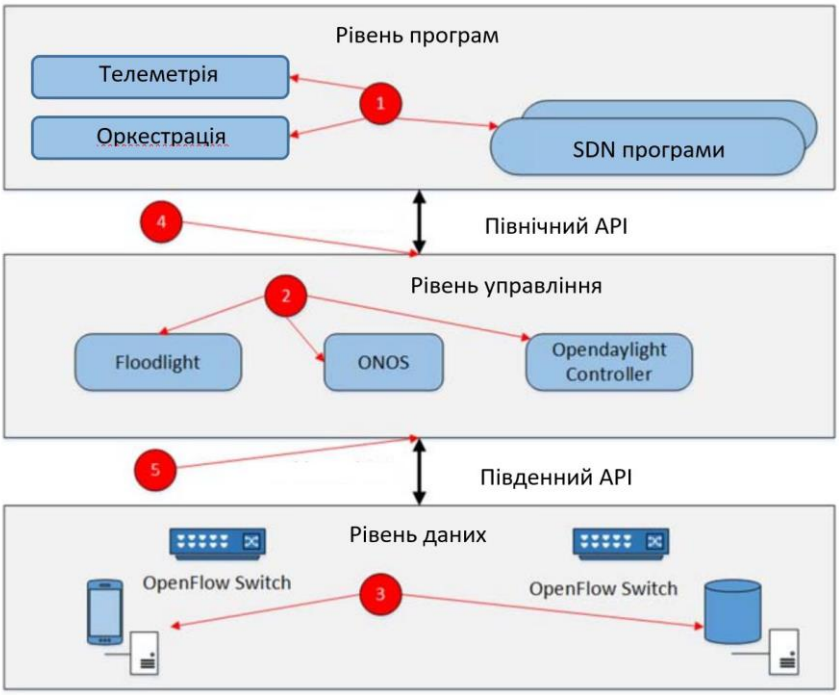


Рисунок 1.2 - Схематичне зображення архітектури SDN

Основні переваги SDN [3]:

1) Підвищення продуктивності. У зв'язку з тим, що з комутаторів знімаються навантаження з обробки лінії управління, програмно-конфігурована мережа дає можливість цим пристроям направити всі свої ресурси на прискорення переміщення трафіку.

2) Спрощення процесу адміністрування. На централізованому контро-

лері програмно-конфігурованої мережі системний адміністратор може спостерігати мережу в єдиному уявленні, що сприяє підвищенню зручності управління, забезпечення безпеки та виконання інших завдань.

3) Прискорення реалізації та тестування нових сервісів. Програмні засоби програмно-конфігурованих мереж дозволяють адміністраторам додавати нові функції до вже наявної мережевої архітектури.
4) Підвищення безпеки. Програмно-конфігурована мережа надає можливість адміністратору чітко бачити всі потоки трафіку, тому він буде набагато легше помічати вторгнення, визначати пріоритети різних типів трафіку, розробляти правила реагування мережі при заторах та проблемах з обладнанням.
5) Застосування хмарних технологій. У хмарах дані та програми розміщені на комп'ютерах, які взаємодіють через мережу, і технологія SDN здатна дати необхідний хмарам рівень «інтелектуальності» мереж.

1.2 Архітектура мереж SDN

Архітектура SDN представляє трирівневу модель, що складається з рівнів інфраструктури, управління та додатків. Якщо докладно розглянути інформаційні потоки в архітектурі SDN (рисунок 1.3), можна побачити два напрями обміну інформацією. Перший потік між рівнем додатків і рівнем управління, другий між рівнем фізичних мережевих пристроїв і рівнем управління. Перший потік отримав назву "північний інтерфейс", а другий - "південний інтерфейс". Як «північний інтерфейс» виступає протокол на основі REST API (REST – загальні принципи організації взаємодії програми з сервером, API – інтерфейс програмування додатків, що складається з набору готових кодів, що надаються додатком для використання у зовнішніх програмних продуктах), а як «південний інтерфейсу» – протокол OpenFlow [4].

Рівень додатків є сукупністю додатків, які безпосередньо взаємодіють із SDN-контролером, запитуючи необхідні ресурси за допомогою API, вирі-

шуючи високорівневі завдання управління мережею. Прикладом таких програм можуть бути засоби моніторингу, аналітики або бізнес-додатки. Наприклад, конкретний бізнес-додаток Microsoft Lynk та його основна роль – зміна мережі в режимі реального часу під поточні потреби програми, що обслуговується (зміна Quality of Service або створення VPN тунелю між двома абонентами) [5].

Рівень керування здійснює низькорівневе логічно централізоване керування, яке забезпечує форвардинг трафіку на інфраструктурному рівні за допомогою відкритого інтерфейсу OpenFlow. Рівень управління постійно здійснює моніторинг усієї мережі та здатний керувати всіма мережевими пристроями.

Рівень інфраструктури складається із середовища передачі даних та комутаторів SDN (OpenFlow-комутатори), які можуть бути як логічними, так і фізичними елементами мережі.

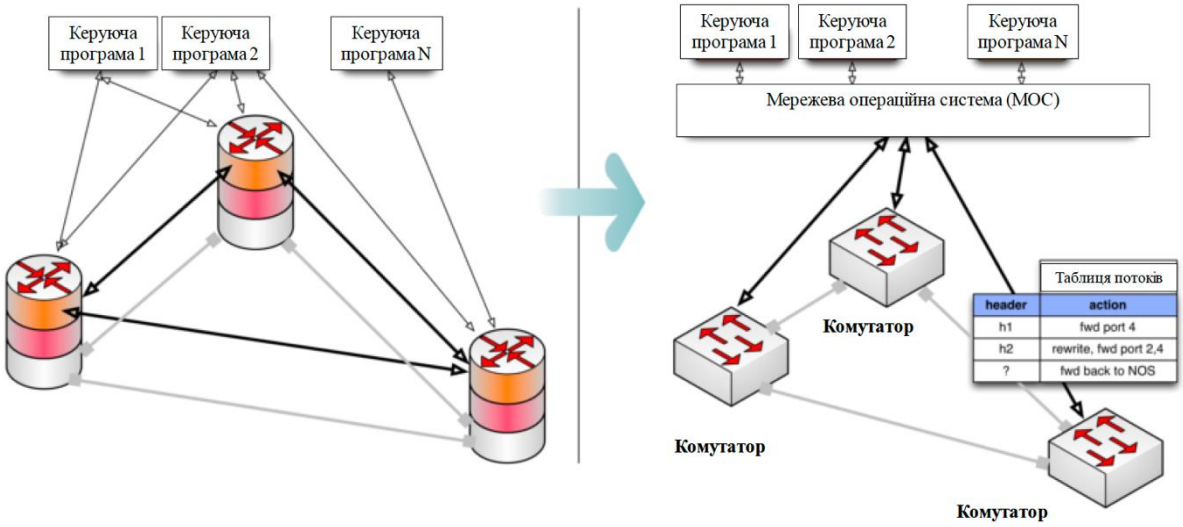


Рисунок 1.3 – Архітектура мереж SDN

Компоненти мережі:

1) Оркестратор – представлена у вигляді апаратного пристрою або програмного забезпечення платформа автоматизації, яка використовується програмами для конфігурування пристроїв рівнів керування та інфраструкту-

ри (наприклад, коригує роботу кількох контролерів) та виконує конкретні сервісні запити.

2) Контролер – обчислювальний пристрій (наприклад, сервер) на якому функціонує спеціалізована платформа, що складається з мережевих керуючих додатків та мережевої операційної системи (МОС):

- МОС – готовий фреймворк, завдання якого надати інтерфейс прикладного програмування для мережевих додатків з контролю та управління мережею цілком, а також реалізація механізмів керування таблицями одного або кількох OpenFlow комутаторів (додавання, видалення, модифікація правил, збір статистики);

- SDN-додаток – програмна реалізація різних мережевих функцій та сервісів (маршрутизація, балансування навантаження, фільтрація трафіку, шлюзи, мережеві екрани, шифрування, DPI, NAT, DHCP, DNS).

3) OpenFlow-комутатор – простий програмований мережевий пристрій, що виконує лише функції комутації даних згідно з інструкціями контролера. Основні складові OpenFlow комутатора, принцип роботи описані в розділі 1.3.

4) Програмний інтерфейс (northbound API) – відкритий інтерфейс програмування, що дозволяє програмувати контролер ззовні. Призначений до створення екосистеми додатків. Користувачами цього API є всі розробники мережних додатків.

1.3 OpenFlow комутатор

1.3.1 Компоненти комутатора OpenFlow

Комутатор OpenFlow складається з однієї або декількох таблиць потоків та групової таблиці, які виконують пошук та пересилання пакетів, а також одного або кількох каналів OpenFlow на зовнішній контролер (рисунок 1.4). Комутатор зв'язується з контролером і контролер управляє комутатором по

протоколу OpenFlow [7].

Використовуючи протокол комутації OpenFlow, контролер може додавати, оновлювати та видаляти записи потоків у таблицях потоків. Кожна таблиця потоків у комутаторі містить набір записів потоків, кожен запис потоку складається з полів зіставлення, лічильників та набору інструкцій, які повинні застосовуватися до пакетів, що зіставляються.

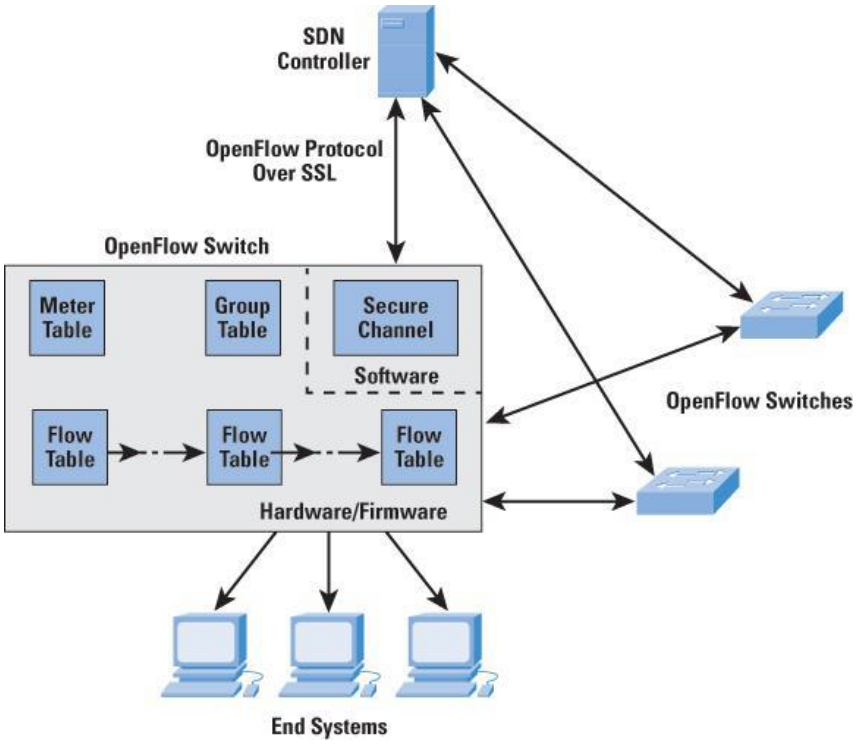


Рисунок 1.4 – Використання протоколу OpenFlow в мережі SDN

Зіставлення починається з першої таблиці потоків і може тривати до додаткових таблиць потоків конвеєра. Якщо знайдено відповідний запис, виконуються інструкції, пов'язані з конкретним записом потоку. Якщо в таблиці потоків збіг не знайдено, результат залежить від конфігурації запису потоку пропущених таблиць (наприклад, пакет може бути перенаправлений на контролери по каналу OpenFlow, відкинуто або може перейти до наступної таблиці потоків).

Інструкції, пов'язані з кожним записом потоку, містять дії або змінюють конвеєрну обробку. Дії, що включені в інструкції, описують пересилання

пакетів, модифікацію пакетів та обробку групових таблиць. Інструкції конвеєрної обробки дозволяють відправляти пакети в наступні таблиці для подальшої обробки та дозволяють передавати інформацію у вигляді метаданих між таблицями. Обробка конвеєра таблиці зупиняється, коли набір команд, пов'язаний із відповідним записом потоку, не вказує наступну таблицю, в цей момент пакет зазвичай модифікується та пересилається.

Записи потоку можуть пересилати порт. Зазвичай це фізичний порт, але це може бути логічний порт, визначений комутатором, чи зарезервований порт. Зарезервовані порти можуть вказувати загальні дії пересилання, такі як відправка на контролер або пересилання з використанням методів не OpenFlow, таких як «звичайна» обробка комутатора, в той час як логічні порти, що визначаються комутатором, можуть вказувати групи агрегації каналів, тунелі або інтерфейси зворотного зв'язку.

Дії, пов'язані із записами потоку, можуть також спрямовувати пакети групу, що визначає додаткову обробку. Групи представляють набори дій, а також складнішу семантику пересилання (наприклад, багатошляховий введення-виведення, агрегація каналів). Як загальний рівень непрямості групи також дозволяють кільком записам потоку пересилати на один ідентифікатор (наприклад, переадресація IP на загальний наступний перехід). Це дозволяє ефективно змінювати загальні вихідні дії в записах потоку.

Таблиця групи містить запис групи; кожен запис групи містить список блоків дій з певною семантикою, яка залежить від типу групи.

1.3.2 OpenFlow порти

Порти OpenFlow нічим не відрізняються від портів звичайного комутатора рівня L2, кожен порт має вхідний/вихідний буфер та унікальний номер, який виступає у ролі імені порту.

Набір портів OpenFlow може не співпадати з набором мережних інтерфейсів, що надаються обладнанням комутатора, деякі мережні інтерфейси

можуть бути відключені для OpenFlow, а комутатор OpenFlow може визначати додаткові порти OpenFlow.

Протокол OpenFlow визначає набір стандартних портів: фізичні порти, логічні порти та зарезервовані порти [8].

Фізичні порти – це порти, визначені комутатором, які відповідають апаратному інтерфейсу комутатора.

Логічні порти – це певні комутатором порти, які відповідають безпосередньо апаратному інтерфейсу комутатора. Логічні порти є абстракціями вищого рівня, які можуть бути визначені в комутаторі з використанням не-OpenFlow методів (наприклад, груп агрегації каналів, тунелів, інтерфейсів зворотного зв'язку).

Логічні порти можуть включати інкапсуляцію пакетів та можуть відображатися на різні фізичні порти. Обробка, що виконується логічним портом, залежить від реалізації і має бути прозорою для обробки OpenFlow, і ці порти повинні взаємодіяти з обробкою OpenFlow подібно до фізичних портів OpenFlow.

Єдині відмінності між фізичними та логічними портами полягають у тому, що пакет, пов'язаний з логічним портом, може мати додаткове поле конвеєра, зване Tunnel-ID, пов'язане з ним, і коли пакет, отриманий на логічному порту, відправляється на контролер, його логічний порт , та його базовий фізичний порт повідомляються контролеру.

Зарезервовані порти визначають загальні дії пересилання, такі як відправлення на контролер, флуд або пересилання з використанням не-OpenFlow методів, таких як "звичайна" обробка комутатора.

Комутатор не повинен підтримувати всі зарезервовані порти, тільки ті, які позначені як «Обов'язкові»:

- обов'язкові: ALL - представляє всі порти, які комутатор може використовуватиме пересилання певного пакета. Може використовуватися лише як вихідний порт. У цьому випадку копія пакета починає вихідну обробку на всіх стандартних портах, крім вхідного порту пакета та портів, які

налаштовані OFPPC_NO_FWD;

- обов'язкові: CONTROLLER – представляє канал управління з контролерами SDN. Може використовуватись як вхідний порт або як вихідний порт. При використанні як вихідний порт пакет інкапсулюється в повідомлення Packet_In і відправляється з використанням протоколу OpenFlow. Коли використовується як вхідний порт, це ідентифікує пакет, що походить від контролера;

- обов'язкові: TABLE – являє собою початок конвеєра OpenFlow. Відправляє пакет до першої таблиці потоків, щоб пакет міг бути оброблений через конвеєр OpenFlow;

- обов'язкові: IN PORT - представляє вхідний порт пакета. Може використовуватися тільки як вихідний порт, який відправляє пакет через вхідний порт;

- обов'язкові: ANY - спеціальне значення, яке використовується в деяких запитах OpenFlow, коли порт не вказаний (тобто порт підстановковий). Деякі запити OpenFlow містять посилання на певний порт, до якого належить лише запит. Використання ANY номери порту в цих запитах дозволяє екземпляру запиту застосовуватися до всіх портів. Не може використовуватися як вхідний, ні як вихідний порт;

- необов'язкові: LOCAL - представляє локальний мережевий стек комутатора та його стек управління. Може використовуватись як вхідний порт або як вихідний порт. Локальний порт дозволяє віддаленим об'єктам взаємодіяти з комутатором та його мережевими службами через мережу OpenFlow, а чи не через окрему мережу управління;

- необов'язкові: NORMAL - представляє пересилання з використанням традиційного не-OpenFlow конвеєра комутатора. Може використовуватися лише як вихідний порт та обробляє пакет, використовуючи звичайний конвеєр. Якщо комутатор не може пересилати пакети з конвеєра OpenFlow до звичайного конвеєра, він повинен вказати, що він не підтримує цю дію;

- необов'язкові: FLOOD - являє собою потокову розсилку з

використанням традиційного не-OpenFlow конвеєра комутатора. Може використовуватись лише як вихідний порт, фактичний результат залежить від реалізації. У загальному випадку пакет відправлятиметься на всі стандартні порти, але не на вхідний порт або порти, які перебувають у заблокованому стані. Комутатор також може використовувати ідентифікатор пакета VLAN або інші критерії, щоб вибрати, які порти використовувати для потокової розсилки.

1.3.3 Конвеєрна обробка

Комутатори, сумісні з OpenFlow, бувають двох типів: тільки OpenFlow та гібридні. У комутаторах, які підтримують лише операцію OpenFlow, всі пакети обробляються конвеєром OpenFlow і не можуть оброблятись інакше [9].

Гібридні комутатори підтримують роботу як OpenFlow, і звичайну комутацію Ethernet, тобто традиційну комутацію Ethernet L2, ізоляцію VLAN, маршрутизацію L3 (маршрутизація IPv4, маршрутизація IPv6), обробку ACL і QoS. Ці комутатори повинні забезпечувати механізм класифікації поза OpenFlow, який спрямовує трафік або до конвеєра OpenFlow, або до звичайного конвеєра. Наприклад, комутатор може використовувати тег VLAN або вхідний порт пакета, щоб вирішити, чи обробляти пакет з використанням одного конвеєра чи іншого, чи може направляти всі пакети в конвеєр OpenFlow. Гібридний комутатор OpenFlow також може дозволити пакету відправлятися з конвеєра OpenFlow до звичайного конвеєра через зарезервовані порти NORMAL і FLOOD.

Конвеєр OpenFlow кожного логічного комутатора OpenFlow містить одну або кілька таблиць потоків, кожна з яких містить кілька записів потоків. Обробка конвеєра OpenFlow визначає, як пакети взаємодіють із цими таблицями потоків (рисунок 1.4) [7].

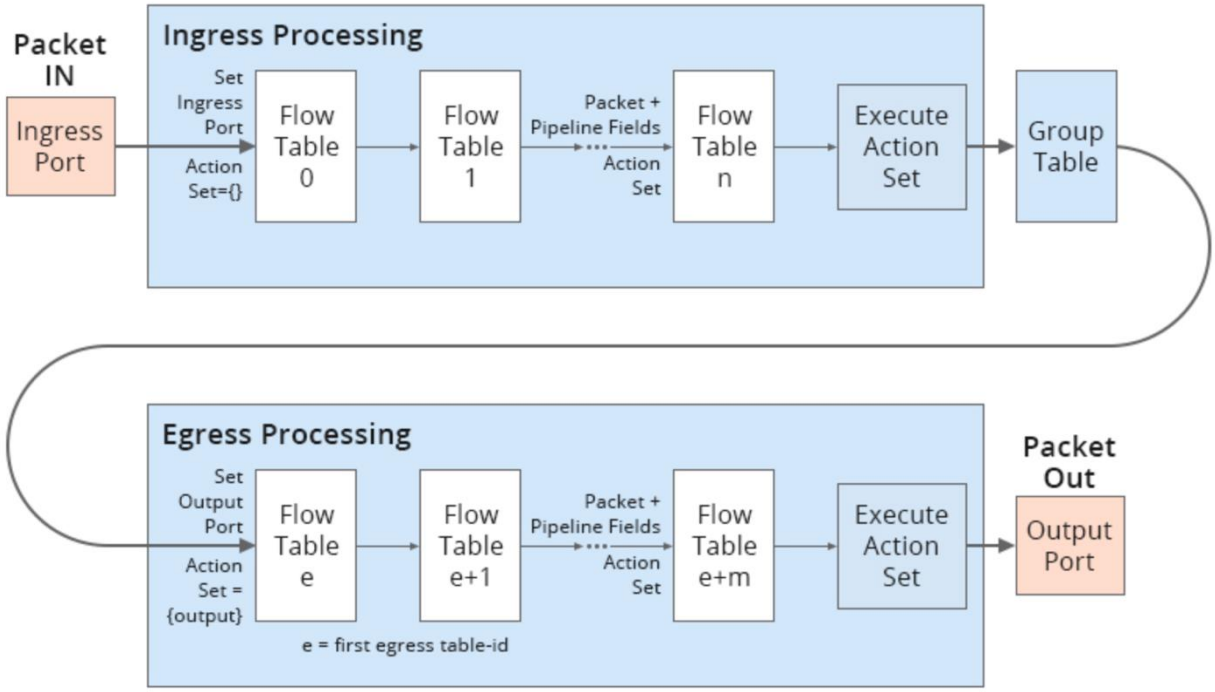


Рисунок 1.5 – Потік пакетів через конвеєр обробки

Таблиці потоків комутатора OpenFlow нумеруються в тому порядку, в якому вони можуть переглядатися пакетами, починаючи з 0. Конвеєрна обробка відбувається у два етапи: вхідна обробка та вихідна обробка.

Конвеєрна обробка завжди починається з вхідної обробки першої таблиці потоків: пакет повинен спочатку зіставлятися із записами потоку в таблиці потоків 0 (рисунок 1.6). Інші таблиці вхідного потоку можуть використовуватись залежно від результату обробки у першій таблиці. Якщо результатом вхідної обробки є пересилання пакета на вихідний порт, комутатор OpenFlow може виконати вихідну обробку у тих вихідного порту.

Вихідна обробка є необов'язковою, комутатор може не підтримувати будь-які вихідні таблиці або може бути не налаштований їх використання. Якщо жодна вихідна таблиця не встановлена як перша вихідна таблиця, пакет повинен оброблятися вихідним портом, і в більшості випадків пакет пересилається з комутатора. Якщо вихідну таблицю встановлено як першу вихідну таблицю, залежно від результату збігу у цій таблиці потоків.

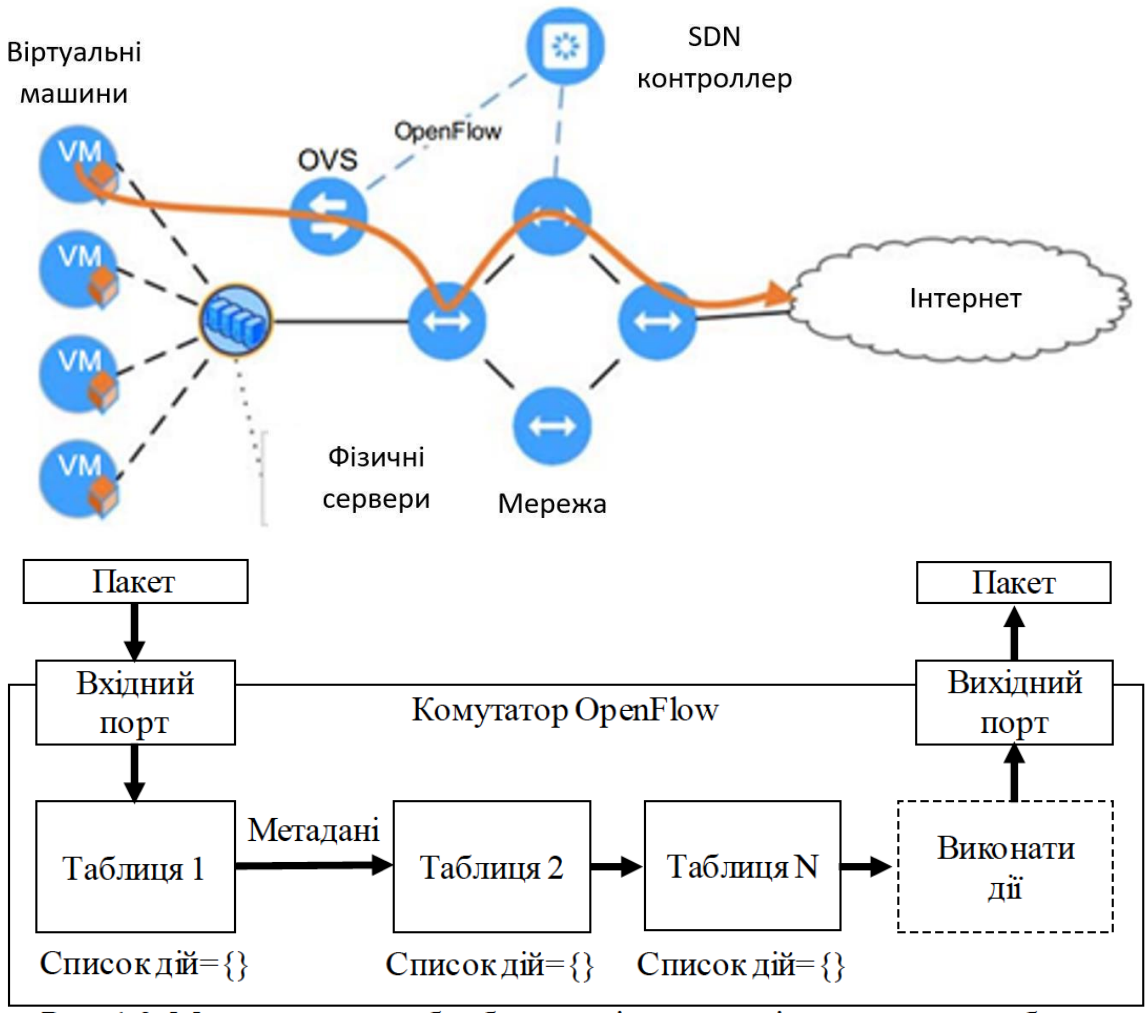


Рисунок 1.6 – Спрощена схема проходження потоку пакетів через комутатор OpenFlow

При обробці таблицею потоків пакет зіставляється із записами потоку цієї таблиці потоків. Якщо запис потоку знайдено, виконується набір інструкцій, включений до запису потоку. Ці інструкції можуть явно направляти пакет до іншої таблиці потоків (використовуючи інструкцію GotoTable), де той самий процес повторюється знову. Запис потоку може тільки направити пакет на номер таблиці потоків, який більше, ніж номер його власної таблиці потоків, тобто конвеєрна обробка може йти тільки вперед, а не назад. Якщо відповідний запис потоку не направляє пакети до іншої таблиці потоків, поточний етап обробки конвеєра зупиняється на цій таблиці, пакет обробляється з відповідним набором дій і зазвичай пересилається.

Якщо пакет не співпадає із записом потоку в таблиці потоків, це озна-

чає пропуск таблиці. Поведінка під час пропуску таблиці залежить від конфігурації таблиці. Інструкції, включені у запис потоку пропущених таблиць, можуть гнучко вказувати, як обробляти не зіставлені пакети. Корисні опції включають видалення пакетів, передачу пакетів до іншої таблиці або відправлення пакетів до контролерів по каналу управління.

1.3.4 Таблиці OpenFlow

Кожен OpenFlow комутатор містить одну або кілька унікальних таблиць, які він заповнює на основі даних, отриманих від контролера. Оскільки по мережі передаються окремі пакети, потоки даних, така таблиця отримала назву Flow table (таблиця потоків). Під потоком розуміється сукупність пакетів (сеансів зв'язку), що визначається за різними ознаками та обмежена лише можливостями реалізації самих таблиць (наприклад, TCP-сесія, пакети з певної MAC або IP-адреси, пакети з однаковим номером порту OpenFlow-комутатора).

Кожен запис таблиці потоків (таблиця 1.1) містить:

- поля відповідності: найменування поля заголовка пакета (field) та рядок із двійкових символів та символу невизначеності (pattern). Запис про потік застосовується до пакета, якщо всі двійкові символи рядка pattern поля field збіглися з відповідним полем пакета, що прийшов, і всі класифікатори (список пар (field, pattern)) сумісні з заголовком пакета;
- пріоритет: натуральне число з певного діапазону, що вказує ступінь значущості потоку в таблиці потоків. Використовується для вибіркового застосування запису про потоки до конкретного пакету (для обробки вибирається запис з найбільшим пріоритетом);
- лічильники: містять статистичні дані про роботу запису (наприклад, інформацію про те, скільки пакетів було оброблено відповідно до цього запису про потік);
- інструкції: для зміни набору дій чи конвеєрної обробки;

- таймаут: мітки, які визначають термін життя запису в таблиці потоків (максимальний час перебування та максимальний термін простою правила);

- cookie: набір певних даних, вибраних контролером. Може використовуватися контролером для фільтрації записів потоку, на які впливають статистика потоку, зміни потоку та запити видалення потоку. Не використовується для обробки пакетів;

- прапори: прапори призначені для зміни запису таблиці потоків, наприклад, прапор OFPFF_SEND_FLOW_REM - видалення запису таблиці потоків.

Таблиця 1.1 – Основні компоненти запису потоку таблиці потоків.

Поля відповідності	Пріоритет	Лічильники	Інструкції	Таймаут	Cookie	Прапори

1.3.5 Інструкції

Інструкція - операція, яка містить або безліч дій, щоб додати їх до списку дій, або дії, які негайно застосовуються до пакета або модифікують процес обробки пакета конвеєрі (рисунок 1.7).

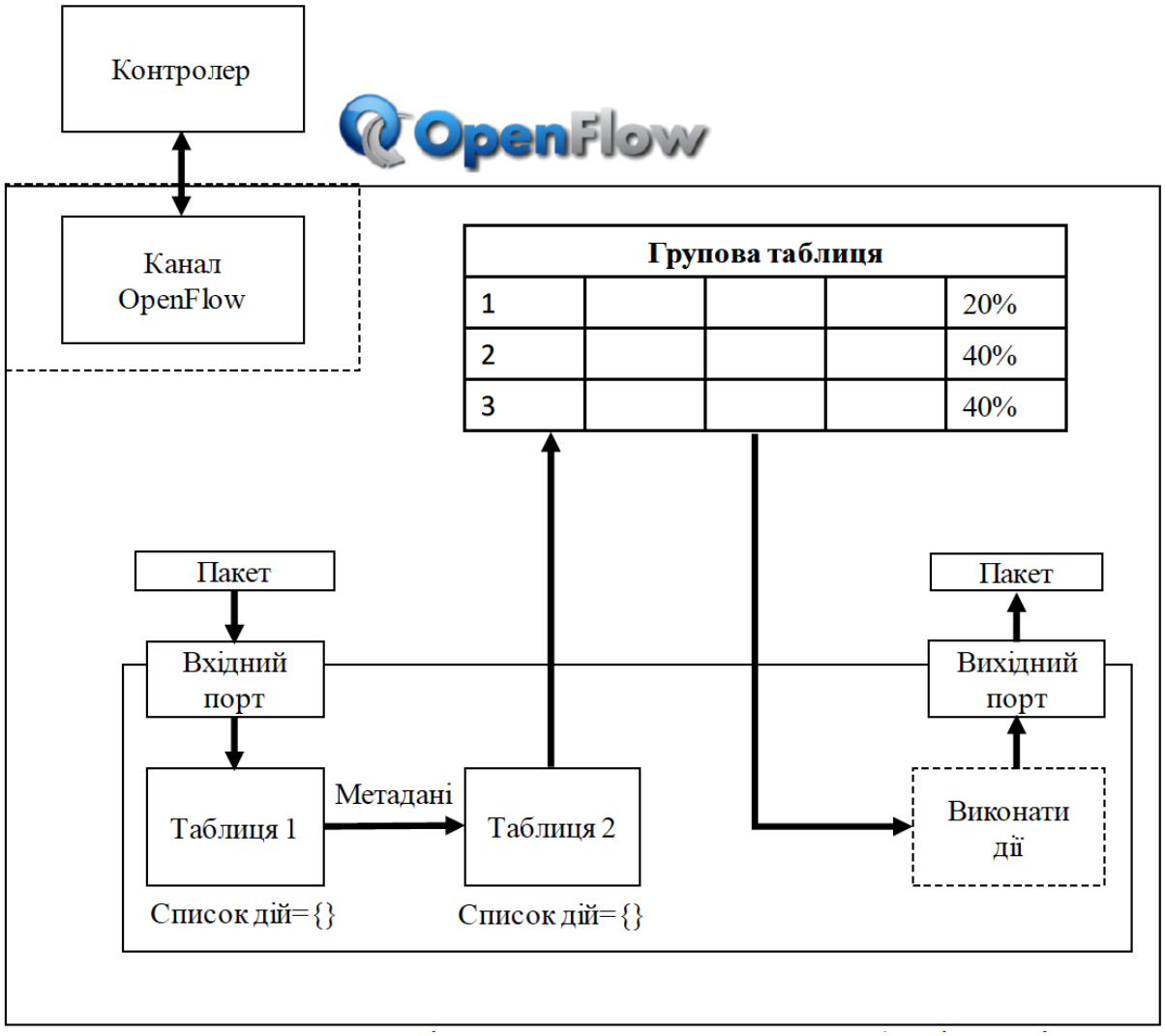


Рисунок 1.7 – Модель комутації з використанням групової таблиці потоків та реалізація балансування навантаження на основі вагових коефіцієнтів шляхів

Розрізняються кілька типів інструкцій [7]:

- Apply-Actions – застосовує конкретну дію негайно, без змін у списку дій. Ця інструкція може бути використана для модифікації пакета між двома таблицями або для виконання кількох дій того самого типу.

- Clear-Actions – видаляє всі дії зі списку дій.

- Write-Actions – заносить дії до списку дій. Якщо дія цього типу є у списку, то вона перезаписується, інакше додається.

Write-Metadata metadata/mask – записує значення метаданих у масці у полі метаданих. Маска вказує, які біти регістру метаданих мають бути зміне-

ні.

Goto-Table – вказує таку таблицю у конвеєрі обробки. Ідентифікатор таблиці повинен бути більшим за поточний ідентифікатор таблиці. Ця інструкція повинна підтримуватись у всіх таблицях потоків, крім останньої, комутатори OpenFlow з однією таблицею потоків не потрібні для реалізації цієї інструкції.

1.3.6 Дії

Кожному потоку в таблиці асоціюється одна або кілька дій, які виконуються, коли потік відповідає запису. Під дією розуміється операція, яка змінює пакет, список дій та/або процес обробки конвеєром, залежно від цього різняться кілька типів дій:

- Output – передати пакет на вказаний порт (фізичний, логічний, зарезервований).
- Set-Queue – встановити ідентифікатор черги для пакета. Drop – відкинути пакет, що належав потоку.
- Group – обробити пакет відповідно до зазначеної групи.
- Push-Tag/Pop-Tag - робота з мітками (VLAN, MPLS).
- Set-Field – змінити значення певного поля заголовка пакета. Change-TTL – змінити значення поля TTL.

У OpenFlow-комутатор дії можуть бути представлені не окремо, а списком дій. Список дій складається з набору дій, пов'язаних з пакетом, який зберігається, поки йде обробка таблицями потоків, і виконується тільки при виході пакета з конвеєра обробки.

1.3.7 Групова таблиця

Група OpenFlow – це абстракція, яка спрощує складніші та спеціалізовані пакетні операції, які нелегко виконати за допомогою запису в таблиці потоків. Кожна група отримує пакети як вхідні дані та виконує будь-які дії

OpenFlow з цими пакетами. Група не може надсилати пакети до інших таблиць потоків. Крім того, очікується, що пакети були відповідним чином зіставлені до входу до групи, оскільки групи не підтримують зіставлення пакетів. Група – це просто механізми виконання додаткових дій чи наборів дій.

Як показано на рисунку 1.8, перевага групи у тому, що вона містить

окремі списки дій, і кожен окремий список дій називається сегментом. Таким чином кажуть, що група містить перелік сегментів. Кожен сегмент або список сегментів може застосовуватись вхідним пакетам (точна поведінка залежить від типу групи) [10].

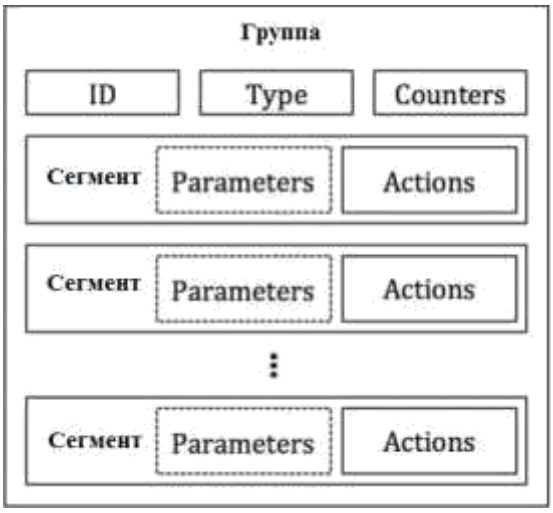


Рисунок 1.8 – Компоненти групи та сегмента

Ідентифікатор групи – це 32-бітове поле, яке визначає номер групи. Лічильники працюють аналогічно лічильникам звичайних таблиць пересилання, зберігаючи статистику групи. Блоки дій є набір інструкцій, пов'язаний з сегментом. Це відповідає набору дій, які змінюють заголовок пакета та/або відправляють пакет на вихідний порт.

Існує чотири типи груп: ALL, SELECT, INDIRECT та FAST-FAILOVER.

Група ALL показана на малюнку 1.9 прийматиме будь-який отриманий пакет як вхідний і дублюватиме його, щоб працювати з ним незалежно кожним сегментом у списку сегментів. Таким чином, група ALL може використ-

товуватися для дублювання та подальшої роботи окремими копіями пакета, що визначаються діями в кожному сегменті. У кожному сегменті можуть бути різні дії, що дозволяє виконувати різні операції з різними копіями пакета.

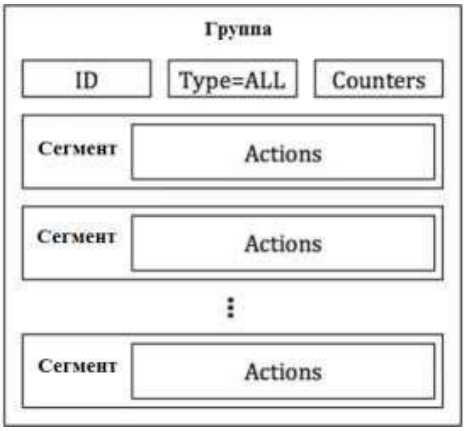


Рисунок 1.9 – Група ALL

Група SELECT насамперед призначена для балансування навантаження. Як показано на рисунку 1.10, кожному сегменту групи SELECT призначено вагу, і кожен пакет, що входить до групи, відправляється в один сегмент. Алгоритм вибору сегмента не визначено і залежить від реалізації комутатора, проте зважений циклічний перебір - це, мабуть, найочевидніший і найпростіший вибір розподілу пакетів по сегментах. Вага сегмента вказується як спеціальний параметр для кожного сегмента. Кожен сегмент у групі SELECT, як і раніше, являє собою список дій, тому будь-які дії, що підтримуються OpenFlow, можна використовувати в кожному сегменті, і як група ALL, сегменти не обов'язково повинні бути однаковими.

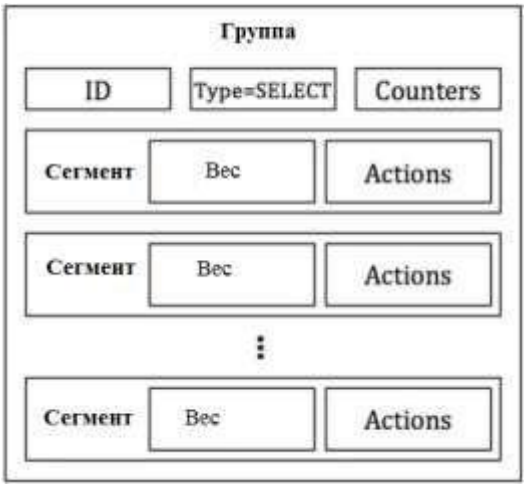


Рисунок 1.10 – Група SELECT

Група INDIRECT (рисунок 1.11) містить лише один сегмент, у якому всі пакети, отримані групою, відправляються у єдиний сегмент. Мета групи INDIRECT – інкапсулювати загальний набір дій, які використовуються багатьма потоками. Наприклад, якщо потоки A, B і C збігаються за різними заголовками пакетів, але мають загальний набір або підмножина дій, ці потоки можуть відправляти пакети в одну групу INDIRECT замість дублювати список спільних дій для кожного потоку. Група INDIRECT використовується для спрощення розгортання OpenFlow та зменшення обсягів пам'яті, що займається набором аналогічних потоків.

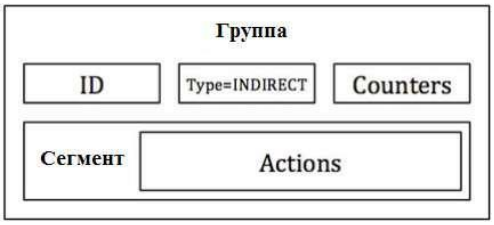


Рисунок 1.11 – Група INDIRECT

Група FAST-FAILOVER розроблена спеціально для виявлення та подолання збоїв портів. Як і групи SELECT і ALL, група FAST-FAILOVER, як показано на рисунку 1.12, має список сегментів. На додаток до цього списку дій кожен сегмент має порт спостереження та/або групу спостереження як спеціальний параметр. Спостережуваний порт/група відстежуватиме працездатність або статус зазначеного порту/групи. Якщо вважається, що життєздатність впала, сегмент не використовуватиметься. Якщо життєздатність визначена як підвищена, сегмент можна використовувати. Одночасно можна використовувати лише один сегмент, і сегмент, що використовується, не буде змінений, якщо поточність порту/групи відстеження використовуваного сегмента не зміниться з високого на низький.

Коли така подія відбувається, уемає жодної гарантії щодо часу перехо-

ду для вибору нового сегмента у разі виникнення збою. Час переходу залежить від часу пошуку активного порту/групи спостереження і зажадав від реалізації комутатора. Однак мотивація використання групи FAST-FAILOVER полягає в тому, що це майже гарантовано буде швидше, ніж звернення до площини управління, щоб обробити подію порту, що не працює, і вставити новий потік або набір потоків. З групами FAST-FAILOVER виявлення збоїв каналу та відновлення повністю відбувається у площині даних.

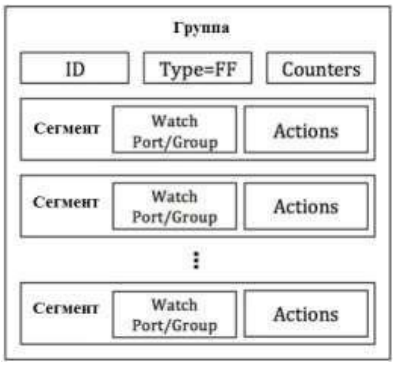


Рисунок 1.12 – Група FAST-FAILOVER

1.4 Класифікатори протоколу OpenFlow

Структура зіставлення OpenFlow використовується для зіставлення пакетів із записами потоку в таблицях. Кожен класифікатор може складатися із заголовка і мати від нуля до деякої кількості відповідних полів класифікаторів, закодованих за допомогою формату OXM TLV [11].

Поля відповідності класифікатора описуються за допомогою формату OpenFlow Extensible Match (OXM). Спільно вони утворюють компактний формат type-length-value (TLV). Класифікатори OXM TLV можуть бути від 5 до 259 (включно) байт довгою.

Перші 4 байти кожного класифікатора OXM TLV – заголовок (рисунок 1.13):

1) OXM class – специфікація OpenFlow поділяє два типи OXM класів:

- ONF reserved – класи, що описуються специфікацією та

зарезервовані для наступних стандартизацій;

- ONF member – класи, які використовуються членами консорціуму ONF. Дозволяють членам консорціуму використовувати свої класифікатори та унікально ідентифікуватися завдяки їм (наприклад, Nicira).

2) OXM field – поле використовується визначення конкретного поля відповідності у кожному типі класу (таблиці 1.2 і 1.3);
3) OXM hasmask – поле може набувати значення 0 або 1. Визначає наявність у полі даних OXM бітової маски;
4) OXM length - описує довжину корисного навантаження OXM TLV в байтах, тобто все, що слідує за 4-байтовим заголовком OXM TLV.

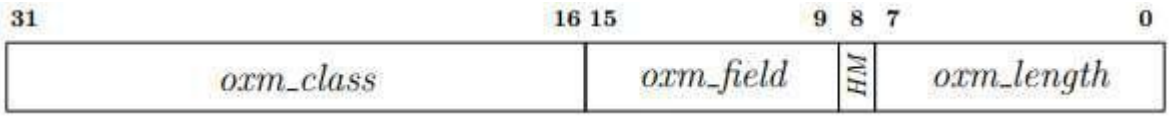


Рисунок 1.13 – Заголовок класифікатора OXM TLV

1.4.1 OXM classes – reserved

Тип OXM-класів – Member – не описується специфікаціями і вимагає розбору у межах даної роботи. Розглянемо класи Reserved з прикладу протоколу OpenFlow версії 1.5.1.

У специфікації протоколу визначено три класи у цьому типі:

- experimentrt – клас використовується для розширення стандартного набору полів відповідності в OpenFlow комутаторі;

- register – клас використовується для зберігання тимчасових значень та інформації, пов'язаної з пакетом у процесі обробки конвеєром;

- OpenFlow basic – основний клас класифікаторів, який містить базовий набір полів відповідності, проте не всі з цього набору полів повинні обов'язково підтримуватись OpenFlow-коммутатором, але кожне обов'язкове match-поле має підтримуватись не менш ніж однією таблицею потоків. Поля відповідності класу OpenFlow basic поділяються на два типи. Усі вони мають

різні розміри, обов'язковість/необов'язковість до реалізації, маску та опис (таблиця 1.2).

Таблиця 1.2 – Поля відповідності класу OpenFlow basic (header match fields)

Math-поле	Розмір, біт	Маска	Опис
1 OFPXMT_OFB_ETH_DST	48	Так	MAC-адреса одержувача
2 OFPXMT_OFB_ETH_SRC	48	Так	MAC-адреса відправника
3 OFPXMT_OFB_TYPE	16	Ні	Ethernet type
4 OFPXMT_OFB_VLAN_VID	12+1	Так	VLAN-ID за стандартом 802.1Q
5 OFPXMT_OFB_VLAN_PCP	3	Ні	VLAN-PCP за стандартом 802.1Q
6 OFPXMT_OFB_IP_DSCP	6	Ні	Diff Serv Code Point (DSCP)
7 OFPXMT_OFB_IP_ECN	2	Ні	ECN біт у заголовку IP
8 OFPXMT_OFB_IP_PROTO	8	Ні	IPv4 чи IPv6-номер протоколу
9 OFPXMT_OFB_IPV4_SRC	32	Так	IPv4-адреса відправника
10 OFPXMT_OFB_IPV4_DSP	32	Так	IPv4-адреса одержувача
11 OFPXMT_OFB_TCP_SRC	16	Ні	TCP-порт відправника
12 OFPXMT_OFB_TCP_DST	16	Ні	TCP-порт одержувача
Продовження таблиці 1.2 13 PXMT_OFB_TCP_FLAGS	12	Ні	TCP-прапори
14 OFPXMT_OFB_UDP_SRC	16	Ні	UDP-порт відправника
15 OFPXMT_OFB_UDP_DST	16	Ні	UDP-порт одержувача
16 OFPXMT_OFB_SCTP_SRC	16	Ні	SCTP-порт відправника
17 OFPXMT_OFB_SCTP_DST	16	Ні	SCTP-порт одержувача
18 OFPXMT_OFB_ICMPV4_TYPE	8	Ні	ICMP-тип
19 OFPXMT_OFB_ICMPV4_CODE	8	Ні	ICMP-код
20 OFPXMT_OFB_ARP_OP	16	Ні	ARP-код відправника/одержувача

21 OFPXMT_OFB_ARP_SPA	32	Так	IPv4-адреса відправника в payload протоколу ARP
22 OFPXMT_OFB_ARP_TPA	32	Так	IPv4-адреса одержувача в payload протоколу ARP
23 OFPXMT_OFB_ARP_SHA	48	Так	MAC-адресвідправників payload протоколу ARP
24 OFPXMT_OFB_ARP_THA	48	Так	MAC-адреса одержувача в payload протоколу ARP
25 OFPXMT_OFB_IPV6_SRC	128	Так	IPv6-адреса відправ-ника
26 OFPXMT_OFB_IPV4_DST	128	Так	IPv6-адреса одержува-ча

Другий тип полів відповідності містить поля, які використовуються в процесі обробки конвеєром (pipeline match fields) і жодним чином не взаємодіють із заголовками пакета (таблиця 1.3).

Таблиця 1.3 – Поля відповідності класу OpenFlow basic (pipeline match fields)

Math-поле	Розмір, біт	Маска	Опис
OFPXMT_OFB_IN_PORT	32	Ні	Вхідний порт. Номер фізичного або логічного вхідного порту
OFPXMT_OFB_IN_PHY_PORT	32	Ні	Фізичний порт. Відповідність фізичного та логічного порту, коли повідомлення надходить на

			логічний порт
OFPXMT_OFB_METADATA	64	Так	Використо-вується для передачі інформації між таблицями
OFPXMT_OFB_TUNNEL_ID	64	Так	Метадані, асоційовані з логічним портом
OFPXMT_OFB_ACTSET_OUT-PUT	32	Ні	Output port from action set Metadata
OFPXMT_OFB_PACKET_TYPE	16+16	Ні	Packet type – canonical header type of outermost header

1.5 Постановка мети і задач дослідження

Сучасне стан мереж показало, що потенціал зростання продуктивності, пропускну здатність з урахуванням традиційних технологій практично вичерпано. Це пов'язано зі зростанням кількості користувачів та трафіку, з труднощами налаштування мережі та управління потоками в ній, особливо з урахуванням нових потреб у якості сервісів для високошвидкісних глобальних мереж та мереж центрів обробки даних. Який впливає також зростання потреб у віртуалізації мереж, тобто. відображення кількох логічно ізольованих мереж із незалежними політиками обслуговування на загальний набір мережевих ресурсів. Такий незадовільний стан справ може змінитися через появу

Важливо нового підходу, званого програмно-конфігурованими мережами. Дотримання такого підходу дозволить підвищити продуктивність мережі, зручність конфігурування, віртуалізацію, але потребує додаткових досліджень та розробок. Зокрема, в області організації мережевих комутаторів, програмних додатків для управління мережею та платформ для їх виконання.

Метою дослідження є визначення характеристик якості роботи програмно-конфігурованих мереж з використанням протоколу OpenFlow.

Мета кваліфікаційної роботи досягається послідовним вирішенням наступних задач як :

- аналіз алгоритму роботи мережі SDN;

- дослідження створення записів у таблиці потоків;
- розробка моделі для аналізу величини наскрізної затримки;
- дослідження величини затримки трафіку, що передається в мережі SDN;
- оцінка впливу оновлення TCAM величину затримки.

2 ДОСЛІДЖЕННЯ РОБОТИ ПРОГРАМНО-КОНФІГУРОВАНОЇ МЕРЕЖІ

2.1 Інструменти дослідження

2.1.1 VirtualBox

VirtualBox – програмний продукт віртуалізації для операційних систем Microsoft Windows, Linux, MacOS та інших. Віртуалізація дозволяє запускати кілька операційних систем на одному комп'ютері: при цьому кожен з екземплярів таких операційних систем працює зі своїм набором логічних ресурсів (процесорних, оперативної пам'яті, пристроїв зберігання), надання яких із загального пулу, доступного на рівні обладнання, управляє хостова операційна система. гіпервізор. Гіпервізор - програма або апаратна схема, що дозволяє одночасне, паралельне виконання декількох операційних систем на тому самому хост-комп'ютері. Гіпервізор також зобов'язаний надавати працюючим під його керуванням на одному хост-комп'ютері ОС засоби зв'язку та взаємодії між собою (наприклад, Емулятор мережі Mininet)

2.1.2 Емулятор мережі Mininet

Мережевий емулятор Mininet дозволяє швидко розгорнути мережу, що складається з різного числа віртуальних хостів, комутаторів, контролерів та каналів між ними [13].

Це програмне забезпечення використовує технології віртуалізації, вбудовані в ядро ОС Linux. Це дозволяє запускати на віртуальних хості стандартні мережеві утиліти Linux. Комутатори, що емулюються за допомогою Mininet, мають підтримку OpenFlow, що дозволяє використовувати даний продукт для дослідження та навчання принципам побудови програмно-конфігурованих мереж.

Більшість коду Mininet написано з використанням мови Python. Топології емульованих мереж, а також параметри мережевих пристроїв можуть бути описані мовою Python. Крім того, існують вбудовані шаблони топологій, наприклад, лінійна топологія чи топологія «дерево». Також для управління мережею, що емулює, існує вбудований інтерпретатор команд.

2.1.3 Контролер мережі SDN OpenDaylight

OpenDaylight (ODL) – це платформа контролерів SDN з відкритим вихідним кодом для налаштування та автоматизації мереж будь-якого розміру та масштабу. Проект OpenDaylight виник із руху SDN із чітким упором на програмованість мереж. З самого початку він був розроблений як основа для комерційних рішень, призначених для різних варіантів використання в існуючих мережевих середовищах [14].

Платформа ODL розроблена, щоб надати наступним користувачам та постачальникам рішень максимальну гнучкість у створенні контролера, який відповідає їхнім потребам. ODL дозволяє будь-якому користувачеві використовувати послуги, створені іншими, писати та включати свої власні та поділитися своєю роботою з іншими. ODL включає підтримку найширшого набору протоколів на будь-якій платформі SDN – OpenFlow, OVSDB, NETCONF, BGP та багатьох інших, які покращують програмованість сучасних мереж та вирішують низку потреб користувачів.

2.1.4 Аналізатор мережевого трафіку Wireshark

Для дослідження процесу обміну трафіком між вузлами мережі використовується аналізатор пакетів мережі Wireshark. Wireshark дозволяє перехоплювати та розпізнавати повідомлення протоколу OpenFlow. Також мережевий аналізатор може визначити структуру повідомлення та дані в ньому [15].

2.2 Дослідження алгоритму роботи мережі SDN

Для аналізу роботи протоколу OpenFlow було використано дві віртуальні машини у програмі VirtualBox. Одна запускає емульовану мережу Mininet, а інша - контролер OpenDaylight (рисунок 2.1). Обидві віртуальні машини підключені до мережі лише хоста, щоб вони могли спілкуватися друг з одним.

У віртуальній машині під керуванням операційної системи Ubuntu, було встановлено контролер SDN OpenDaylight та наступні функції (рисунок 2.2):

- odl-restconf – можливість використання restconf API для керування контролером;
- odl-l2switch-switch – функція комутатора l2, яка потрібна для комутації пакетів через Mininet Open vSwitch, оскільки Open vSwitch є комутатором openflow і не може пересилати пакети без контролера. l2switch – це модуль на контролері, який виконує функцію навчального комутатора для Open vSwitch. Контролер автоматично відправляє потоки до Open vSwitch;
- odl-mdsal-apidocs – функція автоматичного створення документації API;
- odl-dluxapps-applications – функція для графічного інтерфейсу OpenDaylight.

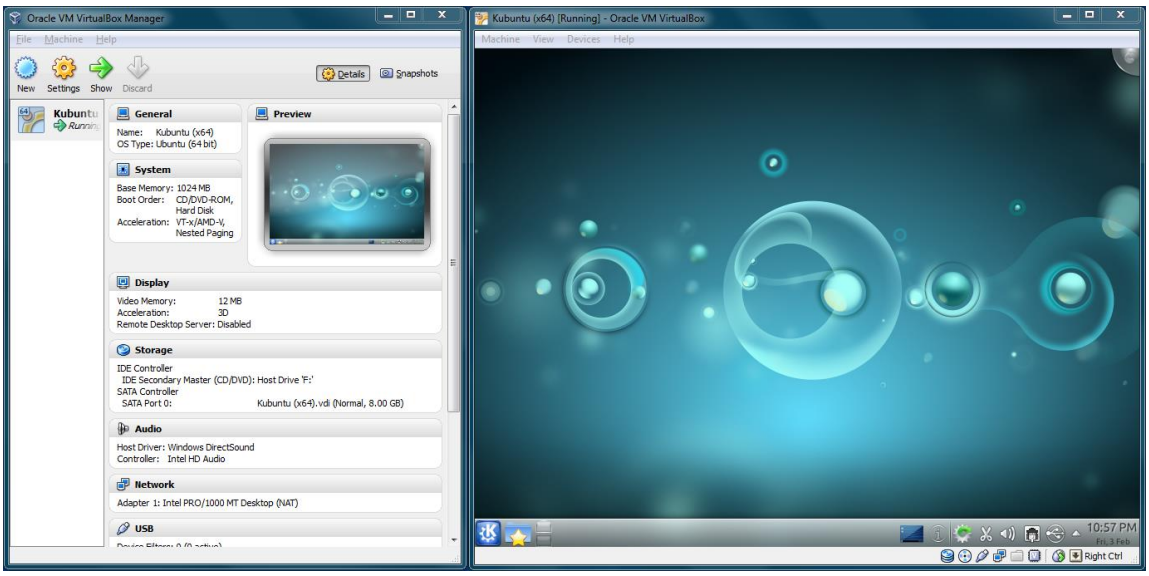


Рисунок 2.1 – Використання VirtualBox

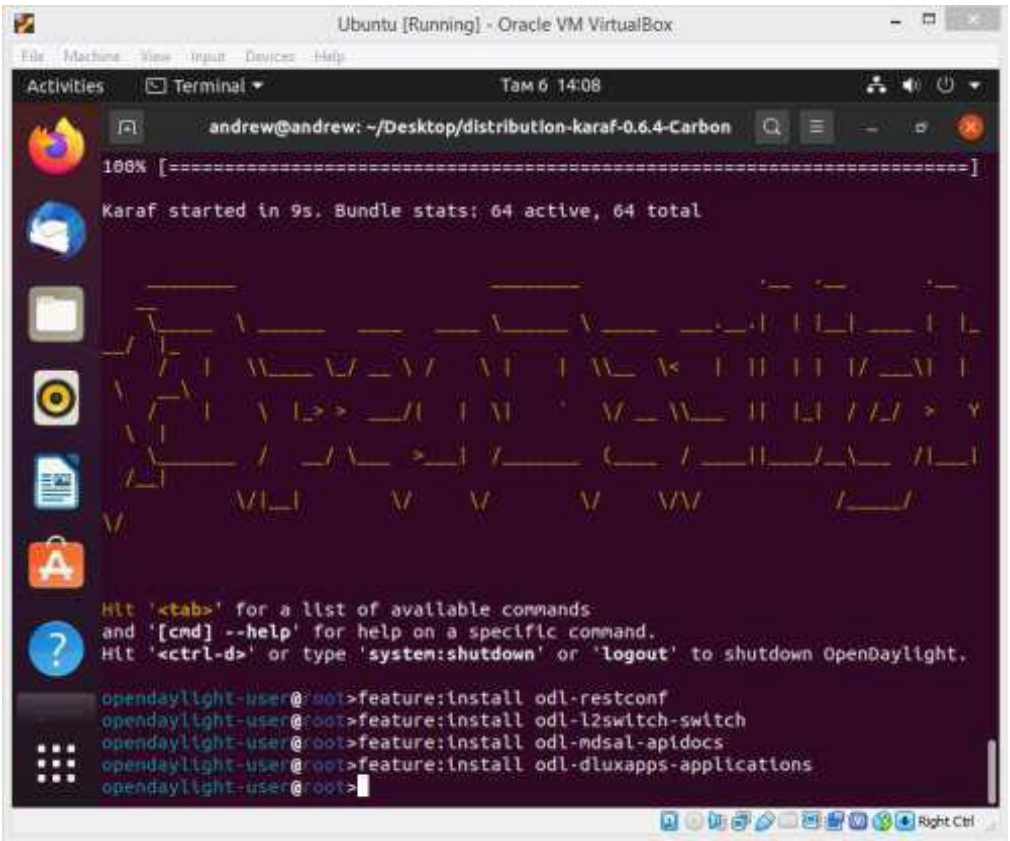


Рисунок 2.2 – Встановлення контролера OpenDaylight

У віртуальній машині Mininet створена мережа з одного комутатора Open vSwitch, який підтримує протокол OpenFlow (ip адреса 192.168.56.101) і трьох підключених до нього хостів під керуванням контролера OpenDaylight

(ip адреса 192.168.56.102), в результаті вводу команди, яка показана на рисунку 2.3.

Параметри команди:

- topo - опція визначення топології мережі. У нашому випадку single,3 означає, що до мережі буде підключено один комутатор і 3 хоста;

- mac – MAC-адреса на кожному хості буде встановлена на просте число;

- controller – опція визначення контролера SDN. У нашому випадку контролер OpenDaylight встановлений на іншій віртуальній машині, тому ми використовуємо опцію remote з IP-адресою віртуальної машини (192.168.56.102), де встановлено OpenDaylight;

- switch – опція визначення віртуального комутатора. Використовується Open vSwitch із Openflow версії 1.3.

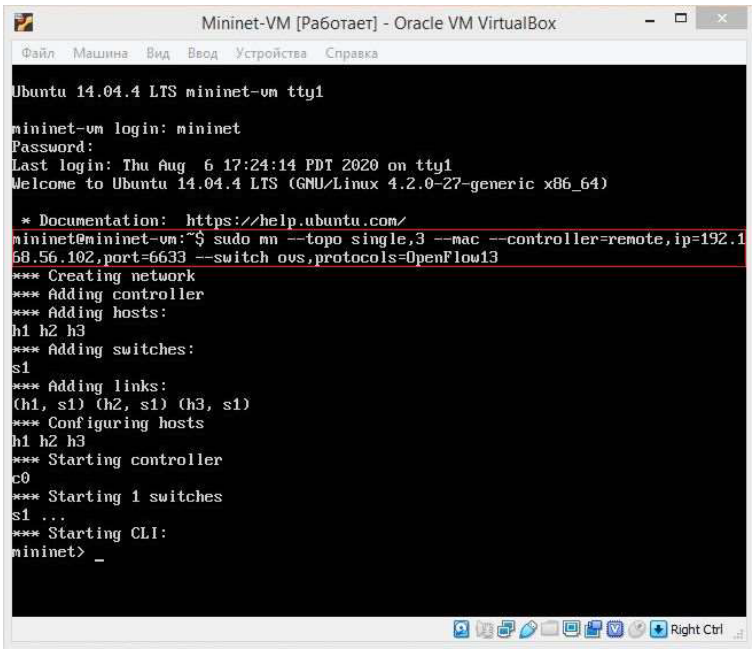


Рисунок 2.3 – Створення мережі

Для перегляду топології мережі, саме зіставлення портів комутатора і хостів використовується команда net (рисунок 2.4).

Рисунок 2.4 – Використання команди net

Для перегляду інформації про вузли, комутатори та контролери в мережі, що моделюється, використовується команда dump (рисунок 2.5).

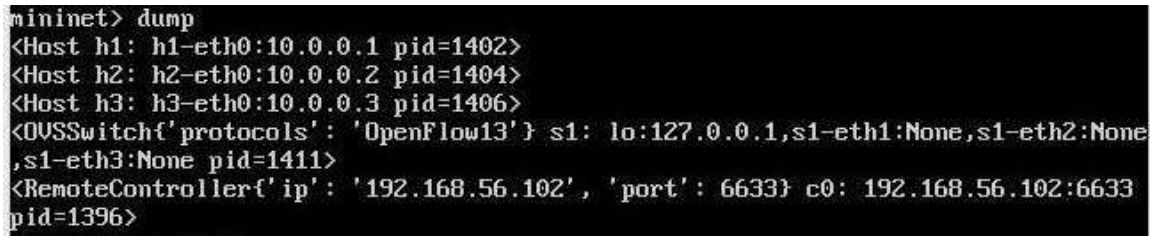


Рисунок 2.5 – Використання команди dump

Для перевірки зв'язку між усіма хостами в мережі використовується команда pingall. Команда pingall запускатиме команду ping на кожному хості і перевірятиме зв'язок з кожним іншим хостом, а потім повідомить про результати в інтерфейсі командного рядка Mininet (рисунок 2.6).

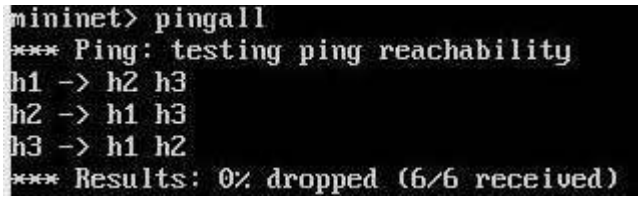


Рисунок 2.6 – Використання команди pingall

Пропінгуючи всі вузли, переконалися, що контролер OpenDaylight працює. Відобразимо нашу топологію мережі в графічному інтерфейсі OpenDaylight. Він працює на віртуальній машині, тому IP-адреса – 192.168.56.102, а порт, визначений додатком – 8181 (рисунок 2.7).

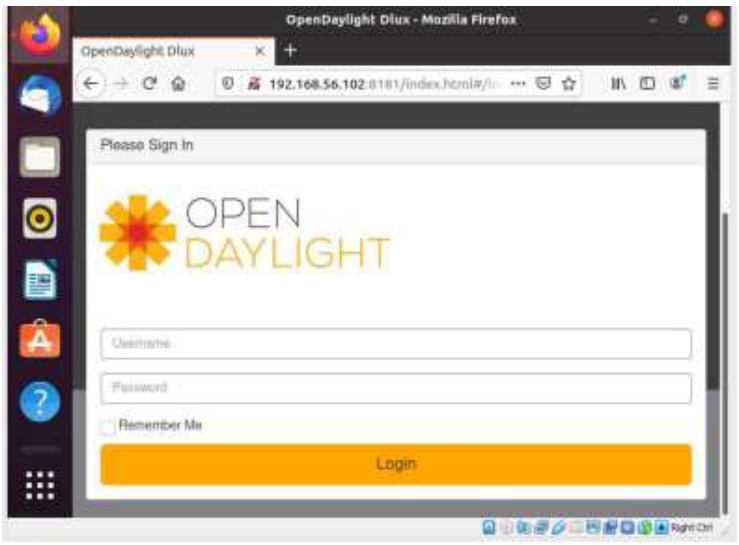


Рисунок 2.7 – Графічний інтерфейс OpenDaylight

Рисунок 2.8 демонструє згенеровану Mininet топологія мережі за допомогою графічного інтерфейсу контролера OpenDaylight.

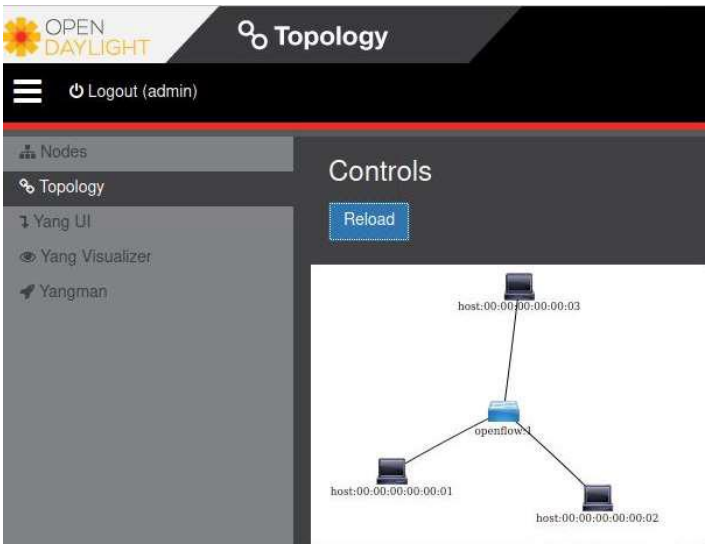


Рисунок 2.8 – Вкладка топології в контролері OpenDaylight

У вкладці «Вузли» можна переглянути інформацію про вузли мережі (рисунок 2.9). Видно, що хост 1 підключений до s1-eth1, хост 2 підключений до s1-eth2, хост підключений 3 до s1-eth3. Зверніть увагу, що існує локальний інтерфейс керування, який є локальною площиною управління комутатора, яка використовується для відправки трафіку безпосередньо на площину управління.

Рисунок 2.9 – Вкладка «Вузли»

Щоб переглянути повідомлення OpenFlow, якими обмінюються контролер та комутатор у мережі, використовується Wireshark.

Кожне повідомлення OpenFlow починається з однакової структури заголовка. Ця фіксована структура виконує три ролі, які не залежать від версії OpenFlow, що використовується. По-перше, поле версії вказує на версію OpenFlow, якій належить це повідомлення. По-друге, поле lenght вказує, де це повідомлення закінчиться в потоці байтів, починаючи з першого байта заголовка. По-третє, xid або ідентифікатор транзакції – це унікальне значення, яке використовується для зіставлення запитів та відповідей. Поле типу, яке вказує, який тип повідомлення є і як інтерпретувати корисне навантаження, залежить від версії (рисунок 2.10).

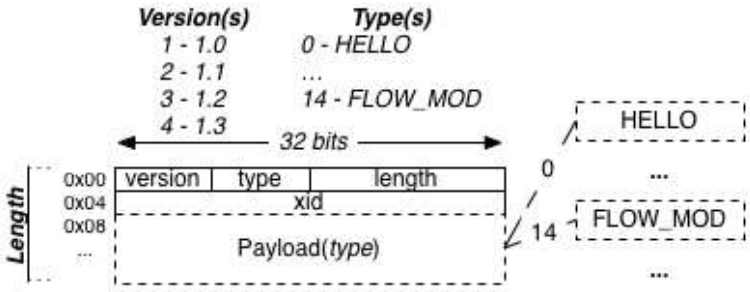


Рисунок 2.10 – Структура заголовка повідомлення OpenFlow

Використовуючи фільтр відображення Wireshark для openflow_v4, що

означає OpenFlow версії 1.3, можна переглянути зловлені пакети (рис. 2.11).

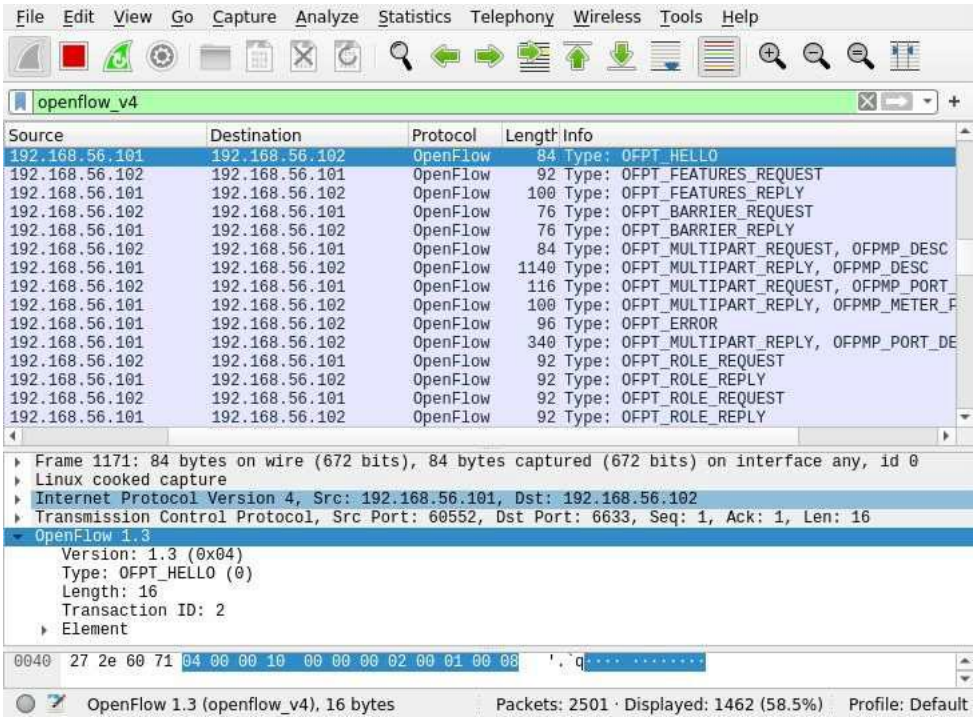


Рисунок 2.11 – Фільтрування пакетів OpenFlow

На рисунку 2.11 видно, що перший спійманий пакет це повідомлення HELLO, відправлене від комутатора (192.168.56.101) до контролера (192.168.56.102). Він використовується для узгодження версії OpenFlow. Якщо узгодження версії протоколу не вдається, надсилається повідомлення про помилку з типом Hello Failed та кодом "Несумісний". Це ініціалізація каналу OpenFlow, який є каналом даних для протоколу OpenFlow між контролером та комутатором.

Наступний тип повідомлення – це OFPT_FEATURES_REQUEST від контролера до комутатора (рисунок 2.12). Коли між комутатором та контролером встановлюється транспортний канал OpenFlow, перша дія – визначення функції. Контролер відправить на комутатор Feature Request транспортним каналом. Запит функції складається лише з повідомлення заголовка OpenFlow зі значенням Feature Request, встановленим у полі типу. Потім комутатор відповість на запит своїми можливостями.

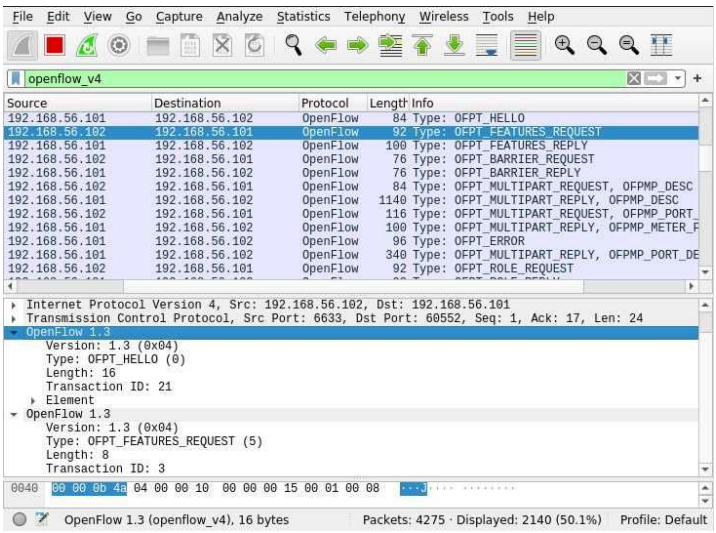


Рисунок 2.12 – Повідомлення OFPT_FEATURES_REQUEST

OFPT_FEATURES_REPLY - це відповідь комутатора контролеру з перерахуванням його можливостей (рисунок 2.13). Datapath-id - це 64-бітове поле, яке слід розглядати як аналог MAC-адреси моста Ethernet-комутаторів, його унікальний ідентифікатор для конкретного керованого конвеєра обробки пакетів. Поле n_buffers визначає, скільки пакетів комутатор може поставити в чергу для дій Packet_In (захоплення та пересилання на контролер). Кількість таблиць в комутатор фіксується n_tables. Capabilities показує які можливості комутатора підтримуються комутатором.

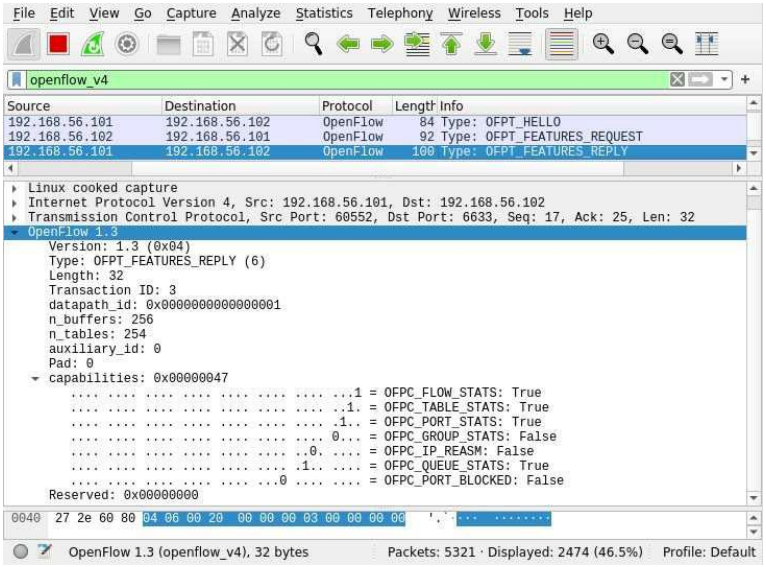


Рисунок 2.13 – Повідомлення OFPT_FEATURES_REPLY

Контролер може запросити у комутатора стан каналу OpenFlow за допомогою повідомлення OFPT_MULTIPART_REQUEST. Типи повідомлень, що обробляються цим повідомленням, включають різну статистику (FLOW/TABLE/PORT/QUEUE/METER тощо) або функції опису (METER_CONFIG/TABLE_FEATURES/PORT_DESC тощо).

У повідомленні OFPT_MULTIPART_REPLY, OFPMP_PORT_DESC розташована інформація про порти комутатора, відправлена на контролер.

У це повідомлення включено все, від типу до статусу та швидкості.

У повідомленні OFPT_MULTIPART_REPLY, OFPMP_DESC, можна побачити такі атрибути, як виробник та версії апаратного/програмного забезпечення (рисунок 2.14).

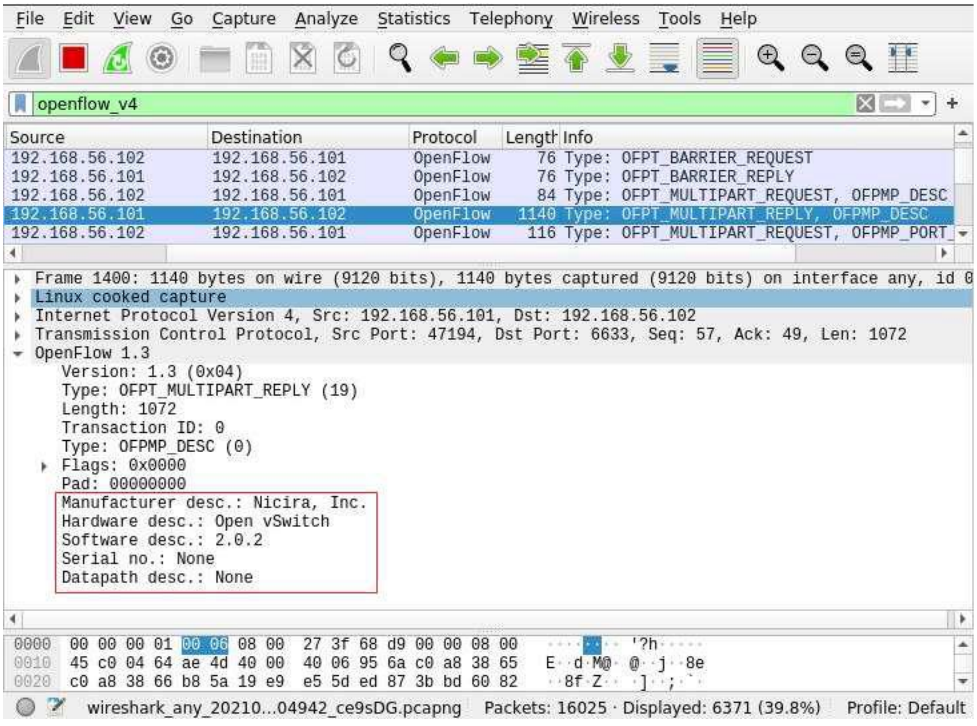


Рисунок 2.14 – Повідомлення OFPT_MULTIPART_REPLY, OFPMP_DESC

Повідомлення Flow_Mod дозволяє контролеру змінювати стан комутатора OpenFlow (рисунок 2.15). Це повідомлення починається зі стандартного заголовка та супроводжується файлом cookie, який є непрозорим полем, встановленим контролером, його маскою, ідентифікатором таблиці та коман-

дою, що визначає тип модифікації таблиці потоків. Далі слідує idle_timeout, hard_timeout і пріоритет. Idle_timeout - це кількість секунд, після яких запис потоку видаляється з таблиці потоків, оскільки жоден з пакетів не відповідає їй. Hard_timeout – це кількість секунд, після яких запис потоку видаляється з таблиці потоків незалежно від цього, збігаються чи пакети з нею чи ні. Пріоритет має на увазі порядок збігів, які перекриваються з вищими числами, що становлять вищий пріоритет. Далі у Flow_Mod йдуть buffer_id, out_port, out_group та прапори. Buffer_id – це ідентифікатор буферизованого пакета. Наприкінці додаються збіг та інструкції.

Рисунок 2.15 – Структура повідомлення Flow_Mod

Повідомлення Packet_In – це спосіб для комутатора відправити захоплений пакет контролеру. Це може статися через явну дію в результаті збігу, що запитує таку поведінку, або через пропуск таблиці.

Повідомлення Packet_In складається з заголовка, за яким слідує ідентифікатор буфера. Ідентифікатор буфера - це унікальне значення для відстеження буферизованого пакета. Довжина захопленого пакета вказується на total_len. Поле причини вказує, чому пакет був захоплений та відправлений. Table_id – це id таблиці, яка переглядалася.

Cookie використовується для ідентифікації певного потоку, який відпо-

відає цьому пакету. Нарешті захоплена частина пакету починається відразу за pad (рисунок 2.16).

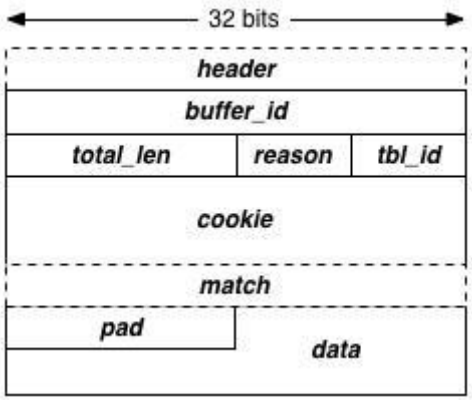


Рисунок 2.16 – Структура повідомлення PacketIn

Було ініційовано пінг з h1 на h2 Mininet (рисунок 2.17).

```
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 2.17 – Пінг з h1 до h2

На рисунку 2.18 показано виведення повідомлення PACKET_IN від комутатора контролеру, оскільки для вхідного пакета на комутаторі не знайдено збігів набору записів і h1 ще не знає, як дістатися до h2. Це повідомлення ARP, інкапсульоване OpenFlow через TCP. Контролер за допомогою OpenFlow-повідомлення типу контролер-комутатор Flow_mod дає команду комутатору запам'ятати розташування вузла h1. Наступним кроком контролер розсилає ARP-запит, інкапсульований у повідомлення Packet_out, тільки ті порти комутатора, до яких підключені вузли.

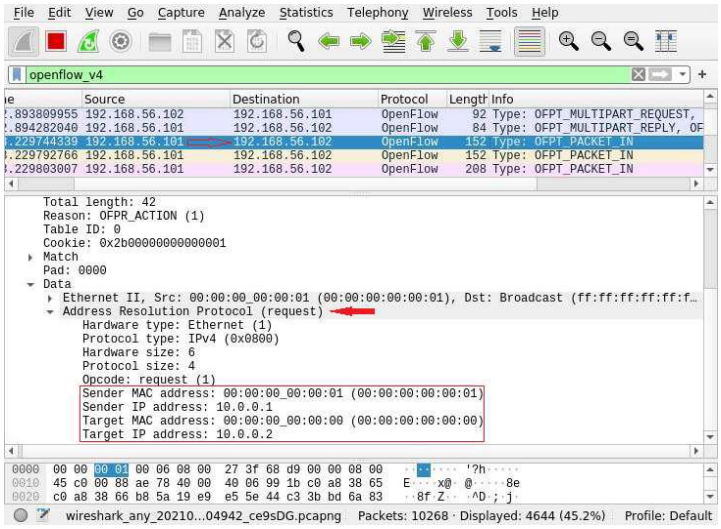


Рисунок 2.18 – ARP запит від h1 до h2

Вузол h2, що виявив збіг шуканої MAC-адреси з власним, відправляє в мережу ARP-відповідь, яка ретранслюється комутатором на контролер. Контролер тепер знає розташування вузла h2 встановлює відповідне правило таблиці потоків комутатора, після чого ARP-відповідь відправляється вузлу h1 (рисунок 2.19).

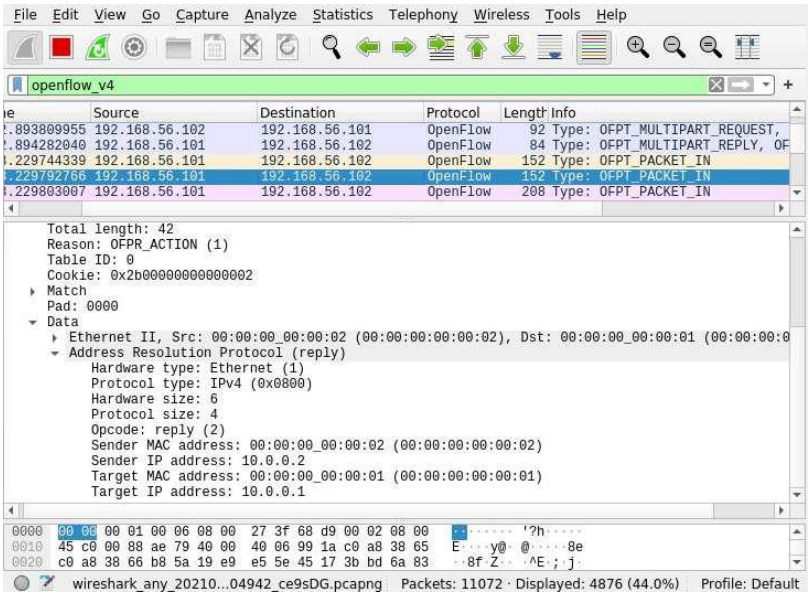


Рисунок 2.19 – ARP відповідь h2

Після додавання запису таблицю потоків, вузол h1 може відправити ICMP пакет вузлу h2 (рисунок 2.20).

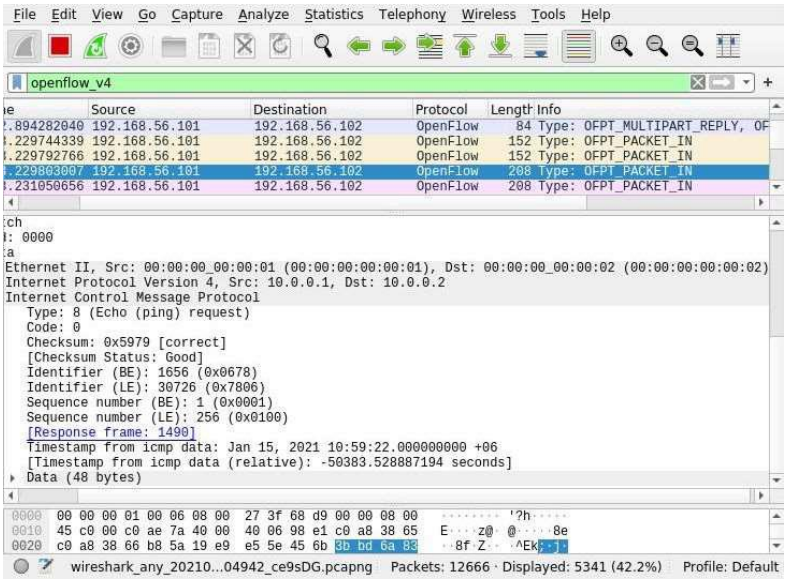


Рисунок 2.20 – ICMP запит

Далі вузлу h2 потрібно відправити ICMP у відповідь пакет вузлу h1 (рисунок 2.21).

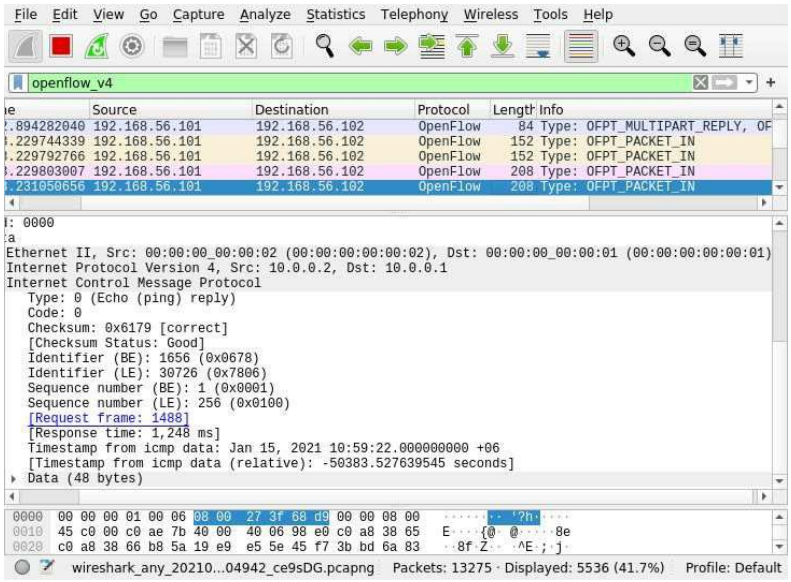


Рисунок 2.21 – ICMP відповідь

2.3 Створення записів у таблиці потоків

Для дослідження впливу записів потоку на трафік у віртуальній машині mininet створено топологію з одним комутатором і трьома хостами в резуль-

таті введення команди (sudo mn - topo = single, 3 - controller = none - Mac), показаної на рисунку 2.22.

Параметр -- controller=none виключає OpenFlow-контролер із топології в даному прикладі. Параметр topo визначає топологію мережі, наприклад, якщо змінимо значення параметра (--topo = 2,3), то топологія складатиметься з двох OpenFlow-комутаторів і кожен з комутаторів матиме по 4 хости. Команда dump дозволяє вивести інформацію про всі вузли мережі та хості з параметрами інтерфейсів. Команда net демонструє як з'єднані хости та вузли.

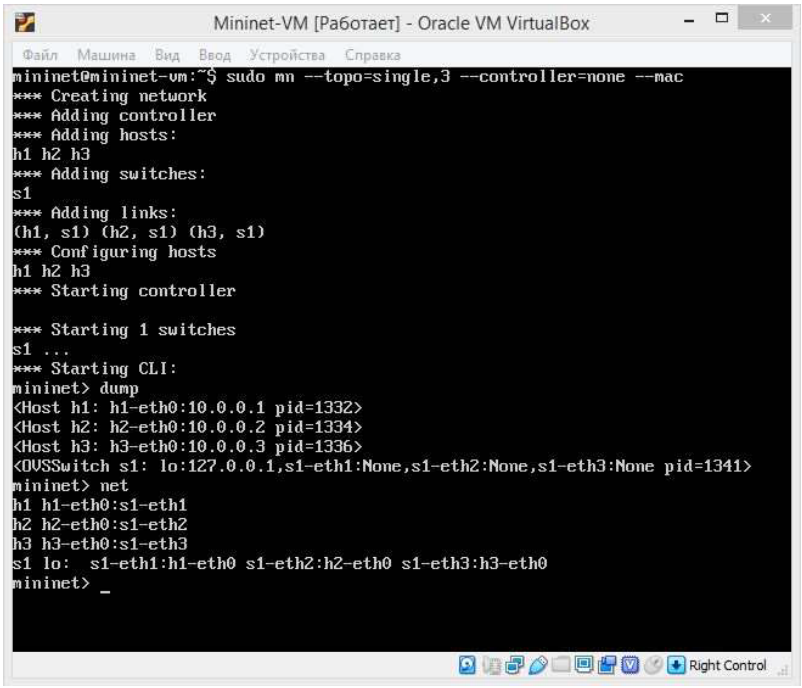


Рисунок 2.22 – Створення найпростішої топології мережі із OpenFlow-коммутатором

Команда (sh ovs-ofctl show s1) дозволяє переглянути параметри OpenFlow комутатора (рисунок 2.23).

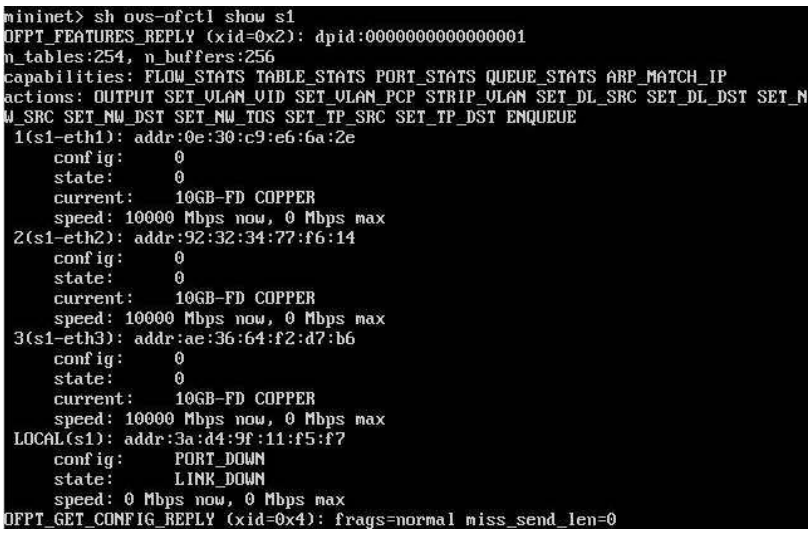


Рисунок 2.23 – Параметри OpenFlow комутатора

Для перевірки зв'язку між хостами потрібно додати OpenFlow запису. З рисунку 2.24 видно, що якщо не задати жодного правила на OpenFlow- комутатор, то зв'язку в локальній мережі не буде. Щоб створити стандартний OpenFlow-запис, потрібно скористатися командою (sh ovs-ofctl add-flow s1 action=normal).

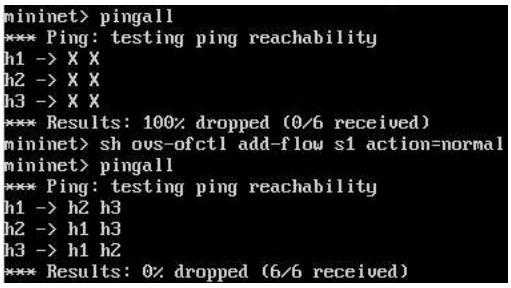


Рисунок 2.24 – Перевірка зв'язку між хостами

Для отримання даних про записи потоку, використовується наступна команда (sh ovs-ofctl dump-flows s1), рисунок 2.25.

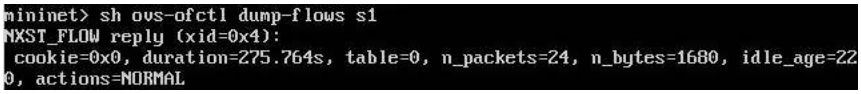


Рисунок 2.25 – Таблиця потоків

Щоб видалити всі записи потоків, необхідно скористатися командою

(sh ovs-ofctl del-flows s1).

Створимо записи у таблиці потоків, які вказуватимуть комутатору який із портів відправляти трафік, що надійшов із зазначеного порту. Такий запис називається Layer 1 matching (рис. 2.26).

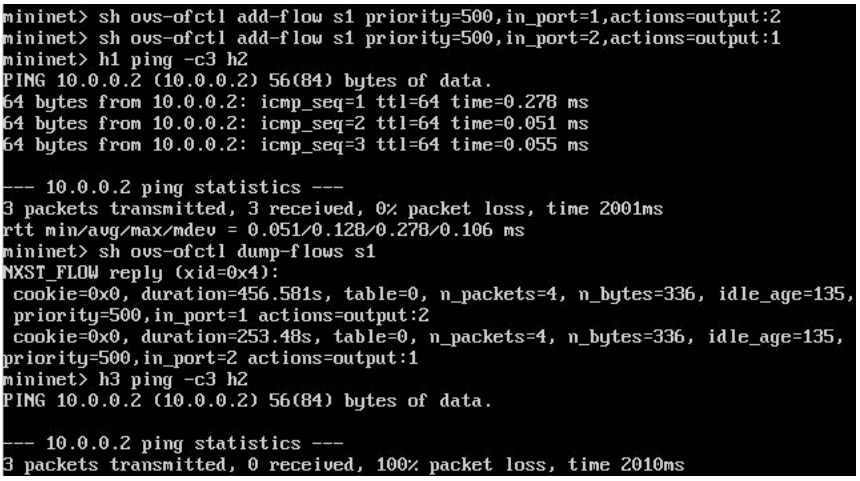


Рисунок 2.26 – Створення записів у таблиці потоків

Тепер OpenFlow комутатор може застосувати певні йому правила при передачі трафіку з порту 1 на порт 2 і навпаки. Як видно з рисунка 2.26, зв'язок з хостом h3 немає саме через відсутність правил для порту, в якому він підключений. Після застосування команди (sh ovs-ofctl dump-flows s1) видно, як змінилася таблиця потоків OpenFlow комутатора.

Управляти правилами таблиці потоків дозволяє полі пріоритету (priority). Подивимося, як впливає додавання правила з великим значенням priority на передачу трафіку, а потім видалимо його за допомогою --strict (рисунок 2.27).

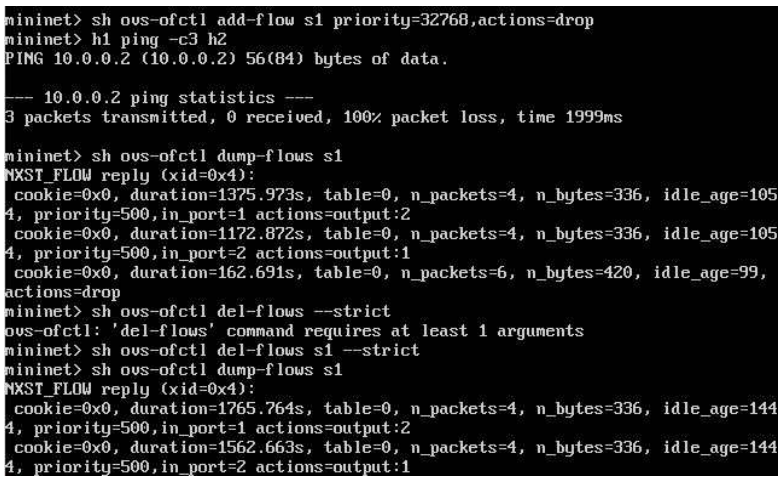


Рисунок 2.27 – Пріоритет записів у таблиці потоків

Наступне правило порівнюватиме MAC-адресу джерела трафіку та MAC-адресу призначення, і відправлятиме трафік, відповідно до значення параметра output. Такий запис називається Layer 2 matching (рис. 2.28).

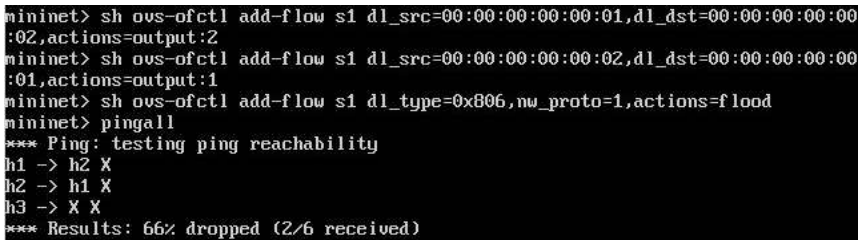


Рисунок 2.28 – Layer 2 matching

Для коректної роботи L2 необхідно прописати правило, яке контролюватиме ARP-запити. Перші 2 рядки команди описують передачу трафіку згідно з фізичними адресами, але для того, щоб OpenFlow-комутатор отримав відомості про те, за яким IP закріплено потрібну MAC-адресу, потрібно описати правило, яке дозволяє широкомовний трафік. В даному прикладі дія flood дає можливість відправляти ARP-REQUEST на всі порти, крім того, з якого було згенеровано запит.

Третій запис у таблиці потоків описує політику передачі трафіку на основі IP адресації та ToS – Layer 3 matching. Запис (sh ovs-ofctl add-flow s1 priority=500,ip,nw_src=10.0.0.3,actions=mod_nw_tos:184,normal). У цьому за-

писі описується ToS, який дозволяє маркувати трафік спеціальною міткою DSCP (Differenciated Services Code Point), що визначає його пріоритет. Якщо зверне увагу на сам запис, параметр priority відрізняється від DSCP тим, що задає порядок порівняння пакетів переданого трафіку із записами в таблиці потоків. Якщо підвищити значення priority у записи для ToS, то пакет спочатку пройде порівняння з цього запису, а потім з усіх інших. На рисунку 2.29 показано приклад створення Layer 3 matching.

```
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 2 .29 – Layer 3 matching

3 МОДЕЛЮВАННЯ НАСКРІЗНОЇ ЗАТРИМКИ В ПРОГРАМНО-КОНФІГУРОВАНІЙ МЕРЕЖІ

3.1 Проактивний та реактивний режим роботи

У SDN, коли контролер заповнює таблиці потоків, існує два основних режими роботи: проактивна установка правил та реактивна установка правил [16].

Реактивний режим таблиця потоків спочатку порожня. Коли пакет прибуває на комутатор, ніяке правило не виконується, і пакет буде інкапсульований як повідомлення PACKET_IN і відправлений на контролер центральним процесором даних у комутаторі. У той же час, комутатор буферизує пакет, чекаючи команди контролера. Контролер завантажує відповідні правила у комутатор. Наступні пакети того ж потоку потім відповідають знову встановленим правилам і передаються на повну швидкість лінії, не порушуючи роботу контролера.

Проактивний режим таблиця потоків встановлюється в комутаторі заздалегідь, до прибуття пакета. Для досягнення продуктивності пошуку на провідній швидкості сучасні комутатори зазвичай використовують TCAM для зберігання цих правил. TCAM може зіставляти кожен пакет з усіма правилами паралельно і виводити результат за один такт. Однак у комутаторі SDN може не вистачити місця для виконання всіх правил, особливо у великомасштабних мережах, з таких причин:

- TCAM коштує дорого і споживає велику кількість енергії. Сучасні набори мікросхем TCAM зазвичай підтримують лише кілька тисяч чи десятки тисяч записів, що від задоволення поточних вимог до управління мережею [7, 8];

- розрив між значним збільшенням трафіку та повільним збільшенням ємності TCAM збільшується. Масштаб таблиці потоків постійно

розширюється;

Розмір таблиці SDN також збільшується. Візьмемо, наприклад, OpenFlow, один з чітко визначених відкритих південних API, кількість полів відповідності становить 12 у версії 1.0, але 45 у версії 1.5. Це означає, що кожне правило займатиме більше місця для зберігання.

Коли TCAM заповнено і потрібно вставити нове правило, потрібно видалити старе правило, щоб звільнити місце для нового правила. Незалежно від того, який алгоритм заміни прийнятий, він неминуче призведе до видалення деяких активних правил таблиці потоків. Наступні пакети, що відповідають цим віддаленим правилам, повинні викликати генерацію вхідних пакетів на контролер.

Наступні два сценарії підкреслюють, що навіть якщо комутатор sdn налаштований у проактивному режимі, все ж таки можливо, що буде активовано налаштування потоків:

1) деякі дослідники, такі як DevoFlow [18], запропонували використовувати грубі правила, щоб максимально скоротити кількість звернень до контролера. Однак це дилема: агресивне використання підстановних знаків відповідності потоку підриває здатність контролера ефективно керувати мережевим трафіком, а також не може проводити точні виміри та збір статистики. Фактично щоб покращити використання мережі та продуктивність додатків та задовольнити такі потреби, як безпека, вимірювання та справедливість у різних мережевих сценаріях, пропонується все більше і більше політик управління трафіком. Ці політики зазвичай перетворюються у детальні правила (наприклад, правило контролю доступу, правила обмеження швидкості та маршрутизації політик) [19]. Кількість деталізованих правил, ймовірно, може досягати сотень тисяч або навіть мільйонів [19], які навряд чи можуть бути збережені в поточному TCAM. Це неминуче призведе до заміни правил та налаштування потоку.

2) Через обмеження ЦП, пам'яті та зміни трафіку віртуальні машини часто потрібні для міграції як усередині, так і між центрами обробки даних

[10]. Однак віртуальні машини зазвичай мають тісний зв'язок із мережею. Наприклад, мережеві політики, такі як QOS, ACL і маршрути політик, у комутаторах зазвичай залежать від віртуальних машин. Отже, міграція віртуальних машин часто спричиняє зміну конфігурації мережевих політик. Відповідні правила комутатора мають бути вилучені, і цей процес викликає оновлення таблиці потоків. Тим часом комутатор пов'язаний з новим сервером динамічно запускає вставку правила для перенаправлення трафіку на новий фізичний інтерфейс.

3.2 Модель наскрізної затримки

Наскрізна затримка кожного пакета – це сума затримок, що випробовуються у послідовності проміжних вузлів на шляху до місця призначення. Кожна затримка являє собою суму двох частин: фіксованої частини, такої як затримка передачі та затримка поширення, і змінної частини, такий як затримка обробки та затримка постановки у чергу у вузлах.

На рисвунку 3.1 показано розбивку затримок при наскрізному проходженні пакетів. Пакети починаються з вихідного хоста, проходять через кілька комутаторів та досягають свого кінцевого хоста. Загальна затримка D включає затримку обробки протоколу (StkD), яка являє собою затримку обробки, пов'язану зі стеком протоколів вихідного і кінцевого хоста; затримку передачі (TD) на вихідному та кінцевому хостах; затримку поширення (PD) у середовищі передачі; затримку комутатора (SD), викликану пересиланням пакетів у комутаторах. Наскрізну затримку можна подати такою формулою:

$$D = StkD_{src} + TD_{src} + PD + SD + TD_{dst} + StkD_{dst}. \qquad (3.1)$$

Наскрізна затримка починається з моменту, коли програма на вихідному хості надсилає повідомлення в інтерфейс сокету. Потім повідомлення обробляється стеком протоколів, і карта інтерфейсу витягує пакет з пам'яті хос-

та і відправляє його на фізичний рівень. Витрати часу в описаній вище процедурі представлені $StkD_{src}$. Процес прийому на хості призначення – процес, зворотний надсилання пакета. $StkD_{dst}$– це величина затримки, з якою пакет обробляється стеком протоколів на хості призначення. $StkD_{dst}$– це часовий інтервал між моментом, коли мережна карта починає копіювати пакет на згадку з використанням DMA, і моментом, коли програма на хості призначення отримує цей пакет.

3.3 Модель адаптації системи моніторингу до властивостей процесу передавання даних

Для забезпечення процесів управління мережею, побудованою за основними принципами SDN, необхідно здійснити модернізацію механізмів моніторингу стану її ресурсів, оскільки при використанні стандартного механізму спостерігається генерація надлишкової службової інформації для елементів, що простоюють. Цей факт може мати негативний вплив на ефективність роботи мережі загалом через завантаженість каналів.

У зв’язку з недоліками існуючих систем моніторингу, визначеними в першому розділі роботи, пропонуємо використовувати метод динамічної адаптації параметрів системи моніторингу. Наприклад, зміну інтенсивності моніторингу стану мережевого елемента залежно від його поточної завантаженості. Інтелектуальний моніторинг на мережі реалізується шляхом встановлення головного додатку моніторингу на контролер та відповідних підконтрольних йому агентів на кожен комутатор.

Процес моніторингу ґрунтується на двох основних підходах:

- перший – зміна інтенсивності моніторингу залежно від попередньо накопиченої статистичної інформації про стан завантаженості в пам’яті керуючого пристрою. На основі аналізу зібраної інформації можна визначати характеристики трафіку в певний період часу та змінювати інтенсивність

відповідно до визначених потреб. Наприклад, у години пікового навантаження (як правило, під час ранкового робочого періоду) слід збільшувати інтенсивність моніторингу, а вночі можна її зменшити;

- другий – динамічна зміна інтенсивності моніторингу залежно від завантаженості елемента мережі на основі зібраних у режимі реального часу даних стану елемента мережі. У випадку наближення завантаженості до критичного значення збільшується частота опитування мережевих вузлів, таким чином здійснюється перехід у стан пильного моніторингу відповідного елемента мережі.

Аналіз моніторингу виконує додаток, встановлений на контролері, а підконтрольні йому агенти виконують функції збору інформації, її модифікацію та відправлення на контролер. Диференціація інтенсивності моніторингу окремих сегментів мережі дає можливість усунути надлишкові процеси моніторингу та обробки даних, виконуючи необхідну інтенсивність лише на тих вузлах, де це необхідно. На основі диференціації інтенсивності моніторингу можна збільшити гнучкість управління елементами мережі, та, як наслідок, ефективніше використовувати обчислювальні ресурси пристроїв рівня управління. Також, володіючи актуальною службовою інформацією, можна здійснювати оптимальний розподіл навантаження на мережі, попереджуючи таким чином негативні явища перевантаження мережевих вузлів. Окрім того, можна переводити елементи, що простоюють, у режим очікування, зберігаючи при цьому можливість надійного і швидкого їх відновлення до нормального режиму роботи. Це значно зменшує витрати на електроенергію та збільшує час «життя» мережевих пристроїв. Повністю відключати обладнання не завжди доцільно, оскільки його ввімкнення потребує певного часу для відновлення таблиць комутації, запуску всіх апаратних та програмних процесів комутатора, що загалом впливає на час перебудови топології мережі та спричиняє стрибок службової інформації між контролером та комутатором.

Особливістю розглянутого механізму є зміна інтенсивності моніторингу лише конкретного мережевого вузла, який цього потребує, однак це не

впливає на інтенсивність моніторингу інших вузлів. Пропонований механізм є адаптованим до сучасного динамічного трафіку та забезпечує хорошу базу для подальшого процесу балансування навантаження. Для прикладу, чим вище миттєве значення завантаження каналу, тим швидше відбудеться наступне опитування. Тобто, чим більше завантаження каналу, тим частіше система моніторингу здійснює опитування щодо кількості переданої інформації. Такий метод можна використовувати стосовно усіх параметрів комутатора у випадку, коли підвищення інтенсивності їхнього опитування не впливає негативно на характеристики продуктивності та якості функціонування пристрою.

Ще одним важливим параметром процесу моніторингу є кількість інформації, що генерується цим процесом. У випадку підвищення інтенсивності моніторингу кількість службової інформації і, як наслідок, навантаження на інформаційний канал зростатиме, що загалом призведе до часткового зниження ефективності використання інформаційного каналу. Функція зміни інтенсивності моніторингу буде різною для кожного окремого параметру, а її максимальне значення залежатиме від рівня навантаження процесу моніторингу на мережеві елементи. Обмеження щодо частоти моніторингу деяких параметрів часто встановлюють виробники комутаторів, вказуючи критичне значення в документації до пристроїв. Емпірична модель адаптації системи моніторингу, зокрема частоти опитування комутатора, до завантаження мережних каналів (3.2):

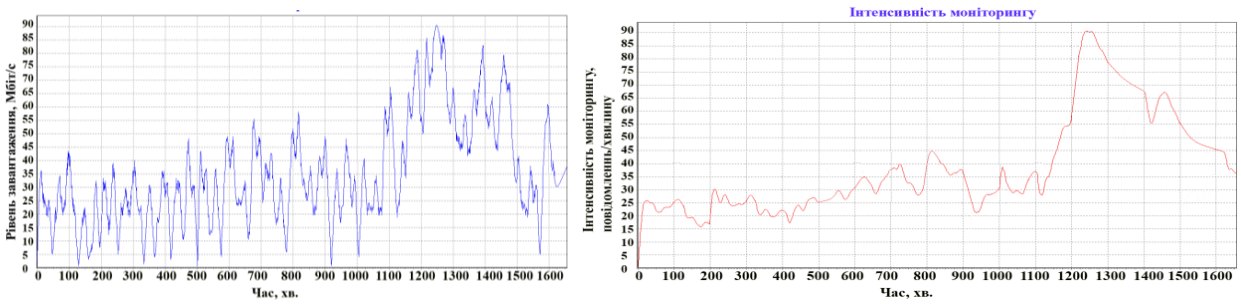
$$f(\rho_i) = \begin{cases} 1 - \dfrac{\rho_i}{2}, & 0 < \rho_i \le 0.5; \\ \dfrac{0.1}{(\rho_i - 0.367)}, & 0.5 < \rho_i \le 1, \end{cases} \tag{3.2}$$

де f – функція завантаженості, що використовується для визначення інтервалу опитування комутатора; ρ – завантаження інтерфейсу комутатора,

безрозмірний коефіцієнт; *i* – порядковий номер опитування.

Всі параметри поділяються на декілька категорій. До першої категорії належать параметри, що фіксуються постійно: завантаження каналу, завантаження комутатора. Ці параметри є критично важливими, оскільки перевантаження вказаних елементів призведе до відмови в обслуговуванні, суттєвих втрат інформації та погіршення якості. До другої категорії відносяться параметри, які необхідно додатково спостерігати у випадку, якщо один з параметрів першої категорії досягнув критичного рівня. Наприклад, у разі перевантаження одного з каналів слід додатково отримати інформацію щодо кожного потоку, який проходить через цей канал (швидкість потоку, втрати). Після цього на основі пріоритетів потоку здійснити моніторинг параметрів якості обслуговування для потоку реального часу. Якщо поточне значення параметрів якості обслуговування не відповідає вимогам, встановленим для цього потоку, тоді інформація про потік передається в частину системи керування, що відповідає за перерозподіл потоків у мережі.

На рисунку 3.1 показано адаптацію частоти моніторингу до завантаженості мережевого елемента. У разі збільшення величини x та падіння навантаження на елемент мережі до 70% інтенсивність процесу моніторингу стабілізується до нормального режиму. Цей механізм збільшуватиме інтенсивність моніторингу не мережі загалом, а лише проблемних ділянок мережі, що не спричинятиме стрімкого росту службової інформації , а також не потребуватиме великої обчислювальної потужності контролера, отже, не перевантажуватиме його.



а б

Рисунок 3.1 - Завантаженість мережевого елемента (а) та зміна інтенсивності моніторингу згідно даної завантаженості (б)

Із наближенням інтенсивності навантаження до пікового значення для даного елементу мережі, здійснюється ряд запобіжних заходів, зокрема: прогнозування, а також приведення в нормальний режим роботи резервних ресурсів мережі. Такі рішення дають можливість швидко та адекватно реагувати на певні критичні випадки, що виникатимуть на транспортній мережі, максимально збільшуючи при цьому стійкість системи до відмов.

3.4 Метод вимірювання затримки передавання пакетів для потоків окремого користувача

Для сервісів потокового характеру втрати пакетів або зміна порядку надходження пакетів спричиняє погіршення якості обслуговування, що проявляється у формі завмирання та спотворення кадру відео чи фрагменту аудіо даних. Для того, щоб забезпечити належний рівень якості обслуговування, а відповідно і сприйняття, моніторинг параметрів якості обслуговування необхідно проводити не окремо на кожному комутаторі, а загалом з моменту входження трафіку в мережу до моменту його виходу з неї.

Методи моніторингу трафіку завжди були актуальними серед науковців у сфері телекомунікацій. Зазвичай, він зводиться до вимірювання обсягів і характеристик трафіку для окремої мережі. На сьогодні велика кількість методів використовується для вимірювання завантаження каналу, затримки передачі пакету з кінця в кінець, а також втрат пакетів. Архітектура ПКМ, завдяки централізованій площині керування, дає змогу управляти багатьма потоками та приймати складні рішення в єдиній логічній точці системи.

Запропонована система моніторингу дає змогу оцінити параметри якості обслуговування для довільних потоків у програмно керованих мережах. Основна ідея вимірювання – використовувати додаток моніторингу як опорну точку. Для цього в мережу вводиться тестовий пакет, заголовок якого від-

повідає заголовку пакетів, що належать цьому потоку. Пакет вводиться в мережу на комутаторі, де потік входить у мережу, і виводиться з площини даних на комутаторі, де потік покидає мережу. При цьому корисне навантаження тестового потоку містить дві важливі змінні: перша змінна включає ідентифікатор потоку; друга – момент введення потоку в мережу. Тестовий пакет періодично вводиться у мережу з наперед визначеним інтервалом, що залежить від рівня завантаження каналу та швидкості потоку, для якого вимірюється затримка. Таким чином, пакет передається шляхом проходження всіх пакетів цього потоку і зазнає такої самої затримки. Схему вимірювання затримки потоку відображено на рисунку 3.2.

На останньому комутаторі тестовий пакет виводиться із площини передачі даних і відправляється в систему моніторингу. Система моніторингу визначає затримку цього пакету як різницю між часом його відправлення, отриманим з корисного навантаження цього пакету, та часом надходження, який фіксується в момент його отримання додатком моніторингу.

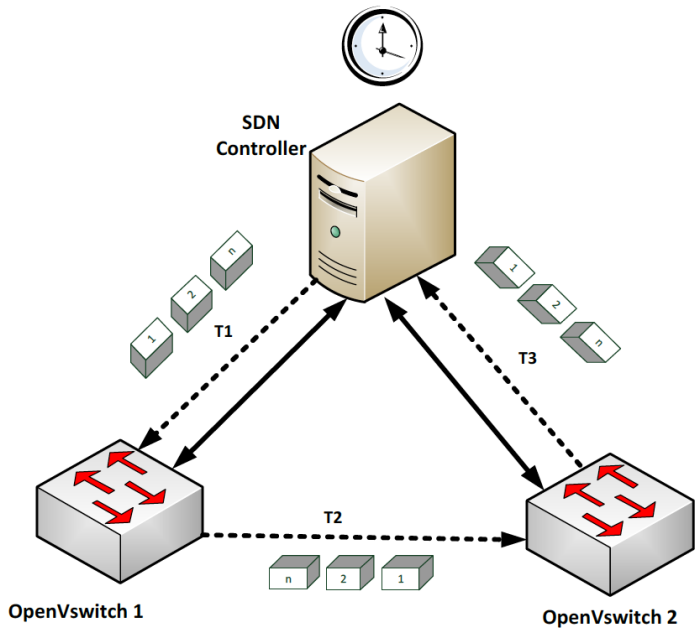


Рисунок 3.2 - Схема вимірювання затримки окремого потоку в програмно-керованій мережі SDN

Алгоритми моніторингу параметрів якості обслуговування представлено на рисунку 3.3. Вся інформація про затримки потоків зберігається в спеціалізованій таблиці. Використовуючи ці дані, інші процеси на контролері мають змогу здійснювати управління трафіком та оптимізацію якості обслуговування для окремих потоків.

Перший алгоритм слідкує за розкладом моніторингу потоків, які були внесені контролером у список таких, що потребують оцінки параметрів якості обслуговування. У випадку, якщо такий список не порожній, алгоритм відправляє тестові пакети в мережу та записує їхні копії в спеціальну чергу. Ці копії використовуються другим алгоритмом для співставлення з отриманими пакетами з мережі.

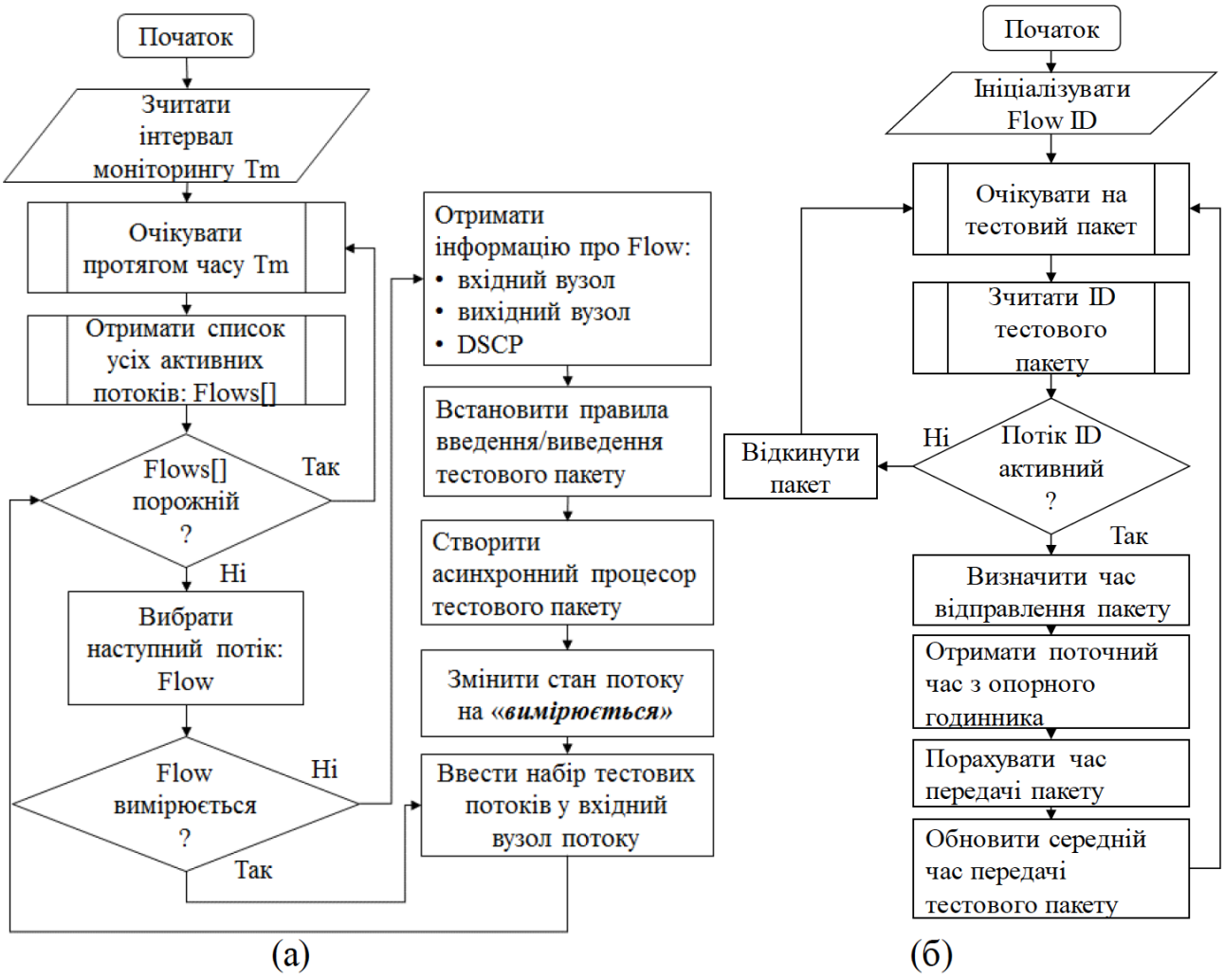


Рисунок 3.3 - Алгоритми відправки (а) та очікування (б) пакету для вимірювання затримки потоку

Особливістю запропонованого алгоритму є те, яким саме чином тестовий пакет передається по мережі. Для цього можуть використовуватися додаткові правила, що встановлюються для комутації такого пакету вздовж шляху, для якого вимірюється затримка. Додаткове правило встановлюється на кожному комутаторі. У запропонованому методі додаткове правило встановлюється тільки на комутаторі, де потік покидає мережу. На всіх інших вузлах пакет порівнюється з тим же ж правилом, з яким порівнюються всі інші пакети потоку. Тому, щоб відокремити тестовий пакет на вихідному комутаторі з потоку, пропонується структура правила, подана у таблиці 3.1

Таблиця 3.1 - Приклад правила в таблиці потоків для вимірювання затримки потоку

Порт	DST MAC	SRC MAC	ETH TYPE	IP TOS	DST IP	SRC IP	IP PROTO	TCP/ UDP SRC	TCP/ UDP DST
*	*	*	*	*	*	*	*	*	40000/ 4

Відомо, що програмно-керовані комутатори мають змогу порівнювати поля заголовків пакетів з полями правил, використовуючи маску. Це, в першу чергу, застосовується для реалізації функцій маршрутизації, коли комутатор порівнює не всю IP адресу, а лише її мережну частину, ігноруючи при цьому біти, що відповідають за адресу станції. Для цього, встановлюючи правило, контролер явно задає комутатору кількість біт адреси, які повинні ігноруватися. Саме ця властивість використовується для вилучення тестових пакетів з потоку. Порівняння на основі TCP чи UDP портів застосовується в мережах вкрай рідко, переважно на крайових вузлах. Такий базовий аналіз протоколів транспортного рівня дає змогу створити найпростіший Firewall на межі мережі та заблокувати проходження окремих потоків, наприклад, заблокувати пакети, які приходять з певної IP адреси та мають встановлені певні порти.

У роботі пропонуємо використовувати діапазон портів від 40000 до 500 0 0, маскуючи останні чотири цифри порту. Коли проводиться вимірювання затримки для конкретного потоку, система моніторингу вибирає для нього порт із зазначеного діапазону та створює відповідне правило, яке повністю дублює правило для передачі цього потоку і порівнює тестовий пакет з вибраним портом.

Схему вимірювання затримки потоку в мережі відображено на рисунку 3.4. Усі тестові пакети, введені в мережу з цим портом, будуть виведені з мережі на кінцевому комутаторі шляху і передані на контролер або ж на систему моніторингу, якщо вона фізично відокремлена.

Зрозуміло, що виміряна затримка враховує тривалість передачі пакету між контролером та вхідним комутатором, а також тривалість передачі між контролером і вихідним комутатором. Тому з кожним десятим тестовим пакетом контролер здійснює оцінку сигнального каналу, а саме – відправляє ICMP запит до обох комутаторів та отримує відповідь.

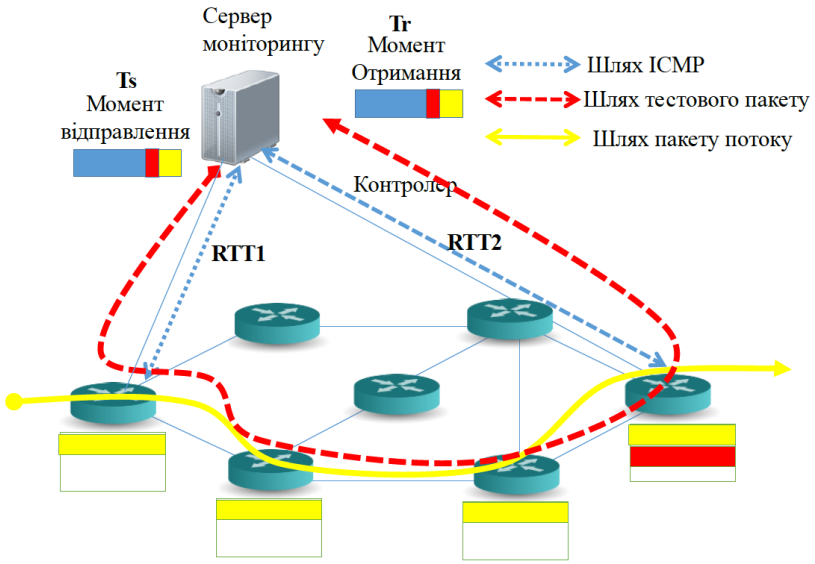


Рисунок 3.4 - Схема вимірювання затримки передачі окремого потоку в мережі

3.5 Алгоритм перерозподілу трафіку на основі відносного пріоритету потоку

Технологія програмно-керованих мереж дає змогу ідентифікувати потік за більшою кількістю ознак, а саме: номером вхідного порту, адресою канального рівня, міткою віртуальної локальної мережі, пріоритетом у віртуальній локальній мережі, типом мережевого протоколу (ARP, ICMP, IP), адресою мережевого рівня (ToS – типом сервісу у випадку, якщо протокол мережевого рівня IP), типом транспортного протоколу (UDP, TCP), значеннями портів транспортного протоколу. Специфікація OpenFlow також передбачає ідентифікацію за міткою MPLS. Аналіз пакету за перерахованими ознаками дає змогу керувати окремими потоками, що належать окремим користувачам, не впливаючи при цьому на всі інші потоки. У такому разі швидкість потоку не має значення, а важливі тільки параметри, що його ідентифікують у мережі.

Для гнучкого управління трафіком важливо врахувати взаємозв'язок між типом трафіку (наприклад, реального часу чи передачі даних з файлового сервера) та мережевих параметрів, зокрема таких: доступна пропускна здатність, втрати пакетів, джиттер та затримка. Аналізуючи типи трафіку, варто зауважити такі моменти:

- VoIP трафік є дуже чутливим до втрат; в ідеалі втрати повинні бути відсутніми, особливо, коли використовується кодек з компресією;

- максимально допустима затримка потокового відео залежить від розміру буфера кінцевих клієнтів, у зв'язку з чим вона може суттєво варіюватися;

- допустимий джиттер не є фундаментальним параметром для передачі потокового відео, а тому до нього немає строгих вимог;

- вимога потокового відео до пропускної здатності залежить від формату кодування та швидкості відеопотоку, а, отже, не є фіксованим значенням.

Більш того, як у мультимедіа, так і в додатках для телефонії, параметри якості обслуговування можуть залежати від типу кодека, механізму виправ-

лення помилок та швидкості складових потоків. Зважаючи на ці зауваження, досить складно точно виміряти якість обслуговування з точки зору користувача, оскільки необхідно врахувати значну кількість різних змінних.

Вимоги щодо якості обслуговування певних класів трафіку, згідно з рекомендаціями ITU-T Y.1541, зведено в таблиці 3.2.

Нульове значення параметра означає, що рекомендація не встановлює ніяких обмежень щодо цього параметра. Його значення може бути довільно вибраним самим оператором мережі. Рекомендований інтервал оцінки параметрів у мережі становить одну хвилину, проте оператори самі можуть вибирати цей інтервал.

Таблиця 3.2 - Класифікація вимог до параметрів якості обслуговування

Параметри функціонування мережі	Класи трафіку					
	Клас 1	Клас 2	Клас 3	Клас 4	Клас 5	Клас 6 (невизначений)
Затримка, мс	100	400	100	400	1000	0
Джиттер, мс	50	50	0	0	0	0
Коефіцієнт втрат, %	0,001	0,001	0,001	0,001	0,001	0
Коефіцієнт помилок, %	0,0001					0

З урахуванням рекомендації щодо важливості тих чи інших параметрів для окремих класів трафіку в роботі сформовано таблицю вимог щодо якості обслуговування існуючих сервісів, що функціонують у мережі Інтернет (таблиця 3.3).

Таблиця 3.3 - Розподіл найпопулярніших сервісів за класами якості обслуговування

Клас	Характеристики	Приклад
1	Реального часу, чутливі до джитеру, інтенсивна взаємодія	VoIP, Skype (аудіо, відео)

2	Реального часу, чутливі до джитте-ру, інтерактивні	VoIP, Інтернет, телеба-чення
3	Транзакції, високоінтерактивні	HTTP, Web-Services
4	Транзакції, інтерактивні	Сигналізація (OpenFlow)
5	Тільки з низькою чутливістю до втрат	FTP, Cloud, потокове (Youtube)
6	Традиційні додатки IP мереж	Будь які інші дані

У результаті аналізу двох таблиць представлено зведену таблиці 3.4 вимог щодо параметрів якості обслуговування для конкретних додатків.

Введемо два додаткових критерії для класифікації потоків: перший критерій – чутливість до неправильного порядку отриманих пакетів; другий критерій – пріоритет окремого користувача.

Кожній сформованій категорії відповідає певна політика керування, згідно якої з потоками конкретної категорії дозволено чи заборонено виконувати певні операції, а також зазначено умови, за яких ці дії можуть бути виконані. Наприклад, у випадку виникнення перевантаження певного шляху в певному комутаторі необхідно здійснити перерозподіл потоків. Ураховуючи те, що потоки першої категорії є чутливими до неправильного порядку пакетів та джиттеру, перенаправлення цих потоків може погіршити якість їхнього обслуговування. Тому розвантаження шляху починається з перерозподілу потоків третьої категорії. При чому, після кожної ітерації перерозподілу чи балансування відбувається вимірювання часових параметрів якості обслуговування спочатку першої категорії, а потім другої. Якщо параметри відповідають затребуваним, і шлях не є перевантаженим, тоді алгоритм перерозподілу завершує свою роботу.

Важливим є те, яким саме чином визначити перевантаження конкретного шляху чи конкретного вузла. Враховуючи максимальну чутливість потоків першої категорії до часових параметрів якості обслуговування, оцінювання якості шляху слід здійснювати саме на їх основі.

У випадку, коли мережа функціонує в нормальному режимі, без пере-

вантаження чи вузьких місць, система моніторингу опитує мережеві пристрої зі стабільно великим інтервалом часу. Зокрема, система моніторингу опитує завантаження мережевих інтерфейсів, завантаження таблиці потоків комутатора та завантаження центрального вхідного буферу маршрутизатора, на якому ідентифіковано перевантаження інтерфейсів.

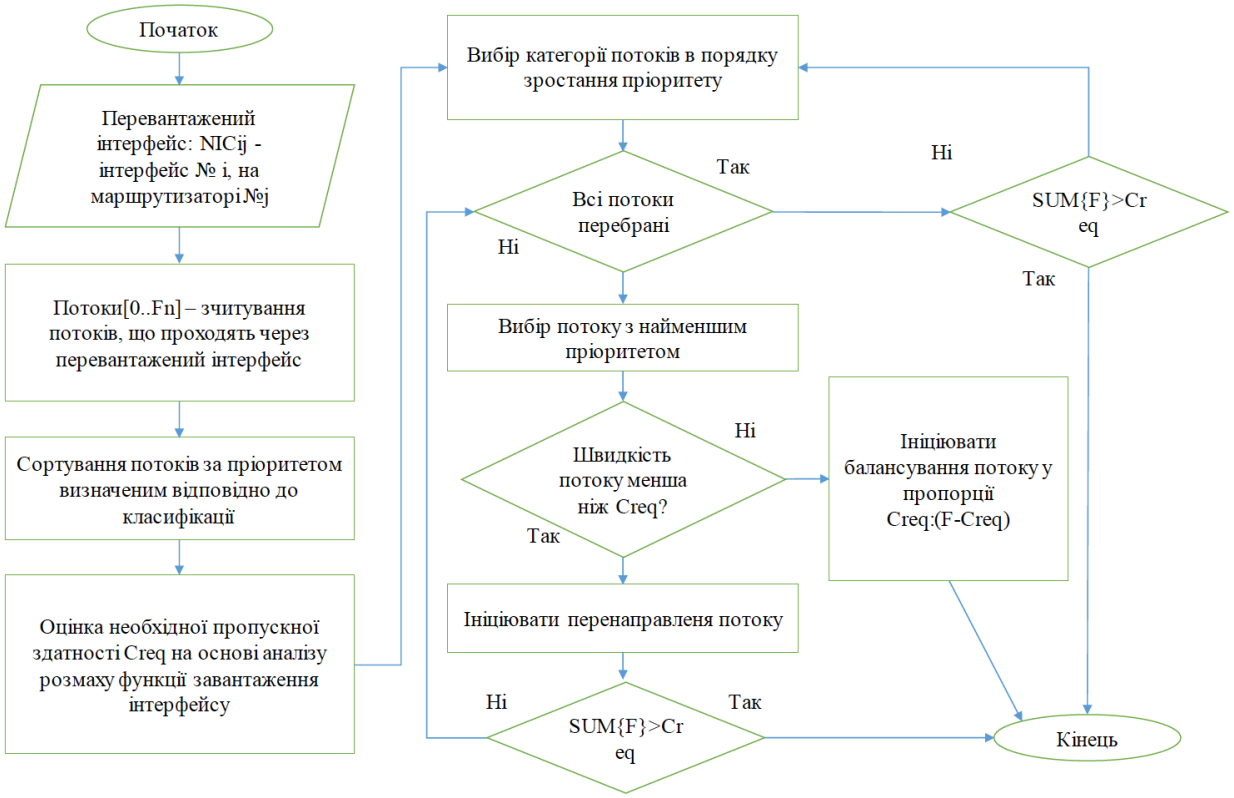


Рисунок 3.5 - Алгоритм перерозподілу потоків у мережі для уникнення перевантаження інтерфейсу

Система моніторингу опитує вузли з певним інтервалом часу та здійснює моніторинг часових параметрів передачі потоків першої категорії. У випадку середньостатистичного зростання завантаження інтерфейсу, затримки передачі та джиттеру система переходить у стан пильного моніторингу зазначеного інтерфейсу та підвищує інтенсивність вимірювання затримки. Крім того, коли рівень завантаження інтерфейсу сягає 0,9, система починає опитувати завантаження буферів пристрою.

В тому разі, коли середньостатистичне завантаження буферів потоку реального часу більше від 1, запускається алгоритм розвантаження інтерфей-

су, описаний вище.

3.6 Експериментальне виявлення залежності затримки передачі службового трафіку від затримки контролера

Для виявлення залежності між затримкою передачі службового трафіку та затримкою контролера в мережі, що функціонує з використанням протоколу OpenFlow у середовищі Mininet, була змодельована деревоподібна топологія, показана на рисунку 3.6. Число комутаторів послідовно ставилося рівним 3, 7, 15 і 31. Виконувалася команда ping з вузла H1 до H2, після чого обчислювалася затримка dτNF, затримка комутатора SNFта затримка контролера CNF. При цьому затримка комутатора SNF вважалася як середня арифметична за всіма OpenFlow-комутаторами (таблиця 3.4).

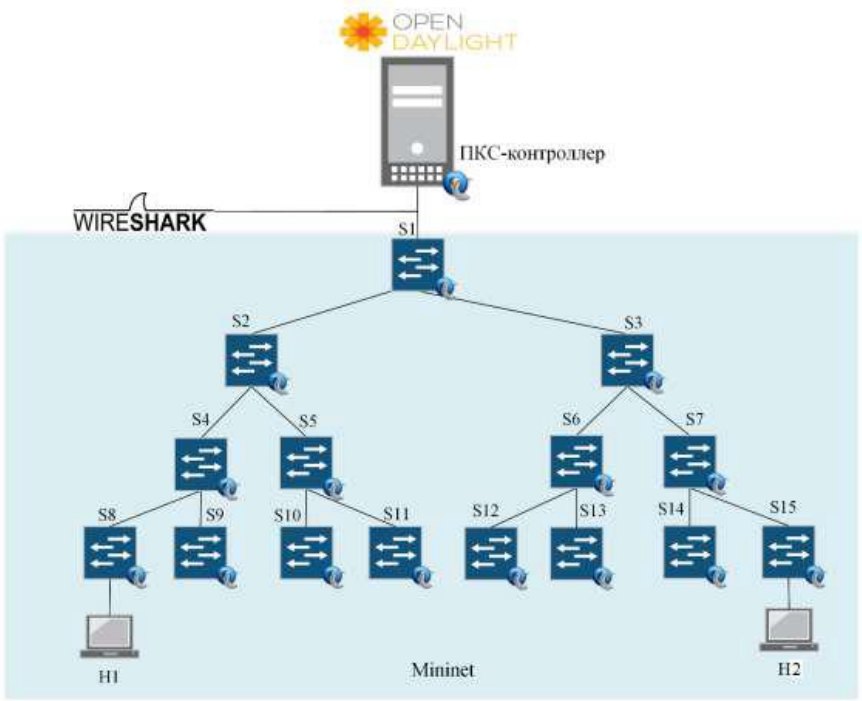


Рисунок 3.6 – Топологія мережі

Таблиця 3.2 – Результати вимірювання, отримані під час експерименту

Кількість комутаторів у мережі, шт.	Затримка $d\tau_{NF}$, мс	Затримка S_{NF}, мс	Затримка CNF, мс

3	9,868	0,184	4,688
7	12,236	0,240	7,433
15	16,252	0,431	6,847
31	23,881	1,870	11,441

За результатами проведеного дослідження було побудовано графік, наведений рисунку 3.7.

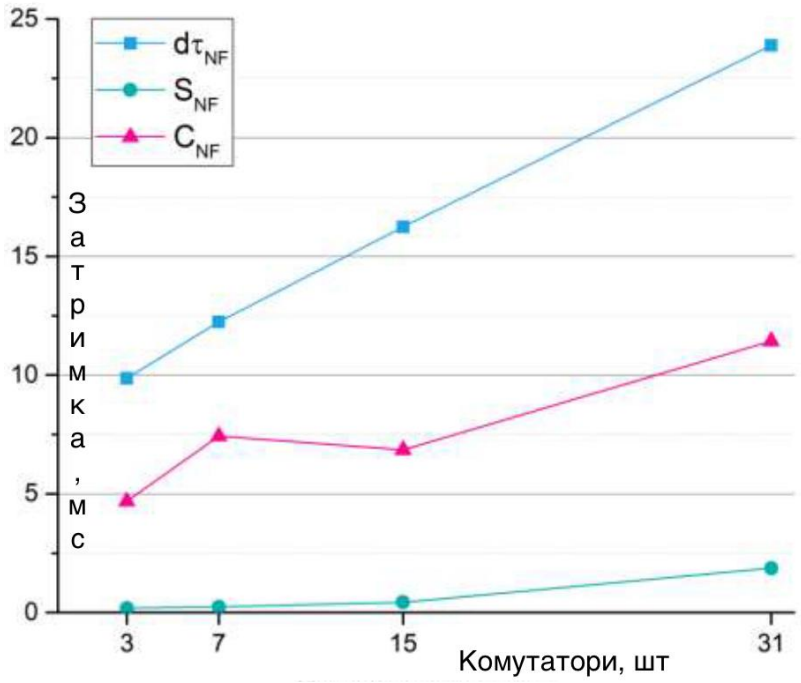


Рисунок 3.7 – Залежність величини затримки від кількості комутаторів у мережі

Виходячи з графіка, висунемо гіпотезу про наявність кореляції між досліджуваними параметрами. Для перевірки цього здійснимо розрахунок коефіцієнтів кореляції попарно для експериментальних значень $d\tau N_F$ та C_{NF}, $d\tau_{NF}$ та S_{NF} згідно з таким рівнянням:

$$r = \frac{\sum xy - \left(\sum x\right)\left(\sum y\right)}{\sqrt{\left(n\sum x^2 - \left(\sum x\right)^2\right)\cdot\left(n\sum y^2 - \left(\sum y\right)^2\right)}} \qquad (3.3)$$

Отримані значення $r(d\tau_{NF}, C_{NF}) = 0.935$, $r(d\tau_{NF}, S_{NF}) = 0.951$ вказують на наявність позитивної лінійної кореляції між досліджуваними параметрами, тобто. зі зростанням/зменшенням значення одного з параметрів зростатиме/зменшуватиметься і інший. Таким чином, зниження затримки, що вноситься контролером, призведе до зниження затримки передачі службового трафіку в мережі SDN.

3.6.1 Експериментальне виявлення затримки трафіку в мережі SDN

Щоб оцінити вплив пакетів на затримку, що надходять, і вплив оновлення TCAM на затримку був створений експериментальний стенд у програмі GNS3. Платформа для виміру показано рисунку 3.8.

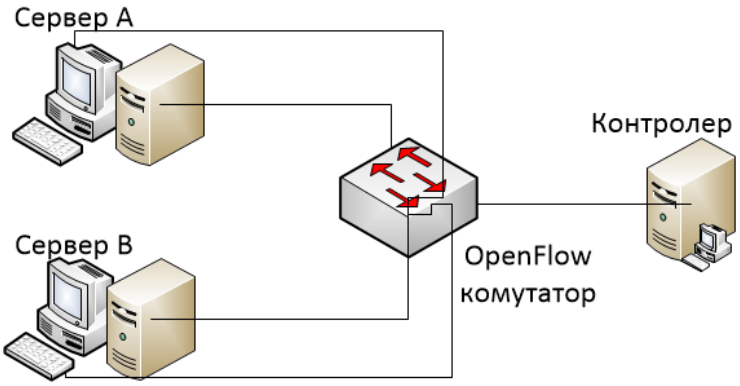


Рисунок 3.8 – Платформа виміру

Експериментальний стенд використовується OpenDaylight (версія Carbon) як працюючий пакет у нашому контролері, комутатор Open vSwitch підтримує протокол OpenFlow. Сервер А використовується для створення пакетів для нових потоків відповідно до різних параметрів. Ці нові потоки будуть викликати налаштування потоку в комутаторі, що тестується. Сервер В використовується для створення базового трафіку. Базовий трафік відноситься до пакетів, які відповідають встановленим правилам TCAM. І сервер А, сервер В встановлено з двома мережевими адаптерами 10 Гбіт/с. Контролер

підключено до порту керування через порт 1 Гбіт/с. Годинник на різних серверах не синхронізується. Щоб уникнути відхилення, викликаного синхронізацією часу, використовується одна мережна карта для надсилання пакетів та інша мережна карта на тому ж комп'ютері для прийому пакетів. Використовується Ostinato для створення певних типів пакетів та потоків трафіку. Будемо використовувати Wireshark для захоплення пакетів.

3.6.2 Вплив надходження пакетів на затримку

У цій частині проведено вимір затримки пакетів, коли пакети packet_in генеруються на контролер. Таблиця потоків у комутаторі спочатку порожня. NIC 1 на сервері A відправляє нові потоки з різними швидкостями на комутатор через порт 1, і нові потоки ініціюють відповідні вхідні пакети packet_in на контролер. Комутатор приймає та аналізує повідомлення flow_mod від контролера та вставляє відповідні записи потоку в TCAM. Встановлено однакові пріоритети всіх цих потокових записів, щоб ізолювати вплив реорганізації TCAM. Потім комутатор пересилає ці пакети на NIC 2. Початкова точка затримки пакета – це момент, коли NIC 1 починає відправлення, а кінцева точка – це момент, коли NIC 2 приймає пакети.

Проведемо три експерименти, щоб показати затримку пакетів з без звернення на контролер при новій швидкості потоку 50, 500 і 5 тисяч пакетів/с. Результати подано на рисунках 3.9, 3.10, 3.11. У кожному експерименті загальна кількість потоків становить 300, а вісь x відноситься до ідентифікатора потоку.

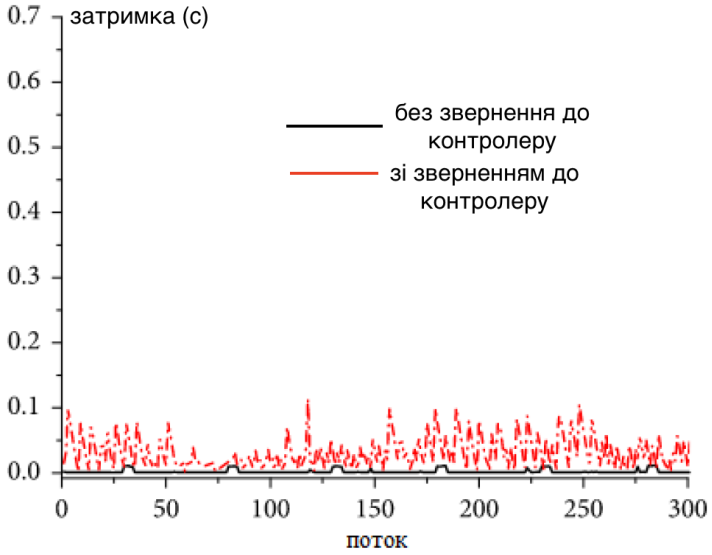


Рисунок 3.9 – Затримка пакетів з/без звернення до контролера, коли нова швидкість потоку становить 50 пакетів/с

Для певної швидкості надходження нового потоку затримка пакетів з зверненням на контролер значно більше, ніж у пакетів без звернення на контролер. Наприклад, коли швидкість надходження потоку становить 5 тисяч пакетів/с, затримка пакета без звернення на контролер є змінною із середнім значенням 0.0859 мс та стандартним відхиленням 0.0452, тоді як середнє значення затримки пакета з зверненням на контролер становить 0.7415 с, а стандартне відхилення 0.5954.

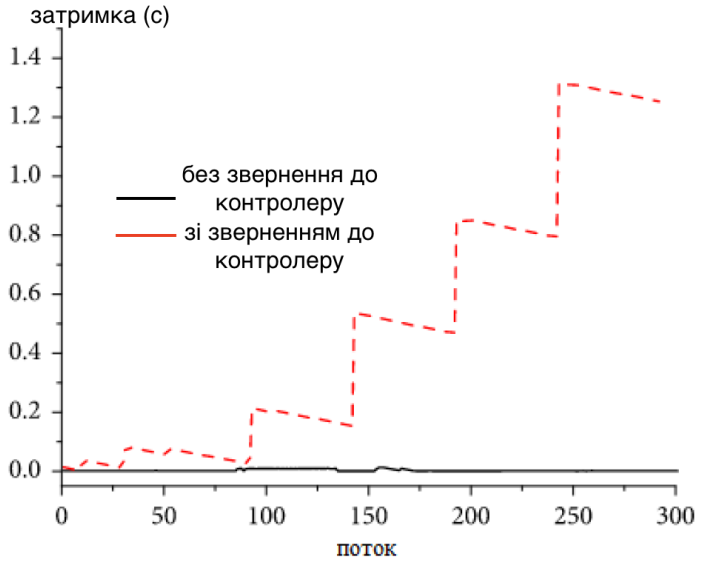


Рисунок 3.10 – Затримка пакетів з/без звернення на контролер, коли нова

швидкість потоку становить 500 пакетів/с

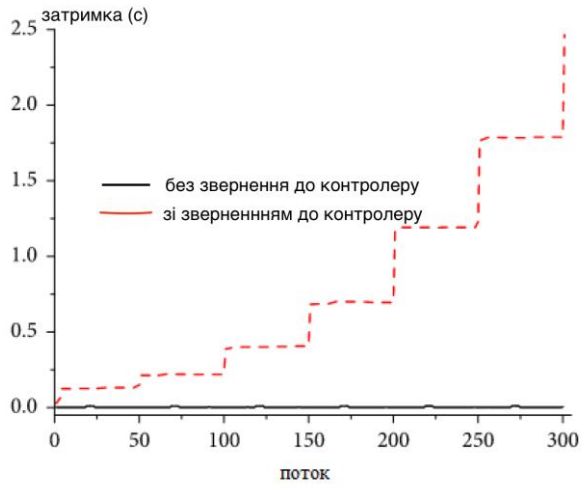


Рисунок 3.11 – Затримка пакетів з/без звернення на контролер, коли нова швидкість потоку становить 5 тисяч пакетів/с

Затримка пакета збільшується зі збільшенням швидкості надходження нового потоку. Наприклад, середня затримка нових потоків зі швидкістю надходження 50 пакетів/с становить 32.291 мс, тоді як затримка нових потоків зі швидкістю надходження 5 тисяч пакетів/с становить 0.7415 с. Максимальна затримка складає 2.4654 с. при 5 тисяч пакетів/с. Також можемо виявити, що розподіл затримки для нових потоків швидкістю надходження 50 пакетів/с відрізняється від розподілу затримки для нових потоків зі швидкістю надходження 500 пакетів/с та 5 тисяч пакетів/с. Значно збільшується затримка для кожні 50 пакетів.

Щоб знайти першопричину цього явища, проведемо наступні дві серії експериментів. З одного боку, використовуємо Cbench інструмент тестування продуктивності контролера SDN, щоб провести вимірювання на контролері. Cbench налаштований як режим пропускної спроможності. Контролер в експериментальній топології управляє 1000 хост через один комутатор. Тест повторюється 10 разів. Експериментальні результати показують, що мінімальна пропускна спроможність становить 451119.36 відповідей/с, максимальна пропускна спроможність становить 473804.17 відповідей/сек, а середня про-

пускна спроможність становить 460435.15 відповідей/сек. З іншого боку, використовуємо Wireshark для захоплення пакетів на мережній карті контролера. Виявляємо, що комутатор відправляє пакети контролеру партіями та розмір партії складає 50 пакетів. Середня затримка кожного нового потоку, що обробляється контролером, становить близько 1-2 мс і суттєво не збільшується. Це означає, що швидкість генерації вхідних пакетів у згаданих вище експериментах не досягає вузького місця продуктивності контролера. У таблиці 3.3 перераховано завантаження ЦП з/без звернення на контролер при новій швидкості потоку 50, 500 та 5 тисяч пакетів/с. З таблиці видно, що завантаження ЦП значно збільшується зі збільшенням нової швидкості потоку, особливо, коли ініціюються вхідні пакети на контролер.

Таблиця 3.3 – Використання ЦП з/без обробки на контролері за різних нових швидкостях потоку

Вид	50 пакетів/с	Швидкість течії 500 пакетів/с	5000 пакетів/с
Без звернення до-контролеру	3% – 4.1%	12%	13.20%
Зі зверненням до-контролеру	24% - 30%	36% - 48%	35.2% - 67%

Грунтуючись на вищезгаданих експериментах, можемо зробити висновок, що локальний ЦП в комутаторі SDN вносить більшу частину затримки. Генерація вхідних пакетів на контролер разом із повідомленням flow_mod споживає дуже багато обчислювальних ресурсів ЦП. Коли швидкість надходження нового потоку збільшується, пакети мають буферизуватися та оброблятися партіями.

3.6.3 Вплив оновлення TCAM на затримку

У цьому розділі досліджуємо затримку комутатора під час оновлення TCAM. Спочатку таблиці потоків зберігається лише одне правило. Це

правило має найнижчий пріоритет і використовується для пересилання базового трафіку. NIC 1 надсилає на комутатор 1600 нових потоків. Швидкість надходження пакетів встановлено на рівні 5 пакетів/с. Нові потоки запускають установку правил, таким чином контролер вставить відповідне правило для кожного потоку TCAM. Потім комутатор пересилає ці пакети на NIC 2, керуючись таблицею потоків. NIC 3 використовується для відправлення базового трафіку. Трафік відповідатиме встановленому правилу. NIC 4 на сервер B отримує ці пакети.

Спершу зосередимося на тому, як пріоритет правила впливає на затримку комутатора. Проводимо три експерименти. Встановлюємо пріоритет правил в одному порядку, порядку зростання та порядку зменшення. Послідовність завантаження правил може бути визначена як r1, r2...rn. Позначимо riяк пріоритет ri. Для зростаючого порядку ri.prio>rj.prio, якщо i>j. Для однакового порядку ri.prio=rj.prio, якщо i>j. У порядку спадання ri.prio<rj.prio, якщо i>j. Результат показаний на рисунку 3.12.

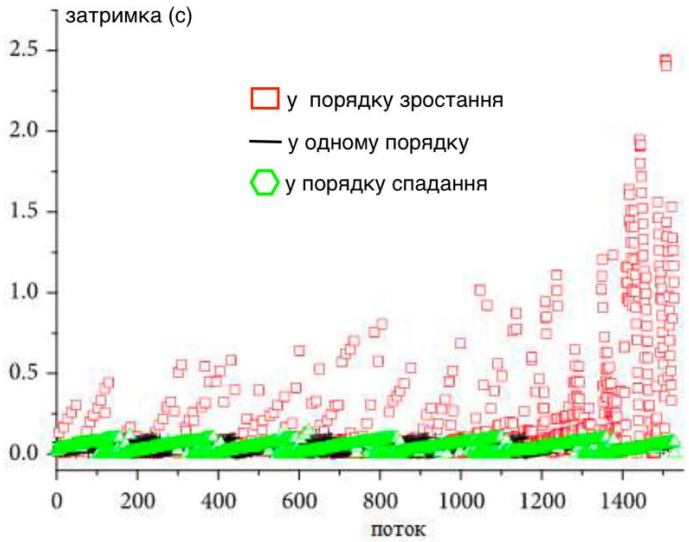


Рисунок 3.12 – Порівняння затримки пакетів, коли пріоритет правил в одному порядку, порядку зростання та порядку зменшення

Основна причина цього явища – механізм організації правил у TCAM. Правило з найвищим пріоритетом завжди зберігається за найменшою

адресою TCAM. Коли вставляється правило з вищим пріоритетом, його позиція вже зайнята правилом відносно низьким пріоритетом. Потім правила, які вже були в TCAM, повинні бути переміщені одне за одним, щоб звільнити позицію з меншою адресою. Ці нові потоки повинні буферизуватися до кінця оновлення TCAM. Якщопослідовність правил завантаження знаходиться в одному порядку або в порядку зменшення, вхідного руху не буде або буде лише кілька вхідних рухів.

Також вивчимо вплив оновлення TCAM на затримку базового трафіку. Початкова точка затримки базового трафіку – це момент, коли NIC 3 на сервер B починає відправляти пакети. Кінцева точка – це момент, коли NIC 4 приймає пакети. Правила вставляють у порядку зростання. На малюнку 3.13 показано порівняння затримки потоку з/без оновлення TCAM. З малюнка видно, що з оновленні TCAM затримка базового трафіку значно збільшується. Середня затримка базового трафіку з оновленням TCAM у 2.42 рази більша, ніж затримка базового трафіку без оновлення TCAM. Тим часом, збільшується завантаження локального ЦП і збільшується лічильник правила на площині даних. Робимо висновок, що під час оновлення TCAM пошук таблиці в TCAM припиняється, базовий трафік буферизується і направляється на локальний ЦП для пошуку таблиці.

Можна зробити висновок, що оновлення TCAM не тільки викликає збільшення затримки нового потоку, а й збільшує затримку базового трафіку. Імовірність втрати пакетів значно зростає, коли необхідно перемістити більше правил, а базова швидкість трафіку висока. Це потенційно знизить стабільність усієї мережі.

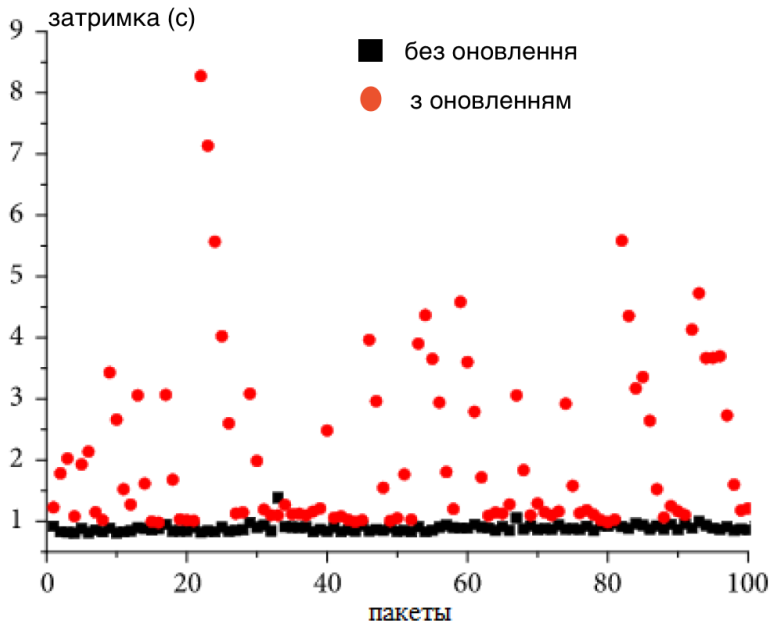


Рисунок 3.13 – Порівняння затримки трафіку з/без оновлення TCAM

ВИСНОВКИ

Переваги, які дає концепція SDN – це централізоване управління, моніторинг та збір статистики у мультивендорному середовищі, незалежність від технологій конкретного виробника, спрощення модернізації та обслуговування мережі.

При цьому на ринку послуг вже зараз спостерігається широке розмаїття мережного обладнання, що підтримує протокол OpenFlow як для організації бездротових мереж, так і провідних залежно від потреб споживача.

Результати дослідження показали, що використання програмно-конфігурованих мереж із протоколом OpenFlow є сучасним, актуальним та перспективним вирішенням питання про ефективність та продуктивність існуючих мереж. Однак, як і кожна технологія, SDN має свої недоліки:

По-перше, кожному комутатору, незважаючи на використання OpenFlow, доводиться обробляти великі заголовки пакетів у пошуках відповідності їх полів у записах своїх таблиць потоків. Зрозуміло, плюсом тут і те, що стандарт OpenFlow розглядає потоки пакетів, а чи не окремі пакети, відповідно обробляється лише заголовок першого пакета потоку.

По-друге, комутаторам необхідно зберігати великі обсяги даних у таблицях потоків, що призводить до необхідності підтримки досить великого обсягу пам'яті і змушує використовувати більш потужні і досить дорогі плати.

По-третє, реалізація всієї функціональності OpenFlow на одному пристрої може піддати мережу збоїв внаслідок навантаження контролера.

У ході роботи було розглянуто пристрій OpenFlow-комутатор та алгоритм роботи мережі SDN. Представлено модель затримки трафіку, що передається по мережі SDN. В результаті дослідження можна зробити висновок, що затримка передачі трафіку в мережі SDN впливає затримка комутатора і затримка контролера. У разі високого значення затримки

трафіку можна знизити шляхом зниження затримки комутаторів чи контролера. Затримка, що вноситься комутатором, визначається такими факторами як продуктивність комутаційної матриці, розміром буфера та наявністю черг пакетів на інтерфейсах.

Затримка, що вноситься контролером, залежить як від продуктивності апаратної платформи, так і від ефективності оптимізації програмного коду МОС контролера і додатків. Для зниження затримки контролера рекомендується вибирати апаратну платформу сервера з максимальною продуктивністю та оптимізувати роботу додатків, що виконують функції керування трафіком, що, у свою чергу, може призвести до зниження навантаження на систему.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Іванісенко І.М., Пушкар А.І. Організація програмно-конфігурованої мережі на базі протоколу Openflow. Проблеми інформатизації: Тези доповідей деся-тої міжнародної науково-технічної конференції. Т.1: секція 2. Черкаси: ЧДТУ; Баку: ВАЗС АР; Бельсько-Бяла: УТіГН; Харків: НТУ «ХПІ»;. Харків: ХНУРЕ; Харків: ДП «ПД ПКНДІ АП». 2022. с. 76.

2. Fei Hu. Network Innovation через OpenFlow and SDN. Principles and Design. CRC Press 1st edition. February 2014. - 520 p.

3. 2 Thomas D. Nadeau та Ken Gray. SDN Software Defined Networks. OReilly Media. September 7, 2013. - 384 p.

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

5. Логінов С. С. Про рівні управління в програмно-конфігурованій мережі (SDN) / T-Comm: Телекомунікації та транспорт. 2017. Том 11. №3. С. 50-55.

6. Wolfgang Braun та Michael Menth. Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices. Future Internet 2014, 6, 302-336 p.

7. OpenFlow Switch Specification – March 26, 2015. URL:https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf. p. 18-30.

8. СеменовЮ.А.Мережеватехнологія OpenFlow(SDN).URL: http://book.itep.ru/4/41/openflow.htm .

9. Владико А.Г, Матвієнко Н.А, Новіков М.І., Киричек Р.В. Тестування контролерів програмно-конфігурованої мережі з урахуванням модельної мережі // Інформаційні технології та телекомунікації. 2016. Том 4. №1. З. 17-28.

10. Ю. Ю. Коляденко, Є. Е. Білоусова. Програмно-конфігуровані мере-

жі з урахуванням протоколу OpenFlow та його характеристики. Scientific Journal "ScienceRise" №3/2(20)2016 р.

11. В. А. Лихачов. Програмно-конфігуровані мережі на основі протоколу OpenFlow. Вісник ВДУ, серія: Системний Аналіз та інформаційні технології, 2014 №1 URL:https://www.virtualbox.org.

12. M. Casado, MJ Freedman, J. Pettit “Rethinking enterprise network control” IEEE/ACM Transactions on Networking, vol. 17, no. 4, pp. 1270-1283, 2009.

13. Paul Goransson. Software Defined Networks: A Comprehensive Approach / Paul Goransson, Chuck Black. — USA: Morgan Kaufmann, 2014. — P. 145-166.

14. Madhusanka Liyanage. Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture / Madhusanka Liyanage, Andrei Gurtov, Mika Ylianttila. — India: Wiley, 2015. — P. 32-33, 235.

15. Dushyant Arora. Live Migration of an Entire Software-Defined Network / Dushyant Arora, Diego Perez-Botero [Електронний ресурс].

16. Eric Keller. Live Migration of an Entire Network (and its Hosts) / Eric Keller, Soudeh Ghorbani, Matt Caesar, Jennifer Rexford [Електронний ресурс].