МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна інженерія»

Група: 2БКС-29

КВАЛІФІКАЦІЙНА РОБОТА

здобувача освіти денної форми навчання

БКС.29.07.000.КРБ

ДЕДІГУРОВА
МИКИТИ
ОЛЕКСАНДРОВИЧА

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: **«Комп'ютерна інженерія»**

Група: **2БКС-29**

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему: **Моделювання потоків даних у програмно-визначуваних мережах**

Проектний матеріал складається з пояснювальної записки на 72 сторінках та графічного (презентаційного) матеріалу на 16 аркушах (слайдах)

Виконавець _____ (Дедігуров М.О.)

Керівник проекту _____ (Суліма Ю.Ю.)

Консультанти:

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувач кафедри _____ (Іванова Л.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «25» 06 2025 р. Протокол ЕК № 1

Оцінка ЕК 5 (відмінно) / 95

Секретар ЕК _____

АНОТАЦІЯ

У кваліфікаційній роботі розглядається питання моделювання потоків даних у програмно-визначуваних мережах і застосовується інтегрований підхід до оптимізації маршрутизації мережевого трафіку. Основна мета роботи – створити централізоване алгоритмічне рішення для моделювання потоків даних в умовах динамічних змін топології мережі.

У рамках проекту проведено ґрунтовний аналіз сучасних методів моделювання мережевих потоків із застосуванням математичного моделювання та алгоритмічного аналізу. Особливу увагу приділено формуванню повної матриці зв’язності мережі шляхом дублювання пів-матриці суміжності, що дозволить точно відобразити всі взаємозв’язки між вузлами. За допомогою централізованого алгоритму Лі побудовано множину маршрутних шляхів, що оптимізують процес передачі даних, забезпечуючи ефективне використання мережевих ресурсів і високий рівень пропускної здатності.

Робота передбачає розробку алгоритмічного та програмного забезпечення з візуальним графічним інтерфейсом, яке забезпечить зручне введення вихідних параметрів, автоматизоване моделювання потоків та наочну візуалізацію результуючих маршрутів. Застосування методів централізованого планування обміну керуючою інформацією між SDN-контролером і маршрутизаторами дозволить оперативно адаптувати маршрутизацію до змін у мережевому навантаженні.

Запропонований підхід, що ґрунтується на поєднанні математичного аналізу, алгоритмічного моделювання та сучасних засобів візуалізації, має сприяти підвищенню ефективності управління потоками даних у програмно-визначуваних мережах та створенню умов для оптимізації роботи сучасних телекомунікаційних систем.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *Комп'ютерних систем* Кафедра *Комп'ютерної інженерії*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"28" 05 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

здобувачеві освіти *Дедігурову Микиті Олександровичу*

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Моделювання потоків даних у програмно-визначуваних мережах*

затверджена наказом по коледжу від "24" 11 2024 р. № 246

2. Термін здачі студентом кваліфікаційної роботи 20.06.25 р.

3. Вихідні дані до роботи *1. Специфікації SDN-мережі; 2. Вимоги до оптимізації потоків даних у мережі; 3. Технічна документація, теоретичні та статистичні дані програмно-визначуваних мереж; 4. Модифікувати алгоритм Лі та виконати побудову графу SDN-мережі; 5. Виконати програмну реалізацію побудови графу з візуалізацією; 6. У якості початкових даних використовувати таблиці зв'язності; 7. Програмну реалізацію виконати мовою Java*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
Аналітичний огляд алгоритмів моделювання потоків даних у програмно-визначуваних мережах; Планування процедури обміну керуючою інформацією між SDN-контролером і маршрутизаторами; Формування маршрутних шляхів для SDN-мережі; Моделювання потоків даних за модифікованим алгоритмом; Розробка візуального інтерфейсу додатку для моделювання; Отримання результатів моделювання та їх аналіз

5. Перелік графічного матеріалу (слайдів мультимедійної презентації) *Архітектура програмно-визначуваних мережах; Принцип передачі пакетів у програмно-визначуваних мережах; Базові інтерфейси у програмно-визначуваній мережі; Таблиця порівняння існуючих алгоритмів розподілення потоків даних; Процедура обміну службовою інформацією у програмно-визначуваній мережі; Маршрутний граф програмно-визначуваної мережі; Таблиця маршрутизації; БСА побудови графа; Об'єктно-орієнтована схема додатку; Діаграма взаємодії користувача і додатку; Скріншоти роботи додатку з результатами моделювання шляхів*

6. Консультанти по кваліфікаційній роботі, із зазначенням розділів, що їх стосуються

Розділ	Консультант	ПІДПИС	
		Завдання видав	Завдання прийняв
Основний розділ	*Суліма Ю.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 6.05.25

Керівник роботи *Суліма Ю.Ю.* _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Вступ. Аналіз технічного завдання	08.06.25	
2.	Огляд методів моделювання потоків даних	09.06.25	
3.	Аналіз способів вирішення проблеми маршрутизації	10.06.25	
	у програмно-визначуваних мережах		
4.	Планування процедури обміну керуючою інформацією	11.06.25	
	між SDN-контролером і маршрутизаторами		
5.	Формування маршрутних шляхів для SDN-мережі	12.06.25	
6.	Оцінка обчислювальної складності алгоритму	13.06.25	
7.	Моделювання потоків даних	14.06.25	
8.	Вибір і аналіз засобів розробки застосунку	15.06.25	
9.	Розробка програмного забезпечення для моделювання	16.06.25	
10.	Тестування розробленого програмного забезпечення	17.06.25	
11.	Аналіз результатів тестування	18.06.25	
12.	Розробка питань з охорони праці та техніки безпеки	19.06.25	
13.	Підготовка матеріалів мультимедійної презентації	20.06.25	

Здобувач освіти _____
(підпис)

Керівник роботи _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 29. 07 000. 00 КРБ	Випускна робота		
А4			БКС 29. 07 000. 00 КРБ ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	БКС 29. 07 000. 00 КРБ		
Розробив		Дедігуров М.О.		20.06	Моделювання потоків даних у програмно-визначуваних мережах	Літ.	Аркуш 4 / Аркушів 72
Перевірив		Суліма Ю.Ю.		20.06			
Н. Контр.		Петрашова В.І.		16.06.25		ВСП "ОТФК ОНТУ" гр.2БКС-29	
Затверд.		Іванова Л.В.					

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого зростання обсягів даних та високолінійної комп’ютеризації суспільства питання ефективного управління мережами набуває особливої актуальності. Програмно-визначувані мережі (Software-Defined Networking, SDN) являють собою революційний підхід до організації мережевої інфраструктури, що дозволяє відділити логіку управління від фізичної реалізації мережевих пристроїв. Така архітектура забезпечує централізоване керування ресурсами мережі, дає змогу гнучко реагувати на зміни у трафіку, оптимізувати використання плодів мережевих ресурсів та значно скоротити витрати на експлуатацію.

Однією з ключових проблем, що стоять перед сучасними SDN-системами, є конструювання маршрутної інформації для потоків даних із забезпеченням заданих параметрів якості обслуговування (QoS). При цьому важливим аспектом є швидке формування численних альтернативних маршрутів доступу до мережевих ресурсів з мінімальними затримками та втратою пакетів, що дозволяє значно підвищити ефективність ремаршрутизації трафіку. Чим більше альтернативних маршрутів з’являється в централізованому SDN-контролері, тим менша ймовірність виникнення затримок або втрати пакетів, що значною мірою сприяє покращенню роботи мережі при непередбачених навантаженнях.

У даній випускній роботі буде досліджено метод моделювання потоків даних у SDN-мережах із застосуванням спеціального підходу до побудови маршрутної інформації. Запропонований спосіб конструювання трафіку буде мати на меті зниження вартості володіння мережевими інфраструктурами та зменшення залежності від окремих вендорів, що є особливо важливим для створення гнучких і адаптивних мережевих рішень.

Для досягнення поставленої мети треба реалізувати програмний застосунок мовою для моделювання та візуалізації мережевої топології. Отримані результати експериментального моделювання дозволять оцінити ефективність запропонованого підходу та його конкурентоспроможність у порівнянні з традиційними методами маршрутизації.

1 ОСНОВНИЙ РОЗДІЛ

Метою роботи є скорочення часу формування маршрутів доступу, спрощення процедуру ремаршрутизації та забезпечення більш рівномірного завантаження каналів передачі інформації у мережі, організованій за принципами SDN. Це дозволить підвищити продуктивність мереж, знизити рівень затримок під час ремаршрутизації.

1.1 Аналітичний огляд алгоритмів моделювання потоків даних у програмно-визначуваних мережах

В цьому підрозділі здійснюється огляд існуючих алгоритмів, які застосовуються для моделювання потоків у SDN-мережах. У науковій літературі запропоновано широкий спектр методів, які можна розділити на декілька основних груп. До першої групи відносяться алгоритми, що базуються на класичних підходах до маршрутизації, наприклад, методи оптимізації шляхів за критеріями мінімальних затримок або мінімальної вартості передачі даних. Ці алгоритми регулярно модифікуються з урахуванням особливостей централізованої обробки інформації у SDN-системах, що дозволяє зменшити час встановлення маршрутів та впровадити динамічні процедури ремаршрутизації. Іншою важливою групою є алгоритми, орієнтовані на аналіз і кластеризацію потоків даних. Вони використовують показники, такі як локальна щільність зв’язків між вузлами, відстань до найближчих вузлів із вищими значеннями щільності, а також інші метрики, що враховують параметри якості послуг (QoS). Результатом таких підходів є створення декількох альтернативних маршрутів, що дозволяє не лише мінімізувати затримки передачі, але й підвищити стійкість мережі у випадках збою окремих ланок або вузлів. Також варто відзначити, що останнім часом значну увагу приділяють алгоритмам, заснованим на евристичних і адаптивних методах, які дозволяють здійснювати моделювання на базі нейронних мереж і методів машинного навчання. Такі алгоритми здатні прогнозувати змінювані параметри мережі та автоматично коригувати конфігурацію маршрутів для зниження затримок і підвищення ефективності передачі даних.

1.1.1 Архітектура програмно-визначуваних мережах та принципи їх функціонування

Програмно-визначувані мережі (SDN) являють собою інноваційний підхід до організації та управління комп'ютерними мережами, в якому керуюча логіка відокремлена від апаратного забезпечення. Таке розділення дозволяє централізовано управляти мережею, що забезпечує гнучкість, масштабованість і спрощену конфігурацію мережевих пристроїв. У цьому підрозділі розглядається структура SDN-мережі, принципи передачі даних у традиційних мережах порівняно з SDN, архітектура типового мережевого пристрою, а також загальна архітектурна концепція SDN.

Рисунок 1.1. Принцип передачі пакетів у традиційній IP-мережі

У традиційних мережах кожен комутатор або маршрутизатор здійснює як функції передачі (передавальний рівень, data plane), так і функції прийняття рішень (керуючий рівень, control plane). Це означає, що вся логіка обробки пакетів — від прийому до маршрутизації — закладена безпосередньо у пристрої, що ускладнює масштабування та адаптацію мережі до змін навантаження. На рис.1.1 зображено, як пакет проходить через всі етапи обробки — від прийому до прийняття рішення.

Рисунок 1.2. Принцип передачі пакетів у програмно-визначуваних мережах

На відміну від традиційного підходу в SDN функції контролю централізовано зосереджені в одному або кількох SDN-контролерах. Пристрої

мережі, як-от комутатори або маршрутизатори, виконують лише завдання передачі даних (data plane) та взаємодіють з контролером через стандартний інтерфейс (наприклад, OpenFlow). Такий розподіл дозволяє централізовано управляти політикою мережі та оперативно реагувати на змінення в мережевому трафіку. На рис.1.2 показано, що пакет пересилається від пристрою до SDN-контролера для прийняття рішення, після чого отримує конкретні інструкції щодо подальшої обробки.

Типовий комутатор або маршрутизатор у традиційній мережі включає інтегрований набір компонентів:

- Контролюючий процесор (Control Processor): відповідає за прийняття рішень щодо маршрутизації пакетів, керування таблицями маршрутизації та реалізацію протоколів управління;

- Передавальний рівень (Data Plane): виконує операції з пересиланням пакетів згідно із заданими правилами, що зберігаються у таблицях швидкого доступу (наприклад, CAM);

- Інтерфейси: мікросхеми та порти, що забезпечують фізичне з’єднання з іншими пристроями мережі.

У SDN-середовищі архітектура комутатора зводиться до реалістичного пристрою передачі даних, у якого практично відсутня інтегрована логіка прийняття рішень. Відповідно, у нього є мінімальна апаратна інтеграція, орієнтована на швидке виконання команд від SDN-контролера. Це дозволяє значно спростити апаратну частину пристроїв, зосередити всю складну логіку у контролері та знизити загальні витрати на розгортання мережі.

Основна ідея архітектури SDN полягає у чіткому розділенні функцій між контролюючим рівнем та рівнем передачі даних. До ключових компонентів архітектури SDN відносяться:

- SDN-контролер (Control Plane): центральний елемент, який вирішує, як саме працюватиме мережа. Контролер отримує інформацію про поточний стан мережі, встановлює політики маршрутизації та видає команди пристроям для здійснення цих політик;

- Пристрої передачі даних (Data Plane): мережеві пристрої (комутатори, маршрутизатори), що виконують інструкції контролера для пересилання пакетів. Вони використовують таблиці потоків, згенеровані контролером, що забезпечує швидке прийняття рішень без власного аналізу;

- Північний інтерфейс (Northbound Interface): комунікаційний інтерфейс, за допомогою якого контролер взаємодіє з програмними додатками управління і orchestrators для отримання політик та моніторингу мережі;

- Південний інтерфейс (Southbound Interface): протоколи, що забезпечують зв'язок між контролером і пристроями передачі даних (наприклад, OpenFlow). Цей інтерфейс визначає, як пристрої отримують інструкції від контролера та повертають інформацію про стан мережі.

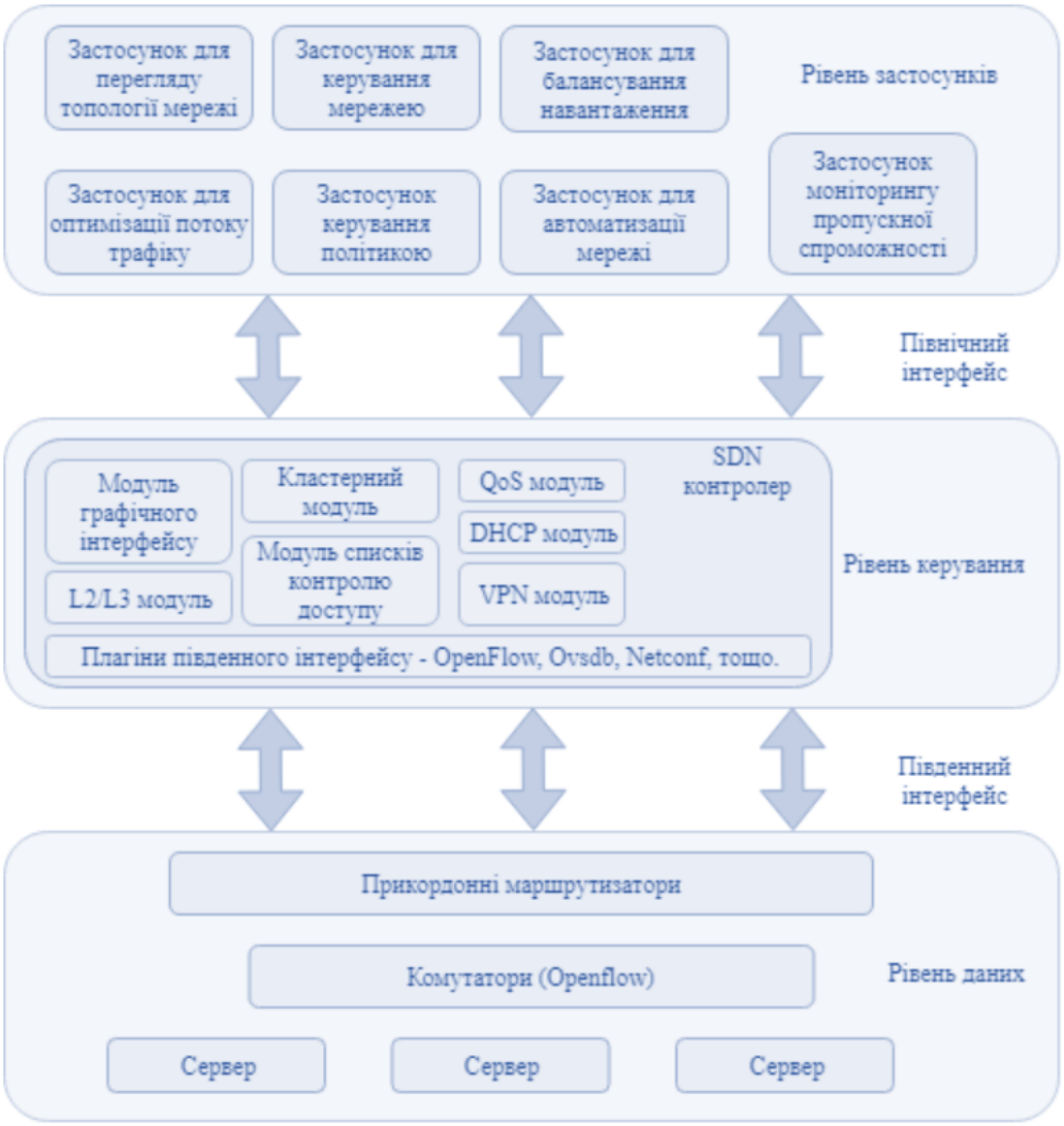


Рисунок 1.3. Архітектура програмно-визначуваних мереж

На рис.1.3 представлено SDN-комутатор, де керуюча логіка зосереджена в SDN-контролері. В результаті застосування SDN-архітектури досягається значне спрощення управління мережею, що дозволяє централізовано визначати множину маршрутів для потоків даних. Це сприяє зниженню затримок у передачі та ремаршрутизації, забезпечує більш рівномірне розподілення навантаження та підвищує загальну стійкість системи. Завдяки можливості формувати альтернативні маршрути у режимі реального часу SDN-контролер може оперативно реагувати на зміни у попиті на ресурси мережі або у випадках відмов окремих компонентів. Крім основних компонентів архітектури SDN, важливим аспектом є забезпечення безпеки та сумісності з постійно розвиваючимися мережевими технологіями. Таким чином, сучасні SDN-системи впроваджують механізми аутентифікації, шифрування та моніторингу, що є особливо важливими в умовах високих вимог до інтернет-безпеки. На рис.1.4 представлений комутатор, реалізований за принципом SDN: мінімальна логіка у пристрої, таблиці потоків та південний інтерфейс для комунікації з контролером.

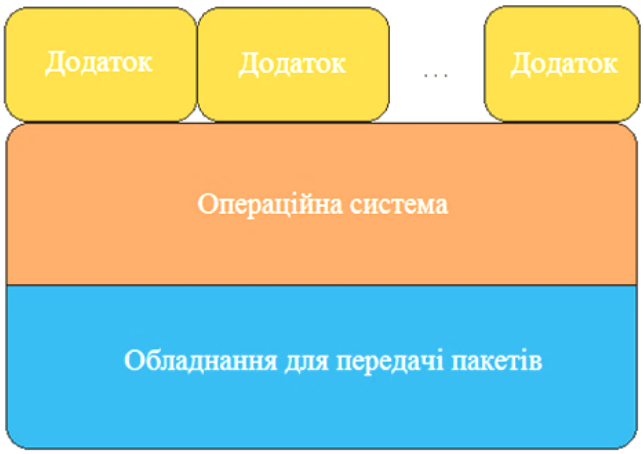


Рисунок 1.4. Архітектура типового SDN-комутатора/маршрутизатора

Таким чином, розділення функціональних частин мережі у SDN дозволяє спростити управління та швидко адаптувати мережу до мінливих умов. Центральний контролер забезпечує високий рівень оперативності при комунікації з мережевими пристроями, що сприяє мінімізації затримок передачі та втрат даних. Далі у роботі буде використано цю архітектуру як базу для моделювання потоків даних та дослідження ефективності конструювання маршрутної інформації з урахуванням вимог today QoS та інтернет-безпеки.

1.1.2 Компоненти програмно-визначуваної мережі

Архітектура програмно-визначуваних мереж (SDN) ґрунтується на розділенні функціональних площин, що дозволяє централізовано управляти мережею та забезпечує гнучкість у розподілі ресурсів. Основні компоненти SDN включають додатки SDN, SDN-контролер, SDN Datapath, інтерфейси зв'язку (як південні, так і північні) та відповідні драйверно-агентські пари, що забезпечують обмін даними між різними площинами. Нижче наведено детальний опис кожного з компонентів.

Додатки SDN – це інтелектуальні програми, які безпосередньо формулюють мережеві вимоги та задають бажану поведінку мережі. Вони взаємодіють із SDN-контролером через північні інтерфейси (Northbound Interfaces, NBI), отримуючи абстрактне уявлення про мережу, включаючи статистику та інформацію про події (рис.1.5). Процес розробки додатків SDN передбачає використання логіки, що відокремлюється від апаратних засобів, для здійснення завдань з управління доступом, балансування навантаження, оптимізації маршрутів і забезпечення політик якості обслуговування (QoS).

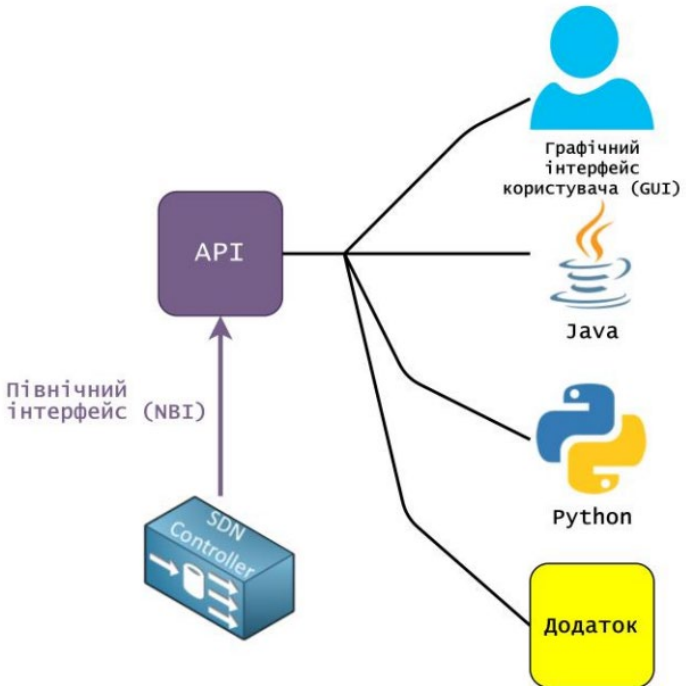


Рисунок 1.5. Схема взаємодії додатків SDN із контролером через NBI

Контролер SDN є логічно централізованою сутністю, яка виконує дві основні функції:

- Переклад мережевих вимог додатків у конкретні інструкції для пристроїв передачі даних (Data Plane);

- Надання абстрактного уявлення про стан мережі додаткам, що дозволяє здійснювати моніторинг і прийняття рішень.

Контролер складається із кількох складових (рис.1.6): агентів NBI, логіки управління та драйвера SDN Control to Data-Plane Interface (CDPI). Хоча контролер логічно централізований, фізична реалізація може включати об'єднання декількох пристроїв, ієрархічне розподілення або віртуалізацію з метою масштабованості та забезпечення високої доступності.

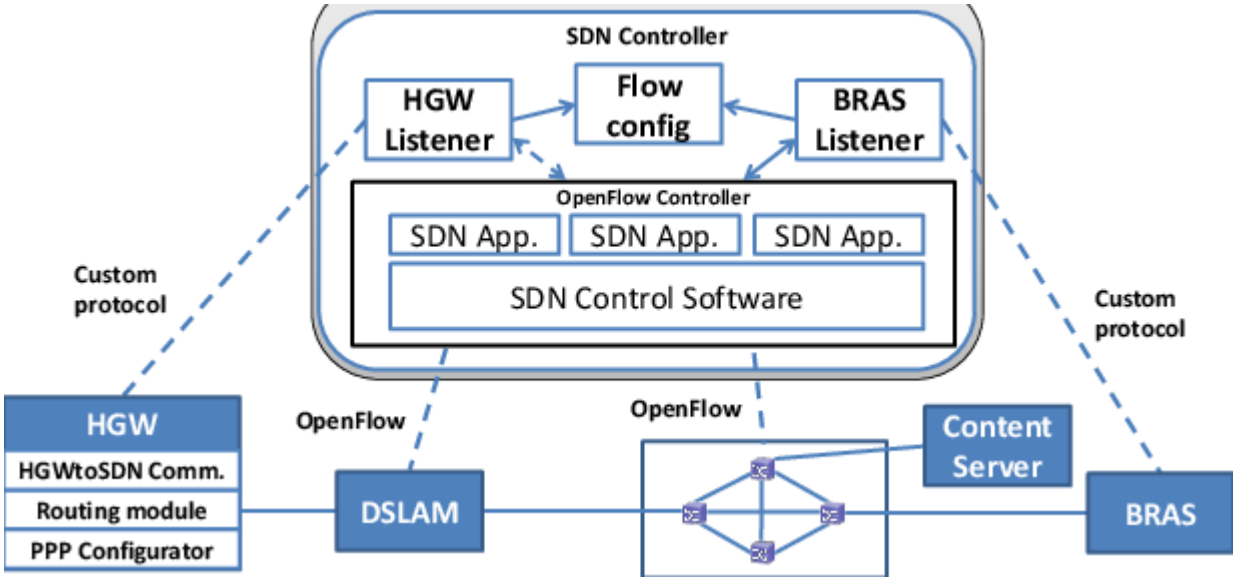


Рисунок 1.6. Загальна схема SDN-контролера з виділенням його підсистем

SDN Datapath (рис.1.7) представляє логічний мережевий пристрій, який відповідає за пересилання даних (Data Plane). Він може включати частину або всі фізичні ресурси мережевої основи, але, головне, виконує лише функції обробки та переадресації пакетів. До його основних складових входить агент CDPI, що забезпечує зв'язок з контролером, та набір механізмів переадресації, які можуть варіюватися від простого пересилання між зовнішніми інтерфейсами до складних внутрішніх операцій із обробки і завершення трафіку.

SDN CDPI – це інтерфейс, що встановлює зв'язок між SDN-контролером і пристроями Data Plane. Він забезпечує програмне керування операціями пересилання, оголошення можливостей, повернення статистики та повідомлення

про події. Важливою властивістю CDPI є його відкритий та нейтральний до вендорів характер, що сприяє сумісності пристроїв різних виробників у єдиній мережі.

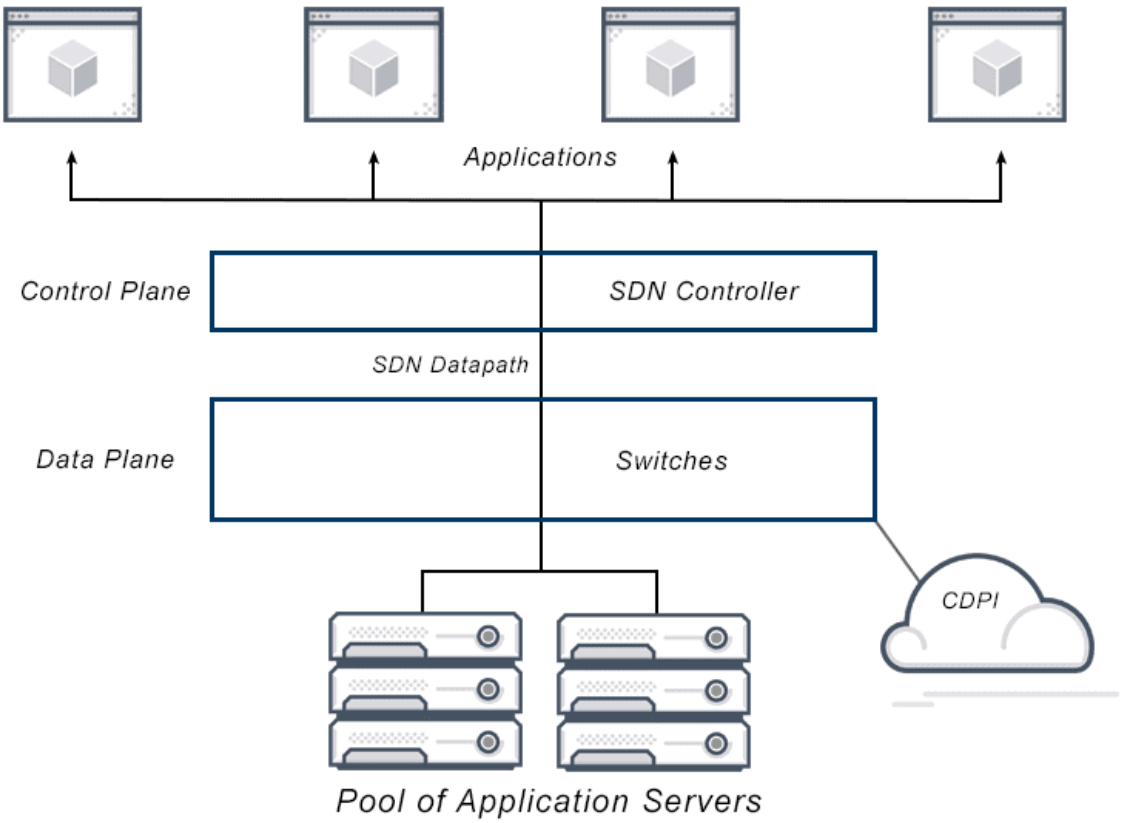


Рисунок 1.7. Схема SDN Datapath

Північні інтерфейси SDN (NBI) забезпечують зв'язок між додатками SDN та SDN-контролером. Вони дозволяють додаткам безпосередньо формулювати свої мережеві вимоги та отримувати абстрактну інформацію про мережу. Завдяки цим інтерфейсам, розробники можуть створювати керуючі програми високого рівня, не концентруючись на специфічних деталях реалізації мережевих пристроїв. Як і CDPI, NBI має бути реалізовано у відкритій та вендоронезалежній манері, що сприяє універсальності застосування SDN.

Кожен інтерфейс у SDN реалізується за допомогою пари драйвер-агент. Агенти представляють «південну» частину, орієнтовану на взаємодію з фізичною інфраструктурою, тоді як драйвери – «північну» частину, яка інтегрується з додатками. Такий підхід ізолює специфіку апаратного забезпечення від логіки додатків, забезпечуючи стандартизовану платформу для розвитку мережевих сервісів і забезпечення сумісності різних виробників.

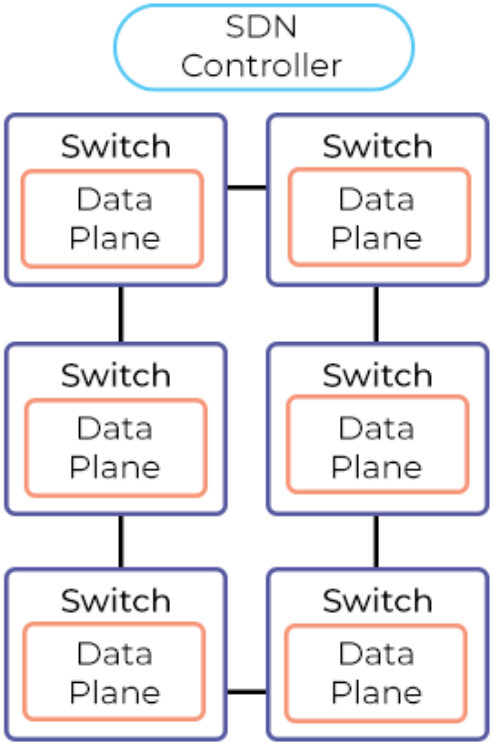


Рисунок 1.8. Схема CDPI у SDN

Площина управління у SDN охоплює завдання, які традиційно виконувалися окремо від операцій передачі даних. Сюди входять управління бізнес-відносинами між постачальниками та клієнтами, конфігурація мережевого обладнання, розподіл ресурсів, облік та координація між логічними і фізичними компонентами мереж. завдяки централізованому контролеру SDN ці завдання можуть бути інтегровані у єдину систему, що спрощує адміністрування та підвищує оперативність прийняття рішень.

У традиційних мережах додатки частково задають свої мережеві вимоги, використовуючи заголовки пакетів для встановлення пріоритетів. Проте мережеві оператори зазвичай не враховують повний спектр вимог користувачів. SDN-додатки дозволяють повністю визначити потрібні параметри, наприклад, пропускну здатність, затримку й доступність ресурсів, що значно покращує адаптацію мережі до змін навантаження та вимог. Завдяки централізованому контролеру SDN можливе оперативне відстеження стану мережі і своєчасна зміна конфігурації для оптимального управління трафіком.

Реалізація наведеної архітектури на практиці дозволяє організаціям отримати вендоронезалежний контроль над мережевою інфраструктурою з

єдиного місця. Це значно спрощує конфігурування та адміністрування мереж, скорочує час впровадження нових сервісів і підвищує загальну стійкість мережі до збою окремих елементів. Основним елементом такої реалізації є протокол OpenFlow, який стандартизує взаємодію між контролером і мережевими пристроями, відкриваючи можливості інтеграції різних виробників в єдину SDN-систему.

1.1.3 Огляд алгоритмів моделювання потоків даних

Сучасні програмно-визначувані мережі (SDN) дозволяють централізовано управляти великими мережами, що відкриває можливості для ефективного моделювання потоків даних. Розробка алгоритмів для моделювання трафіку має на меті оптимізацію шляху маршрутизації, балансування навантаження мережі та забезпечення високої якості обслуговування (QoS). У цьому підрозділі розглядаються та порівнюються основні підходи до моделювання потоків даних, що застосовуються в SDN, із зазначенням ілюстративних схем для кожного методу.

Центральне формування маршрутів. Цей підхід ґрунтується на тому, що центральний SDN-контролер, який має повну інформацію про топологію мережі та її поточний стан, виконує оптимізацію маршрутів для кожного потоку даних. Основна ідея полягає у розбитті наскрізних політик на вкладені таблиці (або «палітру») та їх подальшому розподілі між комутаторами. Контролер максимально спрощує завдання пристроїв Data Plane, надаючи їм лише набір правил, за якими потрібно пересилати пакети.

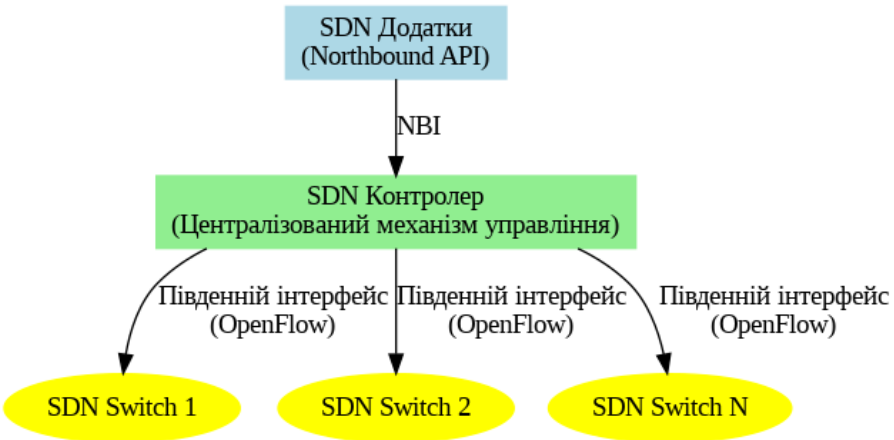


Рисунок 1.9. Схема центрального формування маршрутів у SDN

На схемі зображено, як центральний контролер отримує глобальну інформацію про мережу, розбиває загальні політики на вкладені таблиці та розподіляє оптимізовані правила по комутаторах. Це дозволяє мінімізувати кількість правил, які встановлюються в пристроях пересилання, і забезпечити швидку ремаршрутизацію трафіку.

Мультишляхова маршрутизація. Мультишляхова маршрутизація полягає у формуванні множини альтернативних шляхів між джерелом і призначенням. Основний підхід тут—одночасне створення не перетинаючихся маршрутів, що дозволяє забезпечити резервування і балансування навантаження. Такий підхід сприяє зниженню затримок і мінімізації втрат пакетів, оскільки трафік може автоматично перенаправлятись через менш завантажені канали.

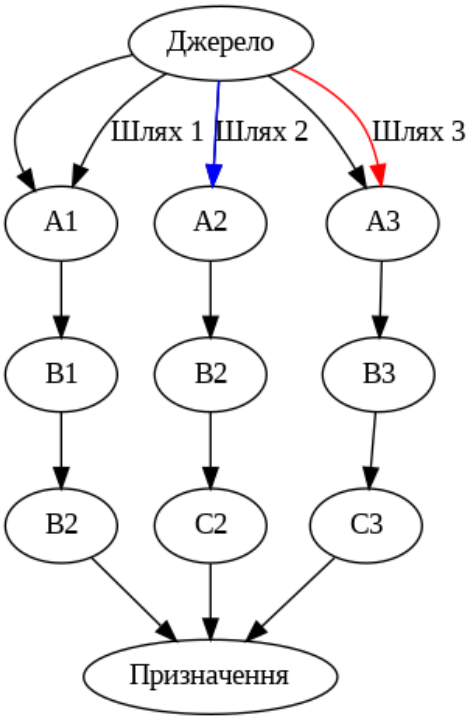


Рисунок 1.10. Схема мультишляхової маршрутизації

На схемі зображено джерело та призначення, між якими існують декілька паралельних шляхів, що не перетинаються. Кожен шлях характеризується власними параметрами затримки, пропускною здатністю та навантаженням, що дозволяє контролеру вибирати оптимальні маршрути в режимі реального часу.

Динамічна адаптація потоків. Цей підхід базується на постійному моніторингу стану мережі шляхом збору статистики використання портів та

пропускної здатності. При досягненні заданого порогу завантаження (наприклад, більше 70% використання ресурсу) система автоматично ініціює перерахунок маршрутів для потоків, що спрямовані через перевантажені сегменти. Динамічна адаптація дозволяє оперативно реагувати на зміни у трафіку та мінімізувати затримки і втрати даних.

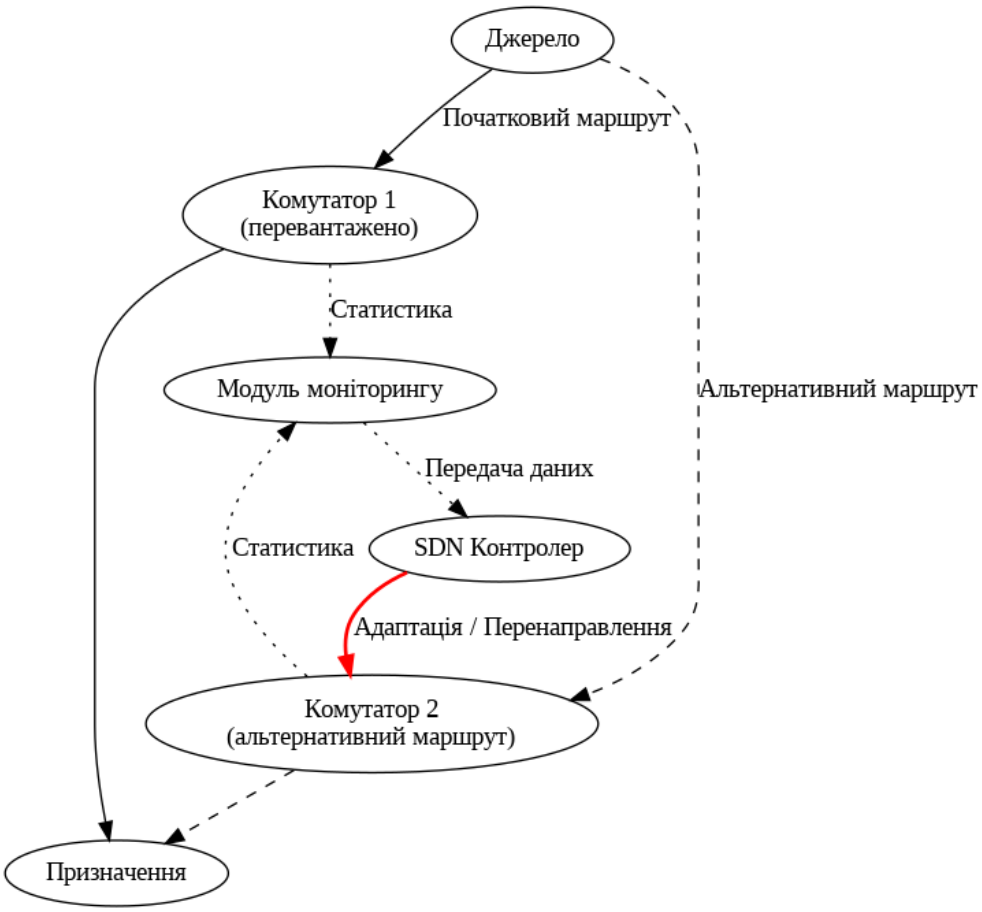


Рисунок 1.11. Схема динамічної адаптації потоків

На схемі показано моніторинг стану мережі, де контролер отримує інформацію про завантаження окремих портів. При перевищенні порогового значення контролер ініціює перерахунок і перенаправлення трафіку на альтернативні шляхи з більшими можливостями.

Гібридні алгоритми оптимізації маршрутизації. Гібридний підхід поєднує централізоване формування маршрутів із локальною оптимізацією через ітеративну корекцію. Спочатку центральний контролер формує базовий набір маршрутів на основі глобальної топологічної інформації. Далі у кожному класі альтернативних шляхів проводиться локальна оптимізація, де окремі сегменти

маршрутів можуть бути скориговані за допомогою локальних алгоритмів, що враховують додаткові параметри, як-от актуальне навантаження, затримки або втрата пакетів.

Рисунок 1.12. Схема гібридного алгоритму оптимізації

На схемі централізований контролер формує початкові маршрути, після чого локальні підсистеми здійснюють корекцію окремих ділянок маршруту, забезпечуючи оптимальний баланс параметрів на глобальному рівні.

Багатошляхова маршрутизація в мережевих центрах даних. Сучасні мережі центрів обробки даних (DCN) характеризуються великою розмірністю та різноманітністю обладнання, включаючи мобільні точки доступу та пристрої бездротового зв'язку. Управління такими мережами ускладнюється через високий рівень динамічності та зміни навантаження. Програмно-конфігурована мережева технологія (SDN) дає змогу інтегрувати централізований контроль за топологією мережі з можливістю організації багатошляхової маршрутизації. Основні принципи:

- Врахування топології: При формуванні альтернативних шляхів враховується специфічна організація мереж, що дозволяє оптимізувати кількість неперетинаючих маршрутів;

- Динамічне управління: Моніторинг доступної пропускної здатності та інформації про працездатність посилань дозволяє SDN-контролеру в режимі реального часу переналаштовувати маршрути для зниження перевантаження;

- Підвищення відмовостійкості: Наявність множини альтернативних шляхів

забезпечує більш стабільну і безпечну передачу трафіку, що особливо актуально для потокових медіа-додатків і обчислювальних центрів.

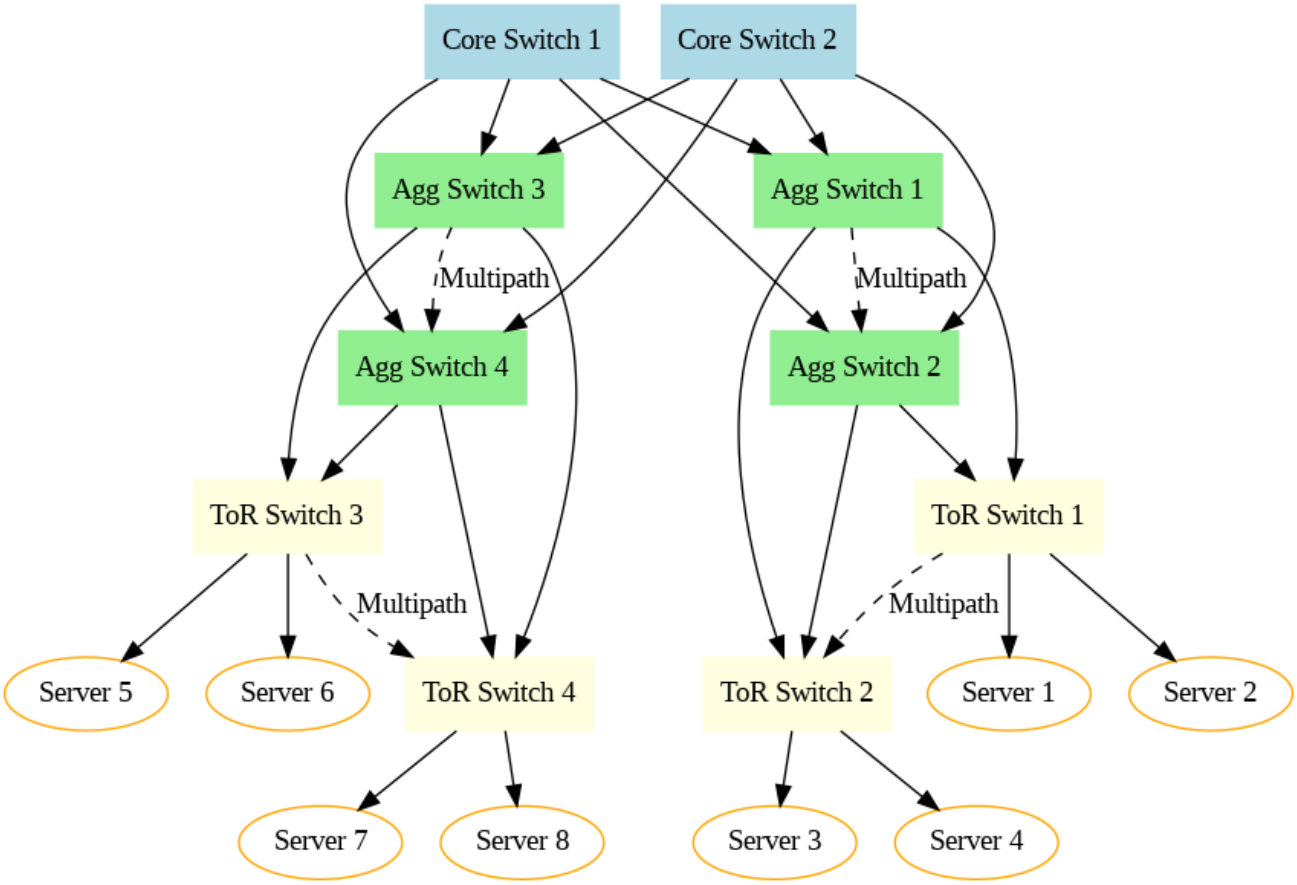


Рисунок 1.13. Схема багатошляхової маршрутизації в мережевих центрах даних

На схемі зображено оверлейну мережу, що складається з трьох типів вузлів: точок входу, проміжних вузлів і граничних точок. Контролер, який знаходиться у центрі, керує ними через спеціалізовані IP-тунелі, забезпечуючи розподіл та балансування потоків даних у мережі. Цей підхід дозволяє використовувати популярні топології, такі як Fat Tree для DCNs, які завдяки своїй самоподібності трансформуються в мережі Dragonfly для великих систем. Завдяки такій структурі забезпечується висока пропускна здатність, надійність та ефективне розподілення ресурсів, що є критичними вимогами для сучасних центрів обробки даних.

Проведений огляд дозволяє зробити наступні висновки:

- Централізоване формування маршрутів забезпечує високу оптимізацію за рахунок використання повної інформації про мережу, але може бути менш гнучким у випадку різких зміни потоків через централізований характер управління;

- Мультишляхова маршрутизація забезпечує підвищену резервованість і балансування навантаження, однак її реалізація потребує додаткових ресурсів для координації паралельних шляхів;

- Динамічна адаптація потоків дозволяє оперативно реагувати на зміну стану мережі, проте може збільшувати обчислювальні витрати при частих переналаштуваннях;

- Гібридні алгоритми поєднують переваги централізованої оптимізації з гнучкістю локальної адаптації, що забезпечує ефективне вирішення задачі, але їх реалізація є більш складною з точки зору алгоритмічної логіки та інтеграції.

1.2 Планування процедури обміну керуючою інформацією між SDN-контролером і маршрутизаторами

У даній роботі пропонується спосіб моделювання трафіку (Traffic Engineering, TE) на основі поєднання переваг централізованих і децентралізованих підходів до маршрутизації. Завдяки використанню SDN-технології, централізований контролер отримує деталізовану інформацію про стан каналів мережі, що дозволяє формувати множини шляхів між різними роутерами. При цьому існує можливість одночасного формування не лише прямого шляху між двома віддаленими вершинами, а й шляхів для їхніх внутрішніх сполучень. Такий підхід дозволяє істотно зменшити часову складність обчислювального процесу за рахунок повторного використання розрахунків для внутрішніх сегментів вже сформованих шляхів.

Основні етапи процедури обміну керуючою інформацією:

1. Збирання інформації про стан каналів. Кожен мережевий роутер (або комутатор), який перебуває на периферії мережі, регулярно передає службову інформацію про свій стан (наприклад, навантаження, пропускну здатність, затримку, показники використання портів тощо) до SDN-контролера. Ця інформація надходить за допомогою стандартних південних інтерфейсів (CDPI) та дозволяє контролеру мати актуальне уявлення про стан кожного елемента мережі;

2. Формування множин шляхів. На основі отриманих даних контролер SDN

застосовує модифікований спосіб формування множин шляхів для встановлення оптимальних маршрутів між різними роутерами. Особливістю даного підходу є те, що при формуванні шляху між двома віддаленими вершинами одночасно формується множина шляхів між їхніми внутрішніми точками з'єднання. Це дозволяє:

- Зменшити часову складність обчислень, оскільки внутрішні сегменти вже враховуються при первинному розрахунку;

- Покращити адаптивність маршрутизації, оскільки в разі зміни стану певного сегмента мережі не потрібно знову рахувати повний шлях, а можна використовувати попередньо обчислені внутрішні маршрути;

3. Оновлення маршрутної інформації. Після обчислення оптимальних маршрутів SDN-контролер здійснює корегування таблиць маршрутизації у відповідних мережевих пристроях (роутерах/комутаторах). Цей етап забезпечує динамічну реконфігурацію трафіку, що дозволяє рівномірно розподілити навантаження по мережі. Оновлення відбувається за допомогою північних інтерфейсів (NBI), через які контролер надсилає команди для внесення змін до конфігураційних параметрів пристроїв.

4. Динамічна адаптація та повторне формування шляхів. У процесі експлуатації мережі деякі маршрути можуть бути перевантажені або їхня ефективність може знизитися внаслідок зміни умов експлуатації. У такому випадку контролер SDN за допомогою періодичного збирання службової інформації ініціює повторне формування множини шляхів з урахуванням нових умов каналу. Це дозволяє швидко переналагодити маршрутизацію до того, як виникнуть помітні затримки або втрати пакетів, забезпечуючи при цьому максимально рівномірне завантаження мережі.

На рисунку 1.14 представлено послідовність обміну службовою інформацією між передавальним роутером R_i та SDN-контролером у контексті Traffic Engineering:

- Роутер R_i спочатку надсилає службову інформацію (статистику використання, рівень завантаження, якість каналу) до контролера;

- Контролер SDN аналізує отримані дані, формує множину оптимальних маршрутів між відповідними роутерами, включаючи внутрішні зв'язки між вершинами сформованого шляху;

- Контролер видає оновлення (команди) комутаторам і роутерам для коригування таблиць маршрутизації, що дозволяє динамічно переналаштувати маршрути трафіку.

Поєднання централізованого формування множин маршрутів із можливістю повторного формування внутрішніх сегментів дозволяє:

- Скоротити час обчислення оптимальних маршрутів, оскільки повторне використання проміжних результатів знижує загальну обчислювальну складність.

- Підвищити адаптивність системи, забезпечуючи оперативне реагування на зміну стану мережі та зниження затримок у передачі даних.

- Забезпечити рівномірне завантаження мережі, що сприяє зниженню вартості володіння та експлуатаційних витрат.

Рисунок 1.14. Обмін інформацією між SDN-контролером і маршрутизаторами

На рис. 1.14 представлено наочну схему процедури обміну службовою інформацією, де видно, як потокова інформація від передавального роутера R_i

надходить до SDN-контролера, який у відповідь виконує оптимізацію маршрутів і відправляє оновлення до мережевих пристроїв. Цей механізм забезпечує централізоване управління мережею із можливістю швидкої адаптації до змін у навантаженні та оптимізації використання мережевих ресурсів.

У даній роботі буде реалізовано модифікований централізований алгоритм Лі формування множини шляхів на основі технології SDN. Основною перевагою цього способу є можливість одночасного формування множин шляхів між різними роутерами мережі. При формуванні шляху між двома віддаленими роутерами одночасно визначаються також маршрути між їхніми внутрішніми підрозділами (внутрішніми роутерами). Такий підхід дозволить суттєво зменшити часову складність формування множини маршрутів за рахунок повторного використання проміжних результатів для внутрішніх сегментів вже сформованого шляху.

В якості метрики для вибору шляху використовується кількість переходів (кількість хопів). Хоча ця метрика дозволяє сформувати шляхи мінімальної довжини, вона не оптимізує мережевий трафік у повному обсязі і не завжди забезпечує рівномірне завантаження мережі. Проте такий підхід значно спрощує процес розрахунку шляхів і дозволяє контролеру оперативно реагувати на змінення умов у мережі.

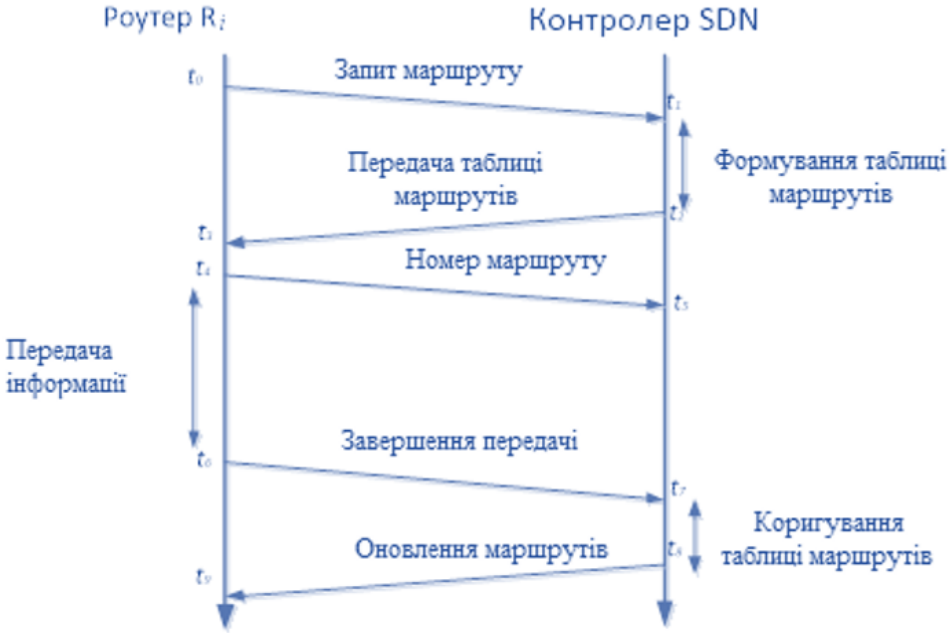


Рисунок 1.15. Процедура обміну службовою інформацією у SDN-мережі

За рис. 1.15 представлено послідовність обміну службовою інформацією між передавальним роутером і SDN-контролером у контексті Traffic Engineering:

- Момент часу t_0 – Ініціалізація запиту: Роутер, який виступає як джерело (обозначимо його як R_s), передає запит маршрутизації. Запит містить інформацію про номер роутера-відправника, номер роутера-одержувача (R_o) та задане значення параметра якості обслуговування (QoS). Цей запит надходить до SDN-контролера через південній інтерфейс, за допомогою якого здійснюється обмін службовою інформацією;

- Момент часу t_1 – Аналіз і формування множини шляхів: Після отримання запиту SDN-контролер проводить аналіз стану мережі та визначає наявність допустимих шляхів між роутерами. Якщо допустимих маршрутів немає, контролер за допомогою свого модифікованого протоколу маршрутизації формує множини допустимих шляхів. Особлива увага приділяється тому, що під час формування шляху між віддаленими роутерами одночасно визначаються маршрути між їхніми внутрішніми вузлами. Це дозволяє знизити загальну тимчасову складність розрахунків за рахунок повторного використання обчислених внутрішніх зв’язків;

- Момент часу t_2 – Оновлення маршрутної інформації: Після формування допустимих множин шляхів SDN-контролер передає сформовані таблиці маршрутів до роутера R_i (який може бути як початковим, так і проміжним елементом мережі). Серед отриманих шляхів роутер R_i вибирає маршрути з мінімальним завантаженням каналів (використання метрики кількості переходів допомагає скоротити кількість хопів, але при цьому завжди проводиться додатковий аналіз завантаження). Після вибору оптимального маршруту R_i повідомляє його номер SDN-контролеру. Роутер починає передачу інформації за обраним маршрутом до наступного суміжного вузла;

- Завершення циклу обміну: Після успішної передачі інформації відбувається коригування таблиць маршрутизації в усіх відповідних мережевих пристроях з метою динамічної реконфігурації трафіку. Контролер оновлює інформацію про маршрути в режимі реального часу, що дозволяє забезпечити

рівномірне розподілення навантаження та дотримання заданих параметрів QoS.

Запропонований модифікований централізований алгоритм Лі дозволяє, використовуючи переваги централізованого управління SDN, швидко і ефективно формувати множини маршрутів між компонентами мережі. Поєднання цього підходу із можливістю одночасного розрахунку маршрутів для внутрішніх вузлів значно зменшує часові витрати на планування мережевих маршрутів. Однак застосування метрики, що базується лише на кількості переходів, може не забезпечити повну оптимізацію мережевого трафіку та рівномірного завантаження; тому подальші дослідження можуть бути спрямовані на врахування додаткових параметрів, таких як затримка, пропускна здатність каналів і поточне завантаження, для покращення якості обслуговування.

Нижче наведено опис процесу передачі інформації, який реалізовано в запропонованому алгоритмі, а також покрокова схема його дії.

Передача інформації відбувається послідовно від передавального роутера R_i до суміжного роутера R_j у напрямку до кінцевого роутера R_a. Суміжний роутер R_j визначається на основі таблиці маршрутів $T_m(i)$ роутера R_i. Алгоритм передачі інформації включає наступні кроки:

1. Ініціалізація алгоритму передачі:

- Призначається початковий роутер R_s, який виступає як передавальний роутер R_i;

- Цей крок задає стартову точку процесу, звідки розпочинається маршрутизація даних;

2. Визначення наявності допустимого шляху:

- За допомогою таблиці маршрутів $T_m(i)$ роутер R_i перевіряє, чи існує шлях з допустимим значенням параметра якості обслуговування (QoS) до кінцевого роутера R_a;

- Якщо в таблиці містяться записи про допустимі маршрути, процес переходить до наступного етапу;

3. Запит до SDN-контролера (за необхідності):

- У випадку, якщо у таблиці маршрутів R_a відсутній допустимий шлях,

роутер R_i надсилає запит до SDN-контролера з проханням сформувати нові множини маршрутів, які відповідають вимогам QoS. Це дозволяє оперативно реагувати на змінення стану мережевих каналів та забезпечити оновлення маршрутної інформації;

4. Вибір оптимального маршруту:

- Серед усіх допустимих шляхів, визначених у таблиці $T_m(i)$ або сформованих контролером, роутер R_i обирає маршрут, що має мінімальне значення метрики (кількість переходів) та мінімальне завантаження каналів, що веде до роутера R_a;

- Використання даної метрики дозволяє отримати маршрут мінімальної довжини, хоча воно може не враховувати всі аспекти оптимізації мережевого трафіку;

5. Передача пакета даних:

- Після вибору оптимального маршруту, роутер R_i передає дані (пакет) до обраного суміжного роутера R_j, який є наступним елементом на шляху до R_a;

6. Ітерація циклу передачі:

- Якщо переданий пакет не є останнім у послідовності або маршрут не завершено до кінцевого роутера R_a, процес повторюється (починаючи з кроку 2) для наступного передавального вузла;

- Таким чином, алгоритм забезпечує послідовну передачу інформації від одного роутера до іншого на шляху до кінцевої точки;

7. Завершення процедури передачі:

- Процес завершується, коли останній пакет даних досягне роутера R_a або коли встановлено, що обрана маршрутизація відповідає зазначеним вимогам QoS.

Постійна перевірка таблиці маршрутів $T_m(i)$ дозволяє виявити, якщо існуючий маршрут не відповідає поточним вимогам, і ініціювати запит до контролера для формування нових маршрутів. Використання метрики «кількість переходів» допомагає формувати маршрути мінімальної довжини, що сприяє зниженню затримок. Незважаючи на це, у подальшому плану можливе розширення метрик для врахування пропускної здатності каналів та поточного

завантаження. Послідовна передача інформації з використанням актуальної таблиці маршрутів забезпечує рівномірний розподіл навантаження та дозволяє швидко адаптувати маршрути в умовах змін у мережі.

Алгоритм передачі інформації:

1. Запуск алгоритму:

Призначення початкового роутера R_s як передавального роутера R_i;

2. Визначення маршруту:

Роутер R_i аналізує свою таблицю маршрутів $T_m(i)$ для виявлення допустимого шляху до роутера R_a із заданими параметрами QoS;

3. Перевірка наявності допустимого шляху:

- Якщо допустимий шлях існує, переходимо до кроку 4;

- Якщо допустимого шляху немає, роутер R_i надсилає запит до SDN-контролера для формування нових маршрутів;

4. Вибір оптимального маршруту:

Серед усіх допустимих шляхів вибирається маршрут, який має мінімальну кількість переходів та найнижче завантаження каналів;

5. Передача пакета даних:

Роутер R_i передає пакет даних до обраного суміжного роутера R_j;

6. Перехід до наступного вузла:

Якщо пакет або його частина не є останньою, процес повторюється, і наступний крок визначається за таблицею маршрутів для R_j;

7. Завершення передачі:

Процедура закінчується, коли вся інформація досягає кінцевого роутера R_a.

1.3 Формування маршрутних шляхів для SDN-мережі

У даній роботі використовуються централізований алгоритм Лі формування множини маршрутних шляхів на основі технології SDN, що дозволяє ефективно та адаптивно конструювати мережеві маршрути між роутерами. Основною перевагою запропонованого підходу є можливість одночасного формування масштабної множини альтернативних шляхів між кількома роутерами мережі.

При обчисленні маршруту між двома віддаленими роутерами алгоритм одночасно визначає маршрути для внутрішніх сполучень цих вузлів. Така особливість дозволяє істотно зменшити часову складність у процесі формування маршрутів, адже обчислення для внутрішніх сегментів використовуються повторно, що сприяє швидкому оновленню інформації і мінімізує затримки.

На первинному етапі в якості основної метрики використовується кількість переходів, що дозволяє отримати маршрути мінімальної довжини. Проте дана метрика має обмеження, оскільки вона не враховує характеристики мережевого трафіку та не забезпечує рівномірного завантаження каналів. Для подолання цих недоліків у цій роботі застосовано комбіновану метрику, яка враховує не лише кількість переходів, але й параметри якості обслуговування (QoS) та завантаження каналів. Комбінована метрика враховує величину трафіку $S(L_{i,j})$, що передається по кожному каналу $L_{i,j}$, пропускну здатність $B(L_{i,j})$ і час затримки при передачі даних. З метою забезпечення рівномірного розподілу навантаження розраховується коефіцієнт завантаження $d_{i,j} = \frac{S(L_{i,j})}{B(L_{i,j})}$, при цьому загальне навантаження маршруту P_i визначається як максимальне значення $d_{i,j}$ серед усіх каналів на шляху. Отримане значення використовується при виборі оптимальних маршрутів з урахуванням як мінімізації кількості переходів, так і рівномірного розподілу навантаження по мережі.

Формування маршрутів здійснюється послідовно між роутерами, що належать до суміжних множин. Під суміжними множинами розуміються множини роутерів, що мають спільні канали зв'язку. Процес маршрутизації розпочинається з роутера-адресата R_s, для якого встановлюється множина $W_1 = R_s$. Множина $W_2 = R_j$ формується як набір роутерів, суміжних з R_s. Далі для кожного роутера з W_2 формуються таблиці маршрутів у напрямку до кінцевого роутера R_a. На кожній подальшій хвилі маршрутизації з роутерів однієї суміжної множини визначаються шляхи до роутерів попередньої множини. Цей процес триває до тих пір, поки не будуть сформовані всі можливі шляхи між R_s та R_a. На рис. 1.16 наведено граф комп'ютерної мережі, який демонструє основну топологію,

використовувану у процесі формування маршрутних шляхів.

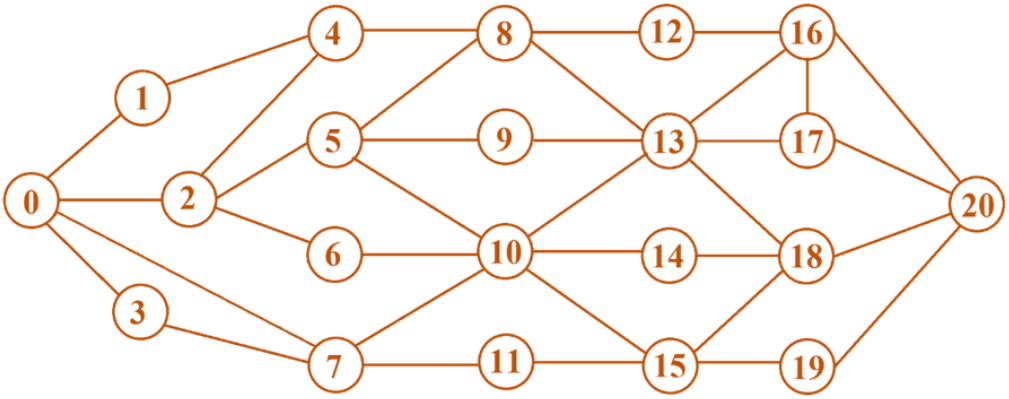


Рисунок 1.16. Маршрутний граф SDN-мережі

За модифікованим алгоритмом маршрутизації формується множина шляхів, що можуть незначно відрізнятися за суміжними сегментами. На рис. 1.17 представлено множину сформованих шляхів, наприклад, між роутерами R_0 та R_{20} з максимальною метрикою, що демонструє варіативність вибору маршруту у межах загальної оптимізації. Така варіативність дозволяє в процесі передачі інформації оперативно змінювати маршрут (як це показано на рис. 1.18) без додаткової затримки або втрати пакетів.

[0, 8, 11, 14, 18, 20] Metric = 19.5 MBIT/S Weight = 0.42
[0, 8, 11, 14, 18, 20] Metric = 19.5 MBIT/S Weight = 0.53
[0, 8, 11, 15, 19, 20] Metric = 19.5 MBIT/S Weight = 0.59
[0, 8, 12, 16, 19, 20] Metric = 19.5 MBIT/S Weight = 0.63
[0, 8, 12, 16, 19, 20] Metric = 19.5 MBIT/S Weight = 0.68
[0, 8, 12, 16, 19, 20] Metric = 19.5 MBIT/S Weight = 0.68
[0, 8, 12, 16, 19, 20] Metric = 19.5 MBIT/S Weight = 0.68
[0, 3, 6, 9, 14, 18, 20] Metric = 17.0 MBIT/S Weight = 0.43
[0, 3, 6, 9, 14, 18, 20] Metric = 17.0 MBIT/S Weight = 0.44
[0, 4, 7, 10, 15, 18, 20] Metric = 17.0 MBIT/S Weight = 0.45
[0, 4, 7, 10, 15, 18, 20] Metric = 17.0 MBIT/S Weight = 0.45
[0, 4, 7, 10, 15, 18, 20] Metric = 17.0 MBIT/S Weight = 0.45
[0, 3, 6, 9, 14, 17, 20] Metric = 17.0 MBIT/S Weight = 0.47
[0, 3, 6, 9, 14, 17, 20] Metric = 17.0 MBIT/S Weight = 0.47
[0, 4, 7, 10, 15, 17, 20] Metric = 17.0 MBIT/S Weight = 0.50

Рисунок 1.17. Сформовані маршрути між маршрутизаторами

Оскільки використання класичних протоколів маршрутизації за станом каналів, наприклад, OSPF, для формування великої кількості маршрутів є неефективним, у запропонованій системі ефективніше використовувати протоколи маршрутизації за вектором дистанції типу RIP. Формування,

оновлення та коригування маршрутної інформації здійснюється централізовано в SDN-контролері, що дозволяє суттєво скоротити час обчислень і запобігти зацикленню маршрутів. Для кожного роутера будується таблиця маршрутів, яка містить дані про суміжні роутери, значення комбінованої метрики та коефіцієнт завантаження, що використовуються під час прийняття рішень про передачу даних.

```
[0, 6, 9, 12, 16, 20]   Metric = 19.0 MBIT/S   Weight = 0.43
[0, 6, 9, 12, 16, 20]   Metric = 19.0 MBIT/S   Weight = 0.47
[0, 6, 10, 13, 17, 20]  Metric = 19.0 MBIT/S   Weight = 0.52
[0, 6, 10, 13, 18, 20]  Metric = 19.0 MBIT/S   Weight = 0.54
[0, 6, 10, 14, 18, 20]  Metric = 19.0 MBIT/S   Weight = 0.61
[0, 6, 10, 14, 18, 20]  Metric = 19.0 MBIT/S   Weight = 0.65
[0, 6, 11, 15, 19, 20]  Metric = 19.0 MBIT/S   Weight = 0.67
[0, 6, 11, 15, 19, 20]  Metric = 19.0 MBIT/S   Weight = 0.71
```

Рисунок 1.18. Зміна маршруту без додаткової затримки або втрати пакетів

Використання протоколів маршрутизації за станом каналів типу OSPF для формування великої кількості шляхів не є ефективним, оскільки передача даних по конкретно обраному шляху ускладнює процес ремаршрутизації при зміні його характеристик. У цьому випадку більш ефективним є застосування протоколів маршрутизації за вектором дистанції типу RIP. Формування та оновлення маршрутної інформації здійснюється централізовано в SDN-контролері, що дозволяє скоротити час цього процесу та виключити проблему зациклення маршрутів. Для кожного роутера R_j формується окрема таблиця маршрутів.

При побудові таблиці застосовуються наступні позначення. Роутер-адресат позначається як R_a; R_i позначає суміжний роутер, через який проводиться передача; $M_{j,a}$ – значення метрики шляху від R_j до R_a; D_l – коефіцієнт навантаження шляху. За допомогою таблиці визначаються суміжні роутери на напрямку до адресата, а також значення метрики та навантаження шляху. Наприклад, для деякого роутера R_{13} може бути сформована наступна таблиця маршрутів.

У даній схемі метрика шляху задається як число переходів, що забезпечує мінімізацію геометричної довжини шляху, але для оптимізації мережевого трафіку також враховується комбінована метрика. Вона враховує величину

трафіку, переданого по каналу $S(L_{i,j})$, пропускну здатність $B(L_{i,j})$ і час затримки при передачі. Коефіцієнт завантаження визначається за формулою $d_{i,j} = \frac{S(L_{i,j})}{B(L_{i,j})}$, а навантаження маршруту P_i визначається як максимум значень $d_{i,j}$ для всіх каналів, що входять до цього маршруту.

Таблиця 1.1. Таблиця шляхів маршрутизатора R_j

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{15}	$M_{j,a}$	0.56
R_{20}	R_{16}	$M_{j,a}$	0.52
R_{20}	R_{17}	$M_{j,a}$	0.60

Таблиця 1.2. Таблиця шляхів маршрутизатора R_{13}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{16}	$M_{j,a}$	0.58
R_{20}	R_{17}	$M_{j,a}$	0.49
R_{20}	R_{18}	$M_{j,a}$	0.63

Процедура формування маршрутної інформації починається із задання початкової множини роутерів, що позначається як $W_1 = \{R_n\}$. На першому етапі кожному маршруту присвоюється початкове значення $D_i = 0$, а також ініціалізується лічильник J = 0. Далі у циклі, збільшуючи значення j на одиницю, формується наступна множина роутерів W_{j+1}, що складається з усіх роутерів R_i (для i = 1, …, k), які є суміжними з роутерами попередньої множини W_j. Тут k визначається як сума ступенів (кількість суміжних зв’язків) роутерів із W_j. Якщо для певного j отримана множина W_{j+1} дорівнює порожній множині (∅), алгоритм переходить до завершального етапу. Якщо множина не порожня, для кожного елемента i від 1 до k формується набір Z_i, який містить значення вершини V_n, значення ребра V_l, метрику $M_{i,n}$ та коефіцієнт завантаження d. При цьому, якщо для даного кроку встановлено, що обчислене значення завантаження d_j перевищує поточне значення D_i, то D_i оновлюється до d_j. Після обробки всіх елементів циклічно формується наступна множина, і алгоритм повторює описані кроки до

завершення обходу всіх можливих шляхів. За допомогою цього алгоритму можна визначити максимальне значення навантаження каналів для кожного маршруту.

Більш детально, алгоритм спочатку здійснює цикл обходу всіх можливих шляхів усередині маршрутного графа. Для кожного шляху проводиться "занурення" по всіх ребрах, щоб обійти кожне значення навантаження. Якщо для окремого ребра значення навантаження перевищує попередньо зафіксоване, це нове значення стає навантаженням усіх каналів цього маршруту. Після завершення обходу поточного маршруту алгоритм переходить до наступного шляху, повторюючи описані дії. Якщо додаткових шляхів не залишається, процедура завершується. Наприклад, при формуванні таблиць маршрутів для передачі інформації від роутера R_0 до роутера R_{20} (рис. 1.16), значення навантаження по каналах L(i, j) (де L(i, j) позначає канал, що зв'язує роутери R_i та R_j) обчислюються відповідно до цього алгоритму, а результати оформлюються у таблицю (табл. 1.3).

Таблиця 1.3. Таблиця маршрутизації при передачі інформації від R_0 до R_{20}

Вузли	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}	R_{16}	R_{17}	R_{18}	R_{19}	R_{20}
R_0	0	0.22	0.09	0.10	-	-	-	0.40	-	-	-	-	-	-	-	-	-	-	-	-	-
R_1	0.22	0	-	0.32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R_2	0.09	-	0	-	0.33	0.20	0.11	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R_3	0.10	0.32	-	0	-	-	-	0.30	-	-	-	-	-	-	-	-	-	-	-	-	-
R_4	-	0.30	0.33	-	0	-	-	-	0.12	-	-	-	-	-	-	-	-	-	-	-	-
R_5	-	-	0.20	-	-	0	-	0.21	0.10	0.41	-	-	-	-	-	-	-	-	-	-	-
R_6	-	-	0.11	-	-	-	0	-	-	0.32	-	-	-	-	-	-	-	-	-	-	-
R_7	0.40	-	-	0.30	-	-	-	0	0.10	-	0.41	-	-	-	-	-	-	-	-	-	-
R_8	-	-	-	-	0.12	0.10	-	-	0	-	0.40	0.50	0.70	-	-	-	-	-	-	-	-
R_9	-	-	-	-	-	0.41	-	-	-	0	-	0.40	-	-	-	-	-	-	-	-	-
R_{10}	-	-	-	-	-	-	-	-	0.40	0.40	0	0.30	0.40	0.50	0.70	-	-	-	-	-	-
R_{11}	-	-	-	-	-	-	-	-	0.50	-	-	0	-	0.40	-	-	-	-	-	-	-
R_{12}	-	-	-	-	-	-	-	-	-	-	0.40	-	0	-	0.50	-	-	-	-	-	-
R_{13}	-	-	-	-	-	-	-	-	-	-	0.50	0.40	-	0	0.40	0.20	0.80	-	-	-	-
R_{14}	-	-	-	-	-	-	-	-	-	-	-	-	0.50	-	0	0.50	-	-	-	-	-
R_{15}	-	-	-	-	-	-	-	-	-	-	-	-	-	0.20	-	0	0.40	0.30	-	-	-
R_{16}	-	-	-	-	-	-	-	-	-	-	-	-	-	0.80	-	-	0	0.40	0.30	0.60	-
R_{17}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.14	0	-	-	0.37
R_{18}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.42	-	-	0	-	0.58
R_{19}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.31	-	-	-	0	0.65
R_{20}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.52	0.38	0.56	0.65	0

Послідовність формування таблиць маршрутів:

На першому етапі встановлюється початковий індекс i = 1, що позначає

початок процесу побудови маршрутної інформації. Визначається множина початкового вузла маршрутизації: $W_1 = \{R_{20}\}$, яка містить лише один роутер-призначення. На основі оновленої таблиці суміжності формується множина роутерів $W_2 = \{R_{16}, R_{17}, R_{18}, R_{19}\}$, що є безпосередньо суміжними з роутером R20. Це означає, що дані вузли мають зв'язок із призначеним роутером та можуть брати участь у формуванні маршрутних шляхів. Для кожного роутера із множини W_2 необхідно скласти таблиці маршрутів з урахуванням навантаження шляху до роутера R_{20}.

Таблиця 1.4. Таблиця шляхів маршрутизатора R_{16}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{20}	$M_{j,a}$	0.58

Таблиця 1.5. Таблиця шляхів маршрутизатора R_{17}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{20}	$M_{j,a}$	0.42

Таблиця 1.6. Таблиця шляхів маршрутизатора R_{18}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{20}	$M_{j,a}$	0.61

Таблиця 1.7. Таблиця шляхів маршрутизатора R_{19}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{20}	$M_{j,a}$	0.68

Після формування маршрутної інформації для всіх маршрутизаторів множини W_2, з таблиці суміжності вузлів мережі (табл. 1.3) вилучається рядок та стовпчик R_{20}. Це означає, що даний маршрутизатор вже отримав необхідну маршрутну інформацію, і його не потрібно враховувати на наступних етапах побудови маршрутів. У результаті формується таблиця суміжності наступного рівня, яка містить оновлені значення зв'язків між залишковими вузлами та

дозволяє продовжити процес побудови оптимальних маршрутів у мережі. На основі цієї таблиці формується множина роутерів $W_3 = \{R_{12}, R_{13}, R_{14}, R_{15}, R_{16}\}$ – це вузли, які суміжні з роутерами з попередньої множини $W_2 = \{R_{16}, R_{17}, R_{18}, R_{19}\}$. Інакше кажучи, роутери W_3 безпосередньо зв'язані з вузлами з W_2, що забезпечує їх участь у передачі даних до кінцевого пристрою. Після цього для кожного роутера з множини W_3 формується власна таблиця маршрутів з урахуванням навантаження шляху до роутера-адресата R_{20} (табл. 1.8-1.11)

Таблиця 1.8. Таблиця шляхів маршрутизатора R_{12}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{16}	$M_{j,a}$	0.58

Таблиця 1.9. Таблиця шляхів маршрутизатора R_{13}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{16}	$M_{j,a}$	0.58
R_{20}	R_{17}	$M_{j,a}$	0.39
R_{20}	R_{18}	$M_{j,a}$	0.78

Таблиця 1.10. Таблиця шляхів маршрутизатора R_{14}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{18}	$M_{j,a}$	0.58

Таблиця 1.11. Таблиця шляхів маршрутизатора R_{15}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{18}	$M_{j,a}$	0.58
R_{20}	R_{19}	$M_{j,a}$	0.69

Після формування таблиць для маршрутизатора із множини W_3 відбувається оновлення маршрутної інформації для деяких вузлів. Зокрема, оновлюються таблиці маршрутів для маршрутизатора R_{16} і R_{17}. Після оновлення інформації з таблиці суміжності вузлів мережі видаляється множина роутерів $W_2 = \{R_{16}, R_{17}, R_{18}, R_{19}\}$. На її основі формується таблиця суміжності наступного рівня, що

використовується для подальшого побудови маршрутних шляхів.

Таблиця 1.12. Таблиця шляхів маршрутизатора R_{16}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{20}	$M_{j,a}$	0.58
R_{20}	R_{17}	$M_{j,a}$	0.09

Таблиця 1.13. Таблиця шляхів маршрутизатора R_{17}

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{20}	$M_{j,a}$	0.38
R_{20}	R_{16}	$M_{j,a}$	0.09

Після формування початкової таблиці суміжності вузлів мережі з неї видаляється множина роутерів W_2, яка складається з $\{R_{16}, R_{17}, R_{18}, R_{19}\}$. Це означає, що вузли, для яких вже сформовано таблиці маршрутів, більше не враховуються при побудові подальших рівнів таблиць. На основі оновлених даних формується таблиця суміжності наступного рівня, в якій відображено зв'язки між залишковими вузлами мережі. Подальшим кроком є формування множини роутерів W_4. У цій множині містяться вузли $\{R_8, R_9, R_{10}, R_{11}\}$, які є суміжними з роутерами попереднього рівня, що входять до множини W_3 (тобто, $\{R_{12}, R_{13}, R_{14}, R_{15}\}$). Після визначення множини W_4 для кожного роутера з цієї групи централізовано формуються таблиці маршрутів із врахуванням навантаження шляху до кінцевого роутера R_{20}. Це дозволяє зібрати інформацію про можливі маршрути та їх параметри для вузлів W_4. Наступним етапом, аналізуючи дані з оновленої таблиці суміжності, формується множина роутерів W_5, що складається з $\{R_4, R_5, R_6, R_7\}$. Ці вузли є суміжними з роутерами, що містяться у множині W_4 (тобто, $\{R_8, R_9, R_{10}, R_{11}\}$). Для кожного роутера з W_5, із врахуванням навантаження шляху до R_{20}, централізовано формується відповідна таблиця маршрутів. Цей крок допомагає детальніше розглянути внутрішню топологію мережі та визначити оптимальні шляхи передачі даних з урахуванням навантаження каналів. Далі, формується множина роутерів W_6, яка містить $\{R_1, R_2, R_3\}$. Вузли цієї множини є суміжними з роутерами з W_5 (тобто, з $\{R_4, R_5, R_6, R_7\}$). Для кожного роутера з W_6

також централізовано формуються таблиці маршрутів до R_{20} з урахуванням характеристик навантаження, що дозволяє об'єднати дані з попередніх рівнів. Нарешті формується остання множина роутерів W_7, яка містить лише $\{R_0\}$. Цей вузол є суміжним із множиною W_6 (тобто, $\{R_1, R_2, R_3\}$). Для роутера R_0 відбувається коригування таблиці маршрутів, що завершує процес централізованого формування маршрутної інформації. У підсумку дані про маршрути об'єднуються таким чином, що оптимізований шлях передачі інформації формується за наступною послідовністю: $R_0 \rightarrow R_7 \rightarrow R_{10} \rightarrow R_{13} \rightarrow R_{17} \rightarrow R_{20}$. Цей ланцюг відображає остаточний маршрут, який враховує як топологічні зв'язки мережі, так і навантаження каналів на кожному сегменті, забезпечуючи ефективну та адаптивну передачу даних у SDN-мережі.

Суть ремаршрутизації полягає у динамічній зміні таблиць маршрутів, що відповідають за конкретний шлях передачі даних. Контролер SDN отримує інформацію про зміну стану каналів від мережевих комутаторів і відповідно коригує таблиці маршрутів цих пристроїв. Наявність декількох шляхів в кожній таблиці дозволяє оперативно реконфігурувати маршрути без значних затримок і без втрати пакетів.

[0, 7, 10, 13, 16, 17, 20] Metric = 16.5 MBIT/S Weight = 0.38
[0, 1, 4, 8, 12, 16, 20] Metric = 16.5 MBIT/S Weight = 0.62
[0, 1, 4, 8, 13, 16, 20] Metric = 16.5 MBIT/S Weight = 0.62
[0, 2, 4, 8, 12, 16, 20] Metric = 16.5 MBIT/S Weight = 0.62
[0, 2, 4, 8, 13, 16, 20] Metric = 16.5 MBIT/S Weight = 0.62
[0, 2, 5, 8, 12, 16, 20] Metric = 16.5 MBIT/S Weight = 0.63
[0, 2, 5, 8, 13, 16, 20] Metric = 16.5 MBIT/S Weight = 0.61
[0, 2, 4, 8, 13, 16, 17, 20] Metric = 14.3 MBIT/S Weight = 0.38
[0, 2, 5, 8, 13, 16, 17, 20] Metric = 14.3 MBIT/S Weight = 0.38
[0, 2, 5, 9, 13, 16, 17, 20] Metric = 14.3 MBIT/S Weight = 0.38
[0, 2, 5, 10, 13, 16, 17, 20] Metric = 14.3 MBIT/S Weight = 0.38
[0, 2, 6, 10, 13, 16, 17, 20] Metric = 14.3 MBIT/S Weight = 0.38
[0, 3, 7, 10, 13, 16, 17, 20] Metric = 14.3 MBIT/S Weight = 0.38
[0, 7, 10, 5, 8, 13, 17, 20] Metric = 14.3 MBIT/S Weight = 0.38
[0, 7, 10, 5, 9, 13, 17, 20] Metric = 14.3 MBIT/S Weight = 0.38
[0, 1, 4, 8, 12, 16, 17, 20] Metric = 14.3 MBIT/S Weight = 0.49
[0, 2, 4, 8, 12, 16, 17, 20] Metric = 14.3 MBIT/S Weight = 0.49

Рисунок 1.19. Оптимальні шляхи між маршрутизаторами R_0 та R_{20} при перенавантаженні каналу $L_{16,20}$

На рис. 1.19 показано множину оптимальних шляхів між роутерами R_0 та R_{20} при перенавантаженні каналу $L_{16,20}$. Наприклад, якщо пакет знаходиться в роутері R_{16}, вибір маршруту визначається змістом його таблиці маршрутів. Таким чином, може бути вибрано або шлях $R_{16} \rightarrow R_{17} \rightarrow R_{20}$, або шлях $R_{16} \rightarrow R_{13} \rightarrow R_{17} \rightarrow R_{20}$, залежно від поточного завантаження і значення метрики. Оскільки в кожній таблиці зберігається інформація про кілька маршрутів, немає потреби перераховувати маршрути заново, що значно спрощує процедуру ремаршрутизації.

При зміні шляху, наприклад, з прямого $R_{16} \rightarrow R_{20}$ на шлях $R_{16} \rightarrow R_{17} \rightarrow R_{20}$, завантаження каналу оновлюється до нового значення (0.52 замість попередніх значень). Це відображено у таблиці нижче.

Таблиця 1.14. Таблиця шляхів маршрутизатора R_{16} (оновлення)

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{17}	$M_{j,a}$	0.52

Після цього контролер SDN вносить зміни до таблиць маршрутів для роутерів, розташованих на проміжних сегментах шляху між R_0 та R_{16}. Зокрема, оновлюються таблиці для роутерів R_{12} та R_{13}.

Таблиця 1.15. Таблиця шляхів маршрутизатора R_{12} (оновлення)

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{16}	$M_{j,a}$	0.52

Таблиця 1.16. Таблиця шляхів маршрутизатора R_{13} (оновлення)

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{16}	$M_{j,a}$	0.52
R_{20}	R_{17}	$M_{j,a}$	0.42
R_{20}	R_{18}	$M_{j,a}$	0.78

Контролер також оновлює таблиці маршрутів для інших вузлів, що знаходяться на шляху до R_{20}. Наприклад, для роутера R_8 створюється нова таблиця маршрутів, де враховуються нові значення навантаження.

Таблиця 1.17. Таблиця шляхів маршрутизатора R_8 (оновлення)

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_{12}	$M_{j,a}$	0.52
R_{20}	R_{13}	$M_{j,a}$	0.42

Далі контролер вносить зміни у таблиці для роутерів, що знаходяться ще ближче до початкового вузла. Наприклад, для роутера R_4 формується нова таблиця маршрутів з оновленим завантаженням.

Таблиця 1.18. Таблиця шляхів маршрутизатора R_4 (оновлення)

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_8	$M_{j,a}$	0.52

Подібні зміни вносяться і для таблиць роутерів R_1 та R_0.

Таблиця 1.19. Таблиця шляхів маршрутизатора R_1 (оновлення)

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_4	$M_{j,a}$	0.52

Таблиця 1.20. Таблиця шляхів маршрутизатора R_0 (оновлення)

Маршрутизатор (адресат)	*Суміжний* R_i	*Метрика шляху*	*Навантаження шляху* D_l
R_{20}	R_1	$M_{j,a}$	0.52
R_{20}	R_2	$M_{j,a}$	0.42
R_{20}	R_3	$M_{j,a}$	0.70
R_{20}	R_7	$M_{j,a}$	0.68

В результаті трафік перебудовується на мінімально завантажений маршрут за наступною послідовністю: $R_0 \rightarrow R_1 \rightarrow R_8 \rightarrow R_{12} \rightarrow R_{16} \rightarrow R_{17} \rightarrow R_{20}$. Навіть у випадку відключення певного роутера, наприклад, R_{16}, система залишається стійкою.

На рис. 1.20 представлено граф мережі та результат формування альтернативних шляхів від роутера R_0 до R_{20} при відключенні R_{16}. Аналогічно, рис. 1.21 демонструє множину оптимальних шляхів при такій ситуації.

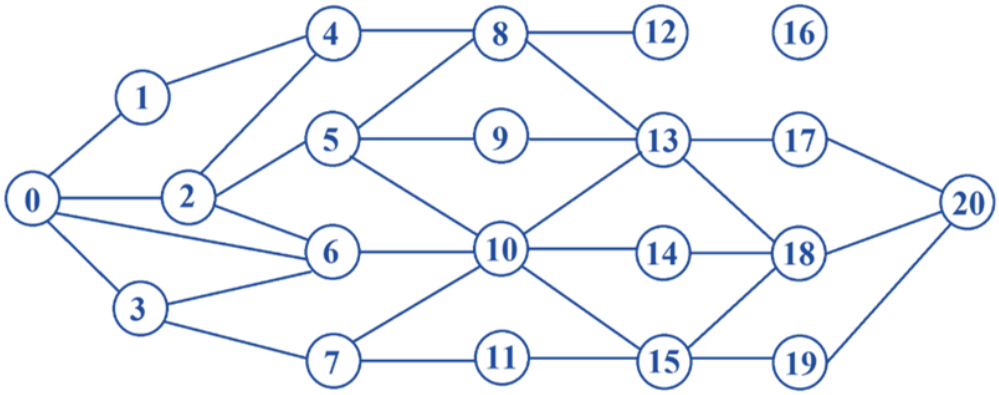


Рисунок 1.20. Маршрутний граф SDN-мережі при вимкненому маршрутизаторі R_{16}

[0, 7, 10, 13, 17, 20] Metric = 20.5 MBIT/S Weight = 0.42
[0, 7, 10, 14, 18, 20] Metric = 20.5 MBIT/S Weight = 0.58
[0, 7, 11, 15, 18, 20] Metric = 20.5 MBIT/S Weight = 0.58
[0, 7, 10, 15, 18, 20] Metric = 20.5 MBIT/S Weight = 0.69
[0, 7, 10, 15, 19, 20] Metric = 20.5 MBIT/S Weight = 0.69
[0, 7, 11, 15, 19, 20] Metric = 20.5 MBIT/S Weight = 0.69
[0, 7, 10, 13, 18, 20] Metric = 20.5 MBIT/S Weight = 0.82
[0, 1, 4, 8, 13, 17, 20] Metric = 17.2 MBIT/S Weight = 0.42
[0, 2, 4, 8, 13, 17, 20] Metric = 17.2 MBIT/S Weight = 0.42
[0, 2, 5, 8, 13, 17, 20] Metric = 17.2 MBIT/S Weight = 0.42
[0, 2, 5, 9, 13, 17, 20] Metric = 17.2 MBIT/S Weight = 0.42
[0, 2, 5, 10, 13, 17, 20] Metric = 17.2 MBIT/S Weight = 0.42
[0, 2, 6, 10, 13, 17, 20] Metric = 17.2 MBIT/S Weight = 0.42
[0, 3, 7, 10, 13, 17, 20] Metric = 17.2 MBIT/S Weight = 0.42
[0, 2, 5, 10, 14, 18, 20] Metric = 17.2 MBIT/S Weight = 0.58
[0, 2, 6, 10, 14, 18, 20] Metric = 17.2 MBIT/S Weight = 0.58
[0, 3, 7, 10, 14, 18, 20] Metric = 17.2 MBIT/S Weight = 0.58
[0, 3, 7, 11, 15, 18, 20] Metric = 17.2 MBIT/S Weight = 0.58
[0, 2, 5, 10, 15, 18, 20] Metric = 17.2 MBIT/S Weight = 0.69
[0, 2, 5, 10, 15, 19, 20] Metric = 17.2 MBIT/S Weight = 0.69
[0, 2, 6, 10, 15, 18, 20] Metric = 17.2 MBIT/S Weight = 0.69
[0, 2, 6, 10, 15, 19, 20] Metric = 17.2 MBIT/S Weight = 0.69
[0, 3, 7, 10, 15, 18, 20] Metric = 17.2 MBIT/S Weight = 0.69
[0, 3, 7, 10, 15, 19, 20] Metric = 17.2 MBIT/S Weight = 0.69
[0, 3, 7, 11, 15, 19, 20] Metric = 17.2 MBIT/S Weight = 0.69

Рисунок 1.21. Оптимальні шляхи при вимкненому маршрутизаторі R_{16}

Перші сім рядків (1–7) демонструють маршрути із вищою метрикою (20.5 MBIT/S) та варіаціями коефіцієнту навантаження від 0.42 до 0.82. Рядки з 8 по 27 представляють альтернативні маршрути з нижчою метрикою (17.2 MBIT/S), де в основному коефіцієнт навантаження залишається 0.44 (винятком деяких незначних відмінностей у перших декількох рядках цієї групи). Це дозволяє бачити різноманіття альтернативних маршрутів, які можуть бути обрані у процесі динамічної ремаршрутизації. Таким чином, динамічна ремаршрутизація

забезпечує перебудовування маршрутів у режимі реального часу з мінімальними затримками та без втрати пакетів, завдяки чому мережа залишається стабільною навіть при зміні стану її елементів. Контролер SDN постійно аналізує стан каналів і, базуючись на інформації з таблиць маршрутів, автоматично оновлює маршрути для оптимізації передачі даних.

1.4 Моделювання потоків даних за модифікованим алгоритмом

У цьому підрозділі моделюється передача даних за допомогою модифікованого алгоритму Лі, який враховує параметри пропускної здатності, затримок і фактичного навантаження каналів, а також забезпечує динамічну ремаршрутизацію. Алгоритм побудови маршрутів базується на ієрархічному формуванні таблиць, що дозволяє швидко перемикатися між альтернативними шляхами без затримок і втрати пакетів, навіть при змінних умовах роботи мережі.

Для реалізації моделі використано мову програмування Java, яка завдяки багатоплатформності, широкому вибору бібліотек та сучасним інтегрованим середовищам розробки (наприклад, IntelliJ IDEA) є оптимальним вибором для створення ефективних і адаптивних рішень. Використання JavaFX (входить до складу JDK 8) для візуалізації результатів моделювання дозволяє створити зручний графічний інтерфейс, а рекомендовані технічні параметри (процесор не менше 3 ГГц, 8 ГБ оперативної пам’яті, JVM 8 або вище) гарантують стабільну роботу програми. Таким чином, запропонований підхід дозволяє ефективно аналізувати поведінку мережі, виявляти «вузькі місця» та оперативно адаптувати маршрути для підвищення продуктивності реальних SDN-мереж.

1.4.1 Розробка БСА для побудови графа програмно-визначуваної мережі

БСА для побудови графа програмно-визначуваної мережі та конструювання трафіку буде складатися з кількох чітких етапів, кожен з яких буде виконувати свою роль у формуванні оптимальних маршрутів для програмно-визначуваної мережі. Нижче наведено детальний аналіз кожного етапу алгоритму.

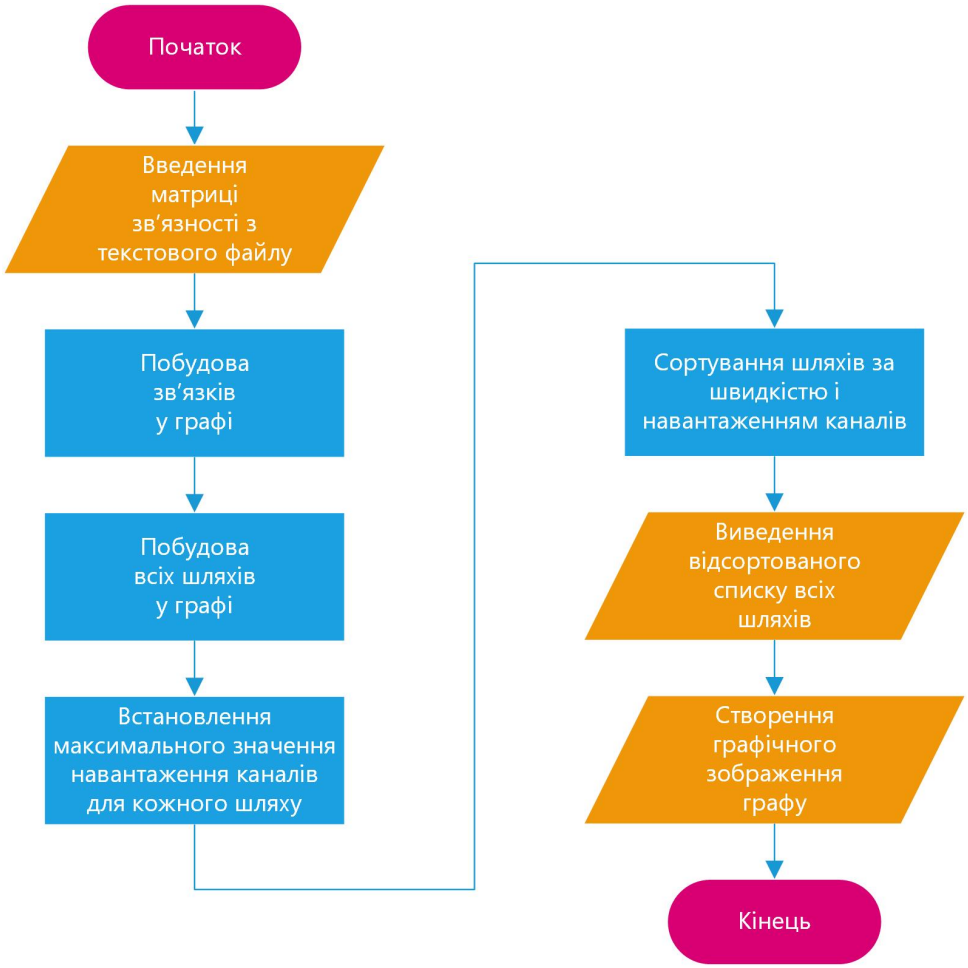


Рисунок 1.22. БСА побудови графа програмно-визначуваної мережі

Вхідні дані та побудова графа. На першому етапі здійснюється зчитування матриці суміжності з текстового файлу. Цей файл містить дані про зв'язність вузлів мережі. Алгоритм формує граф із використанням структури даних, що базується на списку суміжності (реалізованому через Map<Integer, List<Integer>>). Метод додавання вершин і ребер гарантує, що всі зв'язки між вузлами будуть представлені, а відсутність повторення даних очищає структуру перед подальшим аналізом.

Пошук всіх шляхів між заданими вершинами. Для знаходження всіх можливих маршрутів між вихідною та кінцевою точками використовується алгоритм пошуку у глибину (DFS). Рекурсивний виклик методу дозволяє обійти всі доступні гілки графа, формуючи перелік шляхів. Попри те, що перебір усіх шляхів може бути обчислювально важким (зокрема, для сильно зв'язних або великих графів), даний підхід забезпечує повну картину можливих маршрутів, що є важливим фактором для вибору оптимального шляху за критеріями пропускної

здатності та навантаження.

Розрахунок параметрів маршруту. Для кожного знайденого шляху визначається максимальне значення навантаження каналів, що є важливим показником для аналізу пропускної здатності. Розрахунок реальної швидкості передачі (realSpeed) і ваги кожного маршруту (weight) ґрунтується на індивідуальних характеристиках ребер, що входять до складу шляху. Цей крок дозволяє точно оцінити якість кожного маршруту з урахуванням як топологічних, так і експлуатаційних параметрів мережі.

Сортування та вибір оптимальних маршрутів. Після розрахунку параметрів усі маршрути сортуються за двома критеріями: реальна швидкість та вага (нагрузка каналу). Завдяки цій операції формується відсортований список, що містить маршрути від найоптимальніших до менш ефективних. Цей етап є ключовим для прийняття рішень щодо конструювання трафіку, оскільки дозволяє швидко вибирати маршрути, які забезпечують мінімальні затримки та максимальну стабільність передачі даних.

Графічна візуалізація. Останній крок алгоритму полягає у створенні графічного представлення мережі. Візуалізація не тільки полегшує інтерпретацію результатів, але й сприяє виявленню потенційних «вузьких місць» у топології мережі. Цей етап реалізовано через засоби JavaFX і є допоміжним, проте важливим інструментом для аналізу поведінки алгоритму.

Комбінований підхід, який включає побудову графа, повний перебір шляхів за допомогою DFS, розрахунок важливих параметрів кожного маршруту, їх сортування та подальшу візуалізацію, дозволяє отримати детальну інформацію про можливі варіанти передачі даних. Такий метод є ефективним для планування і оптимізації трафіку в SDN-мережах, забезпечуючи можливість швидкого реагування на зміну умов роботи мережі та вибору оптимального маршруту без упущення критичних параметрів. Водночас, варто зазначити, що повний перебір шляхів може бути обчислювально затратним для великих мереж, тому у практичних застосуваннях можливо застосовувати додаткові оптимізаційні методи або евристики для скорочення кількості аналізованих маршрутів.

1.4.2 Створення об’єктно-орієнтованої моделі побудови графа і конструювання потоків даних

Для підвищення ефективності процесу конструювання трафіку в програмно-визначуваної мережі розроблено модифікований алгоритм, який базується на створенні графової моделі мережі та детальному аналізі всіх можливих шляхів між заданими вузлами. Основні етапи реалізації алгоритму поділяються на побудову графа, пошук усіх шляхів між вихідною та кінцевою точкою, розрахунок характеристик кожного маршруту та формування таблиці маршрутів.

Побудова графа. Основну топологічну інформацію про мережу представляє клас *Graph*, який у своїй сутності зберігає набір вершин і ребер. Для зручного представлення графа використовується список суміжності на базі *Map<Integer, List<Integer>>*. Метод *addVertex(int u)* перевіряє наявність вершини у мапі та, якщо її немає, створює новий запис із пустим списком суміжних вузлів. Метод *addEdge(int u, int v)* викликає *addVertex(u)*, а потім додає вершину *v* до списку суміжних вершин для u. Цей підхід дозволяє створювати орієнтовані ребра між вершинами, а також враховувати можливість наявності кількох ребер між однією парою вузлів.

Пошук шляхів у графі. Для виявлення всіх можливих маршрутів між початковим вузлом та кінцевим використовується алгоритм пошуку у глибину (DFS). Виклик методу

public List<List<Integer>> getAllPaths(Integer u, Integer v)

ініціює рекурсивний процес за допомогою допоміжного методу

private void getAllPathsDFS(Integer u, Integer v, boolean[] visited, Deque<Integer> path, List<List<Integer>> result)

При цьому алгоритм починає з відвідування кореневої вершини, додає її до стеку (або дека) та послідовно перевіряє всіх невідвіданих сусідів. Якщо для поточної вершини існує шлях до вузла *v*, цей шлях додається до колекції результатів. Відмічається, що в порівнянні з алгоритмом пошуку у ширину, алгоритм DFS перш за все досліджує всю гілку, що забезпечує повне охоплення доступних шляхів до кінцевого вузла.

Клас *Way* для зберігання інформації про маршрути. Після знаходження всіх шляхів необхідно зберегти параметри кожного маршруту для подальшої обробки. Для цього розроблено клас *Way*, що містить:

- Сформований шлях – список вершин (*List<Integer> wayList*), який задає послідовність переходів;
- Початкову швидкість – показник швидкості передачі між двома вузлами без посередників;
- Довжину шляху – загальна кількість вузлів або відстань маршруту;
- Реальну швидкість – ефективна швидкість передачі даних, що враховує проміжні вузли та зміни ваг ребер. Клас також містить додаткові гетери та методи форматування виводу, що спрощують подальший аналіз і сортування маршруту за критеріями «реальна швидкість» та «вага».

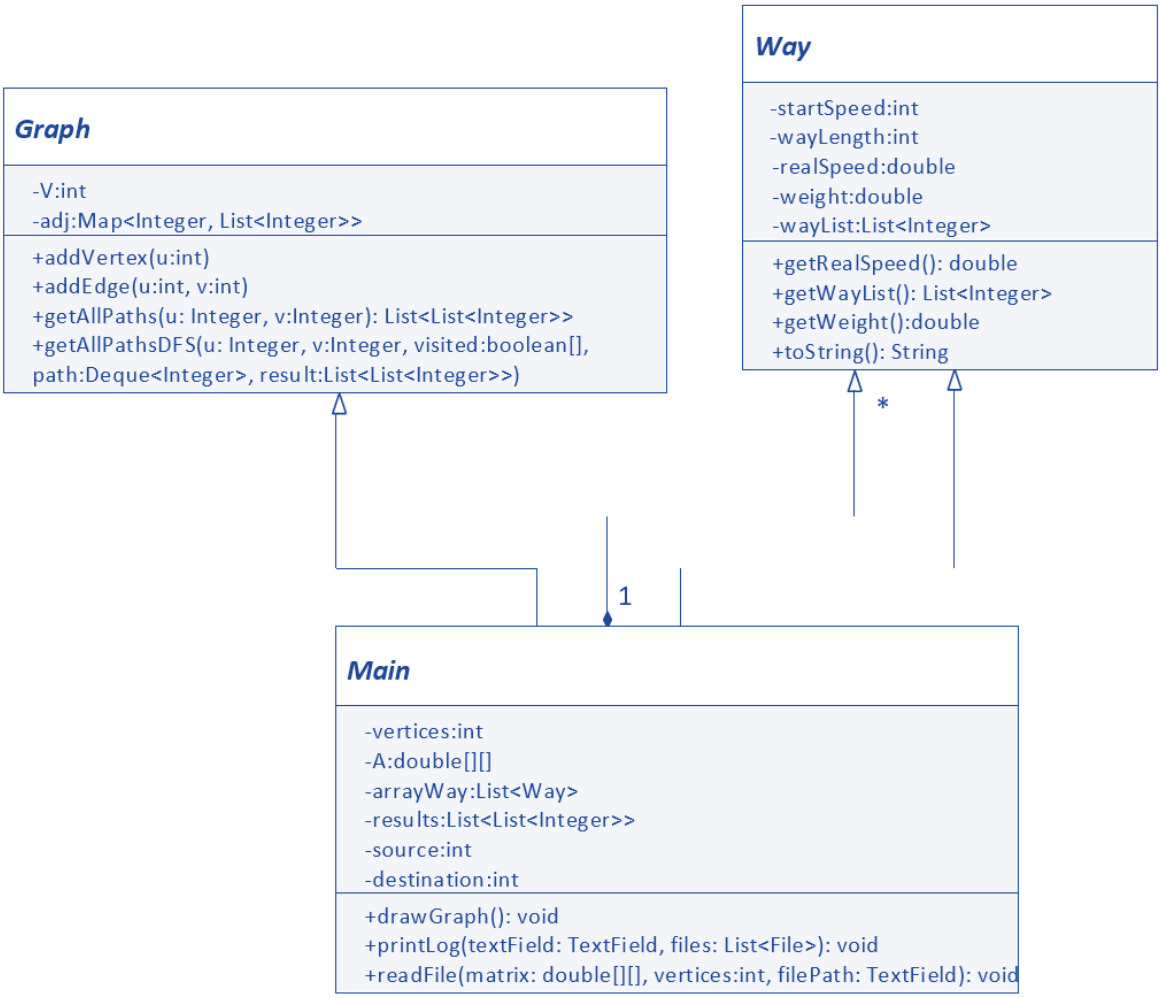


Рисунок 1.23. Об'єктно-орієнтована схема додатку для моделювання потоків даних у програмно-визначуваної мережі

Інтеграція в основний додаток. У класі *Main* реалізовано ініціалізацію та запуск графічного інтерфейсу через JavaFX, що дозволяє користувачу вказати кількість вершин, початкову та кінцеву вершину для пошуку шляху, а також завантажити файл з матрицею суміжності. Метод readFile зчитує матрицю і будує зв'язки між вершинами, після чого всі знайдені шляхи зберігаються у проміжному списку. Для кожного маршруту створюється об'єкт класу *Way*, який додається до колекції *arrayWay* (типу *ArrayList*). Після чого виконується сортування отриманих маршрутів за значеннями реальної швидкості і ваги, що дозволяє сформувати оптимальну таблицю маршрутів для подальшої передачі даних.

Модифікований алгоритм конструювання трафіку, реалізований на базі побудови графа і пошуку шляхів методом DFS, дозволяє систематично аналізувати всі можливі маршрути, розраховувати їхні характеристики та формувати оптимальні таблиці маршрутів відповідно до вимог технології SDN. Цей підхід є ефективним як для попереднього аналізу топології мережі, так і для оперативної адаптації мережевих шляхів у реальному часі. У Додатку А наведено фрагмент коду мовою Java, що демонструє основну реалізацію програми.

1.5 Розробка візуального інтерфейсу додатку для моделювання

Додаток, реалізуємий у даній роботі, призначений для моделювання потоків даних у програмно-визначуваних мережах з використанням модифікованого способу конструювання трафіку. Основне завдання додатку — забезпечення симуляції алгоритму, описаного у підрозділі 1.4, та представлення отриманих результатів у зручному для аналізу вигляді.

Діаграма взаємодії користувача і додатку для моделювання потоків даних у програмно-визначуваної мережі представлена на рис.1.24.

Інтерфейс додатку розділено на дві основні області. Ліва область формується для введення початкових даних. У ній розміщено текстові поля для зазначення кількості вершин графа, початкової та кінцевої точок маршруту, а також кнопку для вибору файлу з матрицею суміжності. Отримані результати обрахунку всіх можливих шляхів також відображаються в межах цієї

зони. Елементи цієї області групуються за допомогою контейнера VBox, наприклад:

VBox description = new VBox(20, textVertices, textSource, textDestination, buttonFile);

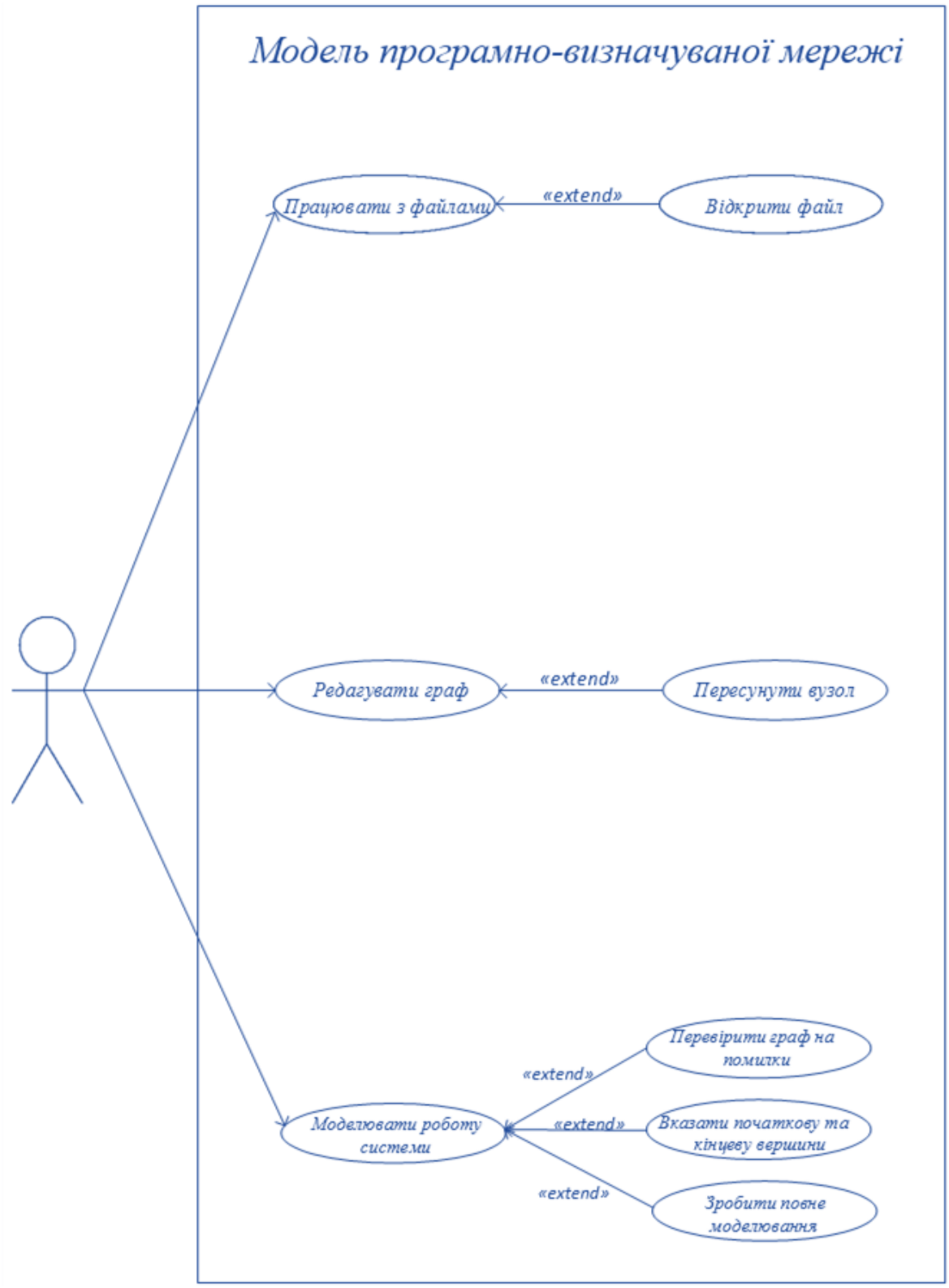


Рисунок 1.24. Діаграма взаємодії користувача і додатку для моделювання потоків даних у програмно-визначуваної мережі

Права область інтерфейсу додатку відводиться для візуалізації графа мережі. Тут формується зображення створеного графа із застосуванням алгоритму KKLayout з бібліотеки JUNG, що дозволяє оптимально розташувати вершини та ребра. Завдяки використанню цієї бібліотеки досягається інтуїтивне зображення топологічної структури мережі, що сприяє легшому аналізу отриманих результатів моделювання.

Розробка інтерфейсу здійснюється з використанням платформи JavaFX. У класі Main методом start створюється основне вікно додатку:

Group root = new Group();

primaryStage.setScene(new Scene(root, 1280, 720));

Для побудови та візуалізації графу використовується Maven як інструмент збірки проекту. Maven забезпечує декларативний опис проекту, автоматичне управління залежностями та завантаження сторонніх бібліотек. У нашому проекті за допомогою Maven було додано бібліотеку JUNG, яка надає необхідний інструментарій для аналізу, моделювання та візуалізації мережевих графів, використовуючи можливості Java API. Таким чином, розробка інтерфейсу забезпечує чітке розмежування функціональних зон:

- Ліва область для введення даних і відображення числових результатів симуляції;
- Права область для створення і візуального відображення графу мережі.

Користувач, використовуючи ліву область інтерфейсу розробленого додатку (рис. 1.25), вводить кількість вершин (6), а також задає початкову вершину (наприклад, 0) та кінцеву вершину (наприклад, 5). Після натискання кнопки «Розрахувати та побудувати граф мережі» програма починає обчислювати всі можливі маршрути між заданими вузлами. У лівій частині вікна відображається цей відсортований список маршрутів разом із зазначенням метрики та ваги для кожного шляху. Це дозволяє користувачу оцінити, який з варіантів є оптимальним з точки зору затримок і навантаження. Для розташування вершин використовується алгоритм KKLayout з бібліотеки JUNG, що забезпечує зрозуміле представлення топології мережі.

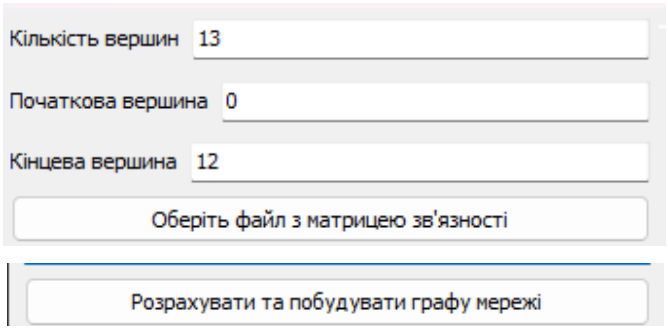


Рисунок 1.25. Елементи інтерфейсу додатку для моделювання потоків даних у програмно-визначуваної мережі

1.6 Отримання результатів моделювання та їх аналіз

У цьому підрозділі описано процес отримання результатів моделювання потоків даних у програмно-визначуваних мережах, алгоритму конструювання трафіку та проведено їх аналіз. Результати моделювання засновані на даних, які надходять із текстового файлу, що містить пів-матрицю суміжності вузлів мережі під головною діагоналлю. Після зчитування файл обробляється алгоритмом, що дублює значення відносно головної діагоналі для отримання повної симетричної матриці, яка відображає усі прямі зв’язки між вузлами.

Для демонстрації роботи програми створено текстовий файл із пів-матрицею зв’язності для графу з 6 вершин:

2
3 4
5 6 7
8 9 10 2
1 3 4 6 7

Алгоритм зчитує ці значення, продублювавши їх за головною діагоналлю, що дозволяє отримати повну матрицю:

Таблиця 1.21. Повна матриця зв’язності мережі

	V0	*V1*	*V2*	*V3*	*V4*	*V5*
V0	0	2	3	5	8	1
V1	2	0	4	6	9	3
V2	3	4	0	7	10	4
V3	5	6	7	0	2	6
V4	8	9	10	2	0	7
V5	1	3	4	6	7	0

Алгоритм перебирає всі можливі шляхи між V0 та V5 за допомогою пошуку у глибину (DFS) і розраховує для кожного маршруту його характеристику: метрику (наприклад, сумарну вартість або затримку) та вагу (коефіцієнт навантаження, що визначає ефективність шляху). Результати обчислень подаються у відсортованій формі, де кращі маршрути розташовано на початку списку.

Таблиця 1.22. Результати моделювання шляхів від V0 до V5

Шлях	*Метрика (мільйон/с)*	*Вага (коеф.)*
V0 → V1 → V3 → V5	12.3	0.45
V0 → V2 → V3 → V5	13.0	0.48
V0 → V2 → V4 → V5	14.0	0.50
V0 → V1 → V2 → V5	15.5	0.55

Паралельно з цим права область інтерфейсу відводиться для візуалізації графу мережі. На основі побудованої повної матриці суміжності створюється граф, де вершинами є вузли 0–5, а ребрами — зв'язки між ними. На рис. 1.26 демонструється приклад такого графічного подання, яке оновлюється разом із обчисленими маршрутами.

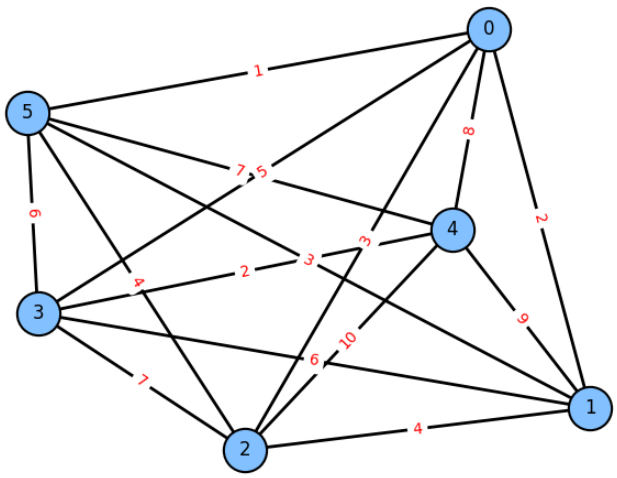


Рисунок 1.26. Результати моделювання шляхів від V0 до V5

Аналіз отриманих результатів полягає у порівнянні характеристик альтернативних маршрутів. Найоптимальніший шлях визначається за мінімальним значенням метрики та меншою вагою, що свідчить про менші затримки та ефективне використання каналів. Список маршрутів і граф дозволяє не лише вибрати найкращий варіант передачі даних, а й ідентифікувати

потенційні «вузькі місця» в мережі. Програмний модуль забезпечує комплексний підхід до моделювання трафіку: від зчитування пів-матриці суміжності, її перетворення в повну матрицю, до обчислення всіх можливих маршрутів і візуалізації графу мережі. Результати моделювання, представлені як числовими таблицями, так і графічно, дозволяють оперативно аналізувати стан мережі та приймати рішення щодо вибору оптимальних маршрутів для підвищення ефективності роботи SDN-мережі.

На рис.1.28 наведено скріншот роботи додатку з результатами моделювання шляхів від V0 до V12 за матрицею зв’язності з рис.1.27.

[0, 4, 8, 12] Metric = 34.0 MBIT/S Weight = 0.15
[0, 2, 4, 8, 12] Metric = 26.0 MBIT/S Weight = 0.35
[0, 3, 6, 8, 12] Metric = 26.0 MBIT/S Weight = 0.45
[0, 3, 6, 10, 12] Metric = 26.0 MBIT/S Weight = 0.45
[0, 3, 7, 10, 12] Metric = 26.0 MBIT/S Weight = 0.45
[0, 3, 7, 11, 12] Metric = 26.0 MBIT/S Weight = 0.45
[0, 1, 4, 8, 12] Metric = 26.0 MBIT/S Weight = 0.55
[0, 4, 8, 9, 12] Metric = 26.0 MBIT/S Weight = 0.55
[0, 2, 6, 8, 12] Metric = 26.0 MBIT/S Weight = 0.65
[0, 2, 6, 10, 12] Metric = 26.0 MBIT/S Weight = 0.65
[0, 2, 5, 10, 12] Metric = 26.0 MBIT/S Weight = 0.75
[0, 4, 8, 6, 10, 12] Metric = 21.0 MBIT/S Weight = 0.35
[0, 1, 4, 8, 9, 12] Metric = 21.0 MBIT/S Weight = 0.55
[0, 2, 4, 8, 9, 12] Metric = 21.0 MBIT/S Weight = 0.55
[0, 3, 6, 8, 9, 12] Metric = 21.0 MBIT/S Weight = 0.55
[0, 2, 6, 7, 10, 12] Metric = 21.0 MBIT/S Weight = 0.65
[0, 2, 6, 7, 11, 12] Metric = 21.0 MBIT/S Weight = 0.65
[0, 2, 6, 8, 9, 12] Metric = 21.0 MBIT/S Weight = 0.65
[0, 3, 6, 7, 10, 12] Metric = 21.0 MBIT/S Weight = 0.65
[0, 3, 6, 7, 11, 12] Metric = 21.0 MBIT/S Weight = 0.65
[0, 3, 7, 6, 8, 12] Metric = 21.0 MBIT/S Weight = 0.65
[0, 3, 7, 6, 10, 12] Metric = 21.0 MBIT/S Weight = 0.65
[0, 4, 2, 6, 8, 12] Metric = 21.0 MBIT/S Weight = 0.65
[0, 4, 2, 6, 10, 12] Metric = 21.0 MBIT/S Weight = 0.65
[0, 4, 2, 5, 10, 12] Metric = 21.0 MBIT/S Weight = 0.85

Рисунок 1.27. Матриця зв’язності з оптимальних шляхів між маршрутизаторами

Результати моделювання демонструють різноманітність маршрутів між вихідним вузлом 0 та кінцевим 12, що були отримані на основі повної матриці зв’язності. Найоптимальнішим з них є маршрут [0, 4, 8, 12] із значно вищою метрикою 34.0 MBIT/S і низькою вагою 0.15. Це свідчить про його перевагу в ефективності передачі даних як за швидкістю, так і за ресурсними витратами.

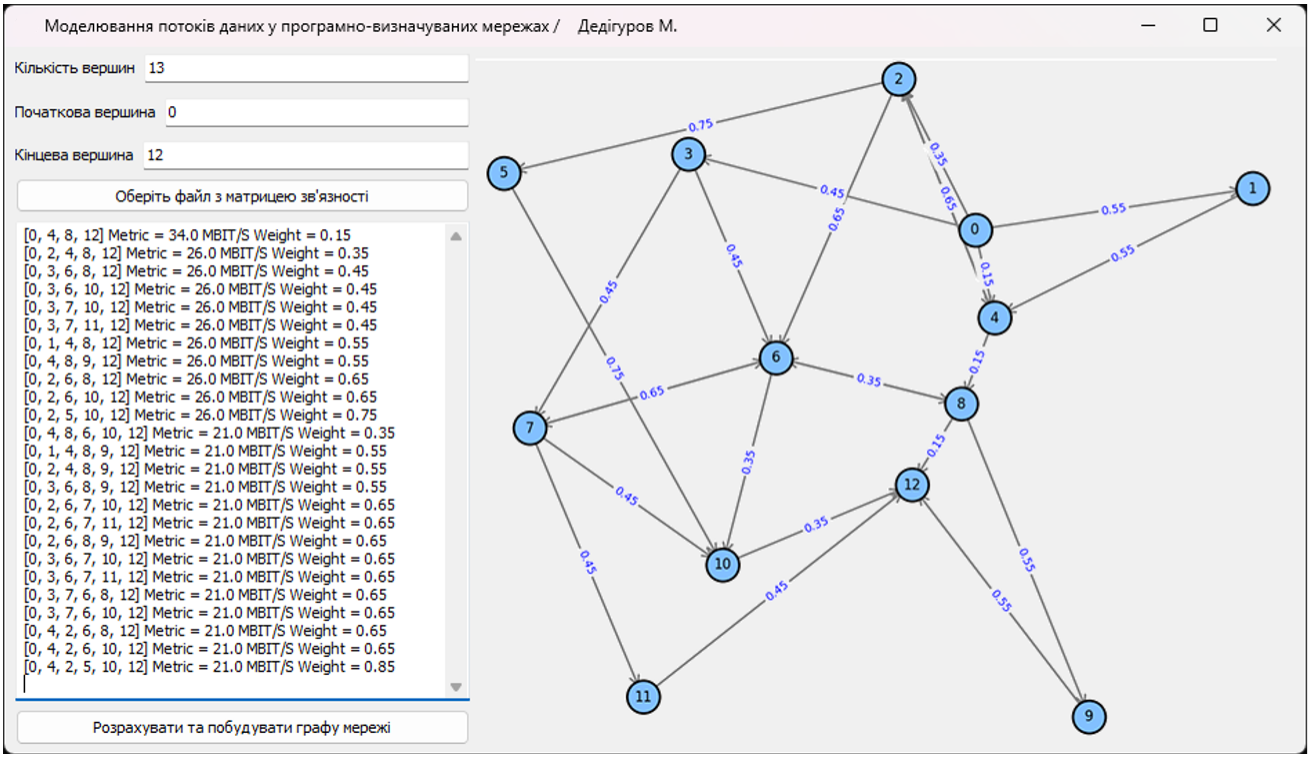


Рисунок 1.28. Скріншот роботи додатку
з результатами моделювання шляхів від V0 до V12

Інші маршрути із значеннями метрики 26.0 MBIT/S або 21.0 MBIT/S мають відповідно вищі коефіцієнти ваги (від 0.35 до 0.85), що свідчить про більшу кількість проміжних вузлів або зростання вартості передачі даних. Таким чином, граф, побудований за результатами моделювання, чітко відображає залежність між кількістю вузлів на маршруті, пропускною здатністю та обчислювальними витратами.

На рис.1.29 наведено скріншот роботи додатку з результатами моделювання шляхів від V0 до V21 за матрицею зв’язності з рис.1.21. Отримані результати моделювання свідчать про наявність кількох альтернативних маршрутів між вихідним і кінцевим вузлами із заданими параметрами (метрика і вага). Наприклад, маршрути з метрикою 20.5 MBIT/S мають вагові коефіцієнти від 0.42 до 0.82, що дозволяє визначити оптимальний шлях за мінімальним навантаженням. Також існують маршрути з меншою пропускною здатністю (17.2 MBIT/S), але з відповідно оптимальними значеннями ваги (найнижча – 0.42). Загалом, аналіз графу свідчить, що система може адаптивно обирати маршрут для передачі даних, враховуючи баланс між швидкістю і навантаженням мережі.

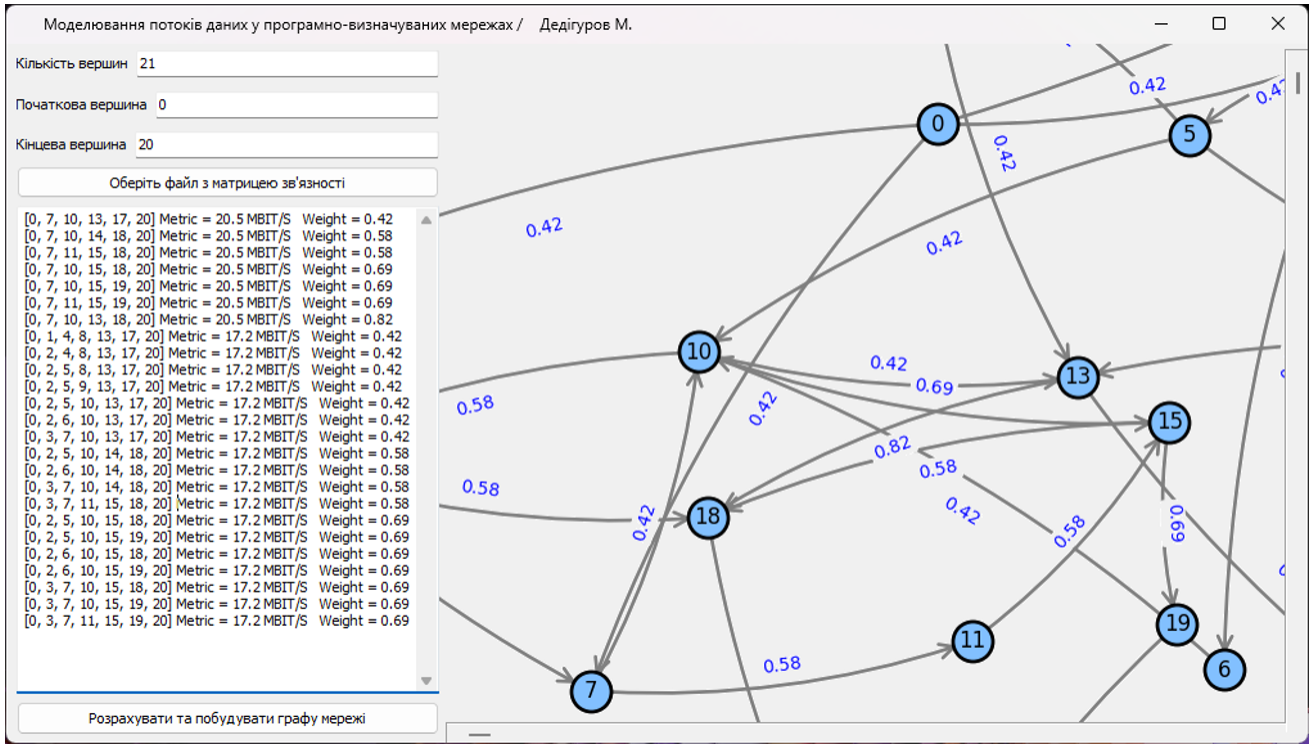


Рисунок 1.29. Скріншот роботи додатку
з результатами моделювання шляхів від V0 до V21

Отримані результати засвідчують, що запропонований алгоритм може ефективно ідентифікувати оптимальні маршрути для передачі мережевого трафіку в умовах зміни навантаження, що є особливо актуальним для сучасних SDN-мереж.

2 РОЗДІЛ ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

В умовах розробки та підтримки програмно-визначуваних мереж (SDN) праця набуває нових форм, де поєднуються інтелектуальна діяльність, моделювання даних та оптимізація обчислювальних процесів. Праця інженера-програміста в цій сфері є не лише засобом до життя, а й способом реалізації творчого потенціалу та самоствердження особи.

Одним із ключових завдань ергономічного забезпечення у сфері SDN є розробка ефективних алгоритмів моделювання потоків даних, що дозволяють оптимізувати робочі процеси та підвищити продуктивність. Взаємодія людини та машинних систем у цьому контексті передбачає розробку таких методів, що мінімізують когнітивне навантаження на програмістів та операторів мереж.

Критично важливим є питання безпеки та охорони праці фахівців, що працюють з програмно-визначуваними мережами. Врахування факторів ергономіки, автоматизації моніторингу стану систем та адаптивного управління потоками даних дозволяє забезпечити комфортні умови роботи та зменшити ризики, пов’язані з високими інтелектуальними навантаженнями.

2.1 Аналіз небезпечних і шкідливих факторів, що впливають на користувача ПК

До основних критеріїв забезпечення гігієни робочого середовища належать інтенсивність освітлення, температура повітря, вологість, рівень шумового забруднення, ступінь вібраційного впливу, токсичність, загазованість, а також обмеження загальної м'язової активності (гіподинамія). Крім цього, враховується дія електростатичного поля та вплив як неіонізуючих, так і іонізуючих електромагнітних випромінювань.

2.2 Гігієнічні вимоги до виробничого середовища

Державні санітарні норми, зокрема ДСАНПіН 3.3.2.007-98 «Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин», спрямовані на запобігання негативного

впливу шкідливих чинників, що супроводжують роботу з візуальними дисплейними терміналами, на здоров’я працівників.

2.2.1 Вимоги до приміщення

Розміщення робочих місць із використанням ВДТ, ЕОМ і ПЕОМ заборонено у підвальних приміщеннях та на цокольних поверхах. Для кімнат, призначених для роботи з візуальними дисплейними терміналами, рекомендується орієнтувати вікна у напрямку півночі або північного сходу. На вікнах повинні бути встановлені регульовані жалюзі або штори, що дозволяють їх повністю закривати для забезпечення оптимальних умов освітлення.

Планувальні рішення будівель і приміщень, де розташовано відеодисплейні термінали, мають відповідати вимогам ДСАНПіН 3.3.2.007-98. Для робочого місця програміста передбачено мінімальну площу не менше 6 кв. м та об’єм приміщення не менше 20 куб. м. Крім того, стіни приміщень повинні бути пофарбовані матовою фарбою, а в приміщеннях з ВДТ обов’язково мають бути передбачені зони для відпочинку та психологічного розвантаження.

2.2.2 Освітлення

Для забезпечення належного освітлення приміщення, де працює програміст, застосовується комбінована система, що поєднує природне освітлення із додатковим штучним світлом. Загальне оздоблення простору виконується за допомогою газорозрядних ламп типу ЛД. Згідно з встановленими нормами, для робочого місця, на якому здійснюються високоточні операції (де мінімальний розмір об'єкта розрізнення становить 0,3–0,5 мм), необхідна освітленість рівномірно має досягати 300 лк. В цілому, ці вимоги щодо освітлення забезпечені.

2.2.3 Шум

У робочих приміщеннях основним джерелом шумового навантаження є звуки, що генеруються ПЕОМ. Крім того, значну частину шуму створюють джерела електромагнітного походження – це коливання компонентів електромеханічних пристроїв під впливом змінних магнітних полів. До того ж, в

приміщеннях виникає структурний шум, який випромінюють поверхні конструктивних елементів (стіни, перекриття, перегородки) у звуковому спектрі частот. Для зниження або усунення негативного впливу шуму доцільно ізолювати робочі зони, розташовуючи їх у частинах будівлі, що знаходяться в глибині та ведуть своїми вікнами у двір – таким чином мінімізується вплив міського шуму. Крім цього, необхідно регулярно перевіряти герметичність корпусів комп'ютерної техніки та своєчасно здійснювати заміну вентиляторів охолодження.

2.3 Вимоги до організації робочого місця працівника

Конструкція робочого місця користувача комп'ютера, з урахуванням розташування сидіння, засобів керування та засобу відображення інформації, розроблена згідно з антропометричними, фізіологічними та психологічними вимогами, а також відповідно до специфіки виконуваної роботи. Робоче меблеве обладнання повинно бути оснащене можливістю індивідуального регулювання, що дозволить адаптувати його під зріст кожного користувача й підтримувати оптимальну, зручну поставу. Робочий стіл рекомендовано обробляти матовим покриттям, що сприяє зменшенню небажаних відблисків. > > Розміщення дисплея організовано таким чином, щоб його верхня межа відповідала рівню очей, а відстань до екрану становила приблизно 70 см – що повністю входить у допустимий інтервал від 60 до 90 см. Частота мерехтіння екрану fмер дорівнює 100 Гц, що значно перевищує мінімальне рекомендоване значення у 70 Гц. Крім цього, робоче місце розташовано перпендикулярно до віконних прорізів, що дозволяє уникнути прямого та відбитого світлового мерехтіння від вікон та джерел штучного освітлення.

2.4 Мікроклімат

Показники мікроклімату, складу іонів у повітрі, а також рівень шкідливих речовин у робочих зонах, де використовуються ПК, мають відповідати вимогам ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Для підтримки нормативних значень мікроклімату та забезпечення оптимального співвідношення позитивних і негативних іонів слід передбачити установку пристроїв зволоження, штучної іонізації або кондиціювання повітря. Крім того, рівні інфрачервоного випромінювання не повинні перевищувати встановлених нормативних меж згідно з ГОСТ 12.1.005. Також вміст озону в робочій зоні не має перевищувати 0,1 мг/м³, оксидів азоту – 5 мг/м³, а концентрація пилу повинна залишатися в межах 4 мг/м³.

2.5 Електробезпека

Приміщення, де використовуються імпульсні джерела живлення згідно з ОНТП24-86 і ПУЕ-87, віднесено до категорії об'єктів, де ризик ураження персоналу електричним струмом не є підвищеним. Це пояснюється тим, що відносна вологість повітря не перевищує 75%, температура залишається нижчою за 35°С, а хімічно агресивні середовища відсутні. Електроживлення обладнання організовано від двофазної мережі з заземленою нейтраллю, при напрузі 220 В і частоті 50 Гц, із застосуванням автоматичних пристроїв токового захисту.

В приміщенні обов’язково має бути встановлена схема заземлення. Ураження електричним струмом може виникнути у випадках: 1) при контакті з відкритими струмоведучими елементами; 2) при торканні неструмоведучих частин обладнання, які, через порушення ізоляції або інші причини, опинилися під напругою.

Відповідно до вимог ГОСТ-12.2.007.0-75 устаткування (за винятком ЕОМ II класу) відноситься до I класу та оснащене робочою ізоляцією згідно з ГОСТ 12.1.009-76. Підключення обладнання здійснено згідно з нормативами ПБЕ та ПУЕ, тому додаткових заходів щодо електробезпеки не вимагається.

2.6 Пожежна безпека

Робоче приміщення, що відповідає вимогам ПБЕ та ОНТП 24–86 у сфері вибухово-пожежної безпеки, класифікується як об’єкт категорії «В».

Основними потенційними причинами виникнення пожежі в такому приміщенні є:

1. Коротке замикання електропроводки;

2. Використання побутових електрорадіоприладів;

3. Недотримання встановлених норм протипожежного захисту.

Відповідно до ПУЕ, для зниження ризику виникнення пожежі необхідно забезпечити комплекс заходів, зокрема: ретельну ізоляцію всіх струмоведучих проводів, що підключені до робочих місць, регулярний огляд та перевірку стану їх ізоляції, а також суворе дотримання норм безпечної експлуатації обладнання.

Для гасіння пожеж на робочому місці користувача ПК застосовують як вуглекислотні, так і порошкові вогнегасники.

– Вуглекислотні вогнегасники випускаються у варіанті ручних пристроїв (наприклад, ВВК-5);

– Порошкові вогнегасники представлені моделями ВП-2, ВП-5, ВП-10 та іншими

Рисунок 2.1. Засоби пожежогасіння

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було реалізовано моделювання потоків даних із застосуванням технології SDN. Основним елементом реалізованої системи є централізований алгоритм Лі, який використовується для формування множини маршрутних шляхів у мережі. Завдяки цьому алгоритму здійснюється детальний аналіз топології мережі на основі побудови повної матриці зв'язності, що дозволяє точно визначити взаємозв'язки між вузлами.

Для виконання аналізу маршрутів створено програмну модель і додаток, який інтегрує централізоване керування потоками даних із зручним графічним інтерфейсом. Користувач має можливість вводити вихідні параметри мережі – кількість вузлів, початкову та кінцеву точки, а також завантажувати вхідні дані у вигляді текстових файлів. Далі система автоматизовано обчислює альтернативні маршрути, кожен з яких характеризується конкретними параметрами передачі даних, такими як пропускна здатність (метрика) та навантаження (вага). Результати моделювання подано як у вигляді таблиць із сортуванням за ефективністю, так і через графічне відображення отриманих маршрутів.

Проведене моделювання дозволило визначити різні сценарії маршрутизації у SDN-середовищі, що характеризуються як високою швидкістю передачі, так і оптимальним використанням мережевих ресурсів. Виявлено, що централізований алгоритм Лі здатний ефективно формувати оптимальну множину маршрутних шляхів, забезпечуючи своєчасну адаптацію до змін топології мережі та умов навантаження. Такий підхід дозволяє оперативно обирати найкращий маршрут для передачі потоку даних і, відповідно, підвищувати загальну продуктивність системи.

Отримані результати підтвердили практичну доцільність і ефективність реалізованого підходу для моделювання потоків даних у програмно-визначуваних мережах. Розглянуте та реалізоване програмно рішення не лише демонструє високу адаптивність і гнучкість у керуванні мережевим трафіком, але й відкриває перспективи для подальшої оптимізації управління ресурсами в умовах змінних мережевих навантажень у сучасних телекомунікаційних системах.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Васильєв І.М. Програмно-визначувані мережі: теоретичний аспект та практичні застосування. – Київ: Технічна література, 2018. – 312 с.
2. Іваненко О.П. Моделювання мережевих потоків: підручник. – Львів: Видавництво «Світ науки», 2019. – 248 с.
3. Петренко С.В., Бойко Н. І. Архітектура програмно-визначуваних мереж. – Харків: Комп’ютерна книга, 2020. – 304 с.
4. Мельник А.І. Інструменти моделювання та оптимізації мережевих потоків. – Одеса: Одеський національний університет, 2021. – 276 с.
5. Скрипник Л.В., Горбунов І. О. Технології SDN: сучасний підхід до мережевої інфраструктури. – Київ: Фенікс, 2022. – 350 с.
6. Гончаренко Є.П. Моделювання потоків даних у телекомунікаційних системах. – Дніпропетровськ: Прогрес, 2018. – 310 с.
7. Захарченко В.О., Синиця І.В. Оптимізація потоків даних у програмно-визначуваних мережах. – Харків: Технічні науки, 2021. – 290 с.
8. Руденко В.М. Аналіз мережевих алгоритмів управління потоками. – Львів: Видавництво «Час», 2020. – 320 с.
9. Сидоренко П.В. Програмно-визначувані мережі та їх вплив на ефективність передачі даних. – Київ: Інститут проблем штучного інтелекту, 2021. – 270 с.
10. Міністерство цифрової трансформації України. Програмно-визначені мережі (SDN) [Електронний ресурс]. – Режим доступу: https://thedigital.gov.ua/sdn (Дата звернення: 20.04.2025).
11. Національний центр інформаційних технологій. Технології SDN у сучасних мережах [Електронний ресурс]. – Режим доступу: https://nci.gov.ua/sdn (Дата звернення: 15.05.2025).
12. Міжнародна асоціація телекомунікаційних інженерів Аналіз потоків даних у програмно-визначуваних мережах [Електронний ресурс]. – Режим доступу: https://itua.org/sdn-analysis (Дата звернення: 05.05.2025).
13. Cisco Introduction to SDN and Traffic Modeling [Електронний ресурс]. – https://ciscoacademy.com/sdn-traffic (Дата звернення: 15.05.2025)

ДОДАТОК А. Фрагменти коду на мові Java додатку для моделювання потоків даних у програмно-визначуваних мережах

```
im import edu.uci.ics.jung.algorithms.layout.FRLayout;
import edu.uci.ics.jung.graph.Graph;
import edu.uci.ics.jung.graph.SparseGraph;
import edu.uci.ics.jung.visualization.VisualizationViewer;
import javafx.application.Application;
import javafx.embed.swing.SwingNode;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import java.awt.Dimension;
import java.util.*;
public class SDNDataFlowModeling extends Application {
   /**
    * Метод buildFullMatrix() формує повну симетричну матрицю зв'язності
    * з пів-матриці суміжності.
    * Матриця задається як масив масивів, де перший рядок містить один елемент,
    * другий – два, і так далі.
    */
   public static int[][] buildFullMatrix(int[][] halfMatrix) {
      int n = halfMatrix.length + 1;
      int[][] fullMatrix = new int[n][n];
      for (int i = 0; i < n - 1; i++) {
         for (int j = 0; j < halfMatrix[i].length; j++) {
            int value = halfMatrix[i][j];
            fullMatrix[i][i + j + 1] = value;
            fullMatrix[i + j + 1][i] = value;
         }
      }
      return fullMatrix;
   }
      /**
    * Реалізація централізованого алгоритму Лі (BFS) для пошуку найкоротшого маршруту.
    *
    * @param matrix повна матриця зв'язності
    * @param start  індекс початкового вузла
    * @param end    індекс кінцевого вузла
    * @return список індексів вузлів, що формують знайдений маршрут;
    *         порожній список, якщо шлях не знайдено.
    */
   public static List<Integer> leeAlgorithm(int[][] matrix, int start, int end) {
      int n = matrix.length;
      int[] dist = new int[n];
      int[] pred = new int[n];
      Arrays.fill(dist, -1);
      Arrays.fill(pred, -1);
            Queue<Integer> queue = new LinkedList<>();
      dist[start] = 0;
      queue.offer(start);
```

```
    while (!queue.isEmpty()) {
        int cur = queue.poll();
        if (cur == end) break;
        for (int neighbor = 0; neighbor < n; neighbor++) {
            if (matrix[cur][neighbor] > 0 && dist[neighbor] == -1) {
                dist[neighbor] = dist[cur] + 1;
                pred[neighbor] = cur;
                queue.offer(neighbor);
            }
        }
    }
            // Відновлення маршруту, якщо шлях знайдено
    List<Integer> path = new ArrayList<>();
    if (dist[end] == -1) {
        return path;
    }
    for (int at = end; at != -1; at = pred[at]) {
        path.add(at);
    }
    Collections.reverse(path);
    return path;
}
    /**
 * Метод buildGraph() створює граф JUNG із заданої повної матриці.
 * Вершини іменуються V0, V1, ..., Vn-1.
 *
 * @param matrix повна матриця зв'язності
 * @return граф типу Graph<String, String>, де ребра отримують унікальні ідентифікатори.
 */
public static Graph<String, String> buildGraph(int[][] matrix) {
    Graph<String, String> graph = new SparseGraph<>();
    int n = matrix.length;
    for (int i = 0; i < n; i++) {
        graph.addVertex("V" + i);
    }
    int edgeId = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (matrix[i][j] > 0) {
                graph.addEdge("E" + edgeId++, "V" + i, "V" + j);
            }
        }
    }
    return graph;
}
    @Override
public void start(Stage primaryStage) {
    // Вхідні дані: пів–матриця зв'язності (як приклад)
    int[][] halfMatrix = {
        {2},
        {3, 4},
        {5, 6, 7},
        {8, 9, 10, 2},
        {1, 3, 4, 6, 7}
    };
    int[][] fullMatrix = buildFullMatrix(halfMatrix);
```

```
        // Попереднє обчислення маршруту алгоритмом Лі між вузлами 0 і 5
        List<Integer> route = leeAlgorithm(fullMatrix, 0, 5);
        // Створення графу з використанням JUNG на основі повної матриці
        Graph<String, String> jungGraph = buildGraph(fullMatrix);
        // Налаштування візуалізації графу за допомогою JUNG (FRLayout)
        FRLayout<String, String> layout = new FRLayout<>(jungGraph);
        layout.setSize(new Dimension(600, 600));
        VisualizationViewer<String, String> vv = new VisualizationViewer<>(layout);
        vv.setPreferredSize(new Dimension(650, 650));
        // Інтеграція JUNG в JavaFX через SwingNode
        SwingNode swingNode = new SwingNode();
        swingNode.setContent(vv);
        // Створення контролів для вводу параметрів і відображення маршруту
        Label routeLabel = new Label("Побудований маршрут: " + route);
        TextField startField = new TextField("0");
        TextField endField = new TextField("5");
        Button calcButton = new Button("Розрахувати маршрут");
        calcButton.setOnAction(e -> {
            int start = Integer.parseInt(startField.getText());
            int end = Integer.parseInt(endField.getText());
            List<Integer> newRoute = leeAlgorithm(fullMatrix, start, end);
            routeLabel.setText("Побудований маршрут: " + newRoute);
        });
        VBox controlPanel = new VBox(10,
                new Label("Початковий вузол:"), startField,
                new Label("Кінцевий вузол:"), endField,
                calcButton, routeLabel);
        // Побудова основного вікна з розташуванням контролів і графу
        BorderPane root = new BorderPane();
        root.setLeft(controlPanel);
        root.setCenter(swingNode);
        Scene scene = new Scene(root, 900, 700);
        primaryStage.setTitle("Моделювання потоків даних у програмно-визначуваних мережах");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

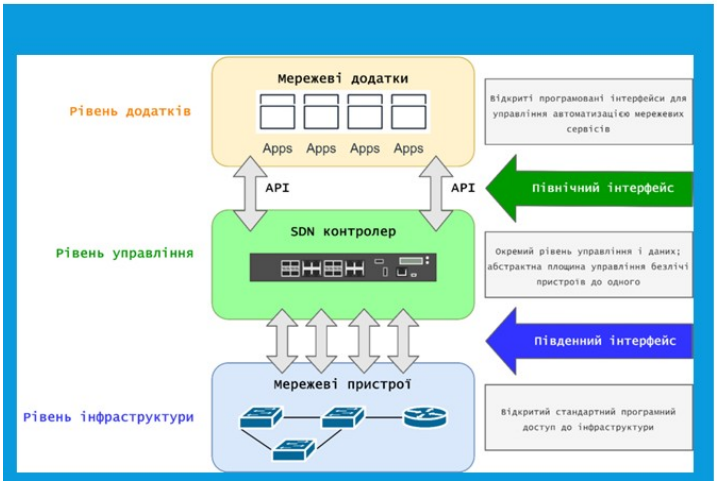
ДОДАТОК Б. Слайди мультимедійної презентації

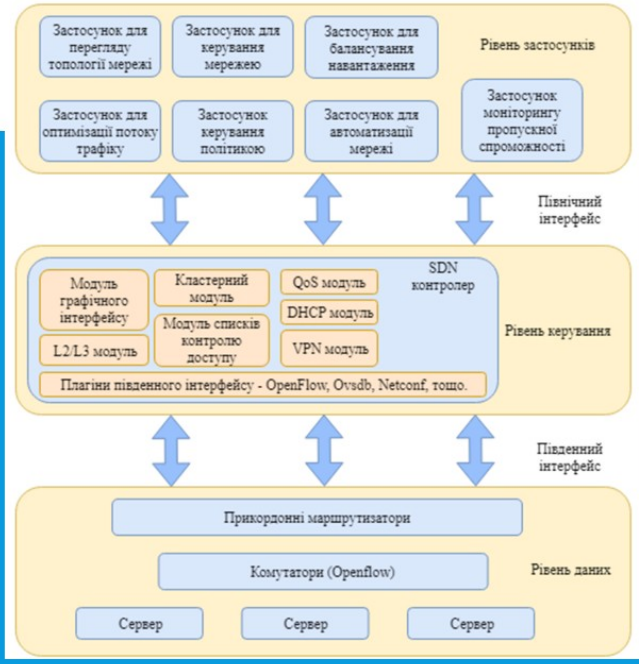
МОДЕЛЮВАННЯ ПОТОКІВ ДАНИХ У ПРОГРАМНО-ВИЗНАЧУВАНИХ МЕРЕЖАХ

Дедігуров
Микита,
гр.2БКС-29

ВСП
“ОТФК ОНТУ”

Архітектура програмно-визначуваних мереж

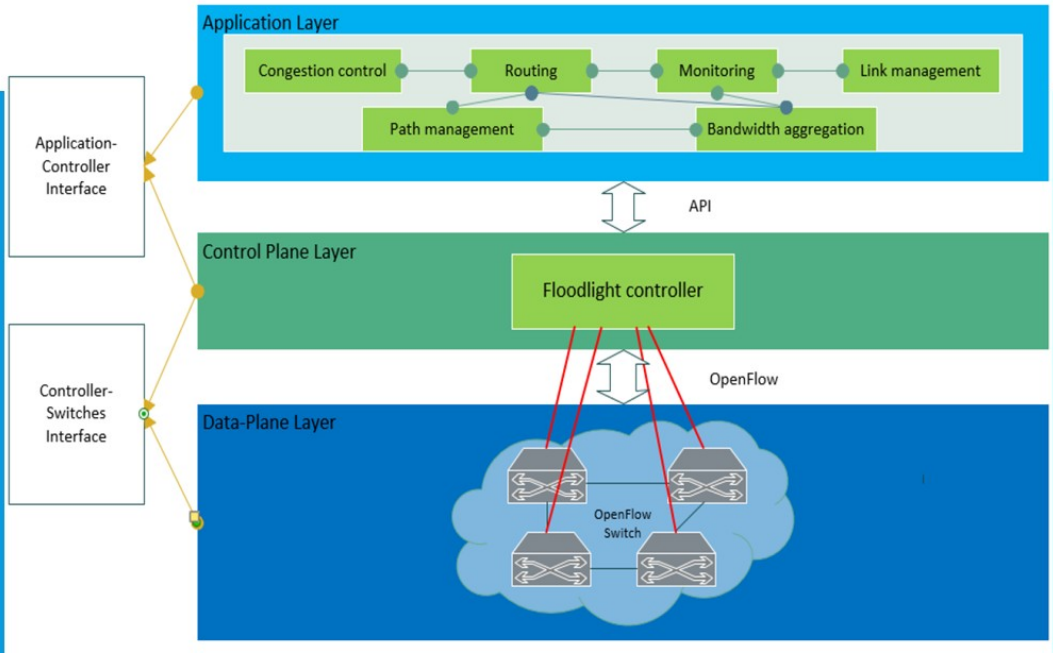




Принцип передачі пакетів у програмно-визначуваних мережах

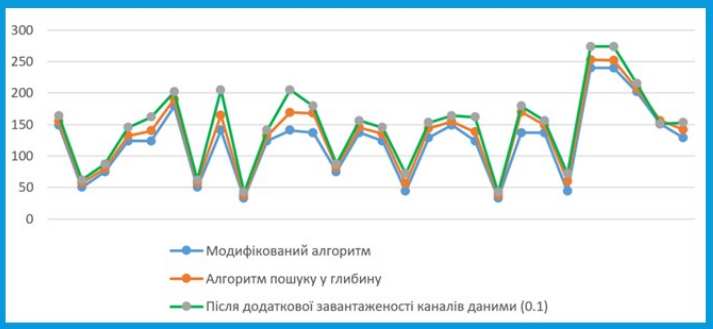
Принцип передачі пакетів у традиційній IP-мережі

Базові інтерфейси у програмно-визначуваній мережі

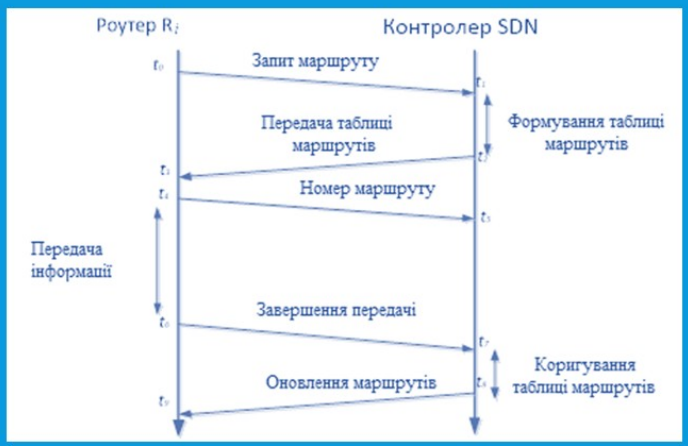


Порівняння алгоритмів розподілення потоків даних

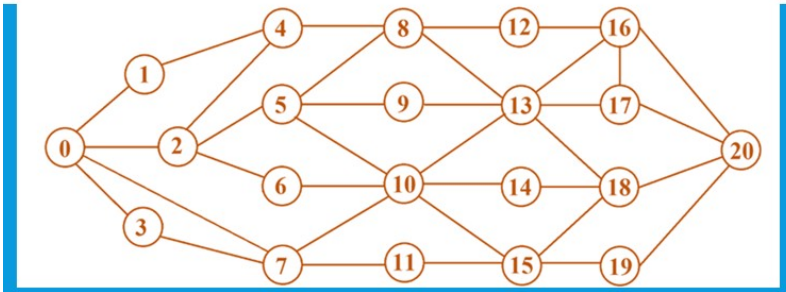
	Швидкість	Точність	Простота алгоритму
Алгоритм пошуку у глибину	Середній	В окремих випадках, некоректні результаті завантаженості каналів	Легка
Модифікований алгоритм пошуку найкоротшого шляху	Швидкий	Точні результати завантаженості каналів	Середня



Процедура обміну службовою інформацією у програмно-визначуваній мережі



Маршрутний граф програмно-визначуваної мережі



[0, 7, 10, 13, 17, 20] Metric = 20.0 MBIT/S Weight = 0.4
[0, 7, 10, 13, 16, 20] Metric = 20.0 MBIT/S Weight = 0.5
[0, 7, 10, 14, 18, 20] Metric = 20.0 MBIT/S Weight = 0.6
[0, 7, 11, 15, 18, 20] Metric = 20.0 MBIT/S Weight = 0.6

Динамічна зміна маршруту

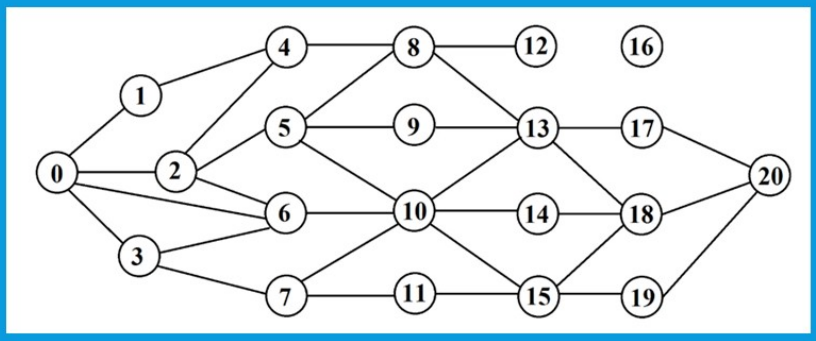
Множина шляхів між роутерами Ro і R20

[0, 7, 10, 13, 17, 20] Metric = 20.0 MBIT/S Weight = 0.4
[0, 7, 10, 13, 16, 20] Metric = 20.0 MBIT/S Weight = 0.5
[0, 7, 10, 14, 18, 20] Metric = 20.0 MBIT/S Weight = 0.6
[0, 7, 11, 15, 18, 20] Metric = 20.0 MBIT/S Weight = 0.6
[0, 7, 10, 15, 18, 20] Metric = 20.0 MBIT/S Weight = 0.7
[0, 7, 10, 15, 19, 20] Metric = 20.0 MBIT/S Weight = 0.7
[0, 7, 11, 15, 19, 20] Metric = 20.0 MBIT/S Weight = 0.7
[0, 7, 10, 13, 18, 20] Metric = 20.0 MBIT/S Weight = 0.8
[0, 1, 4, 8, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 4, 8, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 5, 8, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 5, 9, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 5, 10, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 6, 10, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 3, 7, 10, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 7, 10, 13, 16, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 1, 4, 8, 13, 16, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.5
[0, 2, 4, 8, 13, 16, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.5
[0, 2, 5, 8, 13, 16, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.5
[0, 2, 5, 9, 13, 16, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.5
[0, 2, 5, 10, 13, 16, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.5
[0, 2, 6, 10, 13, 16, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.5
[0, 3, 7, 10, 13, 16, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.5
[0, 7, 10, 13, 17, 16, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.5
[0, 1, 4, 8, 12, 16, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.6

Таблиця маршрутизації при передачі інформації від Ro до R20

Вузли	R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	R_{12}	R_{13}	R_{14}	R_{15}	R_{16}	R_{17}	R_{18}	R_{19}	R_{20}
R_0	0	0.22	0.09	0.10	-	-	-	0.40	-	-	-	-	-	-	-	-	-	-	-	-	-
R_1	0.22	0	-	0.32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R_2	0.09	-	0	-	0.33	0.20	0.11	-	-	-	-	-	-	-	-	-	-	-	-	-	-
R_3	0.10	0.32	-	0	-	-	-	0.30	-	-	-	-	-	-	-	-	-	-	-	-	-
R_4	-	0.30	0.33	-	0	-	-	-	0.12	-	-	-	-	-	-	-	-	-	-	-	-
R_5	-	-	0.20	-	-	0	-	0.21	0.10	0.41	-	-	-	-	-	-	-	-	-	-	-
R_6	-	-	0.11	-	-	-	0	-	-	0.32	-	-	-	-	-	-	-	-	-	-	-
R_7	0.40	-	-	0.30	-	-	-	0	0.10	-	0.41	-	-	-	-	-	-	-	-	-	-
R_8	-	-	-	-	0.12	0.10	-	-	0	-	0.40	0.50	0.70	-	-	-	-	-	-	-	-
R_9	-	-	-	-	-	0.41	-	-	-	0	-	0.40	-	-	-	-	-	-	-	-	-
R_{10}	-	-	-	-	-	-	-	-	0.40	0.40	0	0.30	0.40	0.50	0.70	-	-	-	-	-	-
R_{11}	-	-	-	-	-	-	-	-	0.50	-	-	0	-	0.40	-	-	-	-	-	-	-
R_{12}	-	-	-	-	-	-	-	-	-	-	0.40	-	0	-	0.50	-	-	-	-	-	-
R_{13}	-	-	-	-	-	-	-	-	-	-	0.50	0.40	-	0	0.40	0.20	0.80	-	-	-	-
R_{14}	-	-	-	-	-	-	-	-	-	-	-	-	0.50	-	0	0.50	-	-	-	-	-
R_{15}	-	-	-	-	-	-	-	-	-	-	-	-	-	0.20	-	0	0.40	0.30	-	-	-
R_{16}	-	-	-	-	-	-	-	-	-	-	-	-	-	0.80	-	-	0	0.40	0.30	0.60	-
R17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.14	0	-	-	0.37
R18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.42	-	-	0	-	0.58
R19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.31	-	-	-	0	0.65
R20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0.52	0.38	0.56	0.65	0

Граф мережі при відключеному роутері R16



[0, 7, 10, 13, 17, 20] Metric = 20.0 MBIT/S Weight = 0.4
[0, 7, 10, 14, 18, 20] Metric = 20.0 MBIT/S Weight = 0.6
[0, 7, 11, 15, 18, 20] Metric = 20.0 MBIT/S Weight = 0.6
[0, 7, 10, 15, 18, 20] Metric = 20.0 MBIT/S Weight = 0.7
[0, 7, 10, 15, 19, 20] Metric = 20.0 MBIT/S Weight = 0.7
[0, 7, 11, 15, 19, 20] Metric = 20.0 MBIT/S Weight = 0.7
[0, 7, 10, 13, 18, 20] Metric = 20.0 MBIT/S Weight = 0.8
[0, 1, 4, 8, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 4, 8, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 5, 8, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 5, 9, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 5, 10, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 6, 10, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 3, 7, 10, 13, 17, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.4
[0, 2, 5, 10, 14, 18, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.6
[0, 2, 6, 10, 14, 18, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.6
[0, 3, 7, 10, 14, 18, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.6
[0, 3, 7, 11, 15, 18, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.6
[0, 2, 5, 10, 15, 18, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.7
[0, 2, 5, 10, 15, 19, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.7
[0, 2, 6, 10, 15, 18, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.7
[0, 2, 6, 10, 15, 19, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.7
[0, 3, 7, 10, 15, 18, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.7
[0, 3, 7, 10, 15, 19, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.7
[0, 3, 7, 11, 15, 19, 20] Metric = 16.666666666666668 MBIT/S Weight = 0.7

Множина оптимальних шляхів при відключенні роутеру R16

Схема SDN Datapath

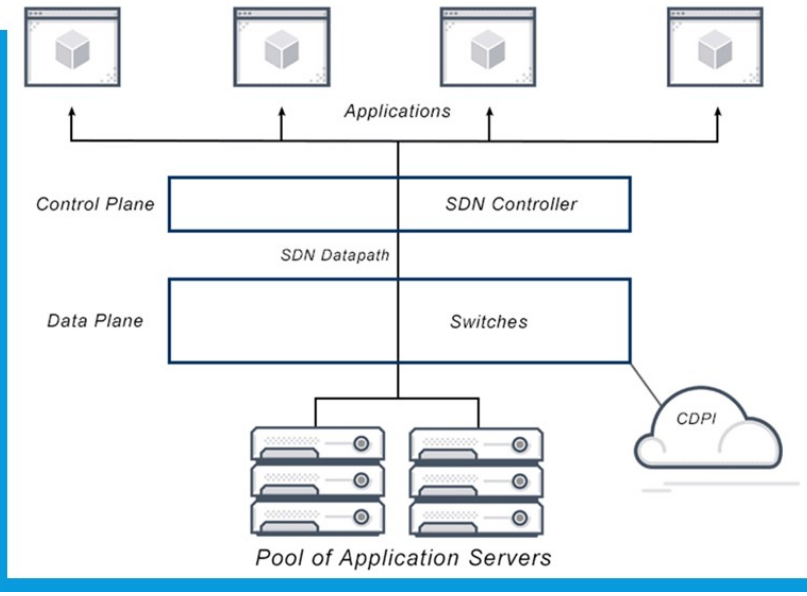
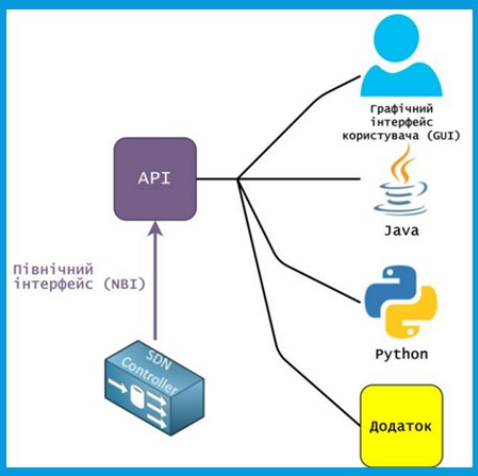
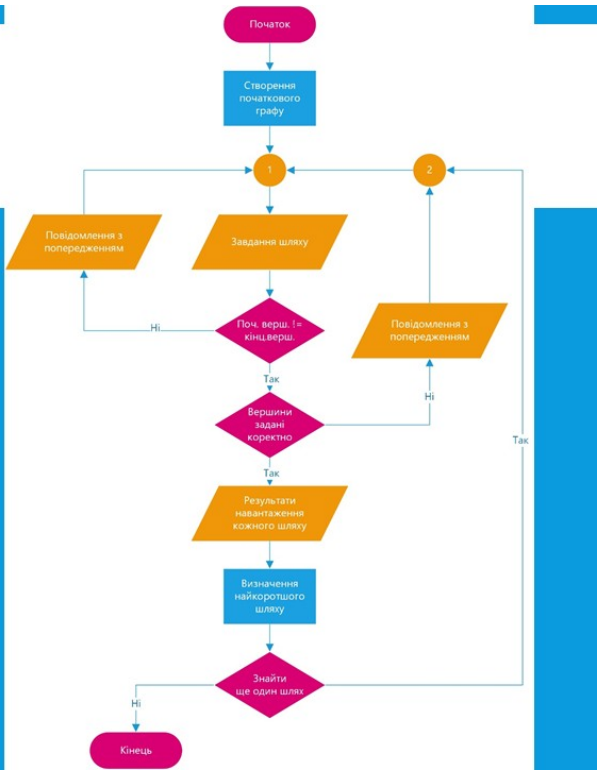


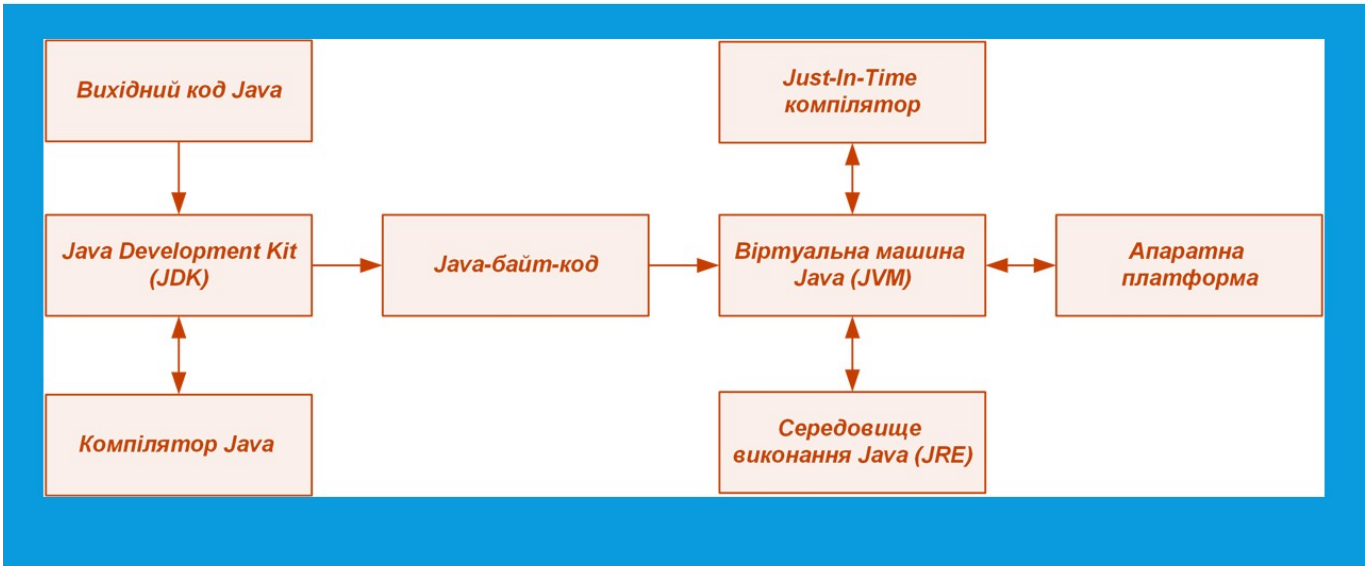
Схема взаємодії додатків SDN із контролером через NBI

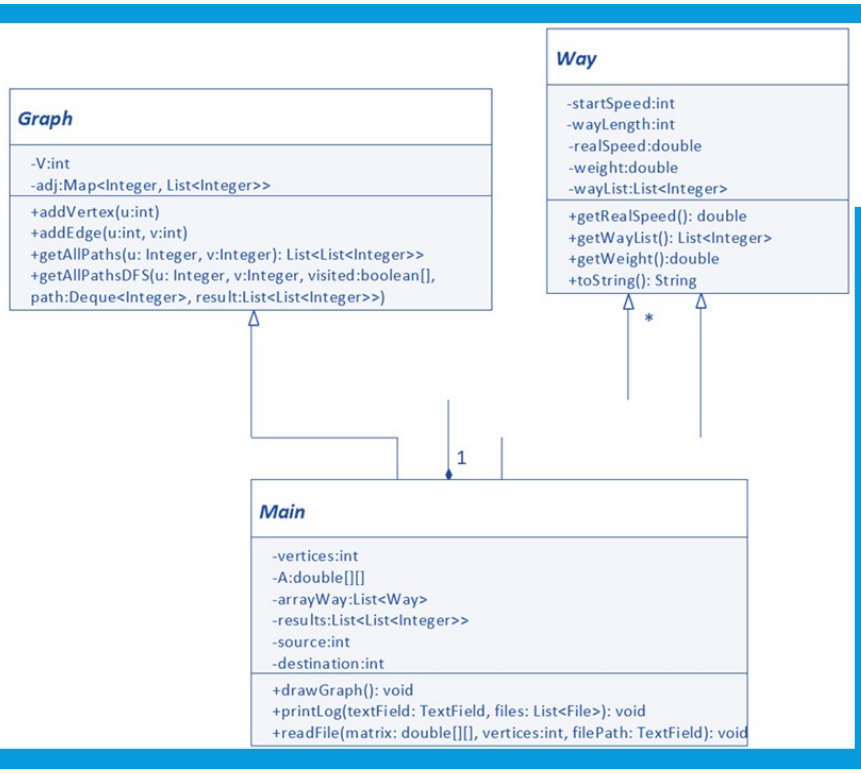


БСА побудови графа програмно-визначуваної мережі



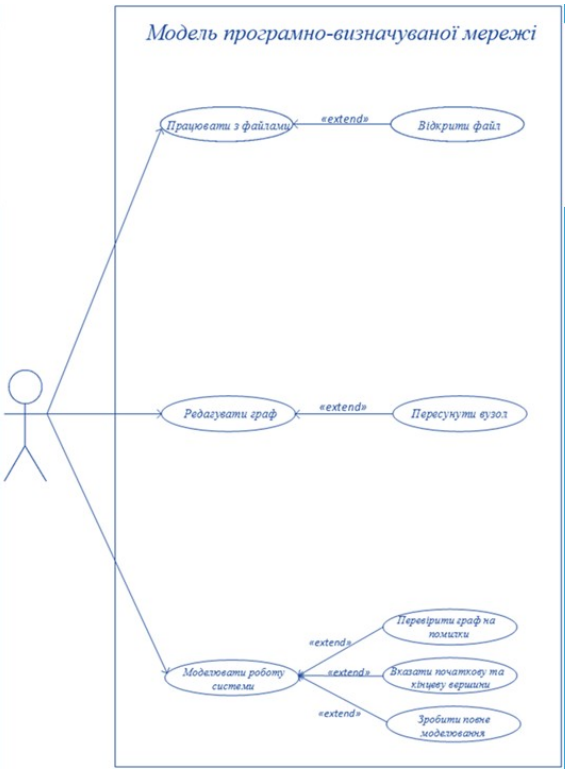
Функціональна схема компіляції програми мовою Java



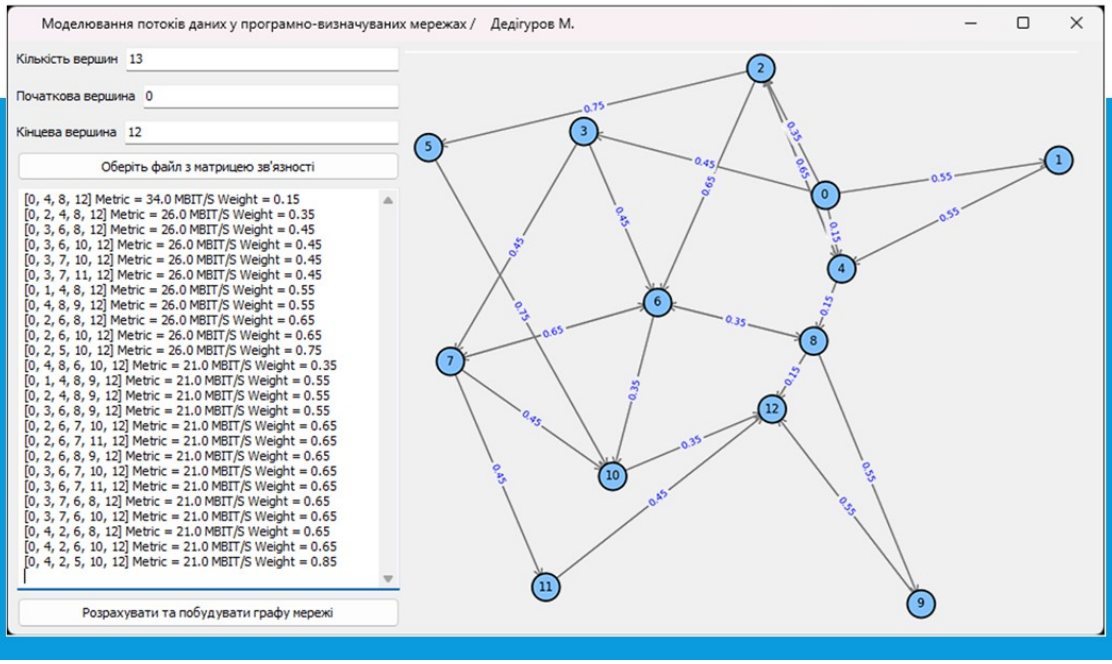


Об'єктно-орієнтована схема додатку для моделювання потоків даних у програмно-визначуваної мережі

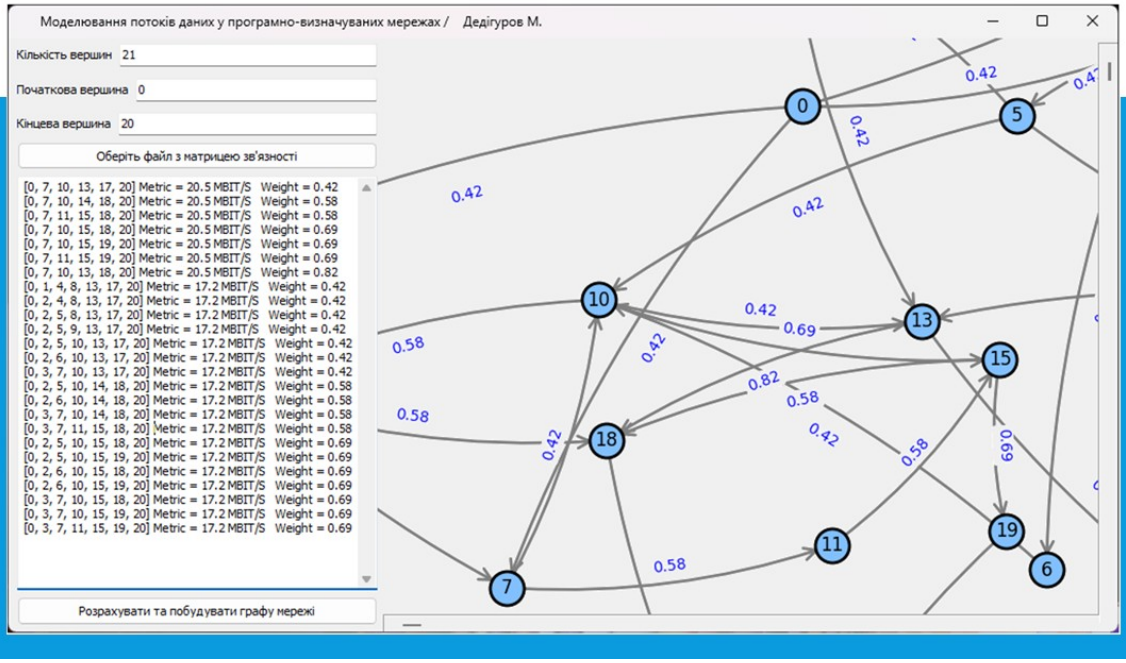
Діаграма взаємодії користувача і додатку для моделювання потоків даних у програмно-визначуваної мережі



Скріншот роботи додатку з результатами моделювання шляхів від V0 до V12



Скріншот роботи додатку з результатами моделювання шляхів від V0 до V21



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу здобувача (здобувачки) освіти
відділення комп'ютерних систем

Дедігурова Микити Олександровича

(прізвище, ім`я та по батькові)

Спеціальність *123 "Комп'ютерна інженерія"*

Освітньо-професійна програма *«Комп'ютерна інженерія»*

Керівник кваліфікаційної роботи *Суліма Юліан Юрійович*

(прізвище, ім`я та по батькові)

Тема кваліфікаційної роботи *Моделювання потоків даних у програмно-визначуваних мережах*

Обсяг розрахунково-пояснювальної записки *72* сторінок

Обсяг графічної (презентаційної) частини *16* аркушів (слайдів)

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаного кваліфікаційної роботи завданню

Представлена на рецензію кваліфікаційна робота бакалавра повністю відповідає меті випускної роботи та технічному завданню. Тематика кваліфікаційної роботи є актуальною для своєї галузі та присвячена моделюванню потоків даних у програмно-визначуваних мережах

б) характеристика виконання кожного розділу кваліфікаційної роботи

Кваліфікаційна робота складається зі вступу, двох розділів, висновків, переліку використаних джерел. У основному розділі виконано аналітичний огляд алгоритмів моделювання потоків даних; планування процедури обміну керуючою інформацією між SDN-контролером і маршрутизаторами; формування маршрутних шляхів для SDN-мережі; моделювання потоків даних за модифікованим алгоритмом; розробка візуального інтерфейсу додатку для моделювання; отримання результатів моделювання та їх аналіз

в) оцінка якості виконання пояснювальної записки та графічної частини кваліфікаційної роботи

Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана охайно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату ідей у роботі не виявлено

г) перелік позитивних якостей кваліфікаційної роботи

Використання комбінованої метрики (врахування не лише кількості переходів, а й характеристик якості обслуговування, завантаження каналів та затримок) є цікавим рішенням для оптимізації маршрутних шляхів у динамічних умовах. Такий підхід дозволяє адаптувати маршрути в реальному часі та сприяє підвищенню ефективності мереж.

д) основні недоліки кваліфікаційної роботи

Використання методу повного перебору шляхів (DFS) для знаходження всіх можливих маршрутів може бути неефективним для великих або густих мереж, оскільки обчислювальна складність зростає експоненційно. Рекомендовано розглянути евристичні методи або обмеження числа аналізованих шляхів для зменшення навантаження.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис:

«23» 06 2025 р.

ВСП «Одеський технічний фаховий коледж ОНТУ»

ВІДГУК

керівника про кваліфікаційну роботу бакалавра

Дедігурова Микити Олександровича

(прізвище, ім`я та по батькові)

Спеціальність *123 "Комп'ютерна інженерія"*

Тема кваліфікаційної роботи *Моделювання потоків даних у програмно-визначуваних мережах*

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) Обсяг і якість виконання роботи (графічного матеріалу і розрахунково-пояснювальної записки) *Випускна робота виконана відповідно технічному завданню. Пояснювальна записка до випускної роботи містить 72 сторінок. У пояснювальній записці розглянуто проблему маршрутизації та оптимального розподілу потоків даних у програмно-визначуваних мережах, реалізовано програмний застосунок, що виконує побудову графу. Графічна частина складається з 16 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра, розробку виконано у повному обсязі.*

б) Самостійність роботи *Протягом виконання випускної бакалаврської роботи Дедігуров Микита поступово та послідовно виконувала всі етапи, проявила ініціативу у створенні загальної концепції та реалізації випускної роботи. Всі роботи вона виконувала самостійно, з оглядом на рекомендації керівника.*

в) Теоретична підготовка здобувача освіти

Дедігуров Микита під час роботи над випускною бакалаврською роботою вивчив достатню кількість літературних джерел за даною тематикою.

Вважаю, що теоретична підготовка здобувача освіти достатня і він готовий до захисту роботи.

г) Вміння розв`язувати виробничі і конструкторські питання на базі останніх досліджень науки і техніки, передових методів виробництва

Під час виконання роботи Дедігуров Микита мав змогу самостійно приймати окремі рішення з виконання програмної частини роботи та показав вміння організовано працювати над поставленою задачею, складати та оформлювати презентацію проекту, користуючись сучасними комп'ютерними програмними засобами, такими як IntelliJ IDEA, JavaFX

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Добре*

Загальна оцінка *Відмінно*

Прізвище, ім`я, по батькові *Суліма Юліан Юрійович*

Місце роботи і посада керівника роботи *ВСП "Одеський технічний фаховий коледж ОНТУ", к.т.н., викладач кафедри комп'ютерної інженерії, заступник директора з НМР*

Підпис

«20» 06 2025 р.

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Дедігуров М.О.,
здобувач освіти гр. 2БКС-29, та

Суліма Ю.Ю.,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до випускної кваліфікаційної роботи бакалавра на тему:

«Моделювання потоків даних у програмно-визначуваних мережах» (автор роботи – Дедігуров М.О., керівник роботи – Суліма Ю.Ю.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи, і даємо згоду на обробку персональних даних.

Виконавець _____ / Дедігуров М.О. /

Керівник _____ / Суліма Ю.Ю. /

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
кафедри комп'ютерної інженерії
здобувача (здобувачки) освіти II курсу
відділення комп'ютерних систем групи 2БКС-29

Дедігурова Микити Олександровича

на тему *Моделювання потоків даних у програмно-визначуваних мережах*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до кваліфікаційної роботи виконана з несуттєвими порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *20.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 31.05.2025 р. значення коефіцієнту подібності в роботі становить 3,48%, коефіцієнт цитування – 0,66%.*

(підпис) *20.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) кваліфікаційної роботи

здобувача (здобувачки) освіти *Дедігурова М.О.* (П.І.Б.)

проведена « *20* » *червня* 2025 р.

Висновки *Пояснювальна записка до кваліфікаційної роботи виконана у повному обсязі. Випускна кваліфікаційна робота відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Зав. кафедри КІ (підпис) *Іванова Л.В.* (П.І.Б.)

StrikePlagiarism.com

Дата звіту 5/31/2025
Дата редагування 6/1/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Моделювання потоків даних у програмно-визначуваних мережах

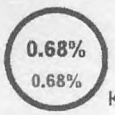
Автор
Дедігуров Микита Олександрович
Науковий керівник / Експерт
Суліма Юліан Юрійович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

КП 1



КЦ

25	14536	114691
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	2
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	58

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Аналіз ефективності алгоритмів розподілу навантаження до контролерів SDN [redacted] 5/31/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	52 0.36 %
2	https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download	52 0.36 %
3	https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download	43 0.30 %

4	http://repo.poltekkesdepkes-sby.ac.id/6330/20/19.%20LAMPIRAN%20KTI.pdf	41 0.28 %
5	https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download	32 0.22 %
6	https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download	30 0.21 %
7	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	27 0.19 %
8	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download	23 0.16 %
9	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download	21 0.14 %
10	https://studfile.net/preview/3741541/page:13/	18 0.12 %

з домашньої бази даних (0.75 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Аналіз ефективності алгоритмів розподілу навантаження до контролерів SDN [redacted] 5/31/2025 **Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")**	**102 (6) 0.70 %**
2	Створення web-застосунку цифрового помічника з використанням Open AI [redacted] 5/28/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	7 (1) 0.05 %

з програми обміну базами даних (1.44 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Конструювання трафіку в мережі SDN [redacted] 3/15/2025 National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute (National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute)	193 (22) 1.33 %
2	asss444.txt [redacted] 2/22/2023 Astana IT University (Astana IT University)	17 (1) 0.12 %

з Інтернету (8.43 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/bitstreams/0e6c3361-ffbf-4469-86a1-fe84a1fe21cd/download	611 (53) 4.20 %
2	**https://card-file.ontu.edu.ua/bitstreams/361286d7-8a03-4221-ad05-db5133ab5f79/download**	**192 (8) 1.32 %**
3	**http://repo.poltekkesdepkes-sby.ac.id/6330/20/19.%20LAMPIRAN%20KTI.pdf**	**84 (7) 0.58 %**
4	https://card-file.ontu.edu.ua/bitstreams/1dff552d-7200-49b8-ae1d-ba76a1335685/download	76 (7) 0.52 %
5	https://card-file.ontu.edu.ua/server/api/core/bitstreams/e86ba9fc-9135-43bb-922a-10bf0bce46b5/content	60 (5) 0.41 %
6	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	33 (4) 0.23 %
7	https://studfile.net/preview/3741541/page:13/	30 (2) 0.21 %
8	**https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download**	**27 (1) 0.19 %**

9	https://vertex-academy.com/tutorials/ru/transponirovanie-matricy-v-java/	25 (2) 0.17 %
10	https://www.nowcoder.com/questionTerminal/17ab1e527c504df09a600e1af09d9a60	21 (2) 0.14 %
11	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	17 (1) 0.12 %
12	http://www.dnaop.com/html/2297/doc-%D0%94%D0%A1%D0%B0%D0%BD%D0%9F%D1%96%D0%BD_3.3.2.007-98	16 (1) 0.11 %
13	http://elartu.tntu.edu.ua/bitstream/lib/31257/2/dyplom_Prystazh_I_2019.pdf	14 (1) 0.10 %
14	https://www.nowcoder.com/questionTerminal/a5c097f75db8418395b6a0e32c608c38	8 (1) 0.06 %
15	https://card-file.ontu.edu.ua/bitstreams/d42aac6d-ab01-4a74-b9cb-ced2a9eff719/download	6 (1) 0.04 %
16	https://github.com/akash-coded/core-java/discussions/7	5 (1) 0.03 %

Список прийнятих фрагментів (немає прийнятих фрагментів)

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Комп'ютерна інженерія» **Група:** 2БКС- 29

КВАЛІФІКАЦІЙНА
РОБОТА

здобувача **освіти денної форми навчання БКС.** 29.07.000.КРБ

ДЕДІГУРОВА
МИКИТИ
ОЛЕКСАНДРОВИЧА м. Одеса
2025 р. МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»
Освітньо-професійна програма: «Комп'ютерна інженерія»
Група: 2 БКС-29

ПОЯСНЮВАЛЬНА **ЗАПИСКА До кваліфікаційної роботи бакалавра на тему:**_____
_____
_____ **Проектний матеріал складається з пояснювальної записки на** _____ **сторінках та графічного (презентаційного) матеріалу на** _____ **аркушах (слайдах) Виконавець** _____ **(Дедігуров М.О.)**

Керівник проекту_____**(Суліма Ю.Ю.)**

Консультанти:

з розділу охорони праці та техніки безпеки _____ **(Чорновол Н.І.)** з нормоконтролю _____**(Петрашова В.І.) старший** консультант _____**(Кривченко Ю. В.) До** захисту допущений
_____ Завідувач кафедри _____ **(Іванова Л.В.) Завідувач** відділення _____
(Краснокутська К.Г.)

_____ Захист «____»_____ 202____ **р.** Протокол ЕК No_____
Оцінка ЕК _____

Секретар ЕК _____

Моделювання потоків даних у
програмно-визначуваних мережах
72
16