

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

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

“___” _____ червня 2021__ р.

**Дипломний проєкт
на здобуття ступеня бакалавра**

по спеціальності

123 «Комп'ютерна інженерія»

на тему: Комп'ютерні засоби моделювання програмно-конфігуровних SDN-мереж

Виконала:

студентка IV курсу, групи КВ-71

Усік Анна Дмитрівна

Керівник доцент кафедри СПіСКС, к.т.н., доцент Орлова М.М

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М.

Рецензент професор кафедри ОТ, д.т.н., проф. Сімоненко В.П.

Засвідчую, що у цьому дипломному проєкті
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____

Київ – 2021 року

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

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____Віталій РОМАНКЕВИЧ

«__20__» жовтня 2020р.

**ЗАВДАННЯ
на дипломний проєкт студента**

Усік Анни Дмитрівни

1. Тема проєкту

Комп'ютерні засоби моделювання програмно-конфігуровних SDN-мереж

керівник проєкту доцент кафедри СПіСКС, к.т.н., доцент Орлова М.М,

затверджені наказом по університету від «25» _05_ 2021 р. №1331-С

2. Термін подання студентом проєкту

3. Вихідні дані до проєкту

Див. ТЗ

4. Зміст пояснювальної записки

- Аналіз існуючих мережевих технологій
- Принцип роботи технології SDN мереж
- Моделювання мережі

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

Структурна схема топології SDN мережі. Структурна схема; Алгоритм моделювання SDN мережі. Схема алгоритму; Архітектура технології SDN мережі. Структурна схема; Алгоритм обробки пакетів OpenFlow комутатором. Схема Алгоритму; Презентація.

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
	Клятченко Я.М., доцент каф. СПіСКС, к.т.н., доц.		

7. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
	Розроблення та узгодження технічного завдання	17.11.2020	
	Установка необхідного програмного забезпечення	08.02.2021	
	Розробка топології	14.04.2021	
	Написання конфігураційних файлів	01.05.2021	
	Моделювання мережі	03.05.2021	
	Тестування	05.05.2021	
	Формування звіту дипломного проєкту	15.05.2021	
	Попередній перегляд проєкту на кафедрі	25.05.2021	

Студент _____

Анна УСІК _____

Керівник проєкту _____

Марія ОРЛОВА _____

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (58 с., 28 рис, 2 додатки).

Об'єктом даного дипломного проєкту є засоби дослідження комп'ютерних засобів моделювання програмно-конфігуровних SDN-мереж. Метою проєкту є симуляція роботи мережі SDN та дослідження принципу технології допомогою емулятора MiniNet.

В бакалаврському проєкті виконано:

- аналіз недоліків та проблем класичних комп'ютерних мереж;
- опис принципів функціонування мереж технології SDN;
- аналіз існуючих засобів моделювання мереж;
- розробку структури топології комп'ютерно-конфігуровної мережі;
- налаштування характеристик мережі;
- тестування працездатності розробленої комп'ютерно-конфігуровної мережі.

За допомогою системи моделювання можна проводити наукові дослідження та експерименти стосовно роботи різноманітних топологій, проєктувати та досліджувати поведінку майбутніх мереж запобігаючи помилок високої вартості. Також цей бакалаврський проєкт можна використовувати у якості методичного матеріалу для виконання лабораторних робіт курсу «Комп'ютерні мережі» щодо проєктування SDN-мереж.

Ключові слова: комп'ютерні мережі, мережі SDN, маршрутизатори, протокол Open Flow, контролер, програмно-конфігуровні мережі.

ABSTRACT

Qualification work includes an explanatory note (58 p., 28 fig., 2 appendices).

The object of this diploma project is tools for research of computer tools for modeling software-defined SDN-networks. The purpose of the project are simulation of SDN network operation and study of the principle of technology using the Mininet emulator.

In the bachelor's project was performed:

- analysis of the disadvantages and problems of classical computer networks;
- description of the principle of operation of the technology of software-defined networks;
- analysis of existing network modeling tools;
- development of the structure of the topology of the software-defined network;
- setting network characteristics;
- efficiency testing of the developed software-defined network.

With the help of the modeling system it is possible to conduct research and experiments on the working of various topologies, to design and study the behavior of future networks preventing high-cost errors. This bachelor's project can also be used as a methodological material for laboratory work in the course "Computer Networks" on the design of SDN-networks.

Keywords: computer networks, SDN networks, routers, Open Flow protocol, controller, software-defined networks.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	A4	ІАЛЦ.045492.002 ТЗ	Комп'ютерні засоби моделювання програмно-конфігуровних SDN мереж	4		
			Технічне завдання			
	A4	ІАЛЦ.045492.003 ТП	Комп'ютерні засоби моделювання програмно-конфігуровних SDN мереж	2		
			Відомість технічного проекту			
	A4	ІАЛЦ.045492.004 ПЗ	Комп'ютерні засоби моделювання програмно-конфігуровних SDN мереж	59		
			Пояснювальна записка			
	A4	ІАЛЦ.045492.005 Д1	Структурна схема топології SDN мережі	1		
			Структурна схема			

					ІАЛЦ.045492.001 ОА									
Змін.	Арк.	№ докум.	Підпис	Дата										
Розробив	Усік А.Д				Комп'ютерні засоби моделювання програмно-конфігуровних SDN-мереж					Літ.	Аркуш	Аркушів		
Перевірив	Орлова М.М											1	2	
Консульт.										КПІ ім. Ігоря Сікорського, ФПМ КВ-71				
Н. контроль	Клятченко Я.М.													
Зав. каф.	Тарасенко В.П.													
					Опис альбому									

[illegible]

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ.....	2
2. ПІДСТАВА ДЛЯ РОЗРОБКИ.....	2
3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ.....	2
4. ДЖЕРЕЛА РОЗРОБКИ.....	2
5. ТЕХНІЧНІ ВИМОГИ.....	3
5.1. Вимоги до програмного продукту, що розробляється.....	3
5.2. Вимоги до апаратного забезпечення.....	3
5.3. Вимоги до програмного забезпечення.....	3
6. ЕТАПИ РОЗРОБКИ.....	4

					ІАЛЦ. 467200.002 ТЗ				
Зм	Лист	№ докум.	Підп.	Дата					
Розроб.	Усік А.Д.				Комп'ютерні засоби моделювання програмно-конфігуровних SDN-мереж Технічне завдання				
Перев.	Орлова М.М								
Н. контр.	Клятченко				НТУУ «КПІ ім. Ігоря Сікорського», ФПМ, КВ- 71				
Затв.	Тарасенко								

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Комп'ютерні засоби моделювання програмно-конфігуровних SDN-мереж».

Галузь застосування: комп'ютерні мережі.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є моделювання роботи технології SDN мереж, що може розглядатись як дослідження поведінки та принципу роботи нової технології, проведення експериментів на базі емулятору мереж Mininet. Також може розглядатися як методичні вказівки на виконання лабораторних робіт з курсу «Комп'ютерні мережі».

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами інформації є публікації та статті у наукових виданнях, технічна документація, патенти, форуми стосовно питань даної області, електронні ресурси та вебсервіс GitHub.

					<i>ІАЛЦ.467200.002 ТЗ</i>	Лист
						9
<i>Зм</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Підп.</i>	<i>Дата</i>		

5. ТЕХНІЧНІ ВИМОГИ

5.1.Вимоги до програмного продукту, що розробляється

- Стабільна робота створеної мережі.
- Робоча топологія мережі.
- Можливість додавання нових конфігураційних файлів.
- Можливість відслідкувати та проаналізувати трафік .

5.2.Вимоги до апаратного забезпечення

- Оперативна пам'ять: 4 Гб.
- Наявність доступу до мережі Wi-Fi (IEEE 802.11 a/b/g/n).
- Дисковий простір: 30 Гб.

5.3.Вимоги програмного забезпечення

- Операційна система Windows.
- Програмне забезпечення віртуалізації VirtualBox.
- Емулятор мереж Mininet.
- Модуль Python 3.

					ІАЛЦ.467200.002 ТЗ	Лист
						3
Зм	Лист	№ докум.	Підп.	Дата		

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів
1	Обрання предметної області та розбір теоретичної частини	12.11.2020
2	Вивчення літератури за тематикою роботи	02.02.2021
3	Розроблення та узгодження технічного завдання	17.03.2021
4	Установка необхідного програмного забезпечення	08.04.2021
5	Розробка топології	14.04.2021
6	Написання конфігураційних файлів	01.05.2021
7	Тестування	05.05.2021
8	Формування звіту дипломного проєкту	15.05.2021
9	Попередній перегляд проєкту на кафедрі	25.05.2021

Зм	Лист	№ докум.	Підп.	Дата

ІАЛЦ.467200.002 ТЗ

Лист

4

[illegible]

Пояснювальна записка до дипломного проєкту

на тему: «Система моделювання програмно-конфігуровних SDN-мереж»

Київ – 2021 року

ЗМІСТ.....	1
ПЕРЕЛІК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ	2
ВСТУП	4
1. Аналіз існуючих технологій, виділення проблем та актуальність обраної теми	5
1.1 Модель OSI	5
1.2 Принцип роботи маршрутизації.....	11
1.3 Аналіз проблем класичних мереж	15
2. Принцип роботи технології SDN	17
2.1 Підстава для появи SDN мереж, їх розвиток.	17
2.2 Принцип роботи SDN мереж.....	24
2.3 OpenFlow.....	32
3. Моделювання SDN мереж.....	37
3.1 Існуючі системи моделювання мереж	37
3.2 MiniNet	41
3.2.1 Установка	42
3.2.2 Створення топологій у середовищі MiniNet	45
3.2.3 Конфігурація параметрів мережі	49
3.2.4 Контролери, їх види	49
3.2.5 Результат роботи та захоплення трафіку у Wireshark.....	53
ВИСНОВКИ.....	56
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	57

ДОДАТКИ

Додаток 1. Копії графічного матеріалу

ІАЛЦ.045492.005 Д1 Структурна схема топології SDN мережі .Структурна схема

ІАЛЦ.045492.006 Д2 Алгоритм моделювання SDN мережі. Схема алгоритму

ІАЛЦ.045492.007 Д3 Архітектура технології SDN мережі. Структурна схема

ІАЛЦ.045492.008 Д4 Алгоритм обробки пакетів OpenFlow комутатором. Схема алгоритму

Додаток 2. Презентація

					ІАЛЦ. 045480.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		1

ПЕРЕЛІК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

API Application Programming Interface – програмний інтерфейс додатку

ARP - Address Resolution Protocol - протокол визначення адреси

ARPAnet - Advanced Research Projects Agency Network - Мережа Агентства передових досліджень — мережа, яку вважають початком Інтернету.

ASCII - American standard code for information interchange - назва таблиці (кодування, набору)

BGP - Border Gateway Protocol - протокол динамічної маршрутизації

DNS - Domain Name System система доменних імен

DoS-attack - Denial of Service attack - відмова в обслуговуванні, напад на

FIND - Future Internet Design - це загальний термін для дослідницької діяльності щодо нових архітектур для Інтернету.

FIRE - Future Internet Research and Experimentation Initiative – ініціатива майбутніх досліджень та експериментів в Інтернеті

ForCES - Forwarding and Control Element Separation - переадресація та розділення елементів управління

GENI - Global Environment for Network Innovations - Глобальне середовище для мережових інновацій

HD - High Density - формат високої щільності запису інформації

HTTP - HyperText Transfer Protocol — протокол передачі гіпертексту

IETF (Internet Engineering Task Force) - відкрите міжнародне співтовариство проектувальників, учених, мережових операторів і провайдерів

IP - Internet Protocol – протокол мережевого рівня

ISO - International Organization for Standardization – Міжнародна організація стандартизації

ISP - Internet Service Provider - організація, що надає користувачам доступ до мережі Інтернет та пов'язані з цим послуги

					ІАЛЦ. 045480.004 ПЗ	Арк.
						2
Змін.	Арк.	№ докум.	Підпис	Дата		

LLC - Logical Link Control - контроль логічного зв'язку

MAC - Media Access Control - контроль доступу до середовища

Model OSI - The Open Systems Interconnection model – мережева модель стека мережевих протоколів

MPLS - Multiprotocol Label Switching — багатопроTOCOLьна комутація за мітками

NFV - Network Functions Virtualization – віртуалізація мережевих функцій

PCE - Path Computation Element - це системний компонент, додаток або мережевий вузол, здатний визначати та знаходити підходящий маршрут для передачі даних між джерелом та пунктом призначення

PDU - Protocol Data Units – блок даних протоколу

QoS - Quality of Service - технологія надання різних класів трафіку різних пріоритетів в обслуговуванні

RCP - Routing Control Platform - логічно централізована платформа, окрема від площини переадресації IP

SDN – Software-defined Networking – програмно-конфігуровні мережі

TCP - Transmission Control Protocol - Протокол керування передачею

TTL - Time to live - граничний період часу або число ітерацій або переходів, за який набір даних може існувати до свого зникнення

UDP - User Datagram Protocol - протокол передачі дейтаграм

UTF-8 - Unicode Transformation Format, 8-bit - формат перетворення Юнікоду

VLAN - Virtual Local Area Network - віртуальна локальна комп'ютерна мережа

VPN - Virtual Private Network — віртуальна приватна мережа

комп'ютерну систему з наміром зробити комп'ютерні ресурси недоступними

					ІАЛЦ. 045480.004 ПЗ	Арк.
						3
Змін.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Сьогоднішній день неможливо уявити без глобальної мережі Інтернет. У сторіччя масової диджиталізації якісний доступ до виходу у мережу є просто необхідною річчю для кожного з нас. З кожним роком кількість користувачів стрімко зростає, зростає також і кількість підключених хостів, адже на кожного користувача загалом припадає 2-3 девайси. Це все призводить до збільшення трафіку.

Особливо зараз, у часи масштабної пандемії і карантинів, коли багато сфер життя знайшли нове життя у віртуальному світі. Освіта, кіно, медицина та багато інших звичних нам речей тепер існують у цифровому форматі: запис до лікаря онлайн у мобільному додатку, лекції та уроки у форматі відеоконференцій, перегляд фільмів вдома на стримінгових платформах і ще безліч таких прикладів.

Забезпечення обробки великих даних, стабільної їх передачі та підтримки користувачів вимагає все більшої пропускнуєї спроможності, високих швидкостей маршрутизації та легшої конфігурації вузлів. Вже на сьогодні класична структура організації мережі не може дати потрібних ресурсів і часто стає перешкодою на шляху розвитку технологій обчислювальної інфраструктури. Саме тому останні декілька років у спільноті мережеских технологій йдуть дискусії щодо програмно-конфігуровних SDN-мереж, які потенційно можуть вирішити ці проблеми. Розглянемо особливості побудови мереж SDN.

					ІАЛЦ. 045480.004 ПЗ	Арк.
						4
Змін.	Арк.	№ докум.	Підпис	Дата		

1. Аналіз існуючих технологій, виділення проблем та актуальність обраної теми

1.1 Модель OSI

Перш за все, щоб розібратися з новою технологією та зрозуміти чому нинішня технологія вже не справляється з навантаженням, потрібно розглянути як влаштована робота класичних мереж, а також їх маршрутизація.

На теперішній день мережа складається з безліч різноманітних елементів: маршрутизатори, комутатори, міжмережеві екрани та багато інших. І кожен з них окремий, незалежний та, як правило, досить складний прилад. Для забезпечення гарної роботи мережі, потрібно щоб кожен елемент міг взаємодіяти з іншими та був правильно налаштований. Для цього на допомогу приходить модель OSI .

Мережева модель OSI (англ. Open systems interconnection basic reference model) - концептуальна модель, яка узагальнює і стандартизує представлення засобів мережевої взаємодії в телекомунікаційних і комп'ютерних системах, незалежно від їх внутрішнього устрою і використовуваних технологій. Модель OSI була розроблена в 1984 році Міжнародною організацією стандартизації (ISO). Основною метою її створення був пошук рішення проблеми несумісності пристроїв, що використовують різні комунікаційні протоколи, шляхом переходу на єдиний, загальний для всіх систем стек протоколів.[1]

Модель OSI представляє у вигляді абстрактної семирівневої моделі мережевої взаємодії. На кожному з рівнів працюють певні набори спеціалізованих протоколів, які забезпечують взаємодію на своєму рівні та з сусідніми рівнями (на одиницю вище/нижче).

Також рівні характеризуються PDU - блоком даних протоколу, якими вони оперують.

					ІАЛЦ. 045480.004 ПЗ	Арк.
						5
Змін.	Арк.	№ докум.	Підпис	Дата		

На цьому рівні інформація представляється у вигляді даних, проте з кожним рівнем вона проходить інкапсуляцію і вже на першому рівні вона представлена у вигляді бітів. Інкапсуляція – процес переходу інформації з даних у біти, декапсуляція – обернений процес до інкапсуляції, коли інформація перетворюється з бітів у вигляд даних. Як ці процеси протікають відносно моделі OSI продемонстровано на рис.1.

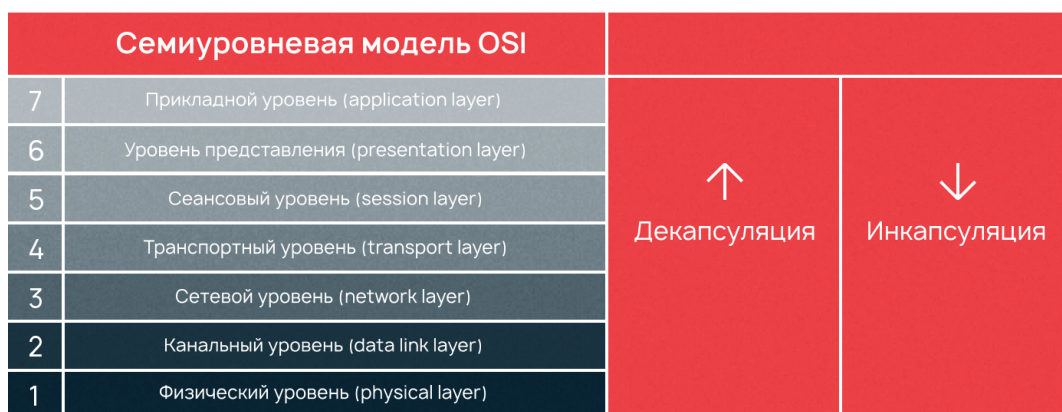


Рисунок 1 – Процеси інкапсуляції та декапсуляції відповідно моделі OSI

Проте інформація не перетворюється одразу з даних у біти, а проходить шлях через кожен рівень і представляється у вигляді певного типу даних, які оброблює кожен з них. Більш детально можна це побачити на рис.2.

Семиуровневая модель OSI		PDU
7	Прикладной уровень (application layer)	Данные
6	Уровень представления (presentation layer)	
5	Сеансовый уровень (session layer)	
4	Транспортный уровень (transport layer)	Сегмент, Датаграмма
3	Сетевой уровень (network layer)	Пакет
2	Канальный уровень (data link layer)	Кадр
1	Физический уровень (physical layer)	Бит

Рисунок 2 – Відповідність PDU до рівнів моделі OSI

Розглянемо кожен з рівнів детальніше.

Фізичний рівень (Physical layer, L1)

Фізичний рівень відповідає за перетворення фізичних сигналів у біти та їх відправку і прийняття. Сигнали передаються через певне фізичне середовище, на даний момент ми використовуємо три типи:

- оптичне волокно (fiber);
- мідний кабель (copper);
- бездротове середовище (wireless).

Для кожного середовища відповідає свій тип сигналу: для оптоволокна це імпульси світла, для мідного кабелю – електричний імпульс, а для бездротового середовища – радіохвилі.

Основні функції фізичного рівня:

- фізичне кодування даних;
- побітова передача даних;
- LLC Error control: Визначення (і іноді виправлення) помилок за допомогою контрольних сум;
- фізичні компоненти.

На першому рівні працюють такі фізичні компоненти, як: концентратори, повторювачі (ретранслятори) сигналу, конектори, через які передаються сигнали.

Канальний рівень (Data Link layer, L2)

Канальний рівень відповідає за передачу даних між вузлами у локальній мережі, за адресацію при передачі даних та за контроль можливих помилок. З рис.2 ми можемо бачити що канальний рівень оперує таким типом даних як кадр (або фрейм). З фізичного рівня він отримує дані у вигляді бітів, упаковує їх у кадри, перевіряє на помилки та виправляє їх при наявності і далі передає на L3.

Канальний рівень складається з двох підрівнів:

- MAC;
- LLC.

MAC підрівень відповідає за присвоєння MAC-адрес , визначення початку і кінця фрейму.

LLC підрівень відповідає за управління передачі даних, логічного з'єднання, перевіркою даних та їх виправленням.

На канальному рівні працюють комутатори(switch).

Мережевий рівень (Network layer, L3)

Мережевий рівень відповідальний за маршрутизацію. Маршрутизація це пошук оптимального маршруту передачі даних. Також на цьому рівні здійснюється перетворення логічних адрес (IP-адреси) у фізичні (MAC-адреси) за допомогою протоколу ARP, тобто виконується інкапсуляція та декапсуляція пакетів, упорядкування, сегментування та об'єднання пакетів.

PDU на даному рівні представлено у вигляді пакетів (packets).

На третьому рівні працюють маршрутизатори, або ж більш розповсюджено роутери (routers).

Транспортний рівень (Transport layer, L4)

Найголовнішою функцією транспортного рівня є доставка даних з різним типом надійності. На цьому рівні використовується два типи PDU, в залежності від необхідної надійності. Це датаграма та сегмент.

Датаграми використовуються протоколом UDP, коли забезпечення надійної цілісної доставки не є першим пріоритетом. Зазвичай використовуються при передачі відео та аудіо матеріалу, де втрата частини даних не є критичною.

					ІАЛЦ. 045480.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		8

При передачі даних протоколом TCP пакет ділиться на сегменти. Відмінність протоколу TCP від UDP полягає в тому, що TCP гарантує надійну доставку цілісної інформації, без втрат одного з сегменту.

Потрібно зауважити що датаграма та сегмент це частини пакету, на які він може ділитися або бути цілком.

Транспортний рівень можна назвати посередником між двома умовними групами, на які можна поділити всю модель OSI (рис.3):

- Media layers (рівні середовища);
- Host layers (рівні хостів).

Рівні середовища відповідають за передачу інформації, зазвичай для кожного рівня є свої мережеві прилади, такі як комутатори, маршрутизатори та інші. В основному обслуговуються та проектується мережевими інженерами.

Рівні хостів як правило використовуються на кінцевих пристроях користувачів, такі як стаціонарні комп'ютери, мобільні пристрої, ноутбуки і т.д. Розроблюються в більшості програмістами, інженерами програмного забезпечення. Транспортний рівень є ніби то посередником між ними двома.

Семиуровневая модель OSI			
7	Прикладной уровень (application layer)	Host layers	
6	Уровень представления (presentation layer)		
5	Сеансовый уровень (session layer)		
4	Транспортный уровень (transport layer)		
3	Сетевой уровень (network layer)	Media layers	
2	Канальный уровень (data link layer)		
1	Физический уровень (physical layer)		

Рисунок 3 – Відповідність рівнів середовища та хостів до рівнів моделі OSI

Сеансовий рівень (Session layer, L5)

Сеансовий рівень є відповідальним за встановлення з'єднання, підтримки сеансу зв'язку та його закінчення, надає можливість синхронізації задач, обміну інформацією. Цей рівень дозволяє додаткам та застосуванням підтримувати взаємодію один з одним протягом тривалого часу. Гарним прикладом роботи сеансового рівня є відеоконференції, коли потрібно встановити сеанс з'єднання між двома користувачами, підтримувати зв'язок, забезпечити синхронність та в кінці роз'єднати їх.

На цьому рівні також визначається тип передачі даних. Їх існує два:

- дуплексний – інформація передається в обидві сторони(прийом та передача одночасно);
- напівдуплексний – інформація передається тільки в одну сторону(тільки прийом або тільки передача).

Рівень представлення (Presentation layer, L6)

Рівень представлення має функції перетворення формату даних у зрозумілий для людини або машини відповідно. Дані отримані з мережі він перетворює у зрозумілі для програм та навпаки. Також усуває проблему несумісності між пристроями, наприклад коли один пристрій може відображати лише один тип кодування, наприклад UTF-8, а інший тільки ASCII. Трансформування з одного кодування на інше виконується саме цим рівнем.

Також на цей рівень займається представленням відео та аудіо форматом (MPEG, QuickTime, WAV, MP3), а також картинками (JPEG, GIF, PNG). Зокрема представлені такі функції, як шифрування даних, стиснення та розпакування даних.

Рівень застосувань (Application layer)

Сьомий рівень можна вважати набором інтерфейсів для взаємодії користувача з мережею. Він надає графічний інтерфейс, доступ до файлової системи, баз даних та інші функції додаткам користувача. Також він відповідає за передачу службової інформації, надає додаткам інформацію про помилки і формує запити до рівня представлення.

1.2 Принцип роботи маршрутизації

Маршрутизація (англ. *Routing*) — процес визначення маршруту прямування інформації між мережами. Маршрутизатор (або роутер від англ. *router*) приймає рішення, що базується на IP-адресі отримувача пакету. Для того, щоб переслати пакет далі, всі пристрої на шляху слідування використовують IP-адресу отримувача. Для прийняття правильного рішення маршрутизатор має знати напрямки і маршрути до віддалених мереж. [2]

Існує два види маршрутизації:

- статична;
- динамічна.

У разі статичної маршрутизації адміністратор власноруч прописує шлях по мережі до пункту призначення.

Щодо динамічної маршрутизації, то в такому випадку маршрут прокладається за допомогою протоколів, таблиць та алгоритмів маршрутизації.

Маршрутизація здійснюється за допомогою такого пристрою, як маршрутизатор. Розглянемо принцип його роботи більш детально.

Маршрутизатором називається електронний пристрій який працює на третьому рівні моделі OSI та є вузлом між двома або декількома мережами, та

					ІАЛЦ. 045480.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		11

його основною функцією є побудова найоптимальнішого маршруту передачі даних по мережі. Зазвичай мають такий вигляд (рис. 4) та (рис.5)



Рисунок 4 – Зовнішній вигляд маршрутизатора

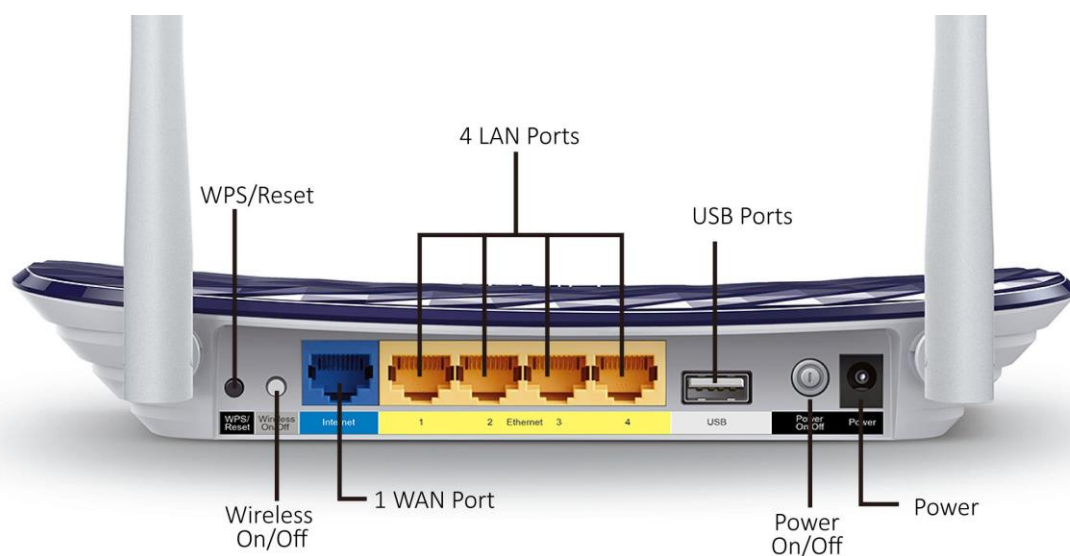


Рисунок 5 – Зворотній вигляд маршрутизатора

Принцип роботи маршрутизатора базується на таблицях маршрутизації.

					ІАЛЦ. 045480.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		12

Таблиця маршрутизації являє собою певну базу даних або ж електронну таблицю, що описує правила поведінки обробки пакету (перенаправлення, відкидування та інше) та відповідності між адресою призначення та інтерфейсом, на який потрібно відправити пакет для подальшого відправлення.

Вона містить у собі такі поля:

- адресу мережі призначення;
- маску мережі;
- адресу шлюзу або шлюз;
- інтерфейс;
- метрику.

Приклад однієї з таких таблиць наведено на рис.6.

Network Address	Netmask	Gateway Address	Interface	Metric
127.0.0.0	255.0.0.0	127.0.0.1	127.0.0	1
0.0.0.0	0.0.0.0	198.21.17.7	198.21.17.5	1
56.0.0.0	255.0.0.0	213.34.12.4	213.34.12.3	15
116.0.0.0	255.0.0.0	213.34.12.4	213.34.12.3	13
129.13.0.0	255.255.0.0	198.21.17.6	198.21.17.5	2
198.21.17.0	255.255.255.0	198.21.17.5	198.21.17.5	1
198.21.17.5	255.255.255.255	127.0.0.1	127.0.0.1	1
198.21.17.255	255.255.255.255	198.21.17.5	198.21.17.5	1
213.34.12.0	255.255.255.0	213.34.12.3	213.34.12.3	1
213.34.12.3	255.255.255.255	127.0.0.1	127.0.0.1	1
213.34.12.255	255.255.255.255	213.34.12.3	213.34.12.3	1
224.0.0.0	224.0.0.0	198.21.17.6	198.21.17.6	1
224.0.0.0	224.0.0.0	213.34.12.3	213.34.12.3	1
255.255.255.255	255.255.255.255	198.21.17.6	198.21.17.6	1

Рисунок 6 – Таблиця маршрутизації

Розглянемо безпосередньо як працює маршрутизація на прикладі маршруту одного пакету.

Нам потрібно відправити пакет, наприклад на якийсь запит до HTTP сервера або DNS. Проте запит у чистому вигляді ми відправити не можемо,

його потрібно інкапсулювати. В нашому випадку запит логічніше обернути в UDP датаграму. Далі він упаковується в IP-пакет, з логічними адресами відправника та отримувача. Далі пакет потрібно обернути в кадр Ethernet, проте в його заголовку повинні міститися MAC адреси, зокрема отримувача, яку ми не знаємо. Далі IP-модуль порівнює логічні адреси і з'ясовує чи знаходиться місце призначення в даній мережі чи ні. Протокол Ethernet може доставляти кадри тільки тим адресатам, які розташовані в межах однієї локальної мережі з відправником. У випадку, якщо отримувач розташований поза локальної мережі, кадр відправляється на маршрутизатор для подальшої маршрутизації (IP адресу маршрутизатора відомо по замовчання, а MAC адресу дізнаємось за допомогою протоколу ARP).

Кадр Ethernet приймається одним з інтерфейсів маршрутизатора. Той аналізує його на помилки, при наявності знищує пакет, в іншому випадку відкидає кінцевик та заголовок фрейму та передає пакет до протоколу IP. В свою чергу протокол виділяє з пакету адресу отримувача і починає аналізувати всі записи у своїй таблиці маршрутизації. Є три можливі варіанти результату пошуку по таблиці маршрутизації.

1. Потрібний маршрут знаходиться у таблиці та пакет надсилається у пункт призначення.
2. Повністю маршрут відсутній у таблиці, проте знаходиться запис який містить адресу мережі призначення.
3. Відповідних записів не знайдено.

З першим варіантом все зрозуміло, давайте розглянемо два інші.

У останньому випадку маршрутизатор формує запит до інших маршрутизаторів чи є в них інформація в таблицях щодо адреси призначення і передає кадр на той маршрутизатор, який відгукнувся.

У другому випадку, якщо в таблиці знайшовся запис з адресою мережі, то розглянемо детальніше цей запис на прикладі рис.6. У першій колонці під назвою Network Address розташовується наша адреса мережі, далі

					ІАЛЦ. 045480.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		14

маршрутизатор з колонки Interface розуміє що всі пакети призначені для даної мережі потрібно надсилати на свій інтерфейс, який вказаний, а у колонці Gateway Adress вказаний інтерфейс наступного маршрутизатора, на який надійде пакет далі.

Таким чином, пакет передається мережею, поки не дійде до свого місця призначення.

1.3 Аналіз проблем класичних мереж

Розглянувши принцип роботи можна зрозуміти, що дана технологія досить складна. На даний момент мережа складається з безліч окремих незалежних пристроїв, кожен з яких досить складний та дорогий в обслуговуванні. На кожному з рівнів середовища моделі OSI зазвичай працює спеціалізований окремий прилад та дуже велика кількість різноманітних протоколів на усіх. І зазвичай з кожним з елементів мережі потрібно працювати окремо. Кожен з вузлів конфігурується власноруч адміністратором через командний рядок або веб інтерфейс, що є досить витратною роботою, адже зазвичай мережа є досить великою. Також є великий шанс помилки, адже людський фактор ніхто не відміняв.

Також розглянувши роботу маршрутизації можна сказати, що цей принцип не є досконалим. Велику проблему складає те, що кожен маршрутизатор не бачить цілісної картини мережі, навіть не дивлячись на те, що вони періодично обмінюються інформацією, до яких мереж під'єднані їх сусіди.

Можна зрозуміти, що кожен маршрутизатор виконує складні функції що призводить до того, що він є доволі дорогим пристроєм. Потрібно зауважити, що з часом мережа росте та потребує все більшої обчислювальної спроможності, що також ускладнює ситуацію, адже кожен раз потрібно або купувати новий пристрій, або модернізувати старий, що є дуже затратним.

					ІАЛЦ. 045480.004 ПЗ	Арк.
						15
Змін.	Арк.	№ докум.	Підпис	Дата		

Ще однією серйозною проблемою є несумісність приладів від різних постачальників. Це добре для бізнесу виробників, адже компаніям увесь час потрібно купувати та модернізувати техніку. Проте це є сильним стримуючим фактором у розвитку мережевих технологій.

Отже, підсумовуючи можна виділити такі основні проблеми:

- статичність класичних мереж;
- складне конфігурування та обслуговування мережі;
- неможливість контролю та прогнозування можливих проблем;
- відсутність віртуалізації всієї мережі цілком;
- несумісність обладнання між виробниками;
- висока вартість мереж.

Через ці перешкоди класичні мережі перестали задовольняти потреби швидко зростаючого бізнесу.

На відповідь кризі мереж прийшла принципово нова технологія, яка обіцяє кардинально інший підхід до побудови мереж – SDN.

Розглянемо особливості побудови і функціонування програмно-конфігуровних мереж SDN.

SDN (Software-defined Networking) – програмно-конфігуровні мережі.

Технологія ПКС націлена на: підвищення ефективності використання пропускної здатності каналів, балансування навантажень в мережі; спрощення управління мережею, підвищення масштабованості мережі; підвищення безпеки мереж; ефективну маршрутизацію; зниження капітальних витрат [3].

Це досить нова технологія яка ще тільки розвивається, проте на неї вже покладено багато надій та сподівань. Чи все так просто, як здається, чи зможе нова технологія дійсно вирішити існуючі проблеми? Як взагалі влаштована робота нового підходу?

Актуальність цього питання дуже висока на даний момент, тому дослідимо цю тему, розберемось з принципом її побудови та зробим висновки про її спроможність.

					ІАЛЦ. 045480.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		16

2. Принцип роботи технології SDN

2.1 Підстава для появи SDN мереж, їх розвиток

Ефективне впровадження нових підходів в межах існуючої концепції побудови мереж є не простою задачею. Адже мережі протягом останніх років концептуально майже не змінювалися – зростала швидкість і з'являлися нові протоколи, але принципи управління і передачі трафіку практично не змінювалися.

Концепція програмно-конфігуровних мереж (Software Defined Networking (SDN)) суттєво змінює принципи функціонування мереж і їх управління. Адже у швидко змінному сучасному світі саме мережі передачі даних названі «тонкою ланкою», яка обмежує зростання продуктивності додатків по мірі зростання кількості мобільних користувачів, масштабування віртуальних середовищ, формування кластерів для Big Data.

У SDN мережах завдання комутації трафіку і завдання управління строго розділені. Весь процес управління централізується і передається контролеру. Таким чином, комутатор у концепції SDN це досить примітивний пристрій, який відповідає тільки за перемикання пакетів на підставі дуже простих правил. Контролер SDN управляє всіма комутаторами в мережі і програмує кожен з них для правильної передачі трафіку.

Зважаючи на вищезазначене, централізація логіки управління на основі технології SDN, дозволяє програмувати мережу, як єдине ціле й, при цьому, спростити операційну модель великих корпоративних мереж, що занадто статичні на даний час та не відповідають вимогам сучасного бізнесу, із властивими йому мобільністю користувачів, пристроїв, додатків, розподілом додатків між віртуальними машинами і інтенсивним обміном даними.

Таким чином, метою роботи є дослідження особливостей, принципів роботи та архітектури SDN мереж.

Незважаючи на те, що концепція SDN стала поширеною в останні роки, сама ідея не зовсім нова і еволюціонує вже більше 20 років. Її відбитки можна простежити навіть у розвитку ранніх телефонних мереж на базі комутації каналів, коли управління мережею було відокремлено від мережі каналної комутації мовного трафіку. Це було зроблено з тією ж метою, що і в SDN – щоб спростити управління і введення нових послуг. Концепція так званих «Програмних комутаторів» Softswitch для телекомунікаційних мереж на базі комутації пакетів також дуже близька до SDN за функціями і реалізацією.

Історію SDN можна розділити на три етапи [4]:

- «активні мережі» (середина 90-х – початок 2000-х років), коли в мережі були введені функції програмування;
- розподіл площин управління і передачі даних (приблизно з 2001 по 2007 р.), коли були розроблені відкриті інтерфейси між цими площинами;
- розробка інтерфейсу API OpenFlow і мережевих операційних систем (з 2007 до приблизно 2010 р.), коли й концепція SDN набула широкого поширення.

При цьому окремо варто розглянути історію віртуалізації мережі, яка пронизує всі три етапи історії SDN і є одним з найважливіших застосувань цієї технології.

Активні мережі. На початку 90-х років стрімко почав розвиватися Інтернет, і з'явилося багато нових додатків, крім передачі файлів і електронної пошти, на які орієнтувався ранній Інтернет. Різноманітність програм викликала розробку нових мережевих сервісів і нових протоколів передачі даних для них. Ці протоколи стандартизувалися в організації IETF (Internet Engineering Task Force). Процес стандартизації був складним і довгим, що часто викликало роздратування розробників. Зважаючи на це, багато

дослідників і розробники стали дотримуватися альтернативного підходу, відмінного від того, який був закладений в Інтернет спочатку.

Прообразом Інтернету в той час виступала мережа військового призначення ARPAnet, основоположною ідеєю якої була можливість збереження працездатності мережі в умовах виходу з ладу як завгодно великої її частини. Тому, кожен мережевий елемент повинен був володіти здатністю самостійно вирішувати, яким маршрутом відправляти наступний пакет. Ця ідея була ефективна для військового застосування, проте в разі цивільного використання, та ще й при великій різноманітності сервісів, вона перетворилася на гальмо, що утруднює інновації і здорожує будівництво і обслуговування мережі.

Традиційні комп'ютерні мережі не були «програмованими». Для концепції «активних мереж» було введено поняття інтерфейсу прикладного програмування API (Application Programming Interface). Цей інтерфейс призначений для того, щоб «показувати» ресурси мережевих елементів (процесори, пам'ять, черги пакетів) на окремих мережевих вузлах для вищого рівня управління, який би через інтерфейс API керував цими ресурсами для виконання певних прикладних програм. Такий підхід не викликав великого зацікавлення в Інтернет-співтоваристві, яке відстоювало «простоту» базової мережі, наголошуючи на тому, що це – основа успіху Інтернет.

Підхід «активної мережі» розроблявся в таких організаціях, як GENI (Global Environment for Network Innovations), NSF FIND (Future Internet Design) у США і FIRE (Future Internet Research and Experimentation Initiative) в Європі. У цих інноваційних організаціях розробляли дві основні моделі програмування мереж [6]:

- «капсульну» модель, де керуючі коди були «зашиті» в пакети з даними (in-band);
- модель «програмованих комутаторів і маршрутизаторів», де використовували окремі пакети, які передавали команди управління (out-of-band).

Однак, такий підхід часто був дуже мовнозалежним (багато виробників розробляли власні версії мов мережевого програмування), вимагав окремого управління для так званих «мідлбоксів» – брандмауерів, проксі-серверів та іншого мережевого обладнання. Уже в цей час було закладено ідейні основи сьогоденішньої технології віртуалізації мережевих функцій NFV (Network Functions Virtualization), яка реалізує функції різних мережевих елементів на віртуальних машинах, що працюють на стандартних серверах.

Однак, технологія «активної мережі» володіла одним суттєвим недоліком. Передбачалося, що кінцеві користувачі будуть розробляти програми і запускати їх на серверах програм, а ті будуть відповідно налаштовувати мережі, щоб програми добре працювали. Для мережі з необмеженими ресурсами і продуктивністю це, можливо, і працювало б, але в умовах обмежених ресурсів, загальне поширення програмування через API призвело б до того, що мережеві ресурси були б узурповані вузьким колом досвідчених користувачів. Крім того, «активна мережа» не достатньо продуктивна і не забезпечує необхідного рівня безпеки.

Зважаючи на вищезазначене, було створено окремий проект Secure Active Network Environment Architecture. Технологія не отримала суттєвого поширення. У лабораторіях все працювало відмінно, але ключових актуальних задач вирішити не вдалося. За таких умов, подальші дослідження зосередилися тільки на розподілі площин управління і передачі даних.

Розподіл площин управління і передачі даних. На початку 2000-х років зростання обсягів мережевого трафіку і вимог до стабільності та продуктивності мережі вимагали кращих підходів до управління мережею, наприклад до управління медіа-потокami, що отримало назву «інжиніринг трафіку» TI (Traffic Engineering). Однак, існуючі мережеві протоколи були для цього погано пристосовані. Причина знову була в тому, що традиційні маршрутизатори і комутатори не було розділено на процеси управління і передачі пакетів даних.

Спроби розподілу площин призвели до двох основних інновацій [4]:

					ІАЛЦ. 045480.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		20

- розробки відкритих інтерфейсів між площинами управління і даних, таких як ForCES (Forwarding and Control Element Separation), стандартизованого IETF, або Netlink для Linux

- розробки протоколу управління маршрутизацією RCP (Routing Control Platform), а також архітектури «програмованих маршрутизаторів» (Soft-Router) і PCE (Path Computation Element).

Ці інновації продиктовані необхідністю управляти процесом маршрутизації на мережі Інтернет-провайдера (ISP). Вони орієнтувалися не на кінцевих користувачів, а на мережесих адміністраторів, передбачали можливість програмування на площині управління, а не на площині даних, що візуалізується, і керованість топології всієї мережі, а не конфігурацію окремих мережесих пристроїв. Прогрес в серверних технологіях уможлилював розрахунок стану всіх маршрутів і рішень по маршрутизації трафіку на всій великій мережі ISP на одному стандартному сервері.

Резервування активності можна було робити на іншому стандартному сервері – резервному, який працює паралельно із основним. Таким чином забезпечувалася надійність управління. Однак, інтерфейс ForCES не був прийнятий більшістю виробників, а в RCP використовувався існуючий протокол площини управління BGP для посилення таблиць маршрутизації в стандартні маршрутизатори. Це суттєво обмежувало керованість всієї мережі.

Спочатку чимало операторів вважали, що поділ площин є неефективним, оскільки ніхто не міг точно сказати, що буде із мережею в разі відмови центрального контролера. Крім того виробників також не зовсім влаштовувала ідея API типу ForCES, оскільки відкриті API уможлилювали проникнення на ринок невеликих компаній-конкурентів.

На основі вищезазначеного за аналогією із об'єктно-орієнтованим програмуванням появилися перші «clean-slate» архітектури й запропоновано чотири площини[4]:

- площину даних (data plane), для обробки пакетів на основі правил конфігурації;

- площину виявлення (discovery plane), для збору інформації про топології і вимірювання трафіку;
- площину поширення (dissemination plane), для встановлення правил обробки пакетів на мережевих вузлах;
- площину рішень (decision plane), що складається із логічно централізованих контролерів, які перетворюють цілі мережевого рівня в стан обробки пакетів.

До таких архітектур відносилися проекти, як приклад, SANE і Ethane. У Ethane маршрутизатори займаються тільки обробкою таблиць потоків трафіку, які роздаються з контролерів на рівні рішень. Цей проект реалізовано в науковому департаменті Стенфордського університету і заклав основи для розробки протоколу OpenFlow.

Розробка інтерфейсу API OpenFlow і мережевих операційних систем.

В середині 2000-х років група дослідників Стенфордського університету заснувала програму Clean Slate Program (програма «з чистого аркуша») і зосередилася на дослідженнях виключно кампусних мереж. До розробки OpenFlow попередники SDN завжди зустрічалися із проблемою невідповідності «всеохоплення» програмованих мереж з її практичним застосуванням [7].

У протоколі OpenFlow було більше функцій, ніж в попередніх контролерах маршрутизації, що дозволило розробляти архітектуру на існуючому комутаційному обладнанні, побудованому на комерційно доступних чіпсетах. Проте, при цьому, дещо обмежувалася гнучкість. OpenFlow можна було дуже швидко розгорнути, що дало можливість SDN стати, як практично застосовним, так і універсальним рішенням. Комутатор OpenFlow містив таблицю правил маршрутизації, де кожне правило містило [4, 7]:

- зразок (Pattern), який відповідав певним бітам в заголовку пакету;

- список дій (Actions, наприклад «відкинути» (drop), «створити розсилку» (flood), передати на певний інтерфейс, модифікувати заголовок пакету, або передати пакет на контролер);
- набір лічильників (counters), для підрахунку байтів і пакетів;
- пріоритет (priority) для усунення можливих неоднозначностей між правилами та зразками, що на них накладаються.

Після отримання чергового пакету, комутатор OpenFlow ідентифікував правило відповідності з найвищим пріоритетом, виконував передбачені дії та інкрементував лічильники.

OpenFlow – це перший протокол із розподілу площин управління і даних. У другій половині 2000-х років дослідницька група в Стенфорді, яку надихнули успіхом SDN в їх кампусній мережі, широко пропагувала зазначену концепцію і для інших кампусних мереж, причому також для мереж великого масштабу, які об'єднують кілька кампусних мереж, де й були реалізовані перші проекти SDN.

OpenFlow використовувався також в інших засобах, наприклад, в мережах дата-центрів для управління великими обсягами трафіку. До недоліків OpenFlow слід віднести відсутність підтримки глибокого аналізу пакетів (DPI) на площині передачі даних, тому OpenFlow сам по собі не міг ефективно задіяти функціональність «мідлбоксів» – допоміжного мережевого обладнання, наприклад, брандмауерів.

Подальша робота по вдосконаленню протоколу OpenFlow призвела до розробки мережевих операційних систем (ОС). Основним завданням мережевої ОС є абстрагування від встановлення станів мережі з боку логіки управління і додатків, які керують поведінкою мережі. При цьому, роботу мережі декомпонували на три рівні:

- площину даних (data plane) з відкритим інтерфейсом;
- рівень управління станами (state management layer), який відповідає за постійне відстежування стану мережі;

- рівень керуючої логіки (control logic), що продукує різні операції в залежності від стану мережі.

2.2 Принцип роботи SDN мереж

Програмно-конфігуровна мережа (SDN) – це підхід, при якому для направлення трафіку в мережі і взаємодії з базовою апаратною інфраструктурою використовуються контролери на базі програмного забезпечення або API-інтерфейси [8, 9, 6]. Така мережа відрізняється від традиційних, в яких для управління мережевим трафіком використовуються спеціальні пристрої (маршрутизатори і комутатори). SDN можна використовувати для розробки та адміністрування віртуальної мережі або для управління звичайною апаратною мережею а основі програмного забезпечення.

У той час як віртуалізація мережі забезпечує можливість розбивати одну фізичну мережу на різні віртуальні мережі або об'єднувати пристрої з різних фізичних мереж в одну віртуальну мережу, SDN пропонує новий спосіб управління маршрутизацією пакетів даних через централізований сервер.

Розглянемо базові принципи роботи SDN. У мережі SDN (як і в будь-якій віртуальній середовищі) програмне забезпечення відокремлено від апаратного. SDN розділяє дві площини мережевих пристроїв: переміщує на програмний рівень площину управління, що визначає напрямки відправлення трафіку, і залишає на апаратному рівні площину даних, яка виконує відправлення трафіку. Це дозволяє адміністратору мережі, який використовує SDN, програмувати і контролювати всю мережу за допомогою єдиної консолі управління замість управління мережевими параметрами окремих пристроїв.

Стандартна архітектура SDN складається з трьох компонентів [9]:

- додатки, що передають запити на ресурси або інформацію про мережі в цілому;

- контролери, які використовують інформацію, отриману від додатків, щоб визначати способи маршрутизації пакетів даних;
- мережеві пристрої, які отримують від контролерів інформацію про те, куди слід переносити дані.

Зазначені компоненти можуть мати різне фізичне розташування. Як фізичні, так і віртуальні мережеві пристрої переміщують дані через мережу. У деяких випадках віртуальні комутатори, які можуть бути вбудовані в програмне або апаратне забезпечення, також працюють як фізичні комутатори і об'єднують всі відповідні функції в єдиний інтелектуальний комутатор. Цей комутатор перевіряє цілісність пакетів даних і їх цільове призначення, а також переміщає пакети. Узагальнену архітектуру SDN мереж зображено на рис.7.



Рисунок 7 – Загальна архітектура SDN

Як видно із архітектури рис.7, крім класичного управління мережею прямими командами системного адміністратора до контролера, SDN контролер підтримує запуск додатків управління мережею.

Відповідно до задуму розробників, SDN дозволяє програмувати мережу як єдине ціле, а адміністраторам не доведеться займатися окремими пристроями. Головним стає контролер: він все бачить, все знає і роздає мережевим пристроям інструкції щодо обробки трафіку. Пристроєм не потрібно розбиратися у протоколах – достатньо слідувати лише за інструкціями контролера. Таким чином, пристрої можуть бути простими і дешевими.

Кожен SDN додаток – це інтерфейс оптимізації мережі під конкретний бізнес-додаток (наприклад Microsoft Lync) і його основна роль – зміна мережі в реальному часі під поточні потреби обслуговується програми. За умов Microsoft Lync – це може бути, наприклад, зміна QoS мережі між двома телефонними абонентами для передачі HD відеодзвінка в реальному часі без затримок або створення VPN тунелю між двома абонентам.

Якщо розглянути більш детально інформаційні потоки в архітектурі SDN, то можна помітити два основних напрямки обміну інформацією: перший – між SDN додатками, а другий – для управління фізичними мережевими пристроями. Компоненти структури SDN проілюстровано на рис.8.

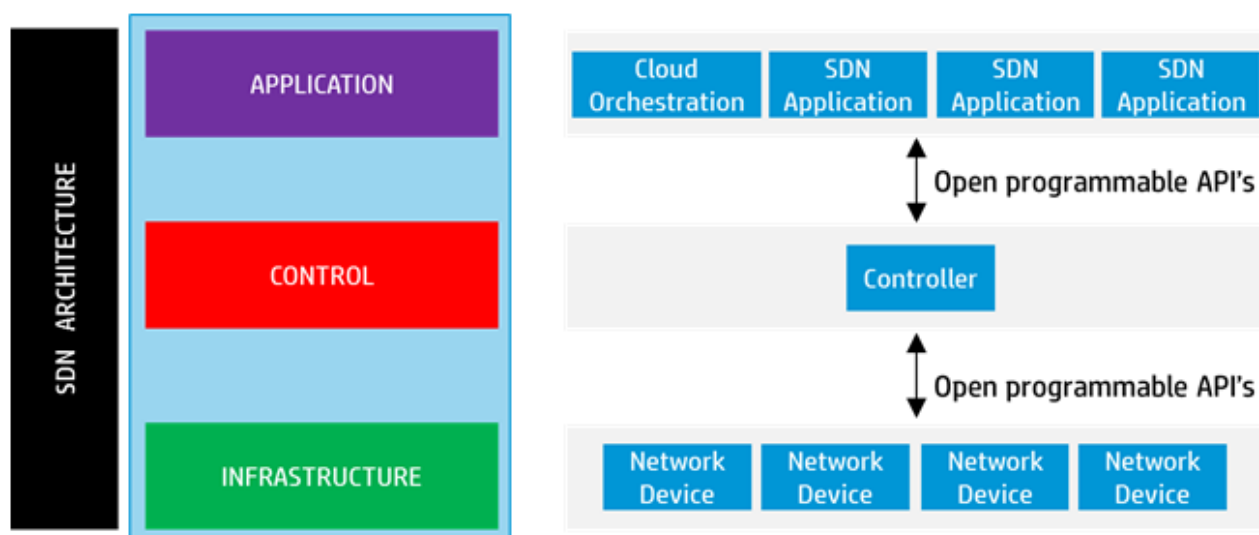


Рисунок 8 – Структура та компоненти SDN

Перший потік отримав назву «північний міст», а другий «південний міст». Як «північний міст» виступає протокол на основі REST API, а в якості «південного моста» – протокол OpenFlow. Схему інформаційних потоків управління контролера SDN наведено на рис.9.

На «північній» стороні контролер надає програмні інтерфейси (API), наявність яких дозволяє власнику мережі або стороннім розробникам створювати програми для управління мережею. Такі програми можуть виконувати різні функції в інтересах бізнес-задач (наприклад, контролювати доступ, управляти пропускнуною спроможністю і т.д.), Причому їх розробникам не потрібно знати деталі функціонування конкретних мережеских пристроїв.

Завдяки контролеру, вся мережа, що складається із великої кількості різнотипних пристроїв різних виробників, виступає для програми, як один логічний комутатор.

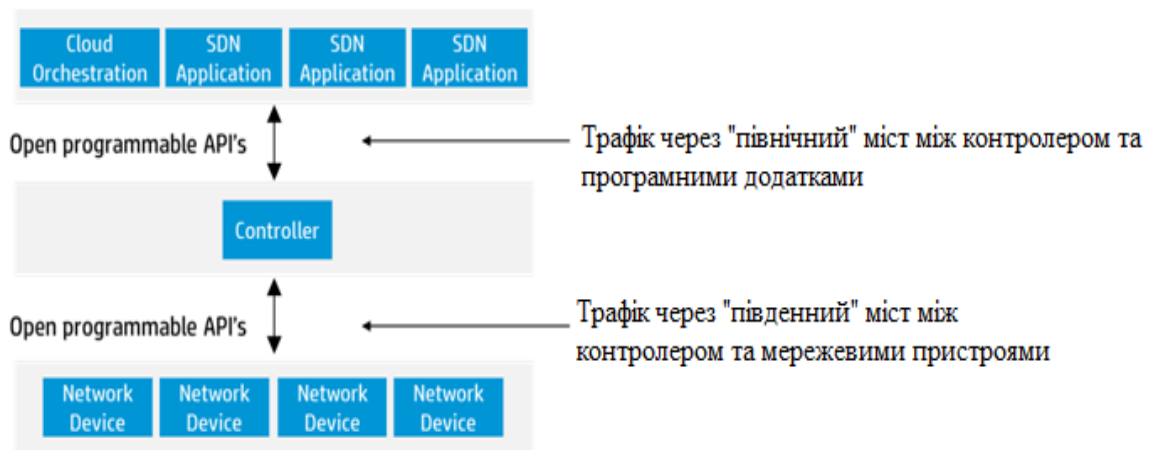


Рисунок 9 – Керуючі інформаційні потоки контролера SDN

Таким чином, весь інтелект (management plane і control plane) переноситься в окремий центральний пристрій – контролер SDN. Логічну модель мережевих пристроїв SDN зображено на рис.10.



Рисунок 10 – Логічна модель мережевих пристроїв SDN

Таким чином, можна виділити три напрямки реалізацій SDN [9]:

- SDN, розроблені на основі первісної, оригінальної версії, запропонованої фахівцями Стенфордського університету (США) в 2007 р .;
- SDN, розроблені на основі існуючих API;
- SDN, розроблені на основі накладених мереж і гіпервізора.

Перший напрямок передбачає переміщення функціоналу управління мережею від мережевих пристроїв до централізованого контролера з використанням протоколу OpenFlow [7]. Така SDN мережа повинна володіти, як вважали п'ятьма фундаментальними властивостями: мати розділені рівні управління, використовувати простіші мережеві пристрої, здійснювати централізоване управління мережею, використовувати автоматизацію і віртуалізацію мережних функцій, бути відкритою для фахівців, розробників. Обов'язковим елементом такої реалізації є наявність контролера, що працює з мережею через інтерфейс OpenFlow.

У другому напрямку при створенні SDN використовують функції API, які можуть бути викликані віддалено, як правило, за допомогою традиційних механізмів, таких як SNMP, CLI або при використанні новіших гнучких механізмів, типу RESTful API.

Реалізація SDN в третьому випадку не залежить від мережевої інфраструктури нижнього рівня: SDN накладається поверх існуючої фізичної мережі.

Отже, на основі застосування технології SDN отримуємо [8]:

- розподіл функцій передачі трафіку і функцій управління (включаючи контроль як самого трафіку, так пристроїв, що здійснюють його передачу);
- єдиний, стандартний, відкритий інтерфейс між пристроями управління і передачі (який отримав назву OpenFlow);
- централізоване управління мережею (контролер SDN);
- віртуалізацію фізичних ресурсів мережі;
- можливості програмування як обладнання (OpenFlow), так і додатків (API – контролер SDN);
- швидше реагування на зміни в мережі;
- оптимізацію передачі трафіку через більшу кількість резервних шляхів;
- простіше і швидше налаштування мережі;

- суттєве зменшення часу розгортання програм;
- спрощення управління мережевими пристроями;
- скорочення витрат на управління мережами;
- централізоване застосування політик, збільшення продуктивності, зменшення затримок призводять до ефективнішої взаємодії користувачів і програм як в корпоративних мережах, так і в мережах дата-центрів;
- простоту управління цілими мережами, а не мережевими пристроями;
- відкриті, засновані на стандартах протоколи дозволять взаємодіяти різним виробникам мережного устаткування між собою, одночасно збільшуючи вибір замовника і конкуренцію між вендорами при зниженні витрат, прискорюючи інновації як в області програмного забезпечення, так і апаратних засобів.
- контролер SDN підтримує відкритий інтерфейс програмування (API), який дозволяє програмувати його ззовні, створюючи середовище для автоматизації і контролю, а також масштабувати функціонал для майбутніх додатків.
- можливість безпосереднього запиту програмою певних вимог до мережі;
- видимість контролером всього трафіку мережі.

Зважаючи на вищезазначене, підхід SDN означає фізичний розподіл площини управління і площини передачі даних у мережах, при якому площина управління відповідає за роботу декількох елементів мережі. Це проста в управлінні, гнучка і економічно ефективна мережева архітектура, що забезпечує високу пропускну здатність і динамічність, що принципово важливо для сучасних програмних додатків.

Застосування SDN сприяє вирішенню проблем віртуалізації. При цьому можливе створення незалежних віртуальних мереж, які спільно використовують єдину мережеву інфраструктуру, причому кожна з віртуальних мереж має набагато простішу топологію, ніж фізична мережа.

					ІАЛЦ. 045480.004 ПЗ	Арк.
						29
Змін.	Арк.	№ докум.	Підпис	Дата		

Віртуалізація мережі не обов'язково вимагає технології SDN, а SDN (як логічний поділ площин управління та передачі даних) не обов'язково означає віртуалізацію мережі. Технологія SDN та віртуалізація мережі співвідносяться в трьох основних іпостасях [8]: як засіб віртуалізації мережі, як засіб тестування на віртуальній мережі, як засіб «нарізки» віртуальної мережі.

У звичайній мережі віртуалізація маршрутизаторів і комутаторів є досить складною, оскільки кожен віртуальний компонент повинен мати власний примірник керуючої програми. Проте віртуалізувати SDN-комутатор набагато простіше.

Концепція розподілу простору мережі на певні «нарізки» – слайси (slices), де кожен слайс має свою частку мережевих ресурсів і керується своїм SDN-контролером використовується для мереж доступу, щоб уможливити інших провайдерів надати послуги локалізованому колу користувачів без розгортання власної мережевої інфраструктури, причому з власною мережевою топологією і адресним простором.

У SDN є чимало переваг у порівнянні з традиційною мережею, зокрема [10]:

- ширші можливості управління з підвищеною швидкістю і гнучкістю. Замість того, щоб вручну програмувати кілька апаратних пристроїв конкретних постачальників, розробники можуть керувати потоком трафіку в мережі, просто програмуючи стандартні контролери на базі програмного забезпечення з відкритим вихідним кодом. Крім того, адміністратори мережі мають ширший вибір мережевого обладнання, оскільки вони можуть вибирати протокол із відкритим вихідним кодом для зв'язку з будь-якою кількістю апаратних пристроїв через центральний контролер;
- Завдяки SDN адміністратори можуть налаштовувати мережеві служби і виділяти віртуальні ресурси, щоб централізовано змінювати мережеву інфраструктуру в режимі реального часу. Це дозволяє адміністраторам мережі оптимізувати потік даних в мережі, віддаючи пріоритет програмам, яким потрібен вищий рівень доступності;

- SDN забезпечує візуалізацію всієї мережі і надає цілісніше уявлення про загрози безпеці. Розробники можуть створювати окремі зони для пристроїв, яким потрібні різні рівні безпеки, або миттєво ізолювати зламани пристрої, щоб не піддавати ризику всю мережу.

Основна відмінність SDN від традиційної мережі полягає у інфраструктурі: мережа SDN заснована на програмному забезпеченні, в той час, як звичайна мережа – на обладнанні. Оскільки площина управління SDN базується на програмному забезпеченні, ця мережа є більш універсальною, ніж традиційні мережі. SDN дозволяє адміністраторам управляти мережею, змінювати параметри конфігурації, формувати ресурси і підвищувати обсяг мережевих ресурсів за допомогою призначеного для користувача централізованого інтерфейсу і без додаткового обладнання.

Ще одна відмінність між SDN та традиційною мережею полягає в різних рівнях безпеки. Завдяки покращеній візуалізації і можливості визначати надійні шляхи SDN забезпечує вищий рівень безпеки в багатьох аспектах. Однак, оскільки в програмно-конфігурованих мережах використовується централізований контролер, його захист критично важливий для підтримки безпеки мережі. Ця єдина точка відмови створює потенційну вразливість SDN. Окрім того, SDN дозволяє швидко переміщати робочі навантаження по мережі, підвищує гнучкість і спрощує масштабування будь-якої мережі в процесі додавання або видалення віртуальних мереж адміністраторами, незалежно від того, чи є зазначені мережі локальними, чи хмарними. Завдяки рівню швидкості і гнучкості SDN уможливорює підтримку нових тенденцій і технологій.

Реалізація концепції SDN на практиці дозволяє підприємствам і операторам зв'язку отримати незалежний контроль над всією мережею з єдиного місця, що значно спрощує експлуатацію. При цьому конфігурація мережі суттєво спрощується і адміністраторам не потрібно вводити сотні рядків коду окремо для різних комутаторів або маршрутизаторів. Характеристики мережі можна оперативно змінювати в режимі реального

часу, відповідно, терміни впровадження нових програм та сервісів суттєво скоротяться.

2.3 OpenFlow

Openflow – стандартний протокол, який є основним елементом концепції SDN, що забезпечує взаємодію контролера з мережевими пристроями [6]. Контролер використовується для управління таблицями потоків комутаторів, на підставі яких приймається рішення про передачу прийнятого пакету на конкретний порт комутатора. Таким чином, в мережі формуються прямі мережеві з'єднання з мінімальними затримками передачі даних і необхідними параметрами.

Таким чином, комутатор OpenFlow складається, як мінімум, з двох компонент [7]:

- таблиці потоків (flow table);
- безпечного каналу (secure channel).

На рис.11 зображено таблицю потоків OpenFlow.

MAC src	MAC dst	IP Src	IP Dst	TCP dport	...	Action	Count
.	10.20..	.	.	.	.	port 1	250
.	.	.	5.6.7.8	.	.	port 2	300
.	.	.	.	25	.	drop	892
.	.	.	192.	.	.	local	120
.	.	.	.	.	.	controller	11

Рисунок 11 – Таблиця потоків OpenFlow

На сьогодні найпоширенішим рішенням для реалізації управління мережових пристроїв в програмно-конфігуровних мережах є протокол OpenFlow. Таке управління може замінити або доповнити працюючу на

мережевому пристрої функцію, що здійснює побудову маршрутів, створення таблиці комутації і т. д.

При такій концепції всієї системи особливо важливим є вибір контролера OpenFlow. Саме його швидкодія стане вирішальним фактором при виникненні будь-яких ситуацій. Саме його алгоритми визначатимуть ефективність роботи всієї мережі і його збій з великим ступенем ймовірності вплине на роботу всієї мережі.

Основна ідея полягає у тому, що відокремивши рівень управління від рівня передачі трафіку в комутаторах, вони, як мінімум, повинні стати простішими. Проте новий протокол, а точніше величезна кількість полів в таблицях маршрутизації, які обліковуються за замовчуванням, висуває високі вимоги до об'ємів пам'яті на комутаторах.

Рівень передачі даних комутатора складається з таких компонентів, який програмується OpenFlow-контролером [7]: OpenFlow-порти (Ports), потоки інформації (Flows), таблиці потоків (FlowTables), класифікатори (Matches), виконавчі дії (Actions).

Пакети, що надходять на OpenFlowпорт, розподіляються на потоки в таблицях потоків за допомогою класифікаторів. Кожен потік складається з набору дій (Actions), що застосовуються для кожного пакету, який повинен відповідати певному правилу.

OpenFlow-порти виконують ті ж функції, що і порти стандартного комутатора. З точки зору вхідних і вихідних потоків трафіку, немає ніякої різниці зі звичайним L2-комутатором традиційної IP-мережі [9]. Вхідний порт одного потоку може бути вихідним для іншого. Процес комутації пакетів в OpenFlow має такий вигляд. Пакети, що опрацьовуються протоколом OpenFlow, приймаються на вхідний порт, опрацьовуються в пайплайн комутатора і направляються далі в вихідний порт.

Пайплайном вважають програмний конвеєр, в якому відбувається послідовний процес обробки даних. Наприклад, значення вхідного порту є одним з властивостей пакету, що проходить через пайплайн комутатора, і

показує на якому OpenFlow-порту цей пакет був прийнятий в обробку [7]. Пайплайн в протоколі OpenFlow є найважливішою частиною, оскільки в ньому приймається рішення про те, відправляти пакети на вихідний порт чи ні, при цьому встановлюючи параметри дій (actions).

Параметри цих дій в свою чергу будуть визначати, як саме пакет буде повернений назад в мережу. Досліджуваний протокол визначає набір стандартних портів: фізичні, логічні і локальні зарезервовані, якщо підтримуються OpenFlow-комутатором. Фізичними портами (physical) є безпосередньо порти самого комутатора. Логічні (logical) – це порти безпосередньо неприв'язані до фізичних інтерфейсів обладнання, як і в традиційних мережах, наприклад, VLAN, Tunnel і Null інтерфейси [7].

OpenFlow-комутатор уможлиблює розробку OpenFlow-портів, доступних для використання в локальних процесах комутатора, що називаються зарезервованими (reserved). Зарезервовані порти використовуються для внутрішньої обробки трафіку і для гібридних типів впровадження. Зарезервовані порти можуть бути обов'язковими або опціональними. Обов'язкові порти включають порти типу ALL, CONTROLLER, TABLE, IN_PORT, ANY, UNSET. До опціональних портів відносять – LOCAL, NORMAL і FLOOD.

Комутатори, які розроблені тільки для роботи з OpenFlow не підтримують порти типу NORMAL і FLOOD, в той час, як гібридні комутатори підтримують їх. Передача трафіку в порти типу FLOOD залежить від того, як налаштований і для чого призначається комутатор. Тут можна згадати про використання групи ALL, яка дозволяє контролеру гнучкіше реалізовувати потокову розсилку. Проте, якщо, наприклад, потрібно зробити тунель між двома кінцевими точками, це неможливо на основі протоколу OpenFlow. Подібні задачі можна виконати за допомогою інших протоколів, наприклад, OF-CONFIG. Порти можуть бути додані або видалені з конфігурації комутатора за допомогою OF-CONFIG, проте не на основі OpenFlow [4].

OpenFlow таблиці. При отриманні пакету з нього витягуються метадані (наприклад, ingress port) й інші поля. Потім ці поля порівнюються із записами в таблиці потоків. Кожен запис в таблиці має відповідне поле «protocol», по якому йде порівняння. Результат з найвищим пріоритетом вважається кращим, і обробка пакету з таким пріоритетом починається першої.

Кожен запис в таблиці потоків повинен мати свої пріоритети, а запис із найвищим пріоритетом визначає те, як надалі буде опрацьовано пакет. Як тільки вибрано правило (Match), до пакету застосовують певну дію (Action). Можливими діями можуть бути: «відправити в порт X», «відкинути» або «відправити на контролер».

За замовчуванням протокол OpenFlow відправляє на контролер всі пакети, які не відповідають жодному правилу. У більш новіших версіях протоколу цей механізм змінено, оскільки це може спричинити DoS-атаки на контролер з боку злоумисників [7].

Перша специфікація протоколу OpenFlow 1.0 передбачала лише можливості «перенаправити на інший порт» або «відправити на контролер», але пізніше прийшли до висновку, що в певних випадках потрібно модифікувати поля в пакеті, наприклад, зменшити TTL або додати тег VLAN. Обмеження на кількість таблиць у наступних версіях протоколу змінено, але це в той же час зруйнувало ідею про те, що OpenFlow може використовуватися на недорогих комутаторах. Наявність декількох таблиць накладає додаткові вимоги до ресурсів комутаторів.

Таким чином, OpenFlow – це метод управління потоками (flow) в мережі. Адже мережеві пристрої (маршрутизатори, комутатори і інші) займаються передачею пакетів і фреймів відповідно до протоколів маршрутизації, проте, програми не використовують певні пакети для надання послуг. Вони обмінюються даними між сервером і клієнтом та створюють потоки пакетів від джерела до одержувача.

OpenFlow визначає стандарт розсилки «правил потоку» (flow rules) в таблицю передачі (forwarding table) кожного мережевого пристрою таким

чином, що площина управління може управляти площиною передачі даних. Ці правила потоку містять установки для мережевих пристроїв, такі як MAC-адреса джерела і призначення (source & destination MAC), IP-протокол для джерела і призначення (Source & destination IP), TCP-протокол для джерела і призначення (source and destination TCP), дані про віртуальну (логічну) локальну мережу VLAN в загальному околі інфраструктури, мітки QoS і MPLS, та іншу інформацію. Правила потоку потім додаються в існуючі таблиці передачі (forwarding table) пакетів зі входу на вихід кожного мережевого пристрою.

					ІАЛЦ. 045480.004 ПЗ	Арк.
						36
Змін.	Арк.	№ докум.	Підпис	Дата		

3. Моделювання SDN мереж

3.1 Існуючі системи моделювання мереж

У сучасному світі стрімкого розвитку інформаційних технологій впливають все більші вимоги до гнучкості і масштабованості комп'ютерних мереж що призводить до їх стрімкого збільшення та ускладнення. Проте побудувати ідеальну мережу з першого разу задача нереальна. Деякі помилки можуть коштувати дуже значну суму та розтягнути процес розвертання мережі на досить тривалий час. Саме через це набули популярності різноманітні системи для моделювання мереж, які дозволяють спроектувати та передбачити поведінку мережі заздалегідь. Також такі системи часто використовуються у навчальному процесі для підготовки спеціалістів у мережевих технологіях. Це дає гарний простір для проведення різноманітних експериментів та розвитку технологій без зайвих витрат.

Отже, усі системи моделювання діляться на дві основні категорії: симулятори та емулятори.

Емуляція - комплекс програмних, апаратних засобів або їх поєднання, призначене для копіювання функцій однієї обчислювальної системи на інший, відмінній від першої, таким чином, щоб емульована поведінка якомога ближче відповідала поведінці оригінальної системи [5].

Емулятори надають досить широкий функціонал.

- Створення власних топологій.
- Можливість підключення різноманітних пристроїв.
- Конфігурація характеристик мережі.
- Емуляція трафіку.

Симуляція у свою чергу це лише імітація процесів, відтворення моделі системи та певних її характеристик та особливостей.

Проте їх перевага полягає у тому, що за допомогою симуляторів можливо змодельовати такі ситуації, які не можливо розглянути навіть при наявності потрібного обладнання.

Отж е, якщо казати узагальнено, то симулятор - це середовище, яке моделює, але емулятор - це середовище, яке копіює використання, як на вихідному пристрої або системі. Зазвичай симулятори використовують з ціллю аналізу та дослідження, а емулятори використовують в якості заміни.

Розглянемо деякі існуючі системи.

GNS3

GNS3 – одна з досить популярних програм, що використовується для емуляції мереж. Це програмне забезпечення надає можливість спостерігати за взаємодією різних елементів мережі у різних топологіях. Приклад вигляду інтерфейсу та робочого середовища з деякою мережею наведено на рис.12.

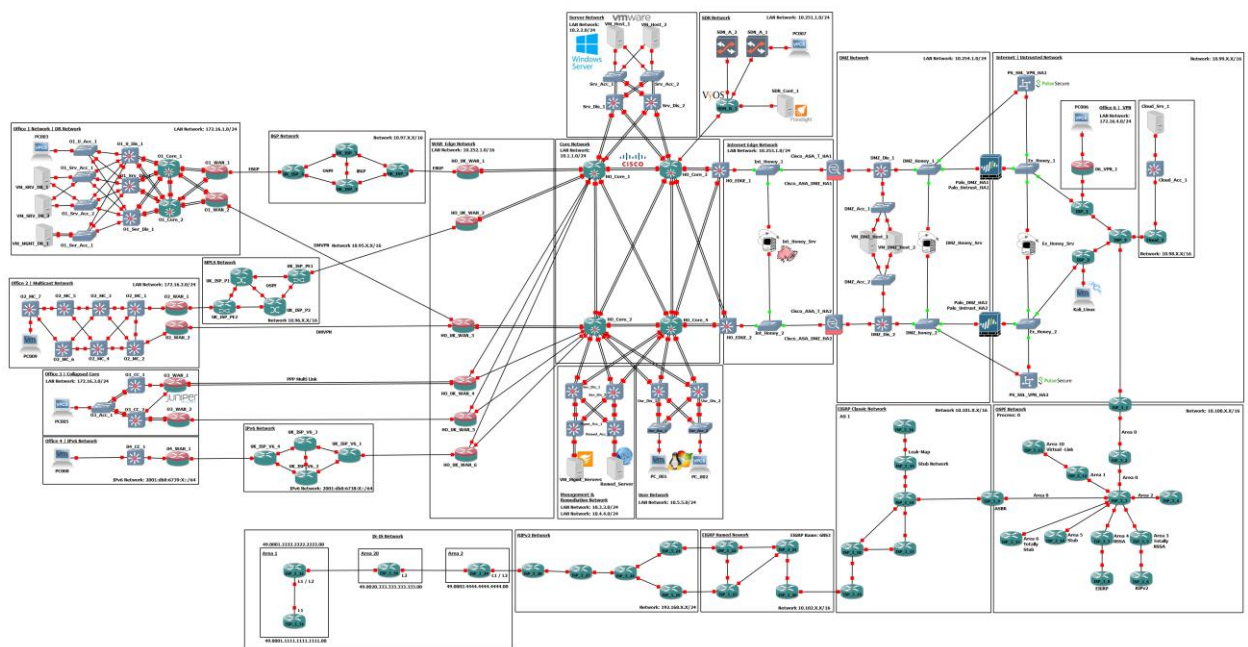


Рисунок 12 – Інтерфейс емулятора GNS3

Це програмне забезпечення, яке є інтегрованим сегментом в міжнародній мережі навчання сертифікації, тому немає ніяких сумнівів у виборі саме цього емулятора для своєї роботи як професіоналам, так і новачкам.

Cisco Packet Tracer

Cisco Packet Tracer - це симулятор мережі, який випускається однією з найвідоміших фірм з постачання мережевого обладнання Cisco Systems.

Найголовнішою перевагою є кроссплатформенність, що і надало цій системі таку популярність. Також є популярним завдяки тому, що він дозволяє працювати та налаштовувати обладнання Cisco за допомогою команд Cisco IOS, що може добре підготувати спеціалістів до реального життя. Вигляд інтерфейсу наведено на рис.13.

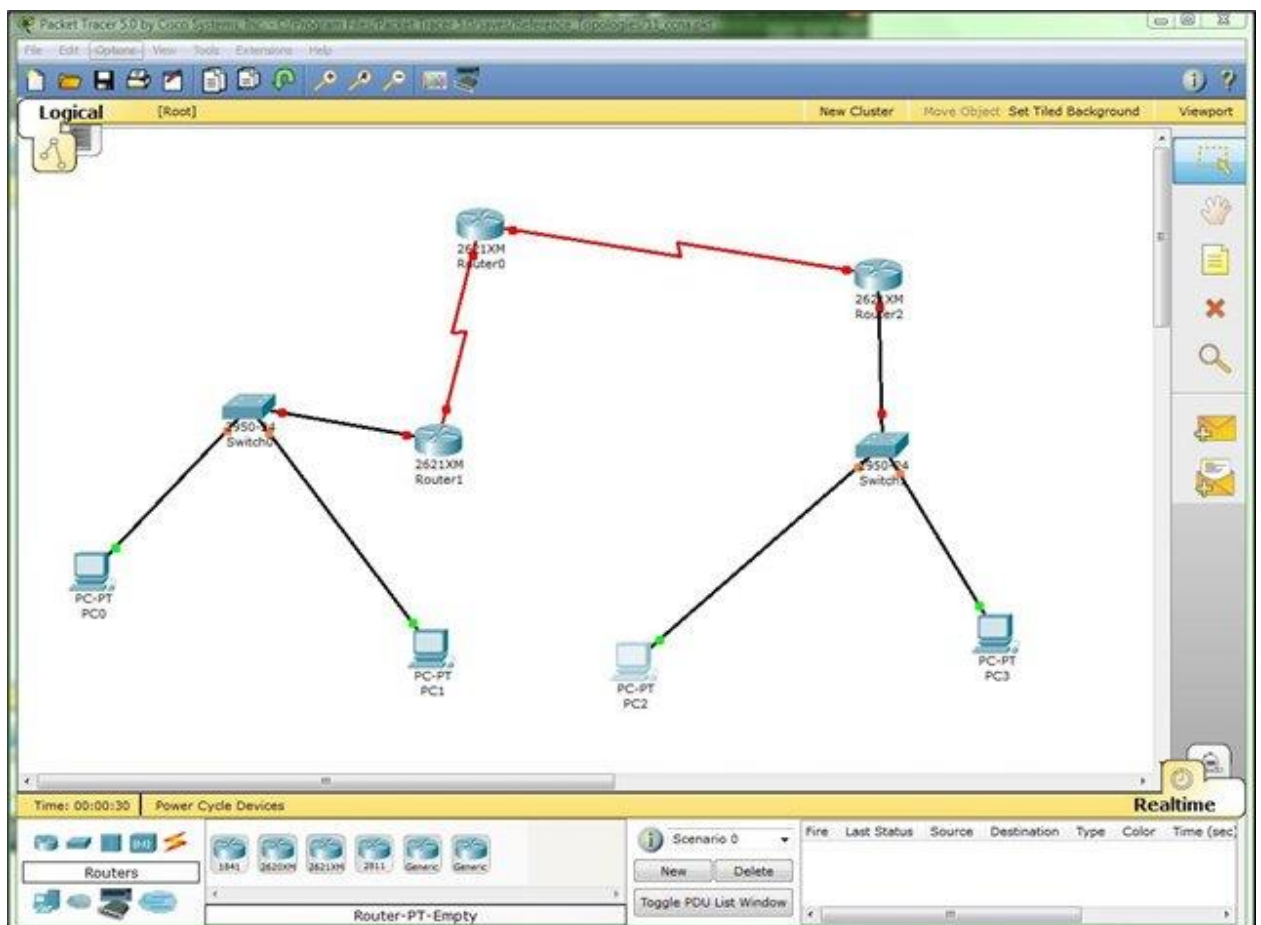


Рисунок 13 – Інтерфейс програмного середовища Cisco Packet Tracer

Його функціонал дозволяє створювати працездатні моделі мережі, налаштовувати маршрутизатори і комутатори, взаємодіяти між декількома користувачами (через хмару).

Змін.	Арк.	№ докум.	Підпис	Дата

ІАЛЦ. 045480.004 ПЗ

Арк.

39

EVE-NG

EVE-NG (Emulated Virtual Environment - Next Generation) - це емулюватися віртуальне середовище наступного покоління, що дозволяє створити повноцінну віртуальну лабораторію з мережевим обладнанням і програмним забезпеченням провідних світових виробників.[11]

Проте основна цільова аудиторія цього програмного забезпечення є невеликі підприємства та приватні особи, адже у програми є дві версії: основна та безкоштовна.

Версія EVE-NG Community Edition є безкоштовною та має обмеження у 63 вузли для кожної віртуальної лабораторії, проте цього цілком достатньо для навчання.

Особливістю цього емулятора є те, що доступ до неї може здійснюватись через веб-браузер. Приклади підключення до емулятору та вигляд інтерфейсу наведено на рис.14 та рис.15 відповідно.

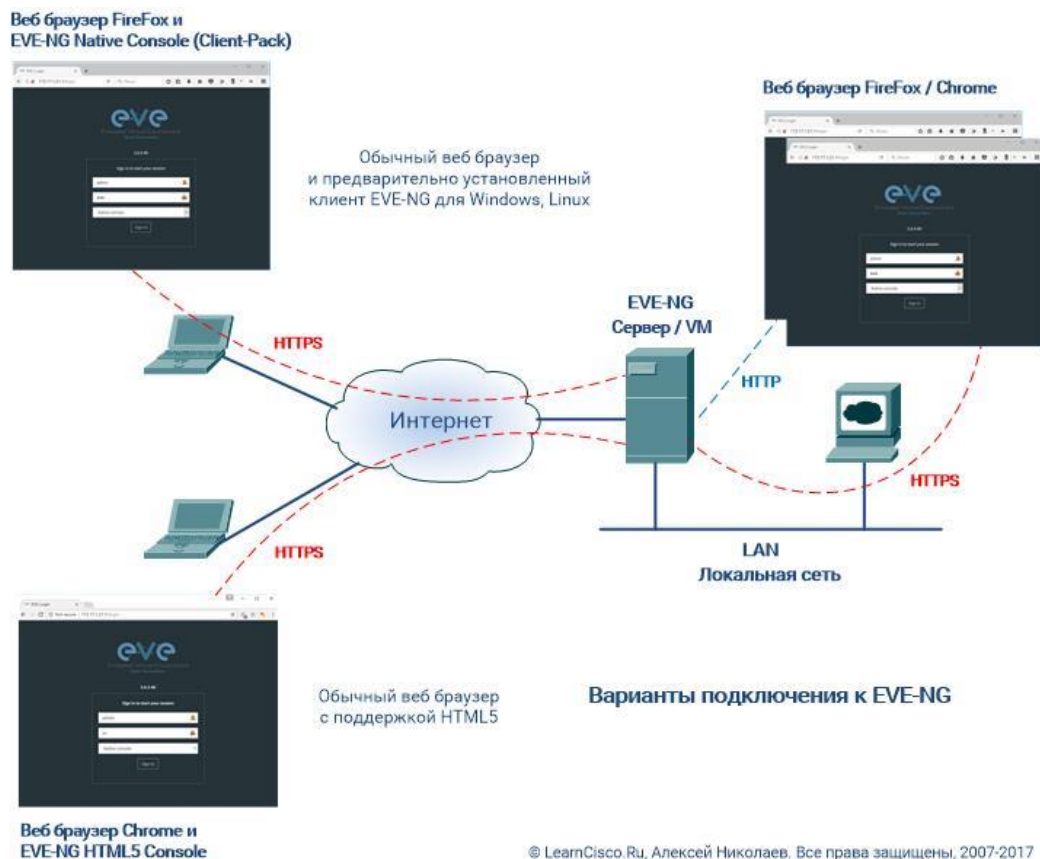


Рисунок 14 – Способи підключення до EVE-NG

					ІАЛЦ. 045480.004 ПЗ	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		40

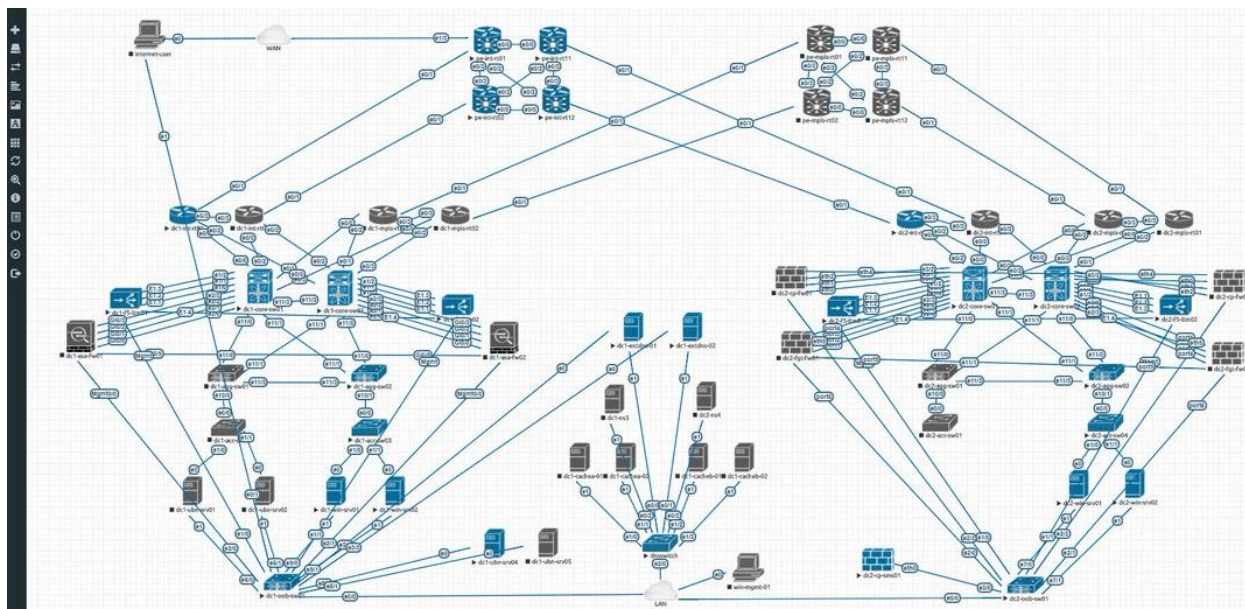


Рисунок 15 – Інтерфейс емулятору EVE-NG

Подібних середовищ існує ще досить багато, але для роботи з програмно-конфігуровними мережами був обран Mininet.

3.2 Mininet

Mininet – це один з небагатьох емуляторів комп’ютерних мереж з підтримкою протоколу OpenFlow. Mininet - це полегшена програмно-конфігуровна мережа і тестова платформа; в ній використовується полегшена технологія віртуалізації, щоб зробити єдину систему схожою на повну мережу для запуску системи ядра і призначеного для користувача коду, про які було задумано[12]. В рамках однієї віртуальної машини за допомогою Mininet можна розвернути мережі з довільною кількістю вузлів, з найрізноманітнішими топологіями за допомогою простого синтаксису в досить примітивному інтерпретаторі.

Великою перевагою є те, що основна функціональність написана мовою Python і багато топологій можна описати за допомогою спеціального синтаксису цієї мови.

Змін.	Арк.	№ докум.	Підпис	Дата

Функції, які реалізовує Mininet:

- підтримка SDN компонентів, а зокрема OpenFlow, OpenSwitch та інших;
- підтримка складних топологій, конфігурованих топологій;
- підтримка Python API;
- висока масштабованість;
- підтримка тисячі вузлів.

3.2.1 Установка

Також однією з найбільших переваг є проста установка. Перш за все потрібно встановити віртуальну машину. В даній роботі використовується VirtualBox. Далі на базі віртуальної машини є два варіанти користуватись емулятором: запустити Mininet безпосередньо на віртуальці чи на базі встановленої операційної системи. Вирішено використовувати операційну систему Linux, як базу для розгортання мережі. Такий підхід має безліч переваг:

- зручна установка та налаштування;
- можливість використання додаткового програмного забезпечення;
- зручна навігація і написання конфігураційних файлів;
- починаючи з версії 2.6.24, ядром Linux підтримуються механізми віртуалізації.

Приклад встановленої віртуальної машини з операційною системою Ubuntu наведено на рис.16.

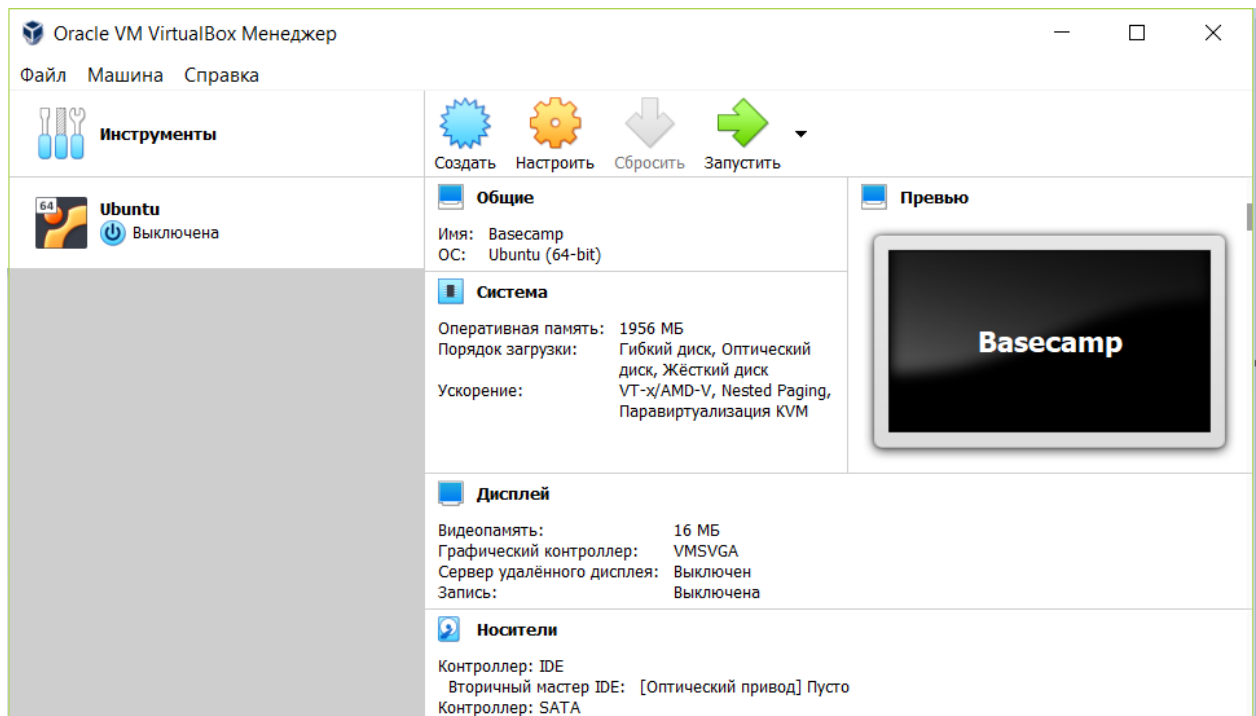


Рисунок 16 – Интерфейс VirtualBox з встановленою системою Ubuntu

Для того, щоб встановити Mininet потрібно запустити Ubuntu та відкрити командну строку Terminal. В роботі використано спосіб установки Mininet за допомогою вихідного коду Git. Для цього потрібно виконати такі команди:

1. `$sudo apt install git`
2. `$git clone git://github.com/mininet/mininet`
3. `$cd mininet`
4. `$util/install.sh -fnv`

Розглянемо кожен рядок.:

1. Встановлення Git для того, щоб можна було встановити вихідний код
2. Клонування вихідного коду Mininet
3. Після завантаження коду, у домашньому каталозі з'являється папка mininet, за допомогою команди cd переходимо до неї
4. Запускаємо інсталяційний файл install.sh з флагом `-fnv` для встановлення Mininet, контрольного контролера OpenFlow та Open vSwitch.

Для перевірки встановлення потрібно ввести команду:

```
$sudo mn
```

Вся робота з віртуальною секцією mininet, а саме розробка мереж бажаних топологій, зміна різних параметрів хостів або комутаторів та т. п., виробляється в простому інтерпретаторі команди – mn [13].

Результатом виконання цієї команди має бути створення найпростішої мережі, що продемонстровано на рис.17.

```
ann@ann-VirtualBox: ~  
ann@ann-VirtualBox:~$ sudo mn  
[sudo] password for ann:  
*** No default OpenFlow controller found for default switch!  
*** Falling back to OVS Bridge  
*** Creating network  
*** Adding controller  
*** Adding hosts:  
h1 h2  
*** Adding switches:  
s1  
*** Adding links:  
(h1, s1) (h2, s1)  
*** Configuring hosts  
h1 h2  
*** Starting controller  
  
*** Starting 1 switches  
s1 ...  
*** Starting CLI:  
mininet> 
```

Рисунок 17 – Результат роботи виконання команди sudo mn

Mininet успішно встановлений, можна переходити до створення власної мережі.

3.2.2 Створення топологій у середовищі MiniNet

Як вже раніше було зазначено, Mininet підтримує можливість створення власних топологій. Проте є варіанти запуску шаблонних топологій з власними параметрами. Розглянемо можливі варіанти.

Після команди `sudo mn` ми задаємо ключ `--topo`, який задає параметри топології та може містити певні аргументи.

`$ sudo mn --topo minimal (1 switch, 2 hosts)`

Параметр `minimal` задає найпростішу топологію з одним комутатором та двома під'єднаними до нього вузлами. Зображення топології наведено на рис.18.

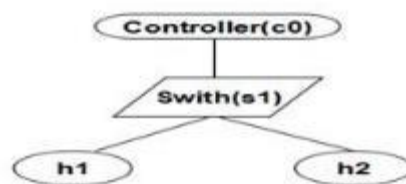


Рисунок 18 – Топологія за параметром `minimal`

`$ sudo mn --topo single, n (1 switch, n hosts)`

При використанні параметру `single`, потрібно задавати певний аргумент `n`. В такому випадку Mininet побудує топологію яка складається з одного комутатора та `n` вузлів. Отже `--topo single, 4` побудує топологію з одним комутатором та чотирма вузлами, як на рис.19.

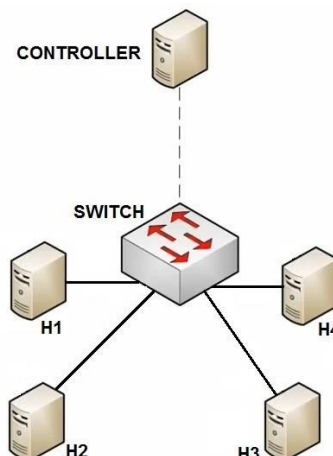


Рисунок 19 – Топологія за параметром `single,4`

\$ sudo mn --topo linear, n (n switches, n hosts)

Параметр linear будує лінійну топологію з n комутаторами та n вузлів підключених до кожного з них відповідно. Тому --topo linear, 3 побудує топологію з трьома комутаторами та трьома підключеними до них вузлами (рис.20).

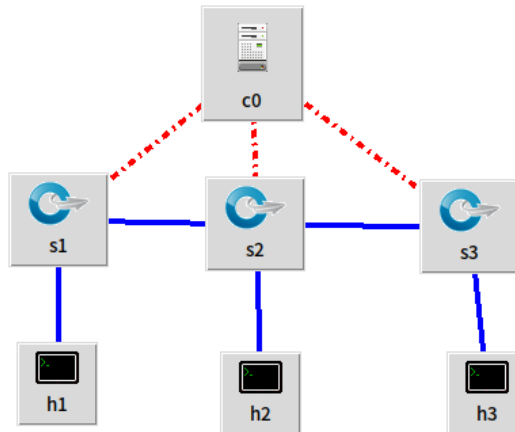


Рисунок 20 – Топологія за параметром linear, 3

\$ sudo mn --topo tree, depth n, fanout n

При заданні параметра tree буде побудована топологія деревовидного типу, при цьому цей параметр має два аргументи: depth n и fanout n.

Аргумент depth n задає значення глибини дерева, або ж кількість рівнів дерева.

Аргумент fanout n задає значення на скільки кожен вузол дерева повинен ділитися.

Приклад створеної топології за параметром --topo tree, depth 2, fanout 2 наведено на рис.21.

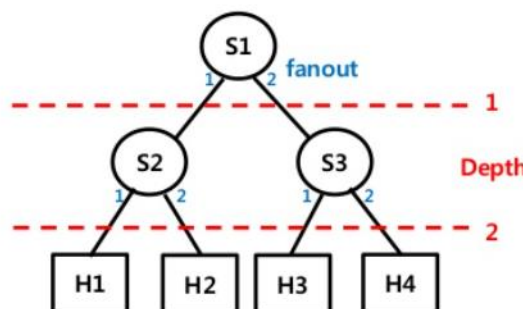


Рисунок 21 - Топологія за параметром --tree, depth 2, fanout 2

```
$ sudo mn --custom file.py --topo mytopo
```

Найцікавішим варіантом є творення своєї топології. Розглянемо детальніше кожен частину команди. Після команди `sudo mn` йде ключ `--custom` з аргументом `file.py`. Що це за файл? Цей файл має містити код Python, в якому прописана певним чином власна топологія. Далі йде вже звичний нам ключ `--topo` з параметром `mytopo` що означає використання власної топології і назва задається у файлі.

Розглянемо детальніше зміст файлу на прикладі написаної топології в ході розробки дипломного проєкту.

MyTopo.py

```
1  #!/usr/bin/python
2
3  from mininet.topo import Topo
4  from mininet.cli import CLI
5  from mininet.net import Mininet
6  from mininet.util import dumpNodeConnections
7  from mininet.log import setLogLevel
8
9  from mininet.node import RemoteController
10
11
12 REMOTE_CONTROLLER_IP = "127.0.0.1"
13
14
15 class MyTopo(Topo):
16
17     def __init__(self, **opts):
18         # Initialize topology and default optioe
19         Topo.__init__(self, **opts)
20         switches = []
21         hosts = []
22
23         # create switches
24         for s in range(3):
25             switches.append(self.addSwitch('s%s' % (s + 1),
26 protocols='OpenFlow13'))
27
28         # create hosts
29         for h in range(4):
30             hosts.append(self.addHost('h%s' % (h + 1)))
31
32         self.addLink(hosts[0], switches[1])
33         self.addLink(hosts[1], switches[1])
34         self.addLink(hosts[2], switches[2])
35         self.addLink(hosts[3], switches[2])
36
37         self.addLink(switches[0], switches[1])
38         self.addLink(switches[0], switches[2])
39
```

```

40 topos = {'mytopo': (lambda: MyTopo())}
41
42 if __name__ == '__main__':
43     # Tell mininet to print useful information
44     setLogLevel('info')
45     topo = MyTopo()
46     net = Mininet(topo=topo,
47                   controller=None,
48                   autoStaticArp=True)
49     net.addController("c0",
50                       controller=RemoteController,
51                       ip=REMOTE_CONTROLLER_IP,
52                       port=6633)
53     net.start()
54     CLI(net)
55     net.stop()

```

Розглянемо структуру написання топології та коду:

- 1-9 рядки – імпортування потрібних бібліотек
- 12 рядок – задання значення глобальної змінної для ір-адреси контролера
- 15-19 рядки – ініціалізація класу та топології
- 20-21 рядки – визначення масивів для комутаторів та вузлів
- 24-26 рядки – цикл для створення трьох комутаторів
- 28-30 рядки - цикл для створення чотирьох вузлів
- 32-38 рядки – створення зв’язків між вузлами та комутаторами
- 40 рядок – задання назви параметру топології для її запуску
- 42-52 рядки – створення мережі, додавання контролера
- 53 рядок – запуск мережі
- 55 рядок – припинення роботи мережі

Ключові функції для побудови мережі:

- self.addSwitch** – функція для додавання нових комутаторів
- self.addHost** – функція для додавання нових вузлів
- self.addLink** – функція для додавання зв’язків
- net.addController** – функція для додавання контролеру
- net** – функція для створення мережі

3.2.3 Конфігурація параметрів мережі

Великою перевагою є те, що при запуску мережі їй можна задавати певні параметри, розглянемо детальніше декілька з них. Для цього, як і для топологій, використовуємо ключі.

--switch – вибір типу програмно-конфігурованого комутатора. Можливі значення: ovsk, kerneluser.

--controller – вибір типу контролера, можливі значення: remote, ref, pox.

--host – задає адресу запуску Mininet

--topo – задає параметр топології

--mac – запускає формування MAC-адрес для хостів

--arp – заздалегідь встановлює arp записи в хости в статичний режим

--ip – встановлює ip-адресу контролера

--prefixlen – довжина маски для автоматичного розподілу адрес

--link tc,bw=10,delay=10ms – встановлення з пропускнуою спроможністю між вузлами обмеженою в 10 Mbit /s і мінімальними затримками в 10 ms.

Також конфігурацією мережі можна вважати запуск певних сервісів на комутаторах чи хостах, наприклад встановлення веб-серверу на хості після запуску мережі (рис.22) чи налаштування комутатора як маршрутизатора.

```
mininet> h1 python -m SimpleHTTPServer 80 & #Open web service на хосте h1
mininet> h2 wget -O-h1 #host h2 загрузить содержимое веб-сайта h1
```

Рисунок 22 – Розгортання вебсерверу на хост1

3.2.4 Контролери, їх види

Контролери SDN (вони ж платформи контролерів SDN) відомі як "мозок" мережі. Контролер SDN є і програма, яка діє як централізований пункт управління в мережі SDN, і фізичний пристрій. Він також керує управлінням потоком до комутаторів / маршрутизаторів. Додатками та бізнес-логікою за допомогою API для розгортання інтелектуальних мереж також

керує контролер. Останнім часом організації розгортають все більше мереж SDN. Контролерам доручено об'єднувати використання загальних інтерфейсів, додатків, включаючи OpenFlow та відкритий віртуальний комутатор бази даних (OVSDB). Різні мережеві завдання виконуються колекцією «підключених» модулів. завдання включають дослідження пристроїв у мережах, пошук можливостей пристроїв та збір статистика мережі тощо. Більш розширені можливості можуть підтримуватись наданням розширень з більшою функціональністю, включаючи алгоритми маршрутизації для аналізу та оркестрування нових правил по всій мережі [14].

NOX

NOX - оригінальний контролер OpenFlow. Він служить платформою управління мережею, яка забезпечує програмний інтерфейс високого рівня для управління та розробки додатків управління мережею. Його загальносистемні абстракції перетворюють мережу на проблему програмного забезпечення.

NOX - спочатку розроблений Nicira Networks, а тепер належить VMware, разом з OpenFlow - був вперше представлений спільноті в 2009 році. Пізніше він був розділений на кілька різних напрямків розробки:

NOX classic: Це версія, яка доступна під GPL з 2009 року.

NOX: "Новий NOX". Містить лише підтримку C++ і має менше програм, ніж класичний; однак ця версія швидша і має кращу кодову базу.

POX

Контролер POX зазвичай називають братом або сестрою NOX, але різниця між ними в тому, що в POX є забезпечення підтримки Python. Контролер POX відповідає за забезпечення зв'язку комутаторів SDN за допомогою протоколу OpenFlow або OVSDB. POX можна використовувати розробниками для створення контролера SDN за допомогою мови програмування Python. Python - це найбільшпопулярний інструмент для розробки та проектування програмного забезпечення мереж та програмування

мережевих додатків. POX є найкращим контролер SDN, який можна використовувати за допомогою компонент, що входять до комплекту. Більш складний контролер SDN також може бути створений шляхом створення нових компонентів POX. Інші мережеві додатки також можуть бути написані за допомогою API POX.

Компоненти POX

Коли POX запускається з командного рядка, то компоненти POX викликаються разом із ним. Вони уявляють собою додаткові Python програми. Вся функціональність мережі в програмно-конфігурованих мережах реалізована за допомогою цих компонент. Деякі компоненти POX є вже доступними.

На веб-сайті NOXhero перераховані наступні функції POX [15]:

- «Пітонічний» інтерфейс OpenFlow.
- вибіркові компоненти для вибору шляху, відкриття топології є багаторазові.
- POX може працювати в будь-якому місці, просте розгортання
- Орієнтовність конкретно на Linux, Mac OS та Windows.
- Графічний інтерфейс і засоби візуалізації POX однакові як NOX.
- Продуктивність програми POX краща ніж додатки NOX, написані на Python.

RYU

Ryu - це програмно-конфігурований контролер на базі компонентів. Компоненти програмного забезпечення з чітко визначеним API RYU допомагають розробникам створювати різні мережеві програми управління та контролю. Різні протоколи, такі як OpenFlow, Netconf, OF-config тощо, підтримуються контролером Ryu для управління мережевими пристроями. Також контролер Ryu підтримує усі розширення Openflow повністю від 1.0 до 1.5 та розширення Nicira. Весь код знаходиться у вільному доступі під ліцензією Apache 2.0. Код Ryu повністю написаний на Python та відкритий для

всіх бажаючих використовувати його. Контролером Ryu підтримуються різні протоколи управління мережею - це NETCONF та OF-config, а також OpenFlow.

Саме цей контролер був обран та встановлен у ході роботи над дипломним проектом. Щоб його встановити, потрібно перейти у командну строку Terminal (у разі використання Ubuntu, як в нашому випадку) та також як і вихідний код Mininet скопіювати з репозиторію GitHub. Для цього потрібно ввести команду:

```
$sudo git clone git://github.com/osrg/ryu.git
```

Для запуску топології з використанням контролеру потрібно при запуску топології використати ключ `--controller remote`. Приклад запуску наведено на рис.23.

```
ann@ann-VirtualBox:~$ sudo mn --topo linear,3 --controller remote --switch ovsk
--mac
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6653
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

Рисунок 23 – Приклад запуску мережі з POX контролером

Далі після запуску мережі потрібно в новому вікні Terminal перейти до каталогу `ryu`, об запусити контролер RYU. Запустимо на ньому простий мережевий додаток `simple_switch_13.py`, який буде емулювати простий перемикач рівня 2, для цього потрібно виконати команду:

```
PYTHONPATH=. ./bin/ryu-manager ryu/app/simple_switch_13.py
```

Якщо порівняти цей результат запуску з рис.17, де був стартовий запуск мережі для перевірки завантаження Mininet, можна побачити різницю, а саме відсутність повідомлення «No default OpenFlow controller found for default switch! Falling back to OVS Bridge» що означає про успішне встановлення та додавання контролеру до мережі.

3.2.5 Результат роботи та захоплення трафіку у Wireshark

Результатом роботи дипломного проєкту є успішно розгорнена SDN мережа на базі емулятора Mininet. Дана мережа складається з одного SDN контролеру RYU, 3 комутаторів з підтримкою OpenFlow протоколу, та 4 вузлів. Схема топології наведена на рис.24.

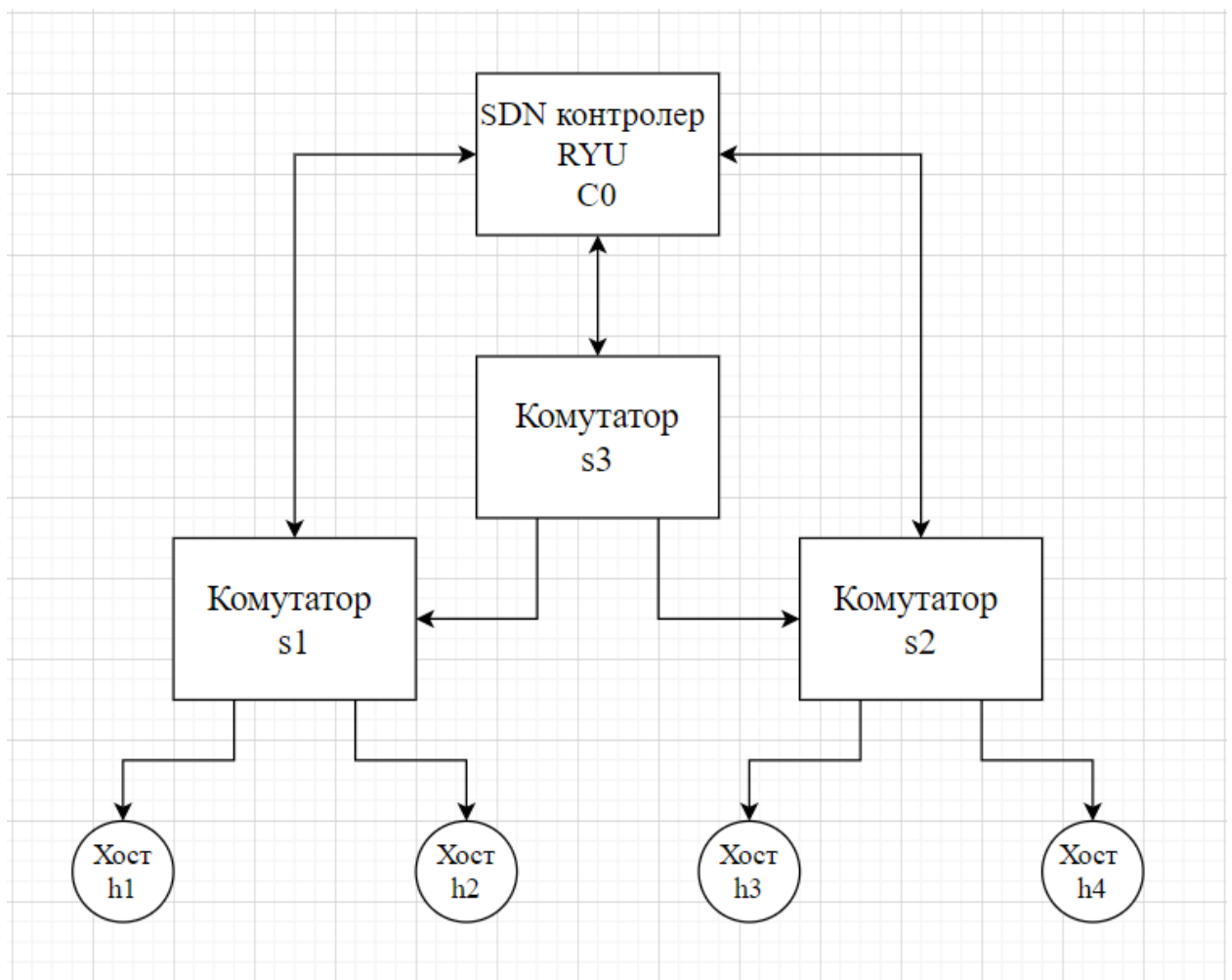


Рисунок 24 – Схема розробленої топології

Для побудови такої топології було використано Python API. Файл з конфігурацією топології написаний мовою Python має назву mytopo.py та приклад його запуску наведено на рис.25(так виглядає топологія з рис.24 у запусненому вигляді у Mininet).

```
ann@ann-VirtualBox:~/mytopo$ sudo mn --custom mytopo.py --topo mytopo --control
ler remote --switch ovsk --mac
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s2) (h2, s2) (h3, s3) (h4, s3) (s1, s2) (s1, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
mininet>
```

Рисунок 25 – Запущений файл mytopo.py з власною топологією

Для перевірки працездатності мережі проведемо найпростішу перевірку. Відправлення запитів на всі вузли- команда pingall - посилає кілька ICMP пакетів між усіма хостами по черзі і показує результат. Результат наведено на рис.26.

```
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)
```

Рисунок 26 – Результат виконання команди pingall

Для того, щоб провести захват трафіку потрібно відправити якісь пакети по мережі. Зробимо це шляхом пінгування одного вузла іншим. Приклад наведено на рис.27:

```

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=0.224 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.056 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.083 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.056 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5124ms
rtt min/avg/max/mdev = 0.053/0.088/0.224/0.061 ms
mininet>

```

Рисунок 27 – Результат виконання команди h1 ping h2

На рисунку 27 можна бачити, що було успішно відправлено icmp пакети, які можна захопити аналізатором трафіку Wireshark. Перейдемо до додатку Wireshark, захоплений трафік продемонстровано на рис.28.

No.	Time	Source	Destination	Protocol	Length	Info
39	8.077419290	127.0.0.1	127.0.0.1	OpenFl...	233	Type: OFPT_PACK
40	8.078571825	127.0.0.1	127.0.0.1	OpenFl...	231	Type: OFPT_PACK
41	8.078583134	127.0.0.1	127.0.0.1	TCP	66	34590 → 6653 [A
42	8.078949003	127.0.0.1	127.0.0.1	OpenFl...	233	Type: OFPT_PACK
43	8.080128397	127.0.0.1	127.0.0.1	OpenFl...	231	Type: OFPT_PACK
44	8.080139386	127.0.0.1	127.0.0.1	TCP	66	34592 → 6653 [A
45	8.425331284	127.0.0.1	127.0.0.1	OpenFl...	178	Type: OFPT_PACK
46	8.426523433	127.0.0.1	127.0.0.1	OpenFl...	176	Type: OFPT_PACK
47	8.426536337	127.0.0.1	127.0.0.1	TCP	66	34594 → 6653 [A
48	8.427044517	127.0.0.1	127.0.0.1	OpenFl...	178	Type: OFPT_PACK
49	8.428289378	127.0.0.1	127.0.0.1	OpenFl...	176	Type: OFPT_PACK

Frame 48: 178 bytes on wire (1424 bits), 178 bytes captured (1424 bits) on interface lo, id 0 Interface id: 0 (lo) Encapsulation type: Ethernet (1) Arrival Time: May 29, 2021 20:12:42.186533502 EEST [Time shift for this packet: 0.000000000 seconds] Epoch Time: 1622308362.186533502 seconds [Time delta from previous captured frame: 0.000508180 seconds] [Time delta from previous displayed frame: 0.000508180 seconds] [Time since reference or first frame: 8.427044517 seconds] Frame Number: 48 Frame Length: 178 bytes (1424 bits) Capture Length: 178 bytes (1424 bits) [Frame is marked: False] [Frame is ignored: False] Protocols in frame: eth:ethertype:ip:tcp:openflow:openflow_v4:eth:ethertype:ipv6:icmpv6 Coloring Rule Name: ICMP						
--	--	--	--	--	--	--

Рисунок 28 – Результат захоплення трафіку

На рисунку 28 можна побачити, що були захвачені пакети протоколу OpenFlow (виділене червоним) типу ICMP (виділене жовтим), а отже мережа працює коректно.

ВИСНОВКИ

В бакалаврському проєкті була розроблена та змодельована структура невеликої програмно-конфігуровної мережі на основі SDN контролера та протоколу OpenFlow. Було розглянуто та продемонстровано роботу гнучкої користувацької топології розробленої за допомогою мови програмування Python. Також на прикладі було ознайомлено з роботою SDN контролера на базі RYU, описано встановлення та запуск на машині Ubuntu та наглядно показано запуск мережевого додатку. Результатом виконаної роботи стала успішно розгорнута мережа, що демонструється захопленням трафіку. У роботі проаналізовано та розглянуто принципи функціонування, особливості роботи, архітектуру, переваги та недоліки технології SDN.

Показано, що SDN – це інструмент для полегшення вирішення проблем мережевого управління на основі розробки програм. Технологія SDN мереж має суттєві переваги у порівнянні із класичними мережами, зокрема: широкі можливості управління із підвищеною швидкістю та гнучкістю, можливість налаштування мережевої інфраструктури, високий рівень безпеки.

Прикладне застосування SDN різноманітне, починаючи від управління конфігурацією і автоматизацією за допомогою програмування мережевих пристроїв із зовнішніх додатків (контролерів), до використання бізнес або мережевих додатків в тісній інтеграції із мережевими пристроями для вирішення задач в окремій галузі.

Моделювання програмно-конфігуровних мереж дає безліч можливостей: запобігання дорогих помилок при розвертанні великих та складних мереж, великий простір для проведення експериментів, що сприяє розвиненню технології, а також гарну базу для підготовки майбутніх мережевих спеціалістів. Отже дану роботу можна розглядати як опорний матеріал при навчанні та виконанні лабораторних робіт.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Модель OSI [Електронний ресурс]. – Режим доступу: http://neerc.ifmo.ru/wiki/index.php?title=OSI_Model
2. Маршрутизація [Електронний ресурс]. – Режим доступу: <https://uk.wikipedia.org/wiki/Маршрутизація>
3. Р.Л. Смелянский, профессор МГУ им. Ломоносова, директор по науке и образованию Центра прикладных исследований компьютерных сетей. Статья для журнала "Вестник связи"/январь, 2014, *Технологии SDN и NFV: новые возможности для телекоммуникаций*
4. Сетевые технологии SDN – Software Defined Networking [Електронний ресурс]. – Режим доступу: <https://habr.com/ru/company/muk/blog/251959/>
5. Словопедія [Електронний ресурс]. – Режим доступу: <http://slovopedia.org.ua/38/53397/379331.html>
6. Панеш А.Х. Содержание и перспективы технологий программно-конфигурируемых сетей и виртуализации сетевых функций // Вестник Адыгейского государственного университета. Сер. Естественно-математические и технические науки. 2014. Вып. 2 (137). С. 120–127.
7. OpenFlow: маршрутизация или коммутация? [Електронний ресурс]. – Режим доступу: <https://shalaginov.com/2019/10/21/6534/>
8. Кому и для чего нужна технология SDN? [Електронний ресурс]. – Режим доступу: https://ko.com.ua/komu_i_dlya_chego_nuzhna_tehnologiya_sdn_80473
9. Основні принципи роботи мережі SDN та її переваги [Електронний ресурс]. – Режим доступу: <https://prezi.com/pmmqvi9yumth/sdn/>

10. Панеш А.Х. Достоинства и недостатки программно-конфигурируемых компьютерных сетей // «Вестник АГУ». 2016. Вып. 3 (186). С. 109-113.
11. Видео курс по установке и работе с виртуальной лабораторией EVE-NG Версия v1.0 [Электронный ресурс]. – Режим доступа: https://learncisco.ru/eve/eve_description.html
12. Мининет SDN [Электронный ресурс]. – Режим доступа: <https://russianblogs.com/article/3305821827/>
13. Mininet — эмулятор компьютерной сети [Электронный ресурс]. – Режим доступа: <https://ivirt-it.ru/mininet/>
14. Software Defined Networks using Mininet, Pramod B Patil., Kanchan S. Bhagat., D K Kirange, S. D. Patil [Электронный ресурс]. – Режим доступа: <https://www.ijrte.org/wpcontent/uploads/papers/v9i1/F9375038620.pdf>
15. Контролер POX [Электронный ресурс]. – Режим доступа: <https://github.com/noxrepo/>
16. Практическое руководство по программно-конфигурируемой сети [Электронный ресурс]. – Режим доступа: https://ozlib.com/950189/tehnika/prakticheskoe_rukovodstvo_programmno_konfiguriruemoy_seti
17. Mininet as Software Defined Networking Testing Platform [Электронный ресурс]. – Режим доступа: <http://www.sbsstc.ac.in/icccs2014/Papers/Paper29.pdf>