МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ АЕРОНАВІГАЦІЇ,
ЕЛЕКТРОНІКИ ТА ТЕЛЕКОМУНІКАЦІЙ
КАФЕДРА ТЕЛЕКОМУНІКАЦІЙНИХ ТА РАДІОЕЛЕКТРОННИХ СИСТЕМ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Одарченко Р.С.
“_____”_____2021 р.

ДИПЛОМНА РОБОТА

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

Тема: «Комп’ютерні моделі програмно-конфігурованих мереж»

Виконавець: _____ Заблоцький К.В.
(підпис)

Керівник: _____ Соловйов Д.О.
(підпис)

Нормоконтролер: _____ Бахтіяров Д. І.
(підпис)

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет аеронавігації, електроніки та телекомунікацій
Кафедра телекомунікаційних та радіоелектронних систем
Спеціальність 172 «Телекомунікації та радіотехніка»
Освітньо-професійна програма «Телекомунікаційні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри

Одарченко Р.С.
“_____”_____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи

Заблоцького Костянтина Васильовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проекту): «Комп’ютерні моделі програмно-конфігурованих мереж»
затверджена наказом ректора від « 06 » квітня 2021 р. №559 / ст.
2. Термін виконання роботи: з 17.05.2021 р. по 20.06.2021 р.
3. Вихідні дані до роботи: архітектура мережі, вимоги до технології передавання даних та налаштування мережі, середовище моделювання.
4. Зміст пояснювальної записки: дослідження перспективи використання SDN, розробка імітаційних моделей для дослідження, дослідження імітаційних моделей.
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: рисунки по запуску та налаштуванню MiniEdit; рисунки імітаційних моделей мережі SDN в Mininet та MiniEdit; рисунки командного рядка Mininet; рисунки командного рядка Xterm; рисунки перехопленого трафіку в програмі WireShark.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Розробити деталізований зміст розділів диплому	17.05.2021-18.05.2021	Виконано
2	Вступ	18.05.2021-21.05.2021	Виконано
3	Дослідження перспективи використання SDN	22.05.2021-31.05.2021	Виконано
4	Дослідження імітаційних моделей	1.06.2021-10.06.2021	Виконано
5	Розробка імітаційних моделей для дослідження	11.06.2021-16.06.2021	Виконано
6	Усунення недоліків дипломної роботи	17.06.2021-20.06.2021	Виконано

7. Дата видачі завдання: “26" квітня 2021 р.

Керівник дипломної роботи _____ Соловйов Д.О.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Заблоцький К.В.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Дипломна робота «Комп'ютерні моделі програмно-конфігурованих мереж» містить 65 сторінок, 83 рисунки, 3 таблиці, 10 використаних джерел.

SDN, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ІМІТАЦІЙНІ МОДЕЛІ, ПРОГРАМНО-КОНФІГУРОВАНІ МЕРЕЖІ, МОДЕЛЮВАННЯ, ТОПОЛОГІЇ, КОМАНДИ, MININET, MINIEDIT, МЕРЕЖЕВІ ЕЛЕМЕНТИ.

Об'єкт дослідження – параметри та процеси імітаційних моделей програмно–конфігурованих мереж.

Предмет дослідження – мережеві топології у програмному середовищі Mininet.

Мета дипломної роботи – розробка імітаційних моделей SDN-мережі для дослідження взаємодії мережевих елементів.

Метод дослідження – моделювання різних топологій мереж SDN за допомогою ПЗ Mininet та MiniEdit.

Матеріали дипломної роботи рекомендується використовувати при дослідженні принципу роботи мереж SDN, що дозволить врахувати усі недоліки при подальшому створенні реальних мереж. На основі досліджень можливо попередньо протестувати мережу та ознайомитися з принципом роботи усіх її компонентів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

ACI (application centric infrastructure) – інфраструктура, орієнтована на додатки

ADC (application delivery network) - мережа доставки додатків

API (application programming interface) - прикладний програмний інтерфейс

CD (continuous delivery) – безперервна доставка

CI (continuous integration) - безперервна інтеграція

IDS (intrusion detection system) - система виявлення вторгнень

IP (internet protocol) – Інтернет протокол

IPS (intrusion prevention system) - система запобігання проникненню

NFV (network function virtualization) – віртуалізація функцій мережі

ns (discrete-event network) – моделювання дискретних подій

ONOS (open network operating system) – відкрита мережева операційна система

SDN (software-defined networking) - програмно-конфігуровані мережі

VM (virtual machines) - віртуальна машина

WOC (WAN optimization controller) - контролер оптимізації глобальної мережі

ВМ – віртуальна машина

ОС – операційна система

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

ПКМ – програмно-конфігуровані мережі

ЦОД - центр обробки даних

ВСТУП

Актуальність теми. Разом з активним розвитком хмарних технологій та обчислень змінюється Іт-інфраструктура з процесами обслуговування в різних напрямках та організаціях, що з часом призводить до оновлення та модернізації існуючого обладнання або заміни на нове. Комутатори та маршрутизатори, які налаштовуються традиційними способами мають схильність до виникнення багатьох помилок і не можуть повністю використовувати можливості мережевих інфраструктур, що існують в даний час. Також самі комп'ютерні мережі мають певні недоліки: висока вартість обладнання, неефективне використання каналів зв'язку, проблеми з передачею великої кількості інформації та складність управління. SDN – комп'ютерна мережа, що створюється, реалізується та керується виключно програмним забезпеченням. Технологія реалізується за допомогою спеціального програмного забезпечення, такого як Open-Flow. Дана мережа надає програмну підтримку зі зменшенням фізичних компонентів,при цьому забезпечуючи підвищення ефективності мережевого обладнання, зниження витрат на експлуатацію існуючих мереж та наданням можливості оперативного створення та завантаження в мережеве обладнання сервісів користувачами.

Основними недоліками традиційних комп'ютерних мереж є:

1. Спадковість мережевої інфраструктури:

Комп'ютерні мережі, як правило, являються сумішшю різних рішень протоколів та різних рішень постачальників самих платформ, що в свою чергу ускладнює інтегрування мережевої екосистеми для багатьох організацій. Конфігурації сотень пристроїв та механізмів залежать від топології мережі, версії ПЗ, виробників та моделей самих пристроїв.

2. Складність налаштування та управління:

У традиційних мережах адміністратори та менеджери IT-систем повинні налаштовувати вручну усі апаратні пристрої, такі як комутатори, маршрутизатори, концентратори та брандмауер. Також вони мають нести повну відповідальність за

оновлення усіх пристроїв і, ґрунтуючись на моніторингу активних сесій та додатків налаштовувати параметри QoS (Quality of Service) та смуги пропускання при змінах запитів користувачів та трафіку.

3. Дефіцит висококваліфікованих спеціалістів:

Складність налаштування та управління разом з усіма недоліками спадковості інфраструктури призводять до виникнення нестачі адміністраторів, які могли б розумітися у цьому всьому. Оскільки відрізняється велика моделей пристроїв та використання багатьох протоколів, збільшується час на навчання майбутніх спеціалістів та зменшується ефективність використання персоналу.

4. Висока вартість проектування, створення та обслуговування мереж:

Складна інфраструктура мережі потребує велику кількість різноманітного обладнання (комутатори, маршрутизатори та мережеве обладнання для трансформації,перевірки та фільтрації трафіку). Усе це обладнання коштує дорого і відповідно збільшується ціна на встановлення та обслуговування усіх пристроїв. Також на вартість вливає необхідність залучення великої кількості кваліфікованих спеціалістів для обслуговування, налаштування, контролю та управління мережами та пристроями.

Мета і завдання дослідження. Розробити імітаційні моделі SDN мереж для вивчення взаємодії між мережевими об’єктами та дослідити усі нюанси різних типів мереж.

Для досягнення поставленої мети вирішуються такі наукові завдання:

1) Дослідження недоліків сучасних мереж, переваг та перспектив використання мереж SDN.
2) Створення імітаційних моделей SDN-мереж за допомогою програмного середовища Mininet.
3) Дослідження роботи імітаційних моделей за допомогою Mininet та MiniEdit.

Об’єкт дослідження – основні параметри взаємодії елементів імітаційних мереж SDN.

Предмет дослідження – компоненти і їх взаємодія у різних мережевих топологіях, що створюються за допомогою ПЗ Mininet.

Методи досліджень. Для забезпечення проведення досліджень та моделювання хостів, контролерів та комутаторів SDN з використанням OpenFlow використовувалися такі програми, як: Oracle VM VirtualBox, Mininet (що включає в себе MiniEdit) , а також додаткові програми (Xterm та WireShark).

Практичне значення отриманих результатів.

В результаті дослідження різних мережевих топологій було створено декілька працездатних мереж SDN. Для кожної з топологій було проведено перевірку з'єднання усіх компонентів, що включає в себе перевірку вузлів, їх пінгування, підключення портів та визначення пропускної здатності. Результати роботи можна застосовувати для створення прототипів реальних мереж, що дозволить спочатку провести усі дослідження над мережею і усунути недоліки, а потім переносити усе на реальне мережеве обладнання.

Апробація отриманих результатів. Основні положення роботи доповідалися та обговорювалися на таких конференціях:

- Науково-практична конференція «Проблеми аеронавігації, електроніки та телекомунікацій», м. Київ, 2020 р.

- Науково-практична конференція «Проблеми експлуатації та захисту інформаційно-комунікаційних систем», м. Київ, 2021 р.

РОЗДІЛ 1
ДОСЛІДЖЕННЯ ПЕРСПЕКТИВИ ВИКОРИСТАННЯ ТЕХНОЛОГІЇ SDN

1.1. Обмеження традиційних мереж

Традиційна взаємодія пристроїв представляє собою вбудовану площину управління, що керує усіма процесами: маршрутизація, комутація, забезпечення інженерної діяльності, а також площину даних, яка на основ трафіку пересилає кадри/пакети . [1]

Візуальне зображення традиційної мережі представлено на Рис. 1.1.:

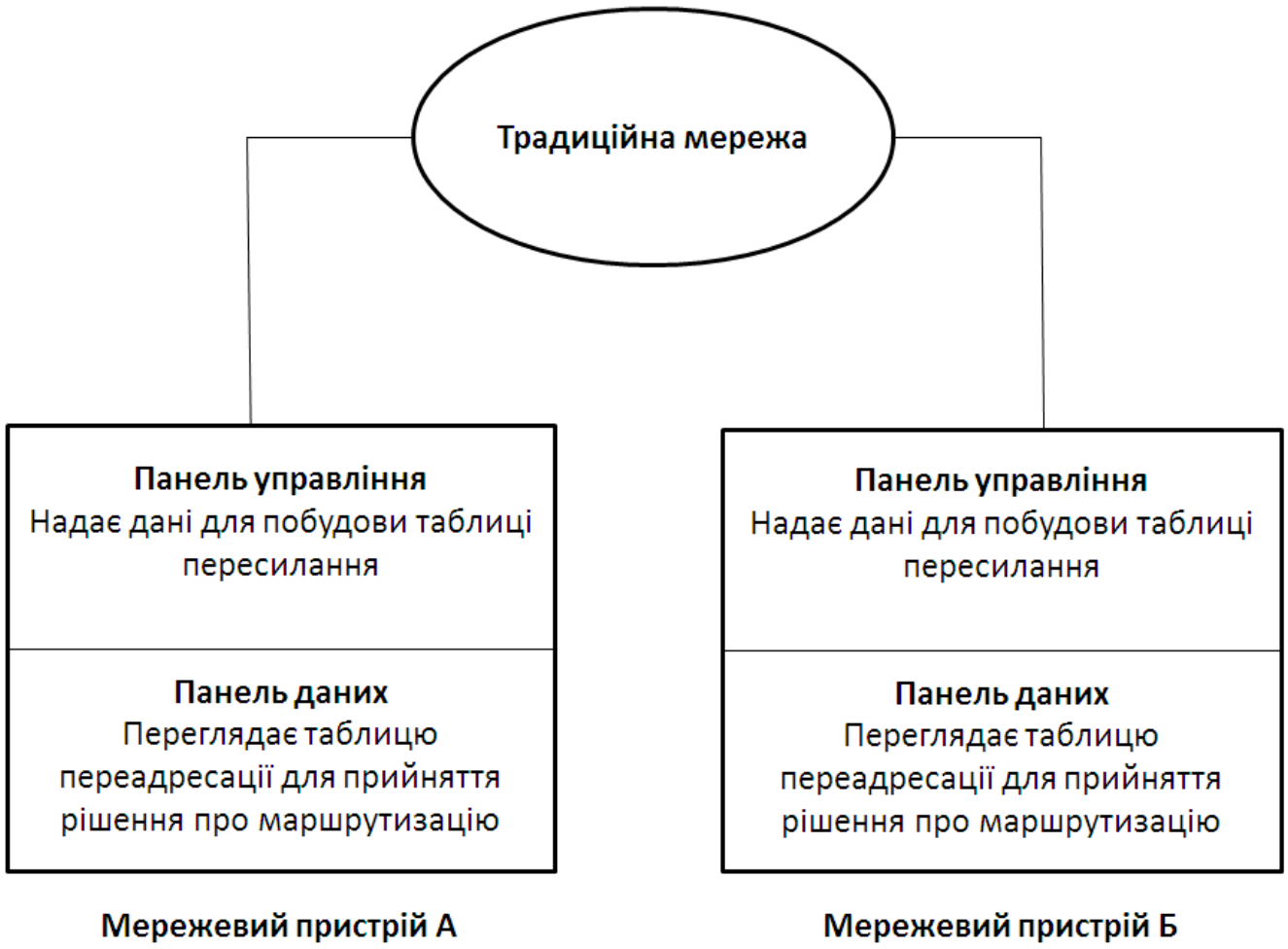

Рис. 1.1. Візуальне зображення традиційної мережі

Тобто, площина управління – це як менеджер трафіку, що надає інформацію, яка буде використовуватись для створення таблиці пересилання. А площина даних є носієм трафіку, що в свою чергу спочатку звертається до створеної таблиці переадресації, а потім приймає рішення, щоб направити кадри/пакети, які надійшли на пристрій. В сумі ці дві площини представлені на мережевому пристрої.

Щоб зрозуміти усю складність роботи протоколів маршрутизації, необхідно перейти безпосередньо до самого складу мережі, яка для забезпечення окремих функцій складається з багатьох окремих елементів: комутатори, маршрутизатори, NAT (Network Address Translation), брандмауери, та балансування навантаження. Тобто, у нас є цільові пристрої, такі як: ПК та сервер разом з проміжними пристроями, які підключені до самої кабельної системи. Увесь процес починається з того, що кадр/пакет надходить на один порт для перевірки маршрутизатором, після чого він відправляє його через порт на наступний етап. Усі маршрутизатори періодично проводять перевірку (опитування) сусідніх маршрутизаторів, з якими він з'єднаний для збору інформації, щоб використати її для створення структури мережі. Постійний обмін топологічними даними маршрутизаторів між собою не дає змогу уникнути повторного розрахунку маршруту. Сусідні маршрутизатори обмінюються результатами перекриття між собою після самостійного розрахунку маршруту. Таке дублювання зменшує ефективність, оскільки необхідна певна потужність для реалізації усіх процесів. Кожен окремий маршрутизатор, що виконує аналогічні обчислення, що й інші маршрутизатори для отримання результатів з мінімальними змінами, є дорогим пристроєм. В кінцевому результаті для впровадження складних алгоритмів маршрутизації для великих мереж вимагається велика обчислювальна потужність на усіх елементах мережі. Разом зі зростанням мережі підприємства з'являється необхідність оновлення ПЗ кожного маршрутизатора для обробки нових розрахунків, при цьому кількість портів та їх тип разом з процесором не змінюється, що часто викликає брак потужності для виконання нових алгоритмів.

Увесь описаний процес вимагає модернізації або повного оновлення для забезпечення необхідного функціонування майбутніх мереж. Для забезпечення

необхідної швидкості зв’язку та з’єднання на довільних відстанях у сучасних мережах існує велика кількість стандартних наборів протоколів, які для вирішення певної проблеми визначаються окремо. Для сучасних динамічних середовищ та віртуалізації серверів не підходить статичний характер мережі, програми мають розподілятися між віртуальними машинами для обміну трафіком між собою.

За допомогою технології SDN ми можемо здійснити перехід до програмованих об’єктів і відмовитись від мережі як платформ та відокремлених елементів. За допомогою додатків, а саме завдяки розподіленню протоколів маршрутизації, оптимізується мережа та її транспортні потоки, що дозволить створювати окремі домени для різних користувачів, забезпечити безперервну мобільність пристроїв та використовувати з’єднання на максимальну потужність. Візуальне зображення мережі SDN представлено на Рис. 1.2.:

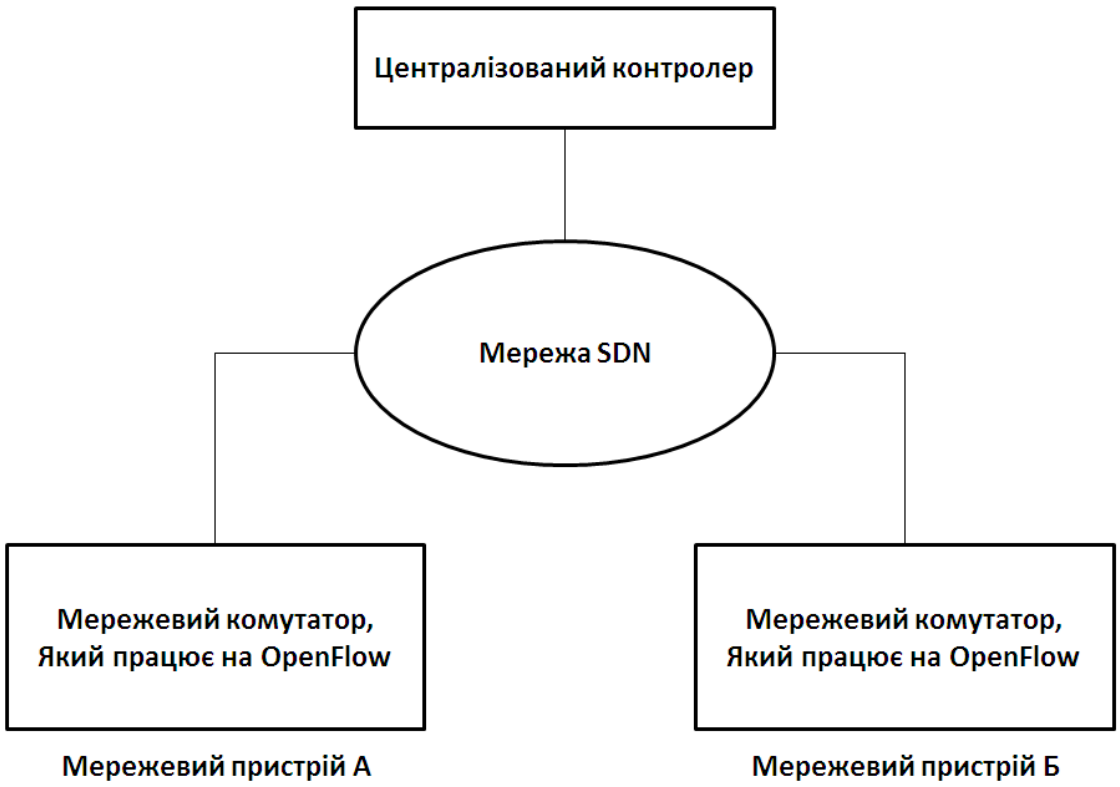

Рис. 1.2. Візуальне зображення мережі SDN

Тому в архітектурі SDN площини управління переносяться з окремих мережевих елементів (пристроїв) і розміщується на окремому централізованому

сервері. Сервером, на якому розташоване ПЗ SDN, може бути контролер SDN. Взаємодія з фізичною або віртуальною площиною даних комутатора відбувається через протокол OpenFlow, який містить інструкції щодо пересилання даних для площини даних. Для забезпечення цього процесу необхідно, щоб мережеві пристрої запускали даний протокол.

1.2. Хмарні технології та SDN

Стрімке зростання хмарних обчислень провокує зміну розробки та впровадження додатків для цих технологій. Швидка зміна та розвиток хмарних технологій все більше та швидше призводить до перетворення більшої кількості фактичного обладнання у ВМ, оскільки віртуалізація стала наступним кроком розвитку мережевої інфраструктури.

Систему хмарних технологій можна розділити на: передній (front end) та задній (back end) кінці (Рис. 1.3.).

До складу передньої частини входять комп'ютери та додатки, необхідні клієнту для доступу до самих хмарних обчислень. До складу задньої частини входять сервери, системи зберігання даних та комп'ютери, з яких складається хмарне обчислення. Для адміністрування клієнтів та системи, контролю потоків трафіку та забезпечення безперебійної роботи існує центральний сервер. Він використовує спеціальне, проміжне ПЗ, що дозволяє взаємодіяти комп'ютерам мережі між собою.

У технології SDN (зображено на Рис. 1.4.) ті самі дії виконує протокол зв'язку OpenFlow. Контролер, без необхідності використання синтаксису командного рядка різних виробників, використовує API і реалізує команди мережі на багатьох пристроях.

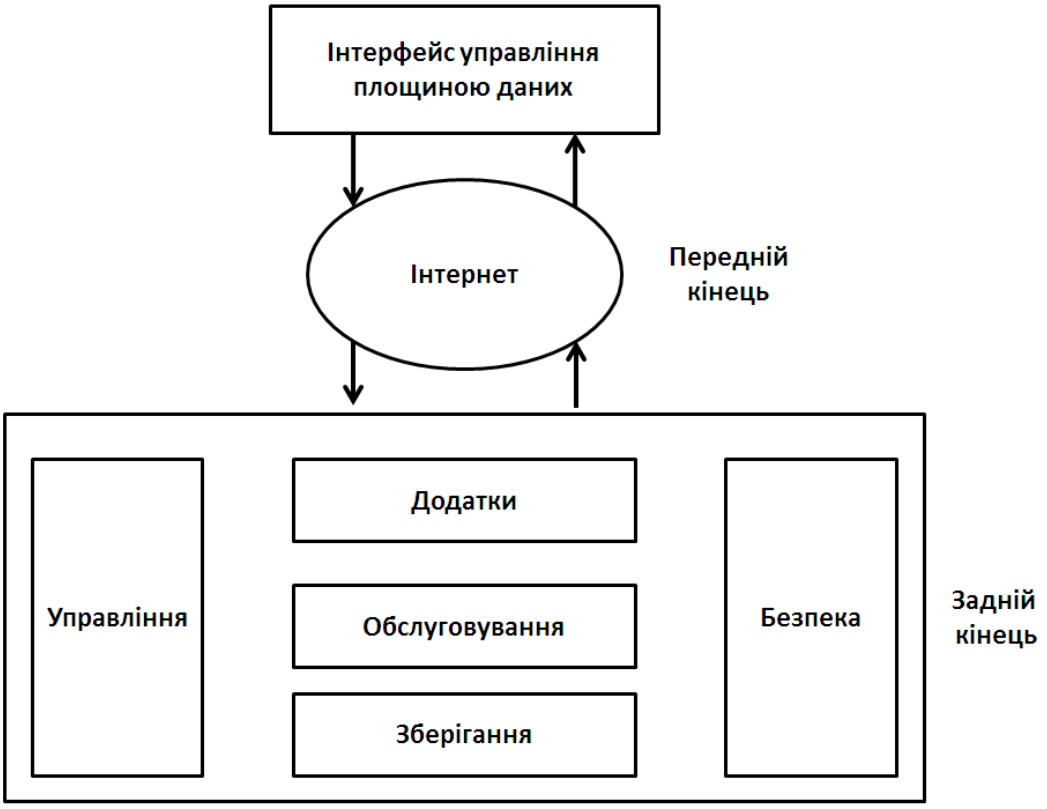

Рис. 1.3. Система хмарних технологій

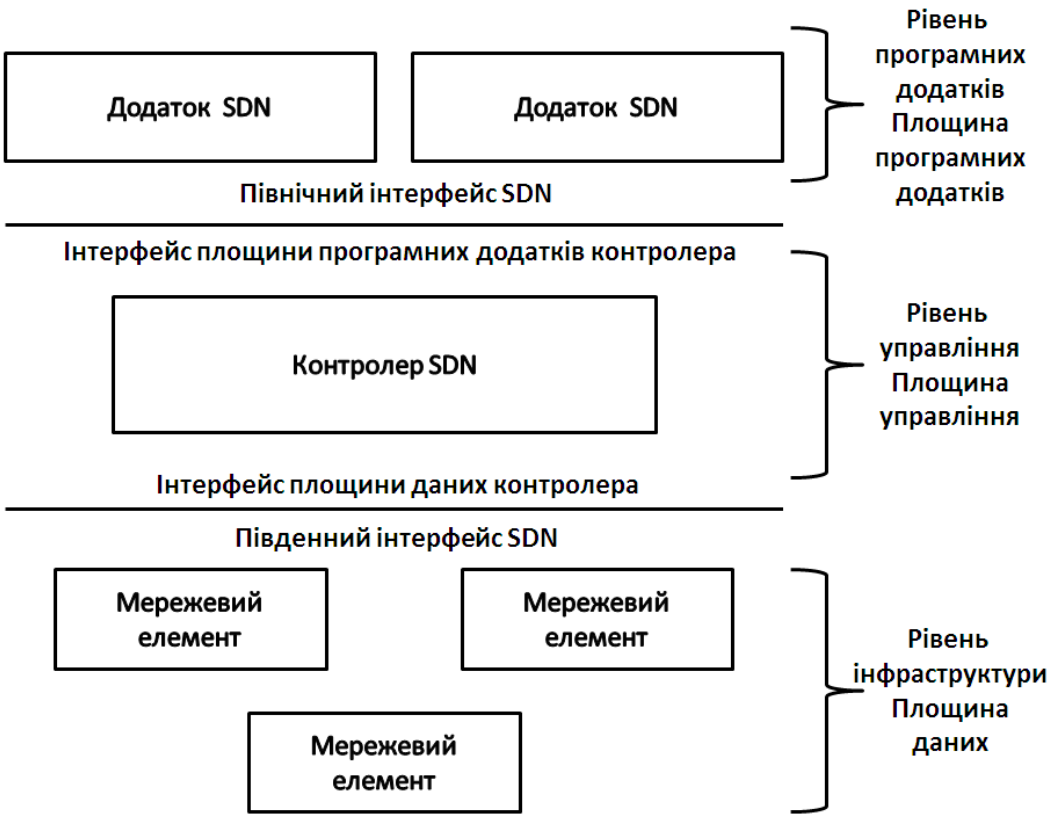

Рис. 1.4. Інтерфейси технології SDN

У інтерфейсі API розрізняють північний та південний інтерфейси. Північний – служить для перетворення бізнес-логіки у мережевій структурі. З його допомогою бізнес-додатки передають інформацію контролеру SDN, який контролює процес подальшого програмування у мережі. Даний інтерфейс дозволяє абстрагувати мережу і на основі вимог програм надає адміністратору вибір серед мережевих ресурсів. Південний інтерфейс за допомогою протоколів взаємодії, таких як OpenFlow, забезпечує зв'язок рівнів управління та інфраструктури між собою.

1.3. Дослідження технології SDN

В основному у ПЗ SDN існує три рівні: управління, дані та мережеві додатки (Рис. 1.5.). Усі рівні доступні через відкриті API і підтримують надійність та безпеку ПЗ.[2]

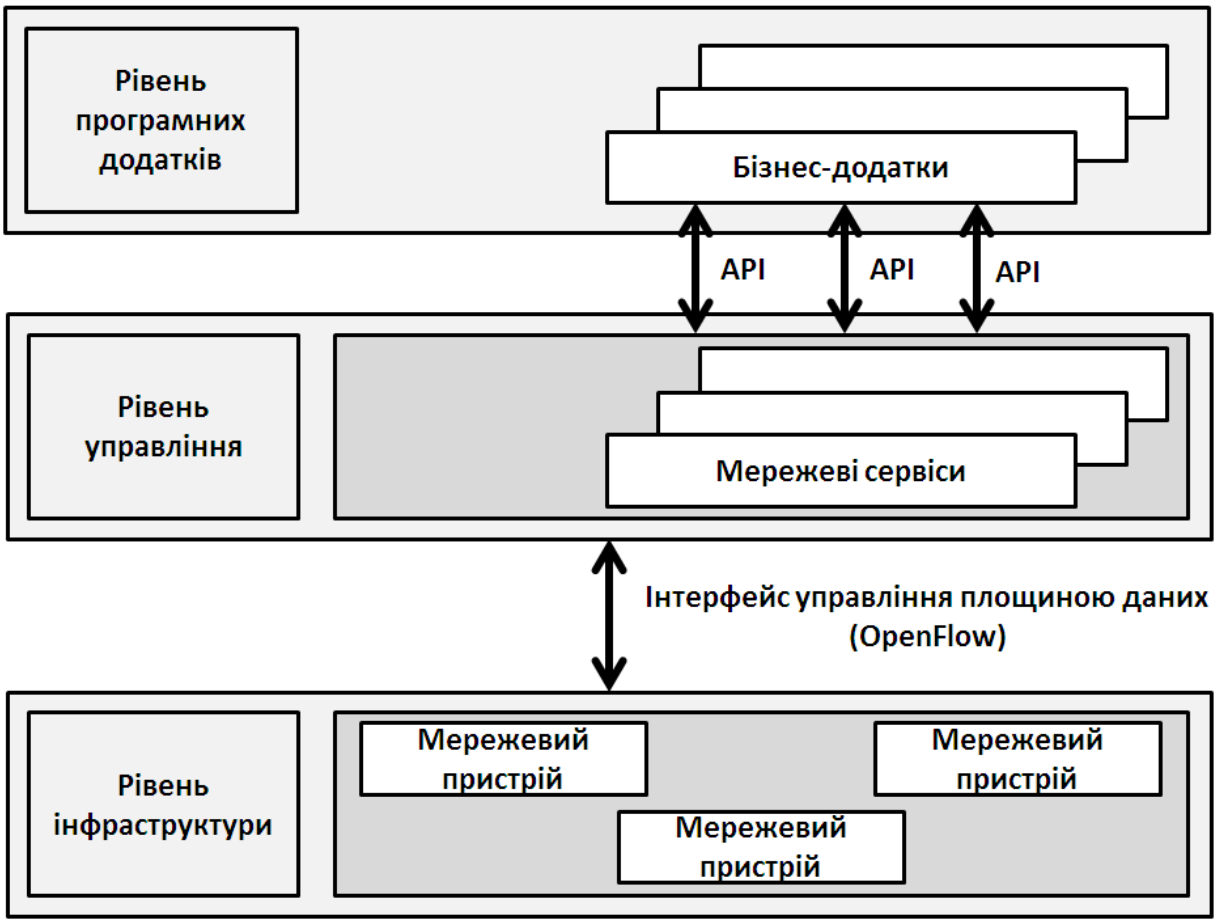

Рис. 1.5. Рівні мережі SDN

1. Рівень програмних додатків.

Забезпечує гнучке та ефективне управління мережею за допомогою різних бізнес-додатків. Функцією даного рівня є контроль за виконанням необхідних завдань відповідно до потреб додатків. Тобто, контролер повідомляє та передає мережі команди за допомогою API.

2. Рівень управління.

Відповідає за маршрутизацію і забезпечує доступ сигнального трафіку. Він включає себе мережеву ОС для керування самою мережею та мережевими пристроями за рахунок API та набору служб через відкритий інтерфейс. Функцією даного рівня є централізована конфігурація та управління, створення пакетів та призначення їх для маршрутизаторів. Тобто, це логічний зміст у мережі, що визначений ПЗ, який передає мережевим компонентам SDN отримані команди та вказівки від рівня даних. Контролер вилучає та передає корисну інформацію та дані з апаратних пристроїв, разом з іншими діями у мережі, для передачі їх до додатків SDN. ПЗ для контролера SDN також розміщується тут і забезпечує заповнення площини даних через централізоване управління пристроями шляхом усунення рівня управління з окремих комутаторів та маршрутизаторів. Також рівень управління за допомогою IPv4, IPV6 та ARP (addres resolution protocol) здійснює керування такими протоколами маршрутизації як: RIP (routing information protocol), OSPF (open shortest path first), EIGRP (enhanced interior gateway routing protocol), та BGP (border gateway protocol).

3. Рівень інфраструктури (даних).

Даний рівень є аналогом фізичного рівня моделі OSI (Open Systems Interconnection) і включає в себе усі мережеві елементи, які забезпечують передачу даних – це фізичні та віртуальні пристрої разом з каналами передачі. Функцією рівня інфраструктури є фізичне пересилання пакетів за допомогою протоколів передачі, які використовує рівень управління, від його інтерфейсу входу до виходу.

До складу рівнів також входять такі основні концепції як: бізнес додатки, мережеві послуги та служби безпеки, перемикач SDN та гібридний комутатор.

- Бізнес додатки – додатки, які включають в себе можливості управління ланцюгом постачання інформації та відносинами з клієнтами, проведення відеоконференцій і при цьому використовуються обмеженими користувачами.
- Мережеві послуги та служби безпеки – функціонал для забезпечення ефективності та безпеки бізнес-додатків. До складу функціоналу входять брандмауери, WOC (WAN optimization controllers), ADC (application delivery network), IDS (intrusion detection system) /IPS (intrusion prevention system) разом з функціями безпеки та захистом від DDoS (distributed denial of servise).
- Перемикач SDN – забезпечує виконання усіх функцій керування звичайного комутатора у централізованому контролері. Протоколи маршрутизації, які застосовуються у створенні інформаційних масивів даних маршрутизації, повністю обмежені рівнем даних.
- Гібридний комутатор забезпечує одночасну взаємодію технології SDN разом з традиційними протоколами для створення можливості виявлення та керування частиною потоків трафіку менеджером мережі. У традиційних мережах решта трафіку довільно направляється протоколами між пристроями мережі.

Створення технології SDN було викликане декількома недоліками традиційної мережі:

- Несумісність обладнання багатьох виробників та унеможливлення проведення досліджень та зміни архітектури через закритість системи.
- Зростання кількості користувачів та потоку трафіку призводить до погіршення мережевої архітектури через проблеми зі швидким оновленням ПЗ та компонентів.
- Збільшення трафіку викликає збільшення кількості елементів мережі, що діють між собою через різні протоколи передачі, що призводить до ускладнення мережі та ускладнює оновлення.

Проаналізувавши технологію SDN, можна охарактеризувати архітектуру наступним чином: [3]

- Централізація логічної взаємодії

Перенесення управління в одне місце, у централізований сервер, надає можливість глобальної мережевої оркестрації. Використання інтерфейсу OpenFlow надає можливість взаємодії елементів мережі без розповсюдження інформації про загальний стан мережі між вузлами та приймати рішення за допомогою глобального уявлення про мережу.

- Автоматизація та програмованість

Централізація логічної взаємодії разом з програмним інтерфейсом забезпечують оновлення, автоматизацію та формування конфігурацій мережевої структури. Перехід до програмного управління від традиційної мережі досягають революції у спілкування програм з мережею через застосування відкритих API.

- Ізоляція пристроїв та додатків

Результатом централізації логічної взаємодії також є те, що бізнес-додатки та використання ВМ абстрагуються від застарілих технологій. Взаємодія між додатками, їх конфігурація, оновлення та взаємодія з іншими додатками різних виробників забезпечується за рахунок віддалення від рівня управління мережі.

1.4. Дослідження ринку SDN в Україні

Стрімкий розвиток хмарних технологій разом зі зростанням трафіку постійно спостерігається у всьому світі, але це не викликає різкого попиту на цифровізацію у сфері телекомунікацій України. Хоча технологія SDN постійно розвивається і активно використовується у різних країнах, вона все ж має недостатню кількість продуктів і рішень для оновлення нашої інфраструктури. Необізнаність замовників через відсутність постійних інновацій в даній структурі призводить до того, що вони не проявляють інтерес до цих технологій. Активізація процесу обговорення по удосконаленню та цифровізації технологій у наукових та підприємницьких спільнотах дозволяє прискорити процес удосконалення мережевої інфраструктури України. Перш за все, необхідно створити умови розвитку промисловості, середовищ проектування та інженерних програмних платформ. Самими перспективними напрямками для розвитку SDN є наукова та комерційна діяльність.

Саме для науки через відкритість структури SDN є можливість дослідження з розрахунком на оновлення мережевого обладнання у телекомунікаційних та мобільних систем. Для комерційної діяльності впровадження нових систем дозволить провести оптимізацію використання ресурсів завдяки зменшення витрат на обслуговування та зменшення кількості обладнання. Оновлення системи дозволить удосконалити управління трафіком та збільшити кількість користувачів.

Завдяки тому, що технологія SDN не стоїть на місці і має певні перспективи удосконалення та розвитку у майбутньому, вона вже має певні рішення і в Україні. У 2015 в оператора зв'язку «Укртелеком» почалося оновлення мережі, в процесі якого з'явилася потреба зміни декількох дата-центрів. Заміна старого обладнання на мережеву архітектуру SDN відбулася за допомогою архітектури Cisco ACI (application centric infrastructure), яке надало усі переваги SDN мереж. Перед впровадження нової системи потрібно було провести повне проектування та забезпечити підтримку усіх специфічних налаштувань та політики безпеки. При виборі обладнання Укртелеком обрав брандмауери Cisco Firepower (зображено на Рис. 1.6.-1.8.), на які було перенесено усі специфічні налаштування старої інфраструктури.[4]

Рис. 1.6. Cisco Firepower 2100

Рис. 1.7. Cisco Firepower 4100

Рис. 1.8. Cisco Firepower 9300

Також під час проведення міграції необхідно було вирішити узгодження роботи та взаємного впливу сервісів на усіх етапах модернізації, після чого перевірити працездатність мережі.

1.5. Перспективи використання SDN

Підхід програмно-конфігурованих мереж за рахунок програмованості мережі дозволяє адаптуватися до потреб користувачів і має усі умови для подальшого розвитку. Новий погляд на реалізацію мережевої інфраструктури надає можливість постійного удосконалення, автоматизації, спрощення та зменшення обмежень мереж. Усі переваги SDN використовуються в майбутньому розвитку даної технології:

- Гнучкість управління надає помітне спрощення мережі, можливість покращення керування, автоматизації управління та адміністрування і створення безлічі нових додатків. Створення і введення додатків та послуг більше не вимагає переналаштування усієї мережі, що дозволить економити час.
- Заміна безлічі розподілених протоколів і вузлів мережі полегшує логіку взаємодії пристроїв мережі, оскільки тепер виконується обробка меншої кількості запитів, інструкцій та стандартів.

- Спрощення структури та логіки мережі забезпечує можливість незалежного розгортання та масштабування рівнів управління та передачі даних.
- Зменшення кількості компонентів мережі за рахунок централізованих контролерів з підтримкою високої пропускної здатності здешевлюють мережі без втрати ефективності.

Перспективи розвитку технології SDN (Рис. 1.9.) мають досить широкий потенціал, розпочавши з OpenFlow, технологія не стояла на місці і з роками продовжувала свій розвиток.[5]

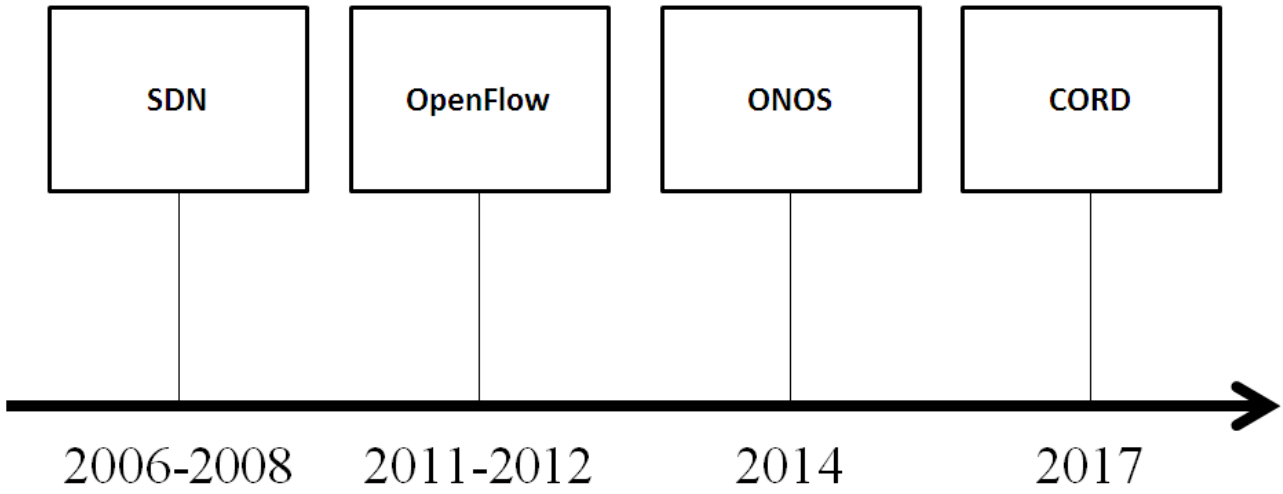

Рис. 1.9. Перспективи розвитку технології SDN

OpenFlow - це перший протокол SDN для відкритих мереж, який в даний час залишається найпопулярнішим рішенням даної технології. Даний протокол описує взаємодію контролерів з мережевими пристроями між рівнями контролю та інфраструктури. Управління здійснюється з централізованого контролера і забезпечує формування прямих мережевих з'єднань з необхідними параметрами з мінімізацією затримки передачі трафіку. Рішення про транспортування пакетів для визначених портів комутатора здійснюється на підставі заповнення таблиць потоків комутаторів.[6]

Наступним етапом після OpenFlow стало створення Операційної системи з відкритою мережею (ONOS). Це наступне покоління провідного контролера з відкритим кодом, яке задовольняє потреби спрощення програмного інтерфейсу та

забезпечення гнучкості системи. Він також підтримує налаштування та управління системою в реальному часі без необхідності запуску протоколів маршрутизації та комутації у самій системі. Дана платформа включає в себе:

- Розширюваний розподілений модульний контролер в якості набору програм платформи.
- Задоволення потреб транспортних середовищ шляхом розширення та покращення масштабованості і стійкості архітектури.
- Спрощення введення в експлуатацію нового ПЗ та обладнання за рахунок удосконалення процесу конфігурації, розгортання та управління.

Розвиток SDN не зупинився на ONOS і продовжив свій логічний рух і було створено проект CORD (central office re-architected as a datacenter) . У ньому задіяні усі попередні технології SDN, OpenFlow та ONOS разом з технологіями NFV та Cloud. Це дозволяє створити гнучку платформу та ЦОД (центр обробки даних), що дозволяє операторам забезпечувати користувачів інноваційними послугами за рахунок удосконаленого обладнання з використанням хмарних технологій.

CORD включає в себе такі програмні підсистеми, як: платформу, профіль, інструментарій CI (continuous integration)/CD (continuous delivery) та робочий процес:

- Загальний базовий рівень платформи містить ONOS та Kubernetes для розміщення контролера SDN та системи управління контейнеризованими додатками.
- У профілі забезпечується розгортання та запуск мікросервісів разом з програмами керування SDN на певних фізичних мережевих пристроях.
- Інструментарій Continuous integrator та Continuous delivery використовується для повної взаємодії з платформою, профілем та робочим процесом. Сюди входить: збирання, розгортання, оновлення та експлуатація різних комбінацій програмних підсистем.
- Функціонування мережевих приладів в мережах різних операторів забезпечується за рахунок логіки інтеграції для розгортання у робочому процесі.

ВИСНОВКИ ДО РОЗДІЛУ 1

У першому розділі було визначено відмінності традиційної мережі від мережі SDN, а саме: у традиційній мережі кожен пристрій містить в собі площину управління і площину даних, що ускладнює роботу мережі. У мережах SDN площина управління та площина даних розділяються і усіма процесами керує централізований сервер. Даний сервер, контролер, містить в собі більшу частину функціональності мережевих елементів і за допомогою протоколу OpenFlow забезпечує зв'язок з площиною даних і використовує програмні додатки у процесі роботи мережі.

Також в даному розділі було порівняно інтерфейси хмарних технологій та SDN, тобто розділення на передній/задній кінці і північний/південний інтерфейси, що в свою чергу означає, що технологія SDN має перспективи розвитку і не відстає від технічного прогресу.

Під час розгляду архітектури програмно-конфігурованих мереж було визначено, що вони складаються з трьох рівнів: управління, дані та мережеві додатки. Ці рівні також містять в собі бізнес додатки, мережеві послуги та служби безпеки, перемикач SDN та гібридний комутатор. Проаналізувавши технологію SDN, можна сказати, що мережі SDN забезпечують централізацію логічної взаємодії, автоматизацію та програмованість, а також ізоляцію пристроїв та додатків. Усе це забезпечує оптимізацію роботи мережі, оскільки є можливість постійного оновлення та конфігурації мережевих елементів.

Дослідження ринку SDN в Україні показало, що компанії тільки починають оновлювати своє мережеве обладнання. Враховуючи технологічність та відношення ціна/якість програмно-конфігурованих мереж, в даний час впровадження даної технології у нашій країні є доволі актуальною темою. Протягом наступних років кількість нових підприємств або тих, що замінюють застаріле обладнання будуть надавати перевагу SDN мережам, що також доводить актуальність теми дипломної роботи.

Також було досліджено перспективи розвитку ПКМ, і враховуючи усі переваги даної технології, було визначено, що вони мають досить широкий потенціал і постійне удосконалення і оновлення дозволяє покращувати роботу попередніх рішень та впроваджувати нові. Створення нових операційних систем та платформ на базі технології SDN дозволяє усувати усі недоліки попередніх рішень.

РОЗДІЛ 2
РОЗРОБКА ІМІТАЦІЙНИХ МОДЕЛЕЙ ДЛЯ ДОСЛІДЖЕННЯ

2.1. Аналіз програмного забезпечення для моделювання мереж

Для розгортання мережі SDN з обмеженнями ресурсів ПЗ на одному комп'ютері необхідно використовувати ВМ та емулятори з відкритим програмним кодом. Для забезпечення проведення досліджень та моделювання хостів, контролерів та комутаторів SDN з використанням OpenFlow нам необхідне таке ПЗ, як: Oracle VM VirtualBox для запуску ВМ, та Mininet (що включає в себе MiniEdit) , а також додаткові програми (Xterm та WireShark).

VirtualBox – це потужне, продуктивне, легке та зрозуміле у використанні ПЗ, що дає можливість застосування, підтримки та віртуалізації гостьових та хостових ОС. До недоліків даної програми можна віднести: нестабільність у використанні багатьох хостових платформ і нестача підтримки сумісності віртуальних дисків інших ВМ та те, що вона поступається великим комерційним платформам. Перераховані недоліки не впливають на роботу з мережами SDN, а підтримка багатьох ОС, збільшення функціоналу програми та темпи розвитку з часом дозволять використовувати платформу і для великих виробничих середовищ. За допомогою даної програми ми можемо встановити та працювати з Mininet, яка встановлюється з Ubuntu (без додаткового встановлення).[7]

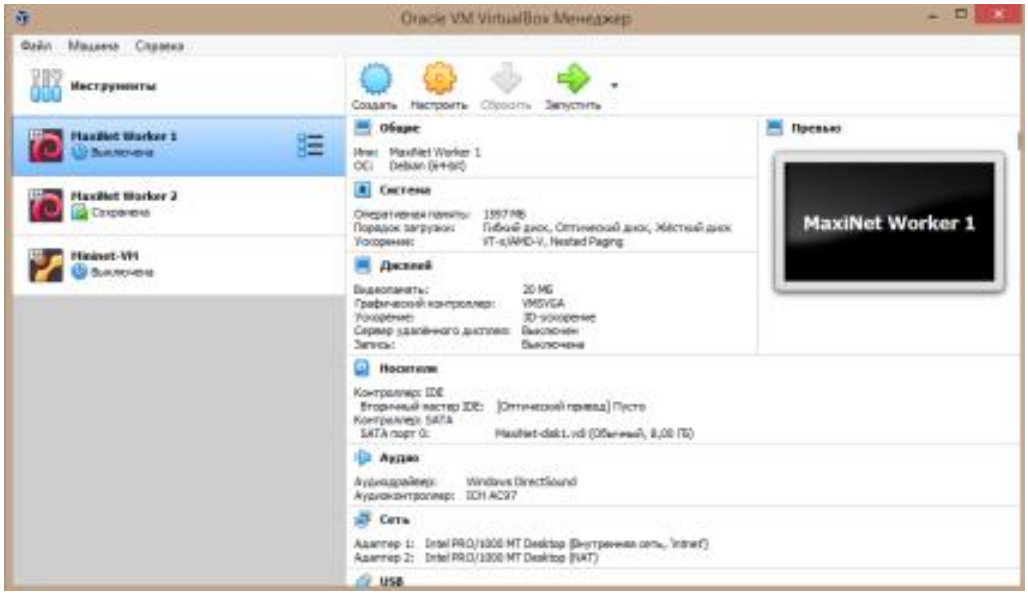

Рис. 2.1. Вікно програми VirtualBox

MiniEdit – частина емулятора Mininet, тобто простий редактор з графічним інтерфейсом для візуалізації та демонстрації розширення Mininet. Простий інтерфейс користувача дозволяє створювати різні топології без застосування командного рядка (усі додані елементи і їх взаємодія автоматично прописуються і можуть використовуватися в подальших маніпуляціях). Усе, що потрібно для використання даного інструменту знаходиться у лівій частині інтерфейсу та не вимагає спеціальних навичок та знання команд програмування.

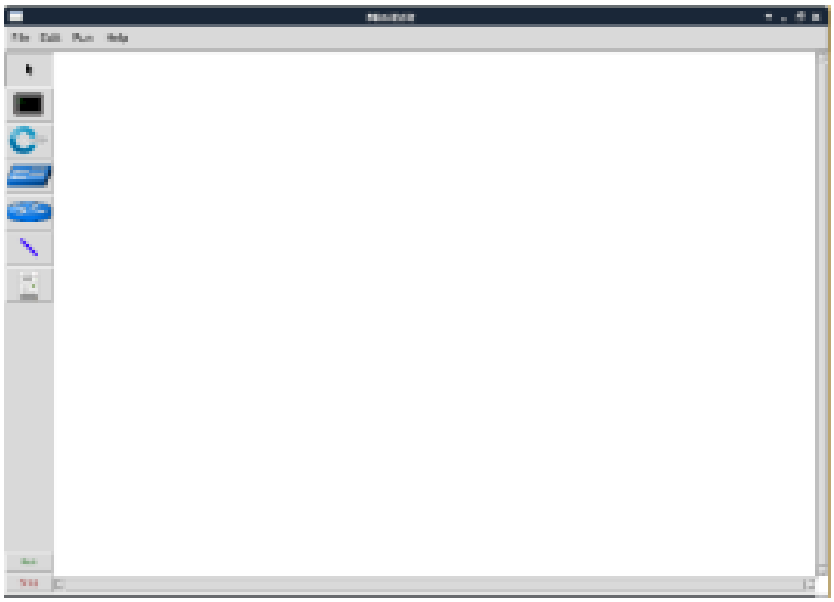

Рис. 2.2. Вікно програми MiniEdit

Xterm – емулятор терміналу, який підключається до графічних додатків. Оскільки немає необхідності постійного запуску терміналів на кожному комутаторі, вікно Xterm відкривається окремо на кожному з хостів у графічному інтерфейсі MiniEdit. Запуск відбувається після на жаття правої клавіші мишки на хості і вибору «Terminal» у відкритому меню. Дана функція корисна для перегляду вихідних даних інтерактивних команд, або відміни їх дії, а також для запису додаткових команд хоста.

Рис. 2.3. Вікно програми Xterm

WireShark – програма моніторингу, що здійснює перехоплення, аналіз та відображення мережевого трафіку у заданій мережі. Проведення таких маніпуляцій дозволяє досліджувати та спостерігати за усіма мережевими протоколами для попередження виникнення проблем та втрати контролю над мережею. У створенні та дослідженні імітаційних моделей SDN це дозволить моніторити та аналізувати процеси між усіма мережевими елементами.

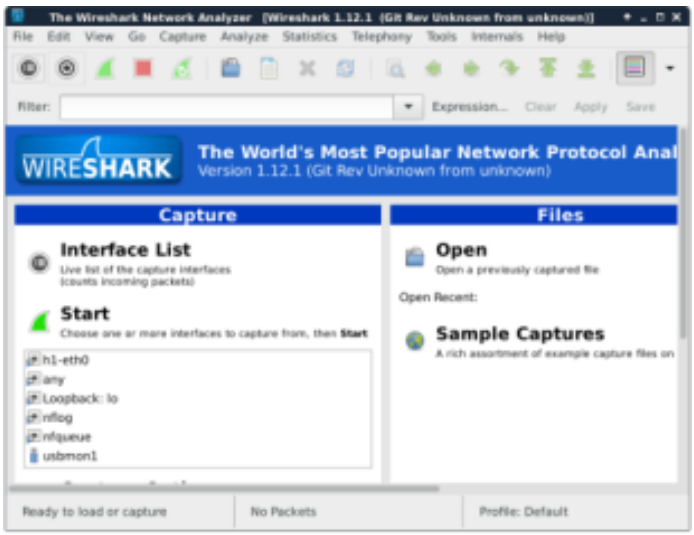

Рис. 2.4. Вікно програми WireShark

Mininet – ПЗ для забезпечення емуляції роботи мереж SDN та OpenFlow. Дана програма дозволяє досліджувати, налаштовувати та проводити різні маніпуляції з віртуальною мережею за допомогою простих інструментів командного рядка та API. Імітація пристроїв OpenFlow та контролерів SDN з легкістю дозволяє розміщувати та тестувати топології, що складаються з тисяч вузлів.

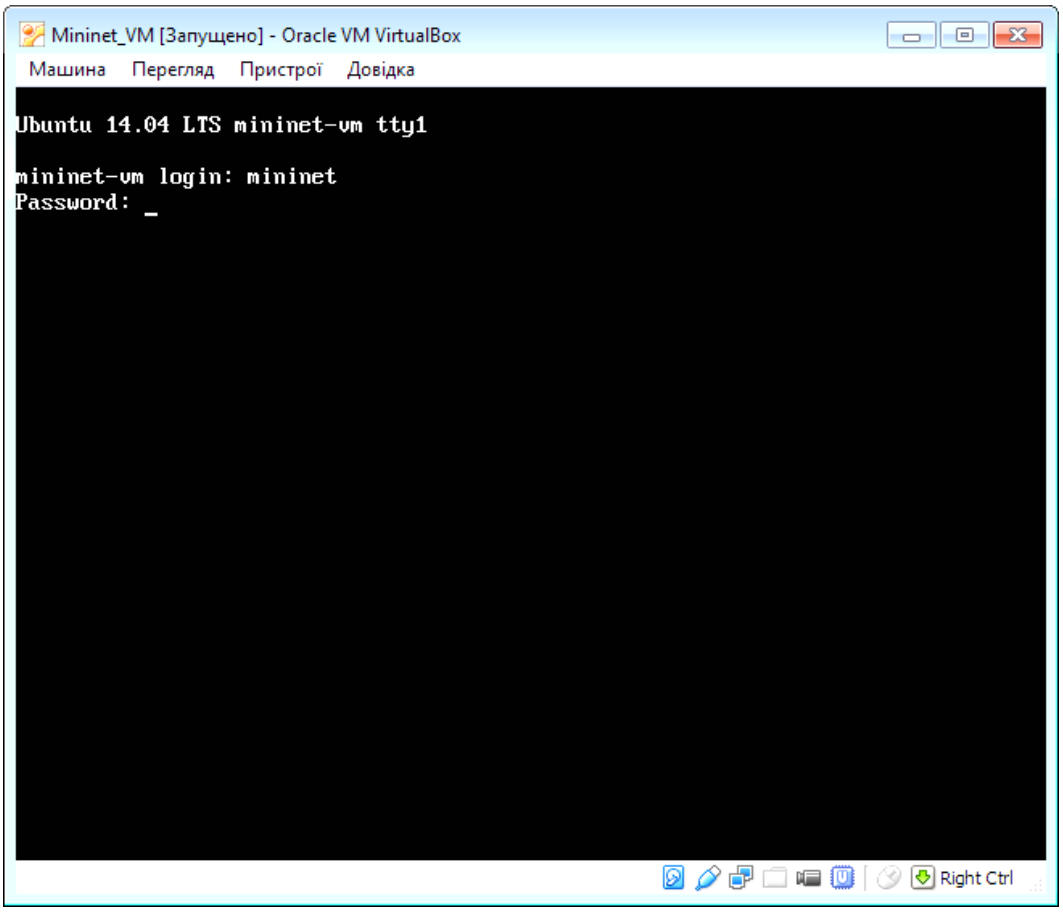

Рис. 2.5. Вікно програми Mininet

Оскільки Mininet постійно розвивається, то існує також ще декілька емуляторів, що включають в себе усі функцій Mininet і також можуть використовуватися для створення мереж SDN. До списку цих емуляторів входять: ns-3, OpenNet, Containernet, Tinynet та Maxinet.

1) Ns-3 – це симулятор DES (discrete-event simulation), моделювання дискретних подій, що використовується для тестування SDN мереж. Даний емулятор має декілька недоліків: відсутність візуалізації та деяких централізованих контролерів.

2) OpenNet включає в себе Mininet та ns-3, об’єднуючи функції обох емуляторів. Тобто, ns-3 надає можливість створення моделей, відсутніх у Mininet, а Mininet надає відсутні контролери, командний рядок та віртуалізацію.
3) Containernet – емулятор, що містить програмну частину Mininet, але працює з контейнеризованими додатками. При створенні імітаційних моделей у якості хостів виступають Docker-контейнери, що дозволяє працювати з хмарними технологіями.
4) Tinynet – найшвидший та найменший емулятор серед аналогів, але при цьому має обмежений функціонал. Даний симулятор підходить для швидкого створення та розгортання маленьких SDN мереж.
5) Maxinet – інструмент для роботи з Mininet на декількох ВМ з розділенням роботи на частини для оптимізації роботи за рахунок розподілення ресурсів. Це чудовий емулятор для роботи з великими мережами та можливістю інтеграції з Docker.

У подальшому дослідженні імітаційних моделей я буду використовувати саме Maxinet, оскільки він є оптимальним рішення для створення та дослідження мереж SDN.

2.2. Команди Mininet та опис MiniEdit

Програма Mininet дозволяє створювати та та запускати віртуальні мережі різних типів, з різною кількістю контролерів та комутаторів. Але для використання усього функціоналу та створення моделей мережі з різними параметрами необхідно використовувати командний рядок, у якому застосовується синтаксис Python та операційна система Linux (Debian), тому необхідно мати базові знання команд. Опис деяких з них представлено нижче в Таблиці 2.1:

Таблиця 2.1

Основні команди Linux

Команда	Опис роботи команди
Команди управління файлами	
LS	Дозволяє переглядати вміст каталогів, одразу показує поточний каталог. Є можливість вказати шлях розташування для перерахування вмісту кінцевого каталогу. Доповнення –l(List) дозволяє дізнатися детальну інформацію за рахунок виводу списку більш детальної інформації. Доповнення –a(All) дозволяє вивести показ прихованих файлів.
CD	Застосовується для переходу до визначеного каталогу від поточного або для повернення до домашнього каталогу у разі запуску без параметрів.
FILE	Застосовується для виводу типу файлу. Вирішує проблему визначення типу користувачем в окремих випадках, оскільки для роботи не завжди необхідні розширення.
CP	Проводить копіювання файлів та каталогів, але по замовчуванню без розташування файлів у директоріях та піддиректоріях. Доповнення –r(Recursive) дозволяє це виправити, а доповнення –a(Archive) додає режим збереження атрибутів та вказує власника.
MV	Дозволяє перейменовувати (переміщувати у ту саму папку, але з іншим ім’ям) файли та каталоги.
RM	Застосовується для видалення конкретних файлів та папок. На жаль, дана команда не має можливості відновлення файлів і дії даного оператора незворотні.
FIND	Проводить пошук у файлах та папках з можливістю виконання заданих команд для файлів, які виявлені в процесі пошуку.

Закінчення таблиці 2.1

Команди управління процесами	
KILL/ KILLALL	Існує для завершення певних процесів, при цьому вони приймають різні параметри ідентифікації.
Команди взаємодії користувача	
SU/SUDO	Перша – перемикає на іншого користувача, друга – виконує певну команду від імені заданого користувача.
UNAME	Дозволяє вивести частину інформації про системні параметри. Доповнення –a(All) дозволяє вивести усю інформацію про ім’я хоста, архітектуру процесу та про ядро.
USERADD/ USERDEL/ USERMOD	Застосовуються для додавання/ видалення/ зміни облікових записів користувачів.
PASSWD	Дозволяє змінювати паролі облікових записів, привілейований користувач має можливість видаляти паролі усіх користувачів.
Команди управління мережею	
IP	Існує для показу списку команд для управління мережею.
PING	Проводить діагностику, надає уявлення про якість зв’язку та перевірку підключень між елементами мережі.
NETHOGS	Показує споживання швидкості та трафіку будь-якої програми.
TRACEROUTE	Модернізована «PING», додається інформація про доступність та час доставки пакетів між вузлами

У Mininet вся робота та маніпуляції з параметрами хостів та комутаторів у віртуальній мережі здійснюється за допомогою команд, що наведені в Таблиці 2.2:[8]

Таблиця 2.2

Основні команди Mininet

Команда	Опис роботи команди
DUMP	Дозволяє відобразити усі IP - адреси та інтерфейс мережевих пристроїв
INTFS	Виводить список задіяних інтерфейсів мережевих пристроїв
IPERF	Використовується для відображення інформації про обмін між хостами по TCP (Transmission Control Protocol)
IPERFUDP	Використовується для відображення інформації про обмін між хостами по UDP (User Datagram Protocol)
NET	Використовується для відображення з'єднання мережевих пристроїв
LINK	Додає нове з'єднання між мережевими пристроями
NODES	Виводить назви мережевих пристроїв у вигляді списку
NOECHO	Дозволяє виконувати команди без відображення результату
PING	Показує результат перевірки з'єднання між мережевими пристроями після пересилання пакетів
PINGPAIR	Показує результат перевірки з'єднання між мережевими пристроями після пересилання ICMP пакетів
PINGALL	Показує результат перевірки з'єднання між усіма пристроями

Програма MiniEdit дозволяє за допомогою графічного інтерфейсу створювати та та запускати віртуальні мережі різних типів, з різною кількістю контролерів та комутаторів, що в свою чергу і є доповненням Mininet.

Для створення мережі немає потреби використовувати командний рядок, із застосуванням синтаксису Python, а лише необхідно використовувати простий інтерфейс, що включає піктограми інструментів.

Опис усіх піктограм, що використовуються для створення мережі, представлено далі:[9]

Таблиця 2.3

Основні команди MiniEdit

Піктограма	Опис роботи
	Виділення Застосовується для зміни розташування мережевих елементів на робочому полотні. Для вибору вузла або посилання на полотні використовується вказівник миші, незалежно від активного інструменту.
	Хост Використовується для створення вузла, що містить функції комп'ютера. Якщо інструмент активний, то є можливість постійного розташування/додавання великої кількості хостів . провести налаштування певного хоста можна за допомогою меню «Властивості», яке появляється після натискання на праву клавішу.
	Комутатор Використовується для імітації підключення OpenFlow між мережевими елементами, при цьому комутатори необхідно підключити до контролера.

Закінчення таблиці 2.3

	Звичайний комутатор Імітація комутатора із заданими налаштуваннями за замовчуванням, що може працювати без контролера.
	Маршрутизатор Імітація базового роутера (у випадку MiniEdit це «Хост», у якому увімкнена зміна (переадресація) IP) із заданими налаштуваннями за замовчуванням, що може працювати без контролера.
	Зв'язок між вузлами Використовується для з’єднання усіх мережевих пристроїв на полотні шляхом перетягування від одного вузла до іншого. Налаштування зв'язку між вузлами проводяться аналогічно до налаштування інструменту «Хост».
	Контролер Застосовується для створення централізованого контролера, або декількох контролерів з можливістю налаштування параметрів. Налаштування властивостей контролера проводяться аналогічно до налаштування інструменту «Хост».

2.3. Запуск та налаштування MiniEdit

В Mininet існує багато додатків, що розташовані в папці mininet/examples/*. Усі ці додатки можна відкривати та використовувати для дослідження. MiniEdit також входить до складу цих прикладів і може бути запущений за допомогою команди з правами root:[10]

```
$ sudo ~/mininet/examples/miniedit.py
```

Рис. 2.6. Команда для запуску MiniEdit

Існує також покрокове відкриття через команду управління файлами у тому випадку, якщо MiniEdit не відкривається:

```
maxinet@worker1:~$ cd mininet
maxinet@worker1:~/mininet$ cd mininet
maxinet@worker1:~/mininet/mininet$ cd examples
maxinet@worker1:~/mininet/mininet/examples$ ls
```

Рис. 2.7. Команди для покрокового запуску MiniEdit

```
maxinet@worker1:~/mininet/mininet/examples$ ls
baresshd.py       controlnet.py      mobility.py        scratchnet.py
bind.py           cpu.py             multilink.py       scratchnetuser.py
clustercli.py     emptynet.py        multiping.py       simpleperf.py
clustercli.pyc    hwintf.py          multipoll.py       sshd.py
clusterdemo.py    __init__.py        multitest.py       test
cluster.py        __init__.pyc       natnet.py          tree1024.py
cluster.pyc       intfoptions.py     nat.py             treeping64.py
clusterSanity.py  limit.py           numberedports.py   vlanhost.py
consoles.py       linearbandwidth.py popenpoll.py
controllers2.py   linuxrouter.py     popen.py
controllers.py    miniedit.py        README.md
maxinet@worker1:~/mininet/mininet/examples$ sudo ./miniedit.py
```

Рис. 2.8. Команди для покрокового запуску MiniEdit

Після успішного запуску має появитися сповіщення:

```
MiniEdit running against Mininet 2.2.1
topo=none
```

Рис. 2.9. Сповіщення про успішний запуск MiniEdit

Для налаштування роботи MiniEdit необхідно у меню програми вибрати «Редагувати/Edit» і перейти до «Налаштування/Settings», після чого появиться діалогове вікно:

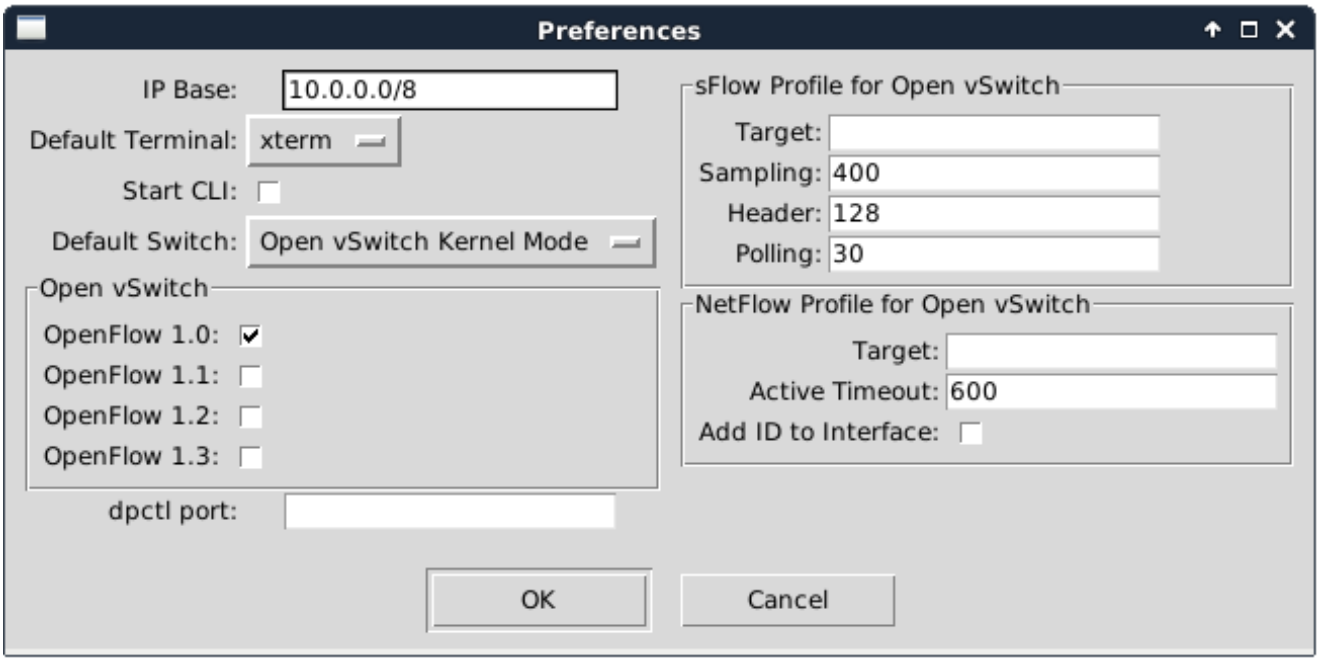

Рис. 2.10. Вікно налаштувань MiniEdit

Для того, щоб у нас був доступ до інтерфейсу командного рядка у процесі роботи MiniEdit необхідно вказати це, поставивши прапорець навпроти пункту «Start CLI» також тут необхідно обрати необхідну версію OpenFlow термінал за замовчуванням:

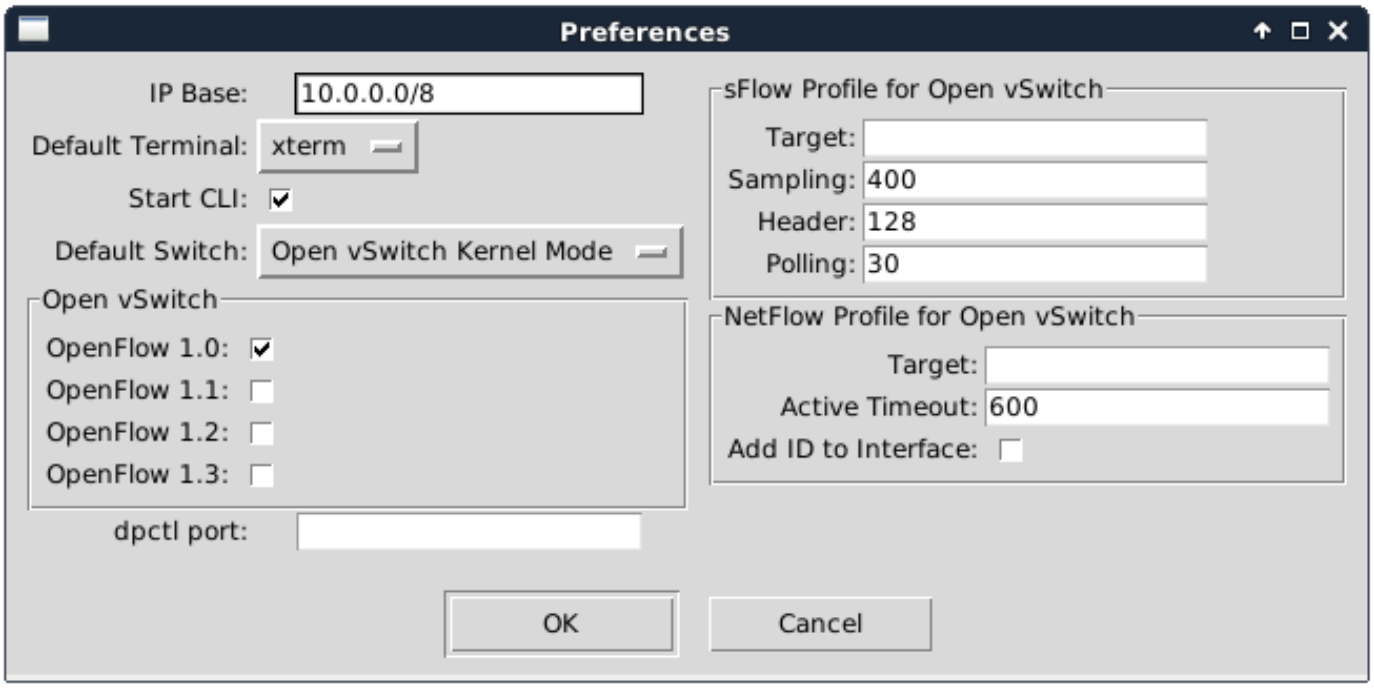

Рис. 2.11. Вікно налаштувань MiniEdit

Усі проведені налаштування забезпечують створення програмного сценарію створюваної мережі, що дозволить взаємодіяти хостам між собою. Налаштування для кожного сценарію зберігаються окремо один від одного, тому для кожної топології можна вказувати різні значення.

2.4. Мережеві топології у Mininet

Для створення імітаційних моделей SDN у Mininet існує декілька топологій за замовчуванням, які можна створити за допомогою командного рядка.

```
--topo=TOPO           linear|minimal|reversed|single|torus|tree[,param=value
                      ...] linear=LinearTopo torus=TorusTopo tree=TreeTopo
                      single=SingleSwitchTopo
                      reversed=SingleSwitchReversedTopo minimal=MinimalTopo
```

Рис. 2.12. Доступні топології у Mininet

Усі створювані топології будуть містити в собі комутатори, що називаються s1…sn та хости, що називаються h1…hn. Також хости містять порти, що містять в своїй назві ім’я хоста та номер Ethernet, що починається з 0. Mininet містить у собі такі базові топології, як: minimal, reversed, single, linear та tree. Далі буде коротко описано кожну з них та представлено візуальне представлення побудови мережі.

I. Minimal – найпростіша топологія серед усіх, що включає в себе 2 хости та 1 комутатор OpenFlow, що поєднує їх та забезпечує зв'язок для взаємодії між ними.

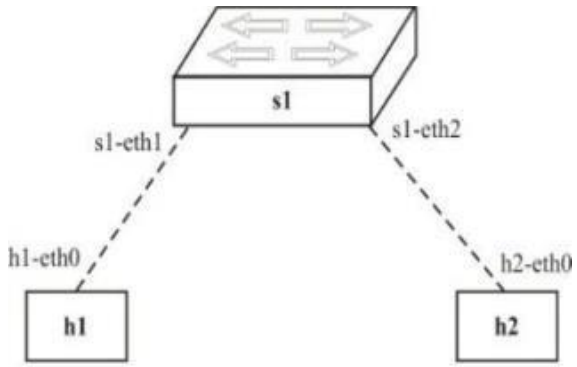

Рис. 2.13. Топологія Minimal

```
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
```

Рис. 2.14. З’єднання портів комутаторів та хостів топології Minimal

II. Single – ускладнена топологія minimal, що містить у собі більшу кількість хостів та один комутатор OpenFlow.

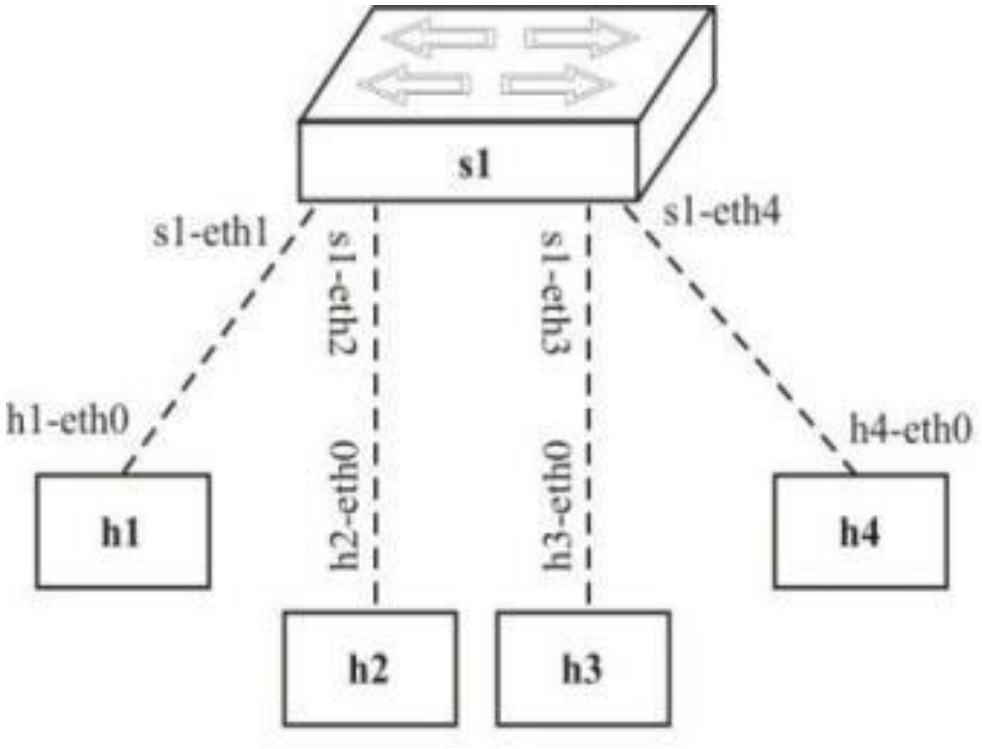

Рис. 2.17. Топологія Single

```
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
h4 h4-eth0:s1-eth4
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0 s1-eth4:h4-eth0
c0
```

Рис. 2.18. З’єднання портів комутаторів та хостів топології Single

III. Reversed – аналогічна топології Single, але має зворотній порядок з’єднання портів між хостами та комутатором.

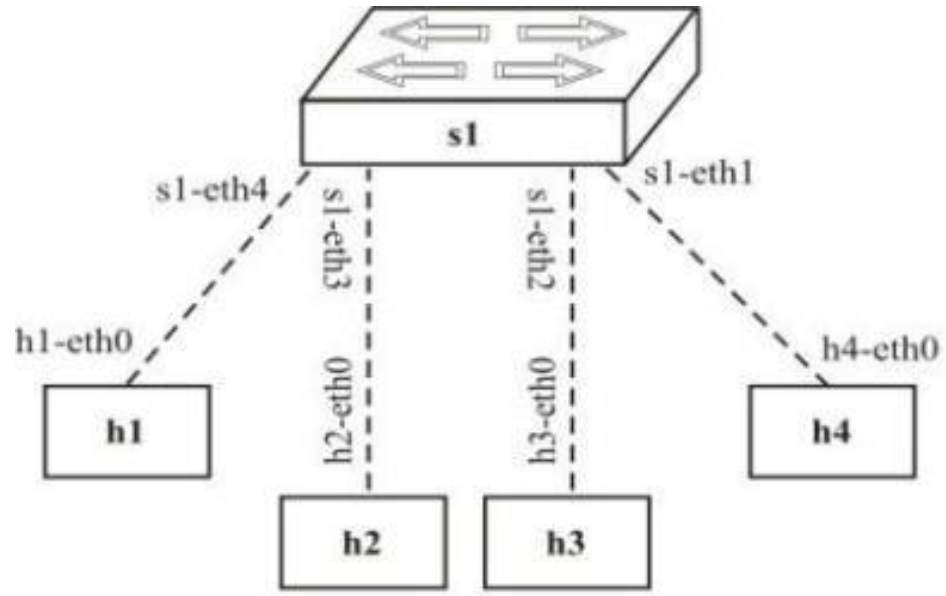

Рис. 2.15. Топологія Reversed

```
mininet> net
h1 h1-eth0:s1-eth4
h2 h2-eth0:s1-eth3
h3 h3-eth0:s1-eth2
h4 h4-eth0:s1-eth1
s1 lo:  s1-eth1:h4-eth0 s1-eth2:h3-eth0 s1-eth3:h2-eth0 s1-eth4:h1-eth0
c0
```

Рис. 2.16. З’єднання портів комутаторів та хостів топології Reversed

IV. Linear – топологія, що містить у собі n-кількість комутаторів та n-кількість хостів. У даній топології забезпечується з’єднання між усіма хостами та комутаторами.

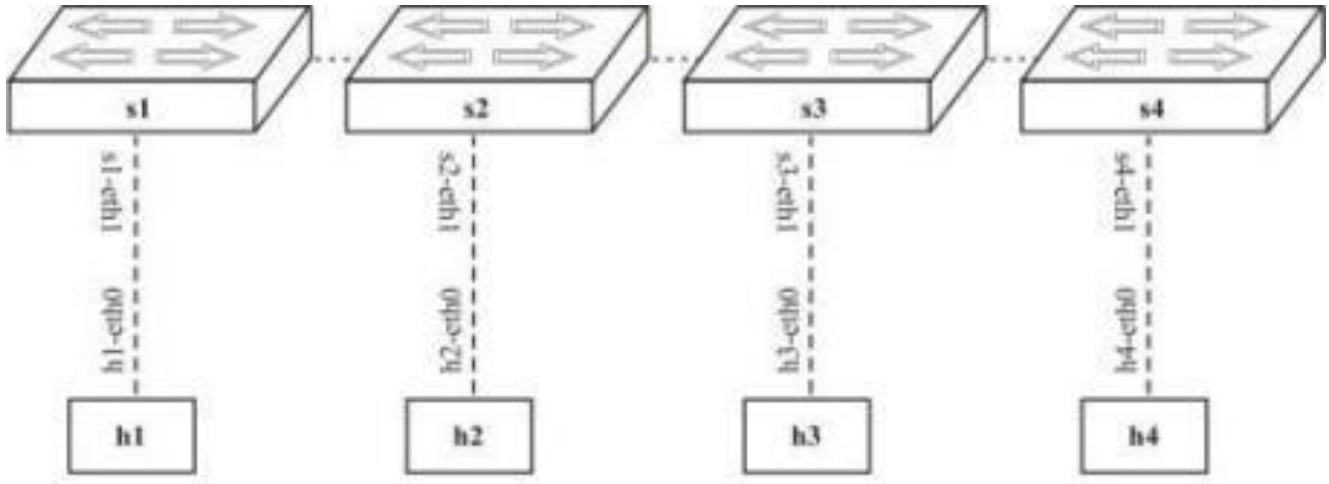

Рис. 2.19. Топологія Linear

```
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
s4 lo:  s4-eth1:h4-eth0 s4-eth2:s3-eth3
c0
```

Рис. 2.20. З’єднання портів комутаторів та хостів топології Linear

V. Tree – топологія, що складається з n-кількості рівнів комутаторів, що поєднані між собою та хостів, що підключені по 2 до кожного комутатора на останньому рівні.

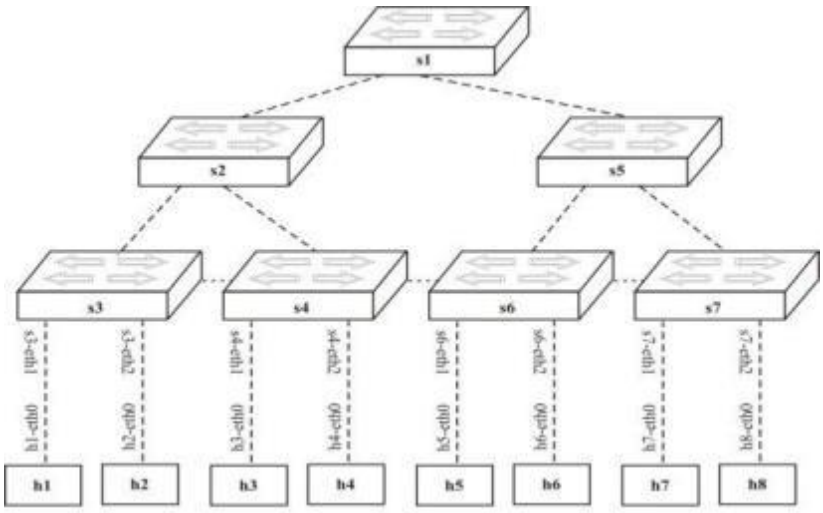

Рис. 2.21. Топологія Tree

```
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
h3 h3-eth0:s4-eth1
h4 h4-eth0:s4-eth2
h5 h5-eth0:s6-eth1
h6 h6-eth0:s6-eth2
h7 h7-eth0:s7-eth1
h8 h8-eth0:s7-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s5-eth3
s2 lo:  s2-eth1:s3-eth3 s2-eth2:s4-eth3 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h1-eth0 s3-eth2:h2-eth0 s3-eth3:s2-eth1
s4 lo:  s4-eth1:h3-eth0 s4-eth2:h4-eth0 s4-eth3:s2-eth2
s5 lo:  s5-eth1:s6-eth3 s5-eth2:s7-eth3 s5-eth3:s1-eth2
s6 lo:  s6-eth1:h5-eth0 s6-eth2:h6-eth0 s6-eth3:s5-eth1
s7 lo:  s7-eth1:h7-eth0 s7-eth2:h8-eth0 s7-eth3:s5-eth2
c0
```

Рис. 2.22. З’єднання портів комутаторів та хостів топології Tree

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі було проведено аналіз програмного забезпечення, що дозволяє досліджувати програмно-конфігуровані мережі. ПЗ, що використовувалося для дослідження різних мережевих топологій: Oracle VM VirtualBox, Mininet, що включає в себе MiniEdit, Xterm, WireShark. VirtualBox – ПЗ, що використовується для запуску віртуальної машини на комп'ютері користувача для роботи з Mininet. Mininet - ПЗ, що дозволяє створювати та досліджувати топології мереж SDN, воно також містить у собі MiniEdit та надає можливість підключення до Xterm та WireShark. MiniEdit – частина Mininet, що являє собою графічний інтерфейс для візуального представлення різноманітних мережевих топологій. Xterm - емулятор терміналу у MiniEdit, який підключається до графічних додатків. Оскільки, у MiniEdit відображається графічне представлення мережі, то Xterm використовується для запуску командного рядка для окремих мережевих елементів. WireShark – ПЗ для моніторингу, аналізу та дослідження повідомлень, що передаються між компонентами мережі.

Також в даному розділі були описані команди Mininet, що використовуються для дослідження імітаційних моделей та інтерфейс MiniEdit, за допомогою якого створюються мережеві топології SDN. Крім цього, були надані рекомендації по запуску із командного рядка Mininet ПЗ MiniEdit, також його налаштування.

Було також проведено аналіз емуляторів, що використовуються для роботи з Mininet, серед яких: ns-3, OpenNet, Containernet, Tinynet та Maxinet. Були досліджені усі плюси та мінуси даних емуляторів і в результаті я для подальшого дослідження імітаційних моделей я обрав Maxinet, оскільки він найкращим для поставлених задач.

У ПЗ Mininet було визначено 5 мережевих топологій, які будуть створюватися та досліджуватися у наступному пункті, серед яких: minimal, reversed, single, linear та tree.

РОЗДІЛ 3
ДОСЛІДЖЕННЯ ІМІТАЦІЙНИХ МОДЕЛЕЙ

3.1. Дослідження мережевої топології minimal

Дана топологія є стандартною у Mininet і містить в собі один комутатор «s1», контролер «c0» і два хоста, «h1» та «h2». Створюється від імені користувача, за допомогою «sudo», та інтерпретатору команд «mn».

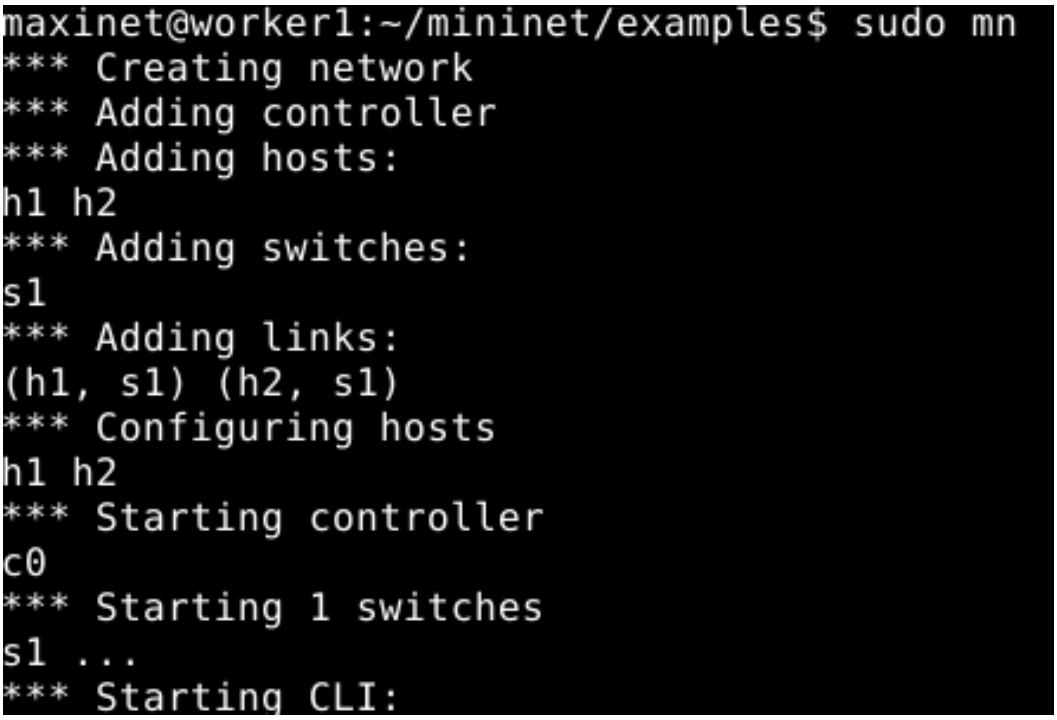

Рис. 3.1. Створення топології

Для дослідження роботи даної топології використовуються наступні команди:
Для перевірки усіх мережевих пристроїв застосовуємо команду «nodes»:

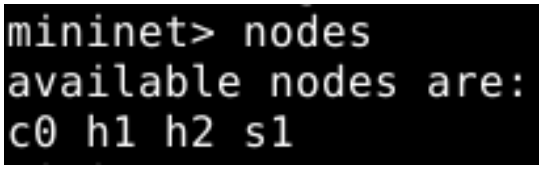

Рис. 3.2. Вузли мережі

Як ми бачимо по Рис. 3.2., у нас доступні: один комутатор «s1», контролер «c0» і два хоста, «h1» та «h2». Тобто, топологія створена правильно і далі необхідно перевірити з'єднання портів комутаторів та хостів, що здійснюється за допомогою команди «net»:

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
```

Рис. 3.3. З'єднання мережевих елементів

Для виводу повної інформації про хост або комутатор використовуємо команду «ifconfig», вказавши назву необхідного хоста чи комутатора:

```
mininet> h1 ifconfig
h1-eth0   Link encap:Ethernet  HWaddr 7e:56:8b:5d:f1:ae
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::7c56:8bff:fe5d:f1ae/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:558 (558.0 B)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Рис. 3.4. Налаштування хоста h1

Після перегляду усього мережевого інтерфейсу перевіряємо маршрутизацію певного хоста за допомогою команди «route» (також з вказаним іменем):

```
mininet> h1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.0.0.0       U     0      0        0 h1-eth0
```

Рис. 3.5. Налаштування маршрутизації хоста h1

Також перевіряємо відправку запитів між заданими хостами, час відправки та отримання, тобто пінгування, вказавши назви двох хостів:

```
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=2.18 ms
^[[A64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.138 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.041 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.037 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.038 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.037 ms
64 bytes from 10.0.0.2: icmp_seq=8 ttl=64 time=0.042 ms
64 bytes from 10.0.0.2: icmp_seq=9 ttl=64 time=0.040 ms
64 bytes from 10.0.0.2: icmp_seq=10 ttl=64 time=0.042 ms
64 bytes from 10.0.0.2: icmp_seq=11 ttl=64 time=0.037 ms
^C
--- 10.0.0.2 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 9998ms
rtt min/avg/max/mdev = 0.037/0.243/2.180/0.613 ms
```

Рис. 3.6. Пінгування хостів h1 та h2

Тестування між усіма хостами мережі, що в результаті покаже % втрачених пакетів та кількість хостів, які успішно виконали перевірку:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

Рис. 3.7. Пінгування усіх хостів

Для визначення пропускної здатності між мережевими елементами використовується команда «iperf»:

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['21.8 Gbits/sec', '21.8 Gbits/sec']
```

Рис. 3.8. Пропускна здатність першого та останнього хостів

3.2. Дослідження мережевої топології single

Топологія «single» містить в собі один комутатор «s1», контролер «c0» і велику кількість хостів. Створюється від імені користувача, за допомогою «sudo», та інтерпретатору команд «mn» з додаванням назви топології та кількість хостів: «--topo = [назва топології], [кількість хостів]». В моєму випадку кількість хостів – 20.

```
maxinet@worker1:~/mininet/examples$ sudo mn --topo=single,20
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1) (h10, s1) (
h11, s1) (h12, s1) (h13, s1) (h14, s1) (h15, s1) (h16, s1) (h17, s1) (h18, s1) (h19, s1) (h2
0, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Рис. 3.17. Створення топології

Для дослідження роботи даної топології також використовуються наступні команди:

Перевірка усіх комутаторів, контролерів та хостів:

```
mininet> nodes
available nodes are:
c0 h1 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h2 h20 h3 h4 h5 h6 h7 h8 h9 s1
```

Рис. 3.18. Вузли мережі

Як ми бачимо по Рис. 3.2., у нас доступні: один комутатор «s1», контролер «c0» і двадцять хостів, «h1 - h20». Тобто, топологія створена правильно і далі необхідно перевірити з’єднання портів комутаторів та хостів:

Рис. 3.19. З’єднання мережевих елементів

Після виводу результатів можна побачити усі 20 хостів і підключені до них порти.

Вивід повної інформації про перший хост:

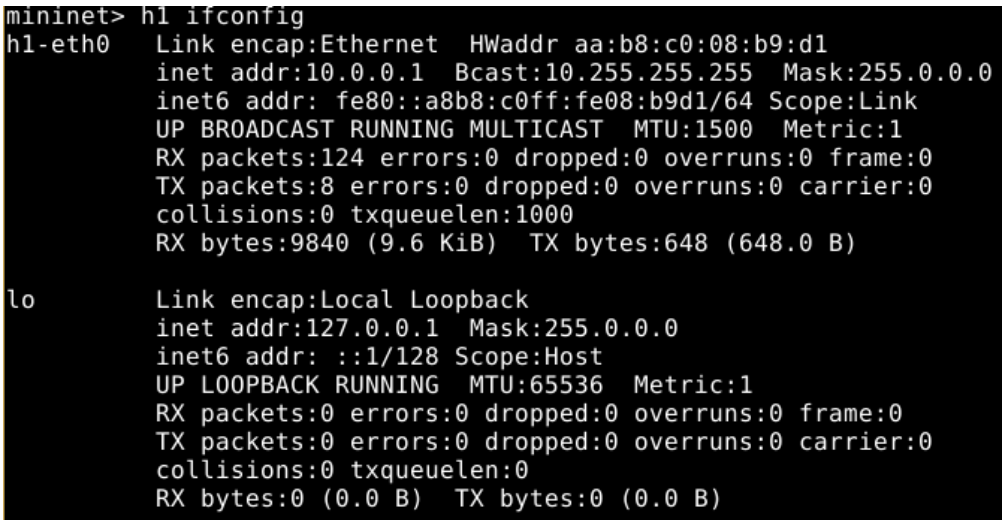

Рис. 3.20. Налаштування хоста h1

Перевірка маршрутизації першого хоста:

Рис. 3.21. Налаштування маршрутизації хоста h1

Відправка запитів між першим та другим хостами, час відправки та отримання:

```
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=2.22 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.169 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.047 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.045 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 4998ms
rtt min/avg/max/mdev = 0.043/0.428/2.221/0.803 ms
```

Рис. 3.22. Пінгування хостів h1 та h2

Тестування між усіма 20 хостами:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12 h13 h14 h15 h16 h17 h18 h19 h20
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h13 h14 h15 h16 h17 h18 h19 h20
h13 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h14 h15 h16 h17 h18 h19 h20
h14 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h15 h16 h17 h18 h19 h20
h15 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h16 h17 h18 h19 h20
h16 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h17 h18 h19 h20
h17 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h18 h19 h20
h18 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h19 h20
h19 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h20
h20 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19
*** Results: 0% dropped (380/380 received)
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h20
*** Results: ['21.3 Gbits/sec', '21.3 Gbits/sec']
```

Рис. 3.23. Пінгування усіх хостів

Визначення пропускної здатності між першим та двадцятим хостами:

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h20
*** Results: ['21.3 Gbits/sec', '21.3 Gbits/sec']
```

Рис. 3.24. Пропускна здатність першого та останнього хостів

3.3. Дослідження мережевої топології reversed

Топологія «reversed» є аналогічною топології «single» і містить в собі один комутатор «s1», контролер «c0» і два хоста, «h1» та «h2», але має обернене розташування портів. Створюється від імені користувача, за допомогою «sudo», та інтерпретатору команд «mn» з додаванням назви топології: «-- topo = [назва топології]».

```
maxinet@worker2:~$ sudo mn --topo=reversed
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

Рис. 3.9. Створення топології

Для дослідження роботи даної топології також використовуються наступні команди:

Перевірка усіх мережевих пристроїв:

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
```

Рис. 3.10. Вузли мережі

Як ми бачимо по Рис. 3.2., у нас доступні: один комутатор «s1», контролер «c0» і два хоста, «h1» та «h2». Тобто, топологія створена правильно і далі необхідно перевірити з'єднання портів комутаторів та хостів:

```
mininet> net
h1 h1-eth0:s1-eth2
h2 h2-eth0:s1-eth1
s1 lo:  s1-eth1:h2-eth0 s1-eth2:h1-eth0
c0
```

Рис. 3.11. З'єднання мережевих елементів

Порівнявши дані результати з топологією «minimal», можна побачити відмінність у розташуванні портів хостів.

Вивід повної інформації про хост «h1»:

```
mininet> h1 ifconfig
h1-eth0   Link encap:Ethernet  HWaddr 2a:0e:70:31:91:74
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::280e:70ff:fe31:9174/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:558 (558.0 B)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Рис. 3.12. Налаштування хоста h1

Перевірка маршрутизації хоста «h1»:

```
mininet> h1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.0.0.0       U     0      0        0 h1-eth0
```

Рис. 3.13. Налаштування маршрутизації хоста h1

Відправка запитів між хостами «h1» та «h2», час відправки та отримання:

```
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=2.04 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.184 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.041 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.043 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.042 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5000ms
rtt min/avg/max/mdev = 0.041/0.399/2.041/0.736 ms
```

Рис. 3.14. Пінгування хостів h1 та h2

Тестування між усіма хостами мережі:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

Рис. 3.15. Пінгування усіх хостів

Визначення пропускної здатності між мережевими елементами:

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['22.5 Gbits/sec', '22.5 Gbits/sec']
```

Рис. 3.16. Пропускна здатність першого та останнього хостів

3.4. Дослідження мережевої топології linear

Топологія «linear» містить в собі задану кількість комутаторів та хостів і контролер «c0». Створюється від імені користувача, за допомогою «sudo», та інтерпретатору команд «mn» з додаванням назви топології та кількість

комутаторів/хостів: «-- topo = [назва топології], [кількість комутаторів/хостів]». В моєму випадку кількість комутаторів – 5.

```
maxinet@worker1:~/mininet/examples$ sudo mn --topo=linear,5
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1 s2 s3 s4 s5
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (h5, s5) (s2, s1) (s3, s2) (s4, s3) (s5, s4)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c0
*** Starting 5 switches
s1 s2 s3 s4 s5 ...
*** Starting CLI:
```

Рис. 3.25. Створення топології

Для дослідження роботи даної топології також використовуються наступні команди:

Перевірка усіх доступних мережевих пристроїв:

```
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 s1 s2 s3 s4 s5
```

Рис. 3.26. Вузли мережі

Як ми бачимо по Рис. 3.2., у нас доступні: 5 комутаторів «s1- s5», контролер «c0» і 5 хостів, «h1 – h5». Тобто, топологія створена правильно і далі необхідно перевірити з’єднання портів комутаторів та хостів:

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
h5 h5-eth0:s5-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
s4 lo:  s4-eth1:h4-eth0 s4-eth2:s3-eth3 s4-eth3:s5-eth2
s5 lo:  s5-eth1:h5-eth0 s5-eth2:s4-eth3
c0
```

Рис. 3.27. З’єднання мережевих елементів

Після виводу результатів можна побачити усі 5 хостів та 5 комутаторів і підключені до них порти.

Вивід повної інформації про перший хост:

```
mininet> h1 ifconfig
h1-eth0   Link encap:Ethernet  HWaddr 2a:0f:31:2a:a1:06
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::280f:31ff:fe2a:a106/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:28 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2232 (2.1 KiB)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Рис. 3.28. Налаштування хоста h1

Перевірка маршрутизації першого хоста:

```
mininet> h1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.0.0.0       U     0      0        0 h1-eth0
```

Рис. 3.29. Налаштування маршрутизації хоста h1

Відправка запитів між 1 та 2 хостами, час відправки та отримання:

```
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=7.17 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.192 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.046 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.050 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.050 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.058 ms
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6001ms
rtt min/avg/max/mdev = 0.046/1.088/7.179/2.487 ms
```

Рис. 3.30. Пінгування хостів h1 та h2

Тестування між усіма 5 хостами:

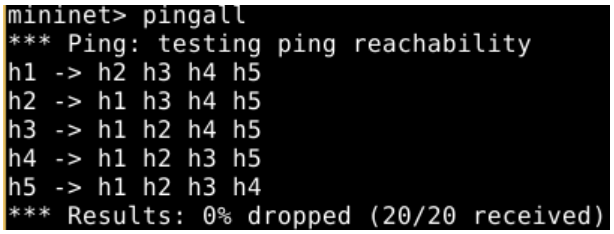

Рис. 3.31. Пінгування усіх хостів

Визначення пропускної здатності між 1 та 5 хостами:

Рис. 3.32. Пропускна здатність першого та останнього хостів

3.5. Дослідження мережевої топології tree

Топологія «tree» містить в собі заданий рівень комутаторів «depth», та кількість хостів «fanout», що підключені до них, а також контролер «c0». Створюється від імені користувача, за допомогою «sudo», та інтерпретатору команд «mn» з додаванням назви топології, рівня комутаторів та рівня хостів: «-- topo = [назва топології], [depth = заданий рівень комутаторів, fanout = кількість хостів]». В моєму випадку кількість рівнів та хостів – 2.

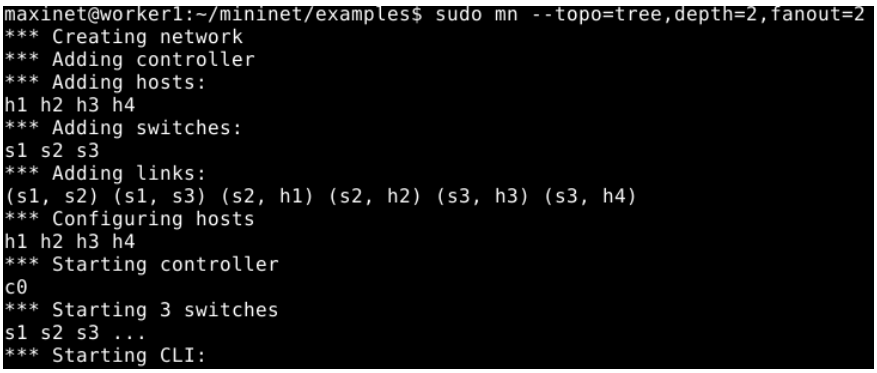

Рис. 3.33. Створення топології

Для дослідження роботи даної топології також використовуються наступні команди:

Перевірка усіх доступних мережевих пристроїв:

```
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 s1 s2 s3
```

Рис. 3.34. Вузли мережі

Як ми бачимо по Рис. 3.2., у нас доступні: 3 комутатори «s1- s3», оскільки на першому рівні – один комутатор і на другому рівні – 2 комутатори, також доступний один контролер «c0» і 4 хости, «h1 – h4», оскільки на другому рівні до кожного комутатора підключено по 2 хости. Тобто, топологія створена правильно і далі необхідно перевірити з'єднання портів комутаторів та хостів:

```
mininet> net
h1 h1-eth0:s2-eth1
h2 h2-eth0:s2-eth2
h3 h3-eth0:s3-eth1
h4 h4-eth0:s3-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s3-eth3
s2 lo:  s2-eth1:h1-eth0 s2-eth2:h2-eth0 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h3-eth0 s3-eth2:h4-eth0 s3-eth3:s1-eth2
c0
```

Рис. 3.35. З'єднання мережевих елементів

Після виводу результатів можна побачити усі 4 хости та 3 комутатори і підключені до них порти.

Вивід повної інформації хост:

```
mininet> h1 ifconfig
h1-eth0   Link encap:Ethernet  HWaddr f6:a9:48:7b:21:9f
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::f4a9:48ff:fe7b:219f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:21 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1674 (1.6 KiB)  TX bytes:648 (648.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Рис. 3.36. Налаштування хоста h1

Перевірка маршрутизації хоста:

```
mininet> h1 route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.0.0        *               255.0.0.0       U     0      0        0 h1-eth0
```

Рис. 3.37. Налаштування маршрутизації хоста h1

Відправка запитів між хостами, час відправки та отримання:

```
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=3.46 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.153 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.040 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.041 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.042 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.035 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.043 ms
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6002ms
rtt min/avg/max/mdev = 0.035/0.546/3.469/1.193 ms
```

Рис. 3.38. Пінгування хостів h1 та h2

Тестування між усіма 4 хостами:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
```

Рис. 3.39. Пінгування усіх хостів

Визначення пропускної здатності між 1 та 4 хостами:

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h4
*** Results: ['20.2 Gbits/sec', '20.2 Gbits/sec']
```

Рис. 3.40. Пропускна здатність першого та останнього хостів

3.6. Дослідження мережевих топологій у MiniEdit

За допомогою графічного інтерфейсу MiniEdit можна створювати топології різних рівнів і з різною кількістю комутаторів і хостів. Для цього необхідно розташувати необхідні нам елементи та з'єднати їх між собою для утворення зв'язку. У моєму випадку було створено топологію, що складається з одного контролера, двох комутаторів і 8 хостів:

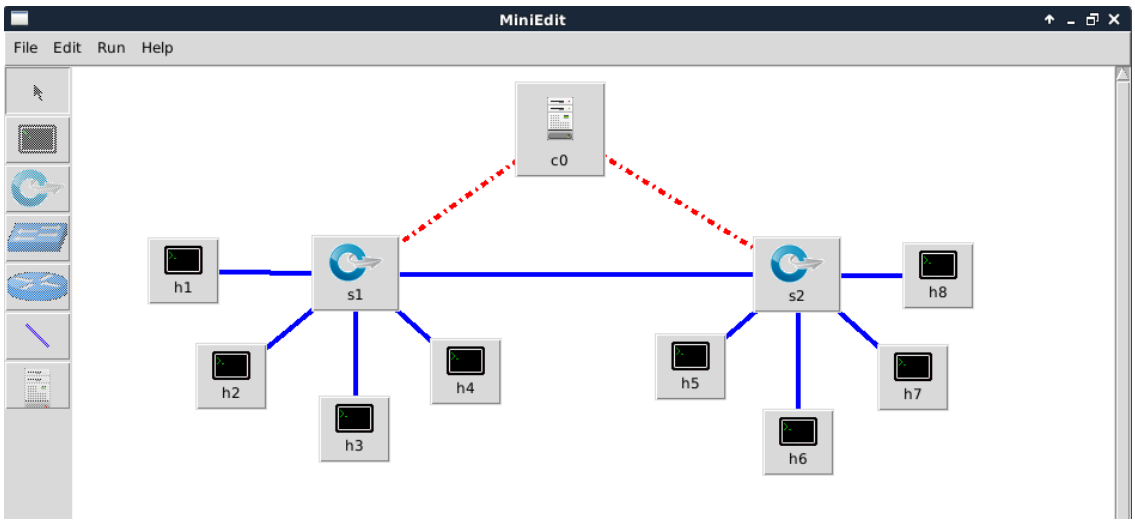

Рис. 3.41. Створення топології у MiniEdit

Після завершення побудови топології у графічному інтерфейсі ми запускаємо моделювання роботи мережі, в результаті чого у командному рядку відображається процес створення мережі:

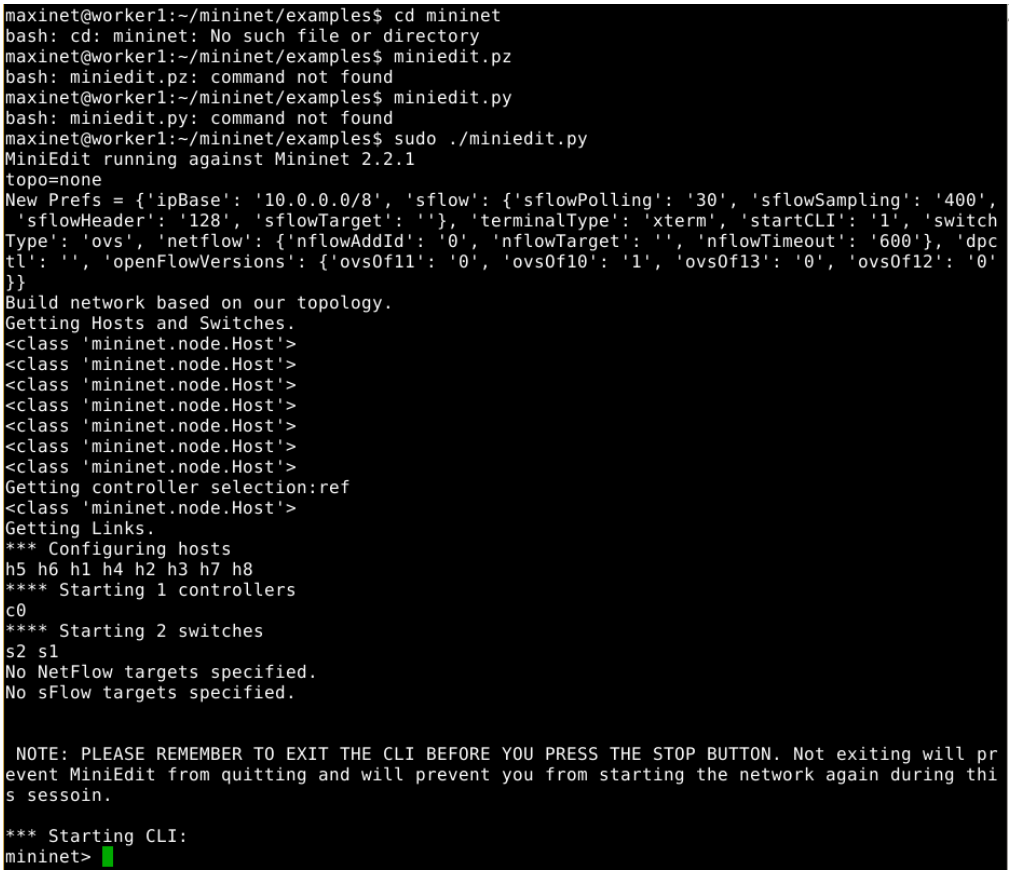

Рис. 3.42. Створення топології у Mininet

Для дослідження роботи даної топології у нас також є можливість використовуються наступні команди:

Перевірка усіх доступних мережевих пристроїв:

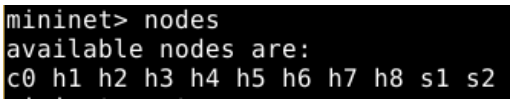

Рис. 3.43. Вузли мережі

Як ми бачимо по Рис. 3.2., у нас доступні: 2 комутатори «s1- s2», один контролер «c0» і 8 хостів, «h1 – h8». Перевірка з’єднання портів комутаторів та хостів:

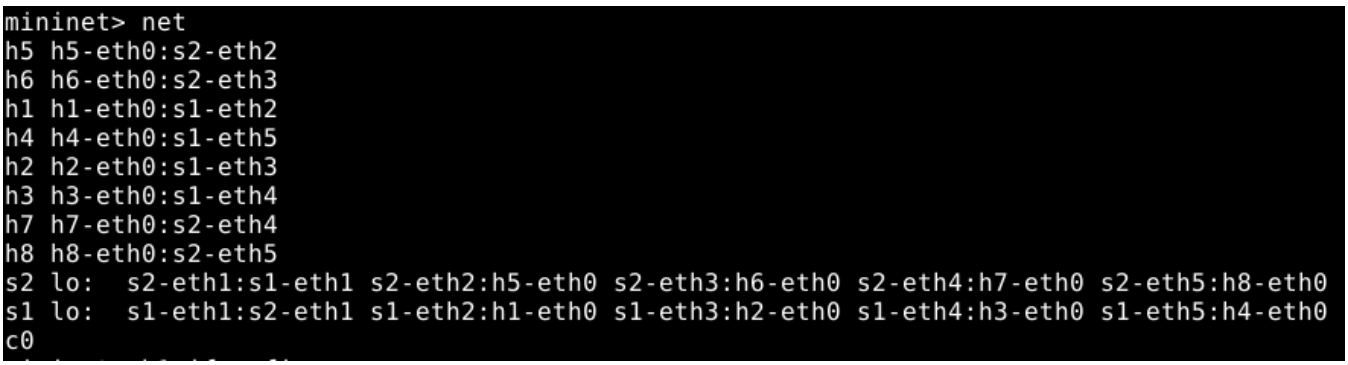

Рис. 3.44. З’єднання мережевих елементів

Після виводу результатів можна побачити усі 8 хостів та 2 комутатори і підключені до них порти.

Вивід повної інформації хост:

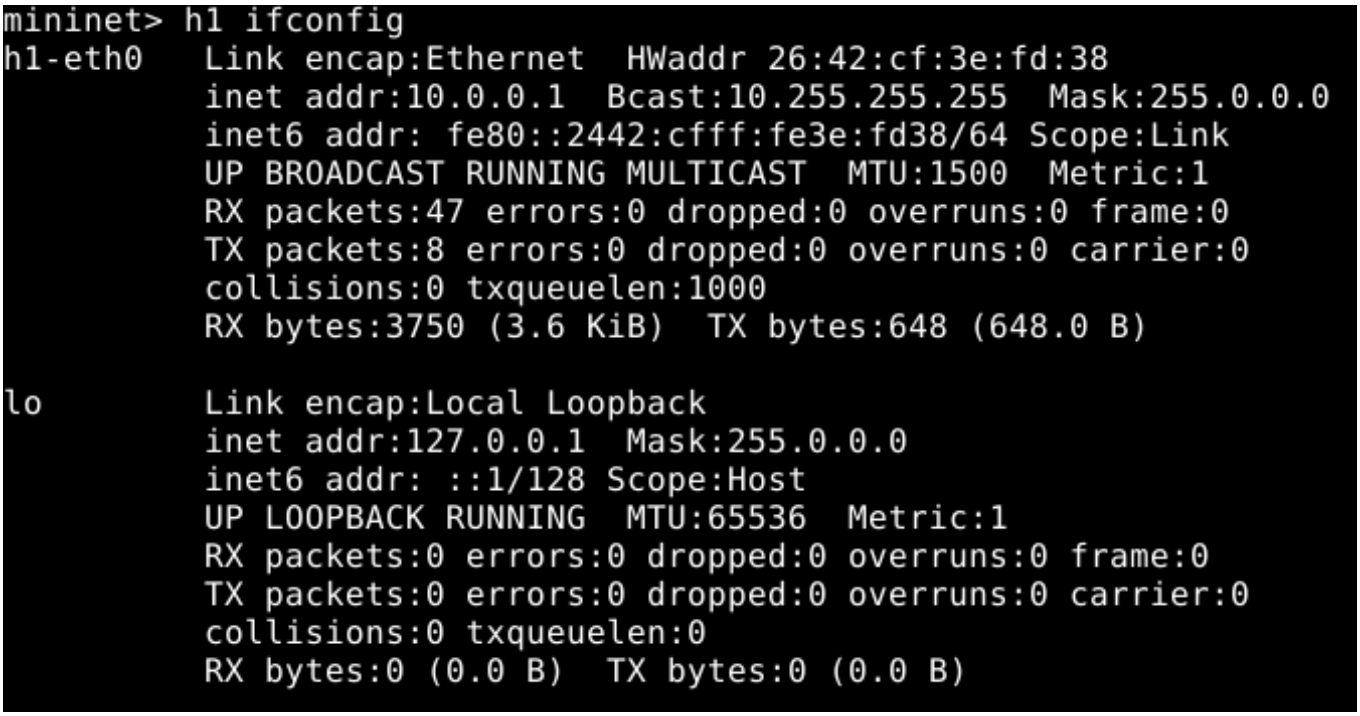

Рис. 3.45. Налаштування хоста h1

Перевірка маршрутизації хоста:

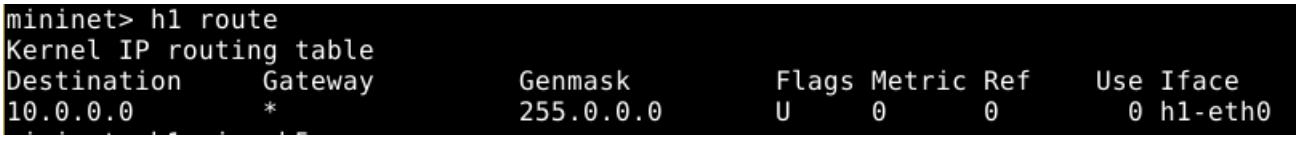

Рис. 3.46. Налаштування маршрутизації хоста h1

Відправка запитів між хостами, час відправки та отримання:

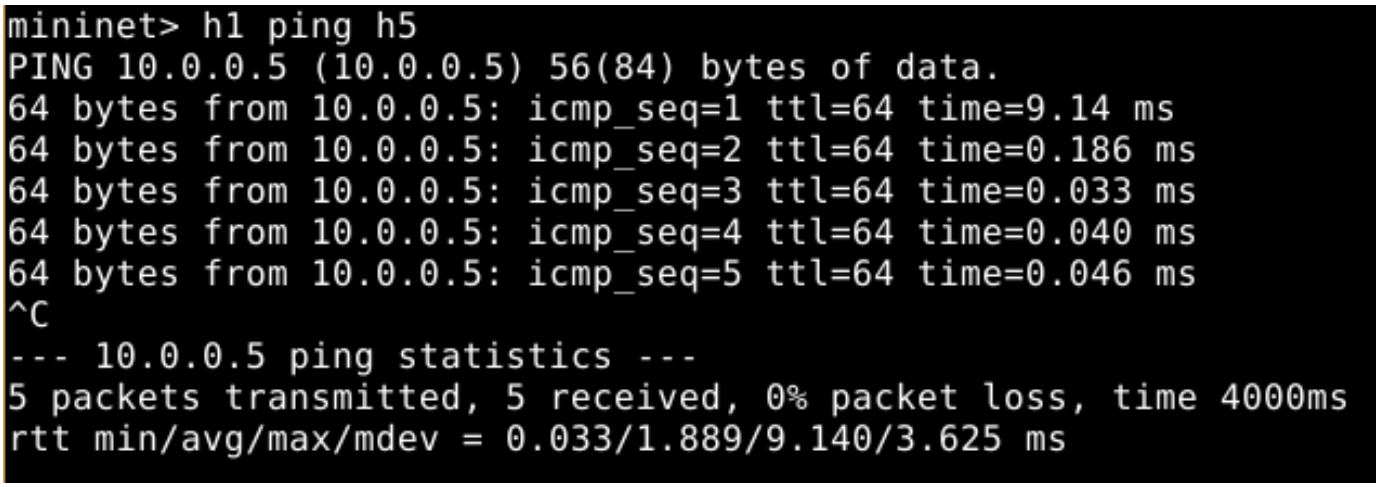

Рис. 3.47. Пінгування хостів h1 та h2

Тестування між усіма хостами:

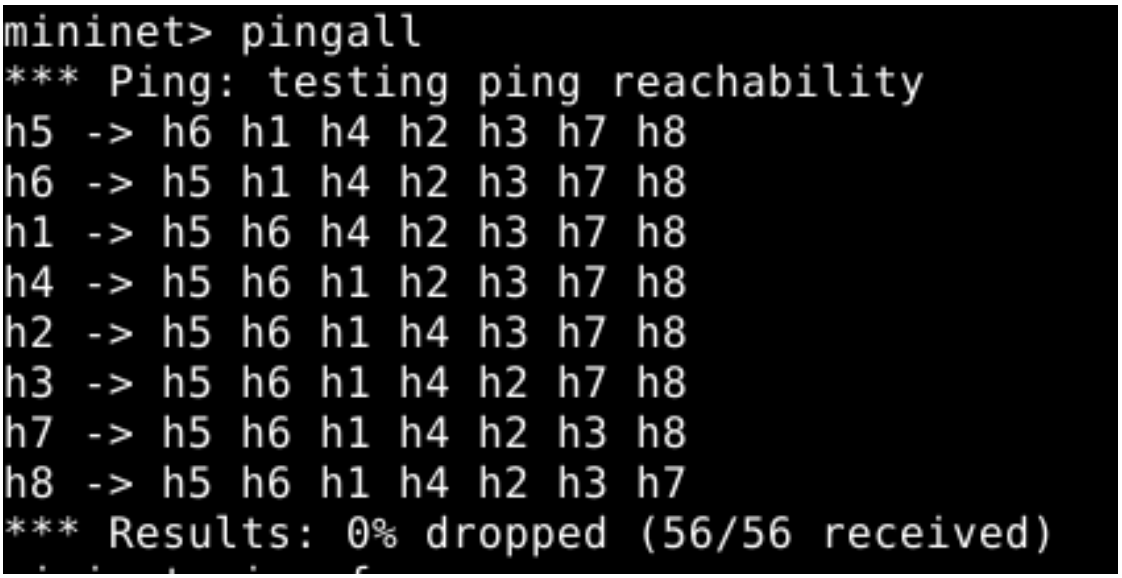

Рис. 3.48. Пінгування усіх хостів

Визначення пропускної здатності між 5 та 8 хостами:

Рис. 3.49. Пропускна здатність першого та останнього хостів

Для перевірки конфігурації комутатора у MiniEdit у вкладці «Run» є пункт «Show OVS Summary», що дозволяє перевірити порти контролерів:

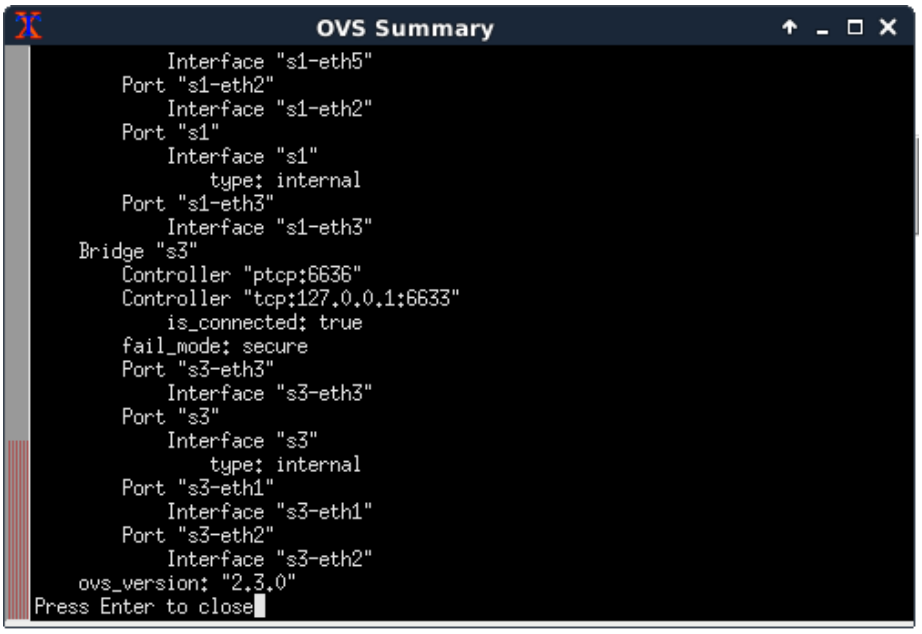

Рис. 3.50. Конфігурації комутатора

Також для кожного хоста ми можемо відкрити термінал Xterm. В одному з терміналів я запустив аналізатор трафіку, а в іншому – трасування пакетів, щоб можна було протестувати зв'язок між хостами:

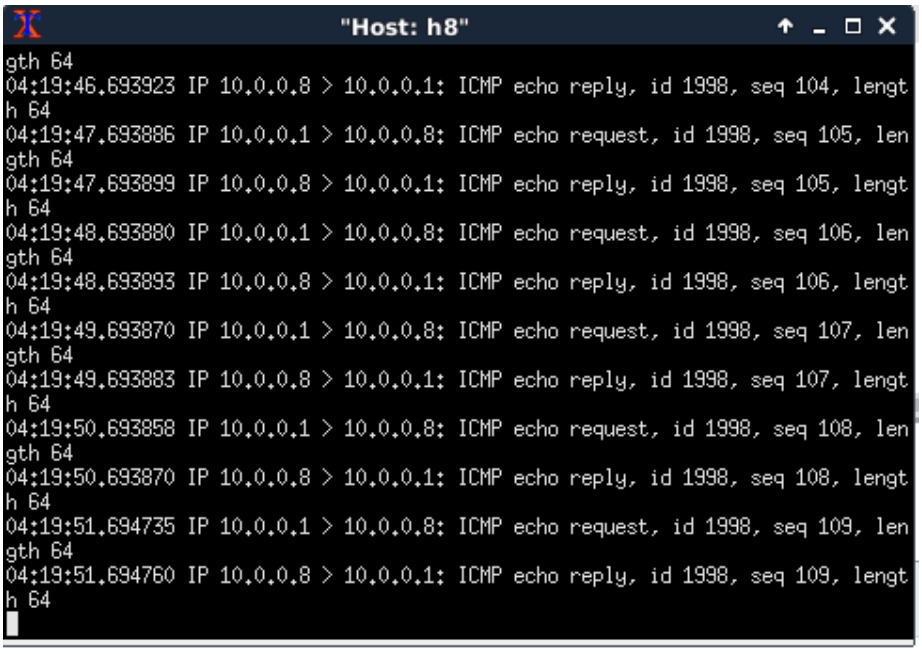

Рис. 3.51. Відправка пакетів

Відповідне відображення у Wireshark:

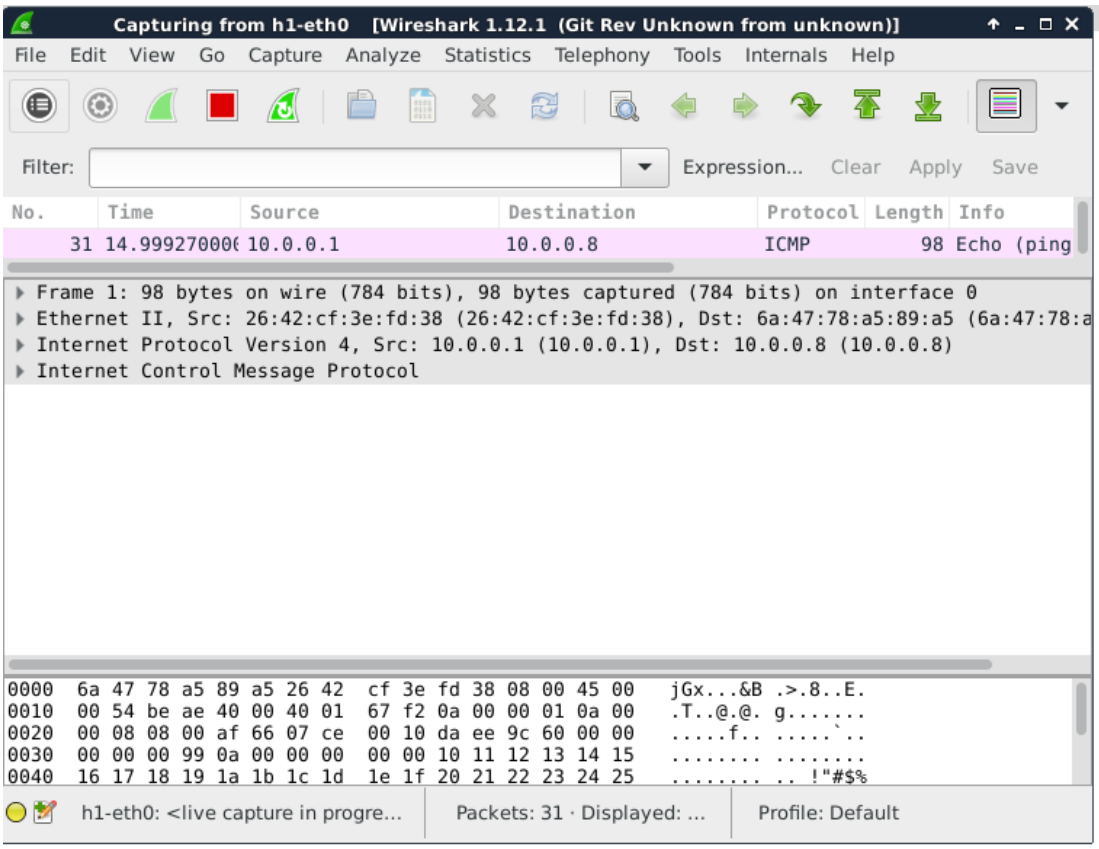

Рис. 3.52. Аналіз трафіку для хоста h1

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі було проведено дослідження 5 мережевих топологій, що створюються у програмному забезпеченні Mininet: minimal, reversed, single, linear та tree. Топологія minimal складається з одного комутатора «s1», контролера «c0» і двох хостів, «h1» та «h2». Топологія reversed також містить в собі один комутатор «s1», контролер «c0» і два хоста, «h1» та «h2», але має обернене розташування портів. Топологія single містить в собі один комутатор «s1», контролер «c0» і в моєму випадку 20 хостів, хоча кількість хостів вказується при створенні мережі. Топологія linear містить в собі 5 комутаторів та хостів, що також вказуються при створенні мережі, і контролер «c0». Топологія tree містить в собі 2 комутатори та 2 хости, що підключені до них, а також контролер «c0», кількість комутаторів та

хостів вказується при створенні. Для кожної з топологій було проведено тестування працездатності і перевірено підключення та налаштування мережевих елементів.

Також в даному розділі було проведено дослідження мережевої топології за допомогою ПЗ MiniEdit, яка складається з з одного контролера «c0», двох комутаторів «s1» та «s2», і 8 хостів «h1 – h8». Для даної топології також було проведено тестування працездатності і перевірено підключення та налаштування мережевих елементів. Також в процесі дослідження мережі було проведено підключення до двох хостів за допомогою ПЗ Xterm для налаштування передачі пакетів між хостами. Після чого через термінал Xterm було проведено запуск ПЗ Wireshark для аналізу даного трафіку.

ВИСНОВКИ

Програмно-конфігуровані мережі в даний час є перспективною технологією і має багато переваг, порівняно з традиційними мережами. Технологія SDN надає програмну підтримку зі зменшенням кількості фізичних компонентів, при цьому забезпечуючи підвищення ефективності мережевого обладнання, зниження витрат на експлуатацію існуючих мереж та наданням можливості оперативного створення та завантаження в мережеве обладнання сервісів користувачами. Гнучкість управління забезпечує спрощення мережі, надає можливість покращення керування, автоматизації управління та адміністрування, а також створення безлічі нових додатків без попереднього налаштування мережевих елементів. Створення і введення нових додатків та послуг у мережу більше не вимагає переналаштування усього з'єднання елементів, що дозволить економити час. Заміна безлічі розподілених протоколів і вузлів мережі на нову архітектуру SDN полегшує логіку взаємодії мережевих елементів, оскільки зменшується кількість запитів, інструкцій та стандартів. Також спрощення структури та логіки мережі SDN забезпечує можливість незалежного створення, розгортання та масштабування рівнів управління і передачі даних. Зменшення кількості компонентів мережі за рахунок централізованих контролерів та заміни безлічі протоколів управління із забезпеченням підтримки високої пропускної здатності зменшуються вартість мережі без втрати ефективності та продуктивності.

У першому розділі було визначено відмінності традиційної мережі від мережі SDN, порівняно інтерфейси хмарних технологій та SDN, розглянуто архітектуру програмно-конфігурованих мереж. Проаналізувавши технологію SDN, можна сказати, що мережі SDN забезпечують централізацію логічної взаємодії, автоматизацію та програмованість, а також ізоляцію пристроїв та додатків. Усе це забезпечує оптимізацію роботи мережі, оскільки є можливість постійного оновлення та конфігурації мережевих елементів. Дослідження ринку SDN в Україні показало, що компанії тільки починають оновлювати своє мережеве обладнання. Враховуючи

технологічність та відношення ціна/якість програмно-конфігурованих мереж, в даний час впровадження даної технології у нашій країні є доволі актуальною темою. Протягом наступних років кількість нових підприємств або тих, що замінюють застаріле обладнання будуть надавати перевагу SDN мережам, що також доводить актуальність теми дипломної роботи. А дослідження перспективи розвитку ПКМ показало, що вони мають досить широкий потенціал і постійне удосконалення і оновлення дозволяє покращувати роботу попередніх рішень та впроваджувати нові. Створення нових операційних систем та платформ на базі технології SDN дозволяє усувати усі недоліки попередніх рішень.

У другому розділі було проведено аналіз програмного забезпечення, що дозволяє досліджувати програмно-конфігуровані мережі. ПЗ, що використовувалося для дослідження різних мережевих топологій: Oracle VM VirtualBox, Mininet, що включає в себе MiniEdit, Xterm, WireShark. Були описані команди Mininet, що використовуються для дослідження імітаційних моделей та інтерфейс MiniEdit, за допомогою якого створюються мережеві топології SDN. Крім цього, були надані рекомендації по запуску із командного рядка Mininet ПЗ MiniEdit, також його налаштування. Проведено аналіз емуляторів, що використовуються для роботи з Mininet, серед яких: ns-3, OpenNet, Containernet, Tinynet та Maxinet. Досліджені усі плюси та мінуси даних емуляторів і в результаті я для подальшого дослідження імітаційних моделей я обрав Maxinet, оскільки він найкращим для поставлених задач.

У третьому розділі було проведено дослідження 5 мережевих топологій, що створюються у програмному забезпеченні Mininet: minimal, reversed, single, linear та tree. Також в даному розділі було проведено дослідження мережевої топології за допомогою ПЗ MiniEdit. Для кожної з топологій було проведено тестування працездатності і перевірено підключення та налаштування мережевих елементів. Також в процесі дослідження мережі у MiniEdit було проведено підключення до двох хостів за допомогою ПЗ Xterm для налаштування передачі пакетів між хостами. Після чого через термінал Xterm було проведено запуск ПЗ Wireshark для аналізу даного трафіку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SDN и другие / Сергей Орлов // Журнал сетевых решений/LAN. – 2014. – № 06.

2. Черняк Л. SDN – от замысла до рынка / Леонид Черняк // Открытые системы. – 2012. – № 09.

3. Смелянский Р. Программно-конфигурируемые сети / Руслан Смелянский // Открытые системы. – 2012. – № 09.

4. Cisco ACI – досвід "Укртелеком" [Електронний ресурс] – Режим доступу до ресурсу: https://it-integrator.ua/project/cisco-aci-dosvid-ukrtelekom.

5. SDN Revolution Started It All [Електронний ресурс] – Режим доступу до ресурсу: https://opennetworking.org/sdn-definition/

6. Сетевая технология OpenFlow (SDN) / Семенов Ю.А. – ИТЭФ-МФТИ, 2014.

7. Download/Get Started With Mininet [Електронний ресурс] / B.Lantz, B. Heller, N. Handigol, V. Jeyakumar – Режим доступу до ресурсу: http://mininet.org/download.

8. Mininet Walkthrough [Електронний ресурс] / B.Lantz, B. Heller, N. Handigol, N. Jeyakumar – Режим доступу до ресурсу: http://mininet.org/download.

9. OpenFlow Tutorial [Електронний ресурс] – Режим доступу до ресурсу: http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial.

10. Mininet Examples [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/mininet/mininet/tree/master/examples.