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

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Комп'ютерна інженерія»

Група: 2БКС-27

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

здобувача освіти денної форми навчання

БКС.27.16.000.КРБ

МІЛЄВА
МИКИТИ ІВАНОВИЧА

м. Одеса
2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

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

Освітня програма: **«Комп'ютерна інженерія»**

Група: **2БКС-27**

ПОЯСНЮВАЛЬНА ЗАПИСКА

До кваліфікаційної роботи бакалавра на тему: **«Дослідження алгоритмів маршрутизації у програмно-визначуваних мережах»**

Проектний матеріал складається з пояснювальної записки на 68 сторінках та графічного (презентаційного) матеріалу на 17 аркушах (слайдах)

Виконавець _____ (Мілєв М.І.)

Керівник проекту _____ (Кривченко Ю.В.)

Консультанти:

з охорони праці _____ (Чорновол Н.І.)

з дотримання вимог ЄСКД _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Завідувач кафедри _____ (Іванова Л.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «24» 06 2023 р. Протокол ДКК № 2

Оцінка ДКК 5 (відм)

Секретар ДКК _____

АНОТАЦІЯ

У кваліфікаційній роботі виконано дослідження особливостей і можливостей алгоритмів маршрутизації і кластеризації програмно-визначуваних мереж та запропоновано удосконалений алгоритм маршрутизації з розподілом щільності розташування контролерів мережі.

Виконано огляд існуючих алгоритмів маршрутизації та кластеризації програмно-визначуваних мереж та визначені їх можливості, переваги та обмеження.

Виконано удосконалення алгоритму маршрутизації та кластеризації у програмно-визначуваних мережах, що дозволяє забезпечити більш рівномірний розподіл навантаження на мережеві контролери і маршрутизатори.

Виконано опис процесу моделювання роботи мережі, маршрутизації та розташування контролерів програмно-визначуваних мереж. Виконано програмну реалізацію розробленого алгоритму, протестовано роботу програми з різними налаштуваннями, описані основні класи та методи, що лежать у основі розробленої мовою Java програми. Проаналізовано роботу програми на заданому прикладі графу мережі, оцінено вплив на ефективність вдосконаленого алгоритму параметра максимального значення найближчої дистанції до маршрутизатора мережі.

Проведено порівняння отриманих результатів моделювання модифікованого алгоритму з іншими існуючими алгоритмами маршрутизації у програмно-визначуваних мережах.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення Комп'ютерних систем Кафедра Комп'ютерної інженерії

Спеціальність 123 «Комп'ютерна інженерія»

Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР Беркань І.В.

"_____" _____ 202____ р.

ЗАВДАННЯ

на кваліфікаційну роботу бакалавра

Здобувачеві (здобувачці) освіти Мілєву Микиті Івановичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Дослідження алгоритмів маршрутизації у програмно-визначуваних мережах

затверджена наказом по коледжу від "17" жовтня 202 2 р. № 235-А2-ОД

2. Термін здачі кваліфікаційної роботи 15 червня, 2023 р.

3. Вихідні дані до роботи 1. Варіанти розташування контролерів у програмно-визначуваних мереж; 2. Модифікований алгоритм має забезпечувати високу ефективність службового трафіку та затримки швидкості; 3. Додаток має створювати, зберігати, редагувати граф мережі, моделювати обраний алгоритм маршрутизації та кластеризації, аналізу результатів моделювання; 4. Мати можливість розширення можливостей за рахунок додавання інших алгоритмів

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Структура програмно-визначуваних мереж; Методи маршрутизації у програмно-визначуваних мережах; Аналіз способів вирішення проблем маршрутизації; Розробка модифікованого алгоритму маршрутизації; Вирішення проблеми маршрутизації у кластерах; Розробка БСА додатку та його ОО-моделі; Розробка та опис інтерфейсу додатку для моделювання алгоритму оптимального вибору маршруту; Порівняння з іншими алгоритмами та аналіз результатів

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Структура програмно-визначуваної мережі за протоколом OpenFlow; Архітектура технології HyperFlow; Кластеризація у мережі; Вихід з ладу деяких вузлів; Маршрутизація без урахування навантаження контролерів; Маршрутизація за методом k-середніх; Маршрутизація мережі за щільністю; Результати роботи удосконаленого алгоритму маршрутизації; Розподіл локальної щільності; БСА програми для моделювання маршрутизації; Результати створення графу мережі; Результати маршрутизації у імітованій мережі; Залежність продуктивності різних алгоритмів маршрутизації від кількості вузлів мережі

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

Розділ	Консультант	ПІДПИС	
		Завдання видав	Завдання прийняв
Технологічний	Кривченко Ю.В.		
Охорона праці	Чорновол Н.І.		
Нормоконтроль	Петрашова В.І.		
Старший консультант	Кривченко Ю.В.		

7. Дата видачі завдання 11.05.23

Керівник роботи Кривченко Ю.В. _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів кваліфікаційної роботи	Термін виконання етапів кваліфікаційної роботи	Примітка
1.	Вступ. Аналіз технічного завдання	4.05.2023	Виконано
2.	Огляд методів маршрутизації	8.05.2023	виконано
3.	Способи вирішення проблеми розміщення	10.05.2023	виконав
4.	у програмно-визначуваних мережах	15.05.23	виконав
5.	Аналіз оптимального розгортання	17.05.23	виконав
6.	контролерів		
7.	Розробка математичної моделі процесу	22.05.23	виконав
8.	кластеризації програмно-визначуваних		
9.	мереж		
10.	Оцінка обчислювальної складності	23.05.23	виконав
11.	розробленого алгоритму		
12.	Моделювання способу кластеризації	26.05.23	виконав
13.	Вибір і аналіз засобів розробки	27.05.23	виконав
14.	Розробка програмного забезпечення для	28.05.23	виконав
15.	маршрутизації та кластеризації	31.05.23	виконав
16.	Тестування методу кластеризації мереж	1.06.23	виконав
17.	Аналіз результатів тестування	2.06.23	виконав
18.	Розробка питань з охорони праці	3.06.23	виконав
19.	Підготовка матеріалів мультимедійної	4.06.23	виконав
20.	презентації		

Виконавець _____
(підпис)

Керівник роботи _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			БКС 27. 16 000. 00 КРБ	Випускна робота		
А4			БКС 27. 16 000. 00 КРБ ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	БКС 27. 16 000. 00 КРБ		
Розробив		Мілєв М.І.			Дослідження алгоритмів маршрутизації у програмно-визначуваних мережах	Літ.	Аркуш
Перевірив		Кривченко Ю.В.				Н Д П	4 / 68
Н. Контр.		Петрашова В.І.				ВСП "ОТФК ОНТУ" гр.2БКС-27	
Затверд.		Іванова Л.В.					

ЗМІСТ

ВСТУП

Через високу складність структур сучасних мереж задачі маршрутизації вирішуються не в повному обсязі та мають великі затрати часу. Сьогодні одним з найбільш поширених рішень для забезпечення необхідного сервісу заданому трафіку в певних технологічних рамках є впровадження технології QoS (Quality of Service). Однак ця технологія не є гнучкою. Важливим є питання підвищення продуктивності використання мережі з зовнішнім каналом, який має низьку пропускну спроможність [1].

Виникає задача відрізнення трафіку та здійснення автоматичного регулювання пропускної здатності для певних його видів. Рівномірне навантаження мережі і зниження подальших енергозатрат – дуже актуальна сучасна задача. Для вирішення подібних проблем є доцільним використання технології програмно-визначуваних мереж [2]. Ця архітектура відокремлює функції мережевого управління і пересилання, дозволяючи мережевому управлінню стати безпосередньо програмованим і абстрагувати базову інфраструктуру для додатків і мережевих служб. Концепція програмно-визначуваних мереж швидко розвивається і дозволяє спростити управління мережею та підвищити гнучкість та масштабованість. Контролер на рівні управління керує мережевими маршрутизаторами, надаючи маршрутизаторам правила, які повинні виконуватися при перенесенні пакетів по мережі. Така схема спрощує функції маршрутизаторів і прискорює їх роботу.

Оптимальна схема розгортання контролерів у великих мережах дозволяє максимально використати існуючу структуру з'єднань між мережевими вузлами.

Для кожного вузла мережі можна обчислити певні показники, пов'язані з затримкою, балансом навантаження, допущеною несправністю та оцінити, наскільки вдалим є алгоритм маршрутизації.

У даній випускній роботі виконується аналіз існуючих алгоритмів маршрутизації у програмно-визначуваних мережах і пропонується один з алгоритмів для рішення проблем маршрутизації у таких мережах.

1 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

1.1 Структура програмно-визначуваних мереж

Правила маршрутизації даних у програмно-визначуваних мереж визначаються централізованим пристроєм – програмованим контролером, тоді як пристрої передачі даних (вузли) у мережі керуються цими правилами і, таким чином, здатні швидше надсилати пакети. Таке управління можливе завдяки програмному інтерфейсу між контролером та вузлами. Найпоширеніший програмний інтерфейс – OpenFlow. Архітектура програмно-визначуваних мереж охоплює три рівні (рис. 1.1):

– інфраструктурний рівень: мережеве обладнання (пристрої та канали передачі даних);
– рівень управління: мережева операційна система, яка забезпечує програмним інтерфейсом для управління мережею;
– прикладний рівень: віртуалізація мережі, програми захисту, управління та інші інструменти для підвищення ефективності управління мережею.

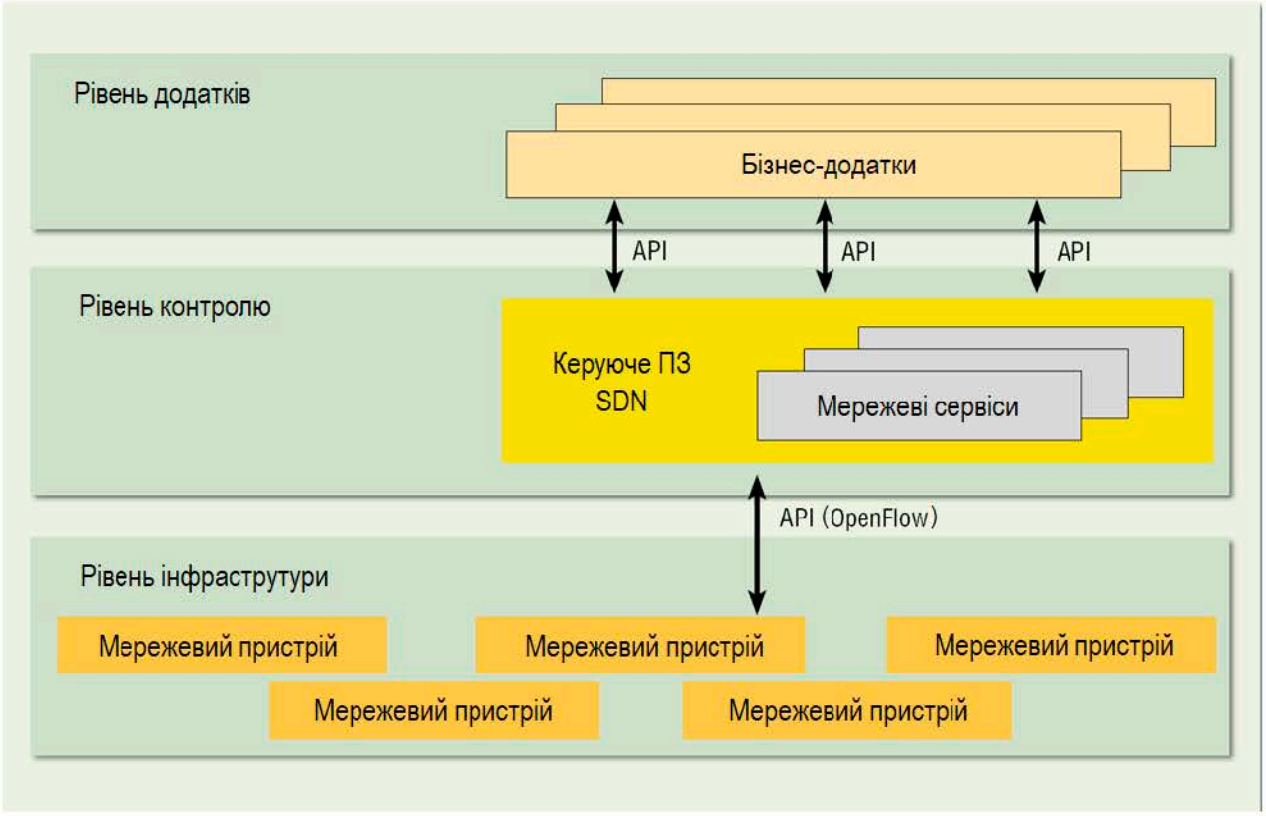


Рисунок 1.1. Структура програмно-визначуваної мережі

Основні ідеї та принципи програмно-визначуваних мереж наступні:

- розділення процесів передачі та управління даними;
- єдиний, незалежний від постачальника інтерфейс між шаром управління та шаром передачі даних;
- централізоване управління мережею за допомогою контролера із встановленою мережевою операційною системою та встановленими мережевими додатками;
- віртуалізація мережевих фізичних ресурсів.

Виділення функції управління на окремий шар та її перенесення до центрального компонента (мережевого контролера) спрямоване на оптимізацію конфігурації мережі для задач додатків. Це, зокрема, дозволяє отримати такі переваги, як відсутність обмежень обладнанням або виробником на формати даних, правила обробки та технології передачі в маршрутизаторі. Окрім того, передача може бути визначена багатовимірним вектором, що охоплює області з різних рівнів моделі мережевої взаємодії. Автоматизовані методи корекції потоку залежно від завантаження компонентів та інших критеріїв можуть застосовуватися при встановленні правил передачі пакетів. Мережеві контролери можуть бути інтегровані у мережеві домени, що дозволяють оптимізувати та резервувати канали передачі.

На рис. 1.2 показана ідея програмно-визначуваних мереж створити єдиний, незалежний від постачальника мережі інтерфейс між контролером та мережевим транспортним середовищем, що відображене у протоколі OpenFlow. Це дозволяє користувачам визначати та керувати, з ким, за яких умов, з якою якістю відбудеться взаємодія у мережі. Цей протокол підтримує три типи повідомлень: асинхронне, симетричне та контролер-маршрутизатор.

Ініційовані маршрутизатором асинхронні повідомлення оповіщують контролер про події в мережі (прихід пакету або видалення запису з таблиці коли час виходить) та зміни стану або помилки маршрутизатора.

Симетричні повідомлення використовуються для встановлення з'єднання, а також для контролю затримок з'єднання контролер-маршрутизатор, для

вимірювання пропускної здатності або для перевірки активності з'єднання. Їх можна запускати без потреби в запиті.

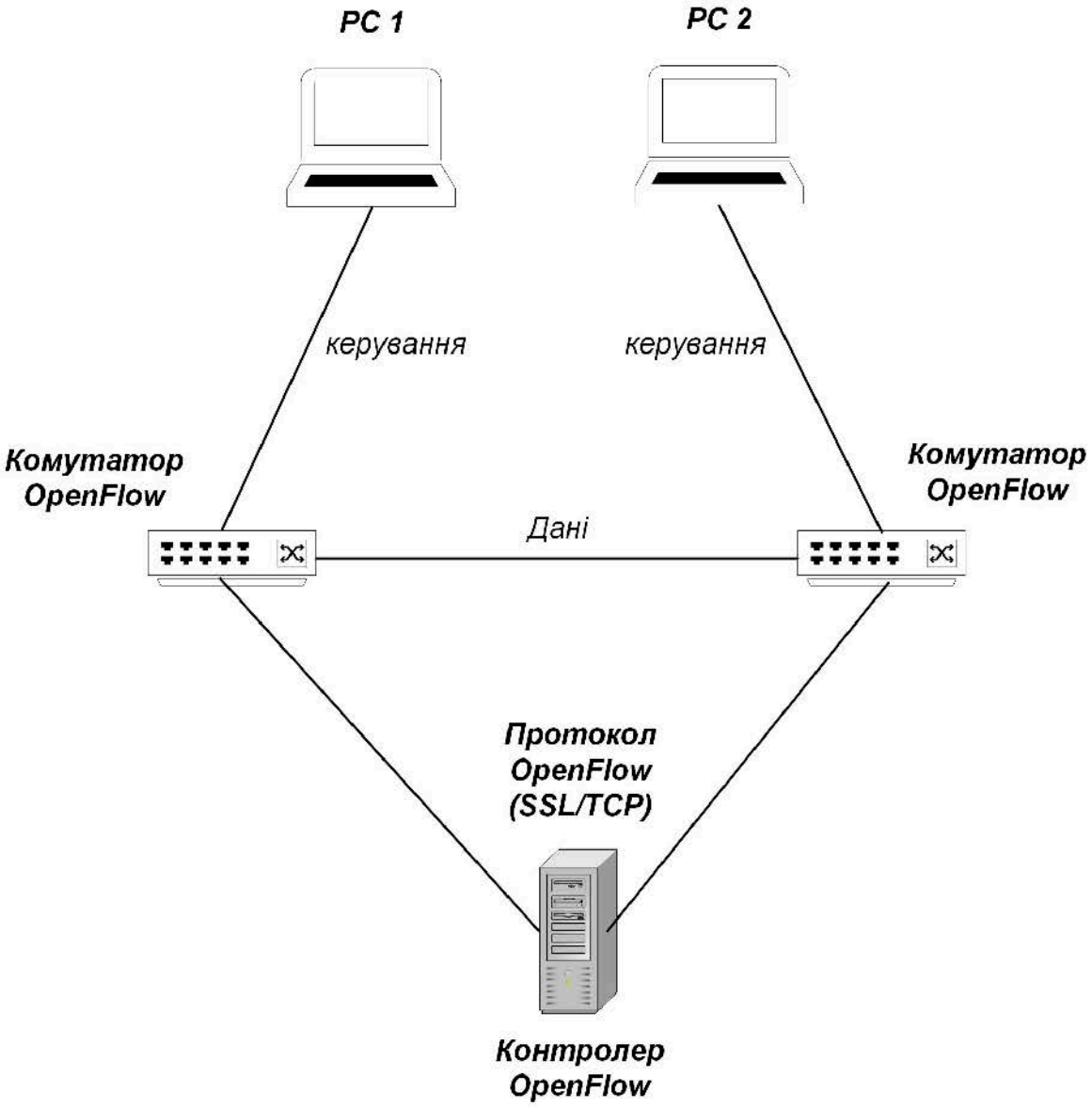


Рисунок 1.2. Структура мережі за протоколом OpenFlow

Повідомлення типу контролер-маршрутизатор ініціює контролер і використовує їх для безпосереднього управління та контролю стану маршрутизатора. Цей тип повідомлень може використовуватися контролером для встановлення параметрів конфігурації маршрутизатора, додавання, видалення та зміни записів у тематичних таблицях, збору статистики.

Віртуалізація мереж є однією з ідей програмно-визначуваних мереж з метою більш ефективного використання мережевих ресурсів. Віртуалізація мережі – це

ізоляція мережевого трафіку – мультиплексування декількох потоків даних з різними характеристиками в межах однієї логічної мережі, яка може спільно використовувати одну фізичну мережу з іншими логічними мережами або фрагментами мережі. Кожен з цих фрагментів надає власні адреси, алгоритми маршрутизації, управління якістю обслуговування тощо, які може використовувати.

Віртуалізація мережі дозволяє:

- підвищити ефективність розподілу мережевих ресурсів і балансувати навантаження;
- ізолювати потік різних користувачів та додатків у межах однієї фізичної мережі;
- використовувати політику маршрутизації та політику управління потоками даних адміністраторам різних фрагментів;
- використовувати в кожному розділі лише послуги, необхідні для спеціальних програм;
- проводити мережеві експерименти з використанням реальної фізичної мережевої інфраструктури.

У якості засобів віртуалізації програмно-визначуваних мереж використовуються проксі-програми, засновані на протоколі OpenFlow. Вони працюють на рівні між маршрутизаторами та контролерами програмно-визначуваних мереж. За допомогою цих програм можна створювати логічні сегменти, які забезпечують взаємну ізоляцію таких логічних мереж, використовуючи різні алгоритми управління потоком даних.

Кожен контролер керує лише власною логічною мережею і не може впливати на роботу іншого. Для контролера, який спілкується з пристроями OpenFlow за допомогою проксі-програми, повідомлення з'являється так, ніби контролер підключений до звичайної програмно-визначуваної мережі. Таким чином, віртуалізація не змінює спосіб його роботи.

Початкова реалізація протоколу OpenFlow передбачала єдиний контролер у мережі, однак оскільки масштаби мереж OpenFlow швидко зростали, стало

зрозуміло, що одного контролеру для всієї мережі замало з декількох причин:

- трафік у напрямку централізованого контролера збільшується із кількістю маршрутизаторів;
- якщо мережа має великий діаметр, деякі вузли дуже довго чекатимуть інструкцій від контролера незалежно від місця розташування контролера;
- складність системи полягає е роботі контролера, створюючи таким чином чергу, яка може загальмувати всю мережу.

Таким чином, розподілена модель програмно-визначуваних мереж має на меті виправити проблему, що може призвести до несправності всієї мережі, та покращити розширення, розподіливши навантаження між кількома контролерами.

Розподілені рівні управління програмно-визначуваних мереж були розроблені таким чином, щоб вони були досить чутливими для управління локальними подіями в центрах обробки даних, де контролери обмінюються великою кількістю інформації для забезпечення глобальної та детальної цілісності мережі.

Зокрема, розподілена архітектура програмно-визначуваних мереж для багатодоменних програмно-визначуваних мереж, у яких використовуються різні технології, починаючи від широкосмугової технології оптичного волокна до безпровідних ланок з обмеженою смугою пропускання, легко адаптується до потреб користувачів і додатків. Окрім того, розподілений контролер здатний швидше та ефективніше керувати глобальними подіями, є більш чутливим і надійним.

Існуючі розподілені рішення можна розділити на три типи:

1. Рішення, спрямовані на підвищення продуктивності окремих контролерів, таких як Maestro та McNettle, застосовуючи паралелізм на рівні маршрутизатора;
2. Архітектури із розподіленим контролером, такі як FlowVisor, Onix, HyperFlow, Devoflow. Підтримується логічно централізоване управління. Рівні управління розподіляються між різними частинами мережі:

використовуються розподілена синхронізація між локальними подіями, розподілені хеш-таблиці, для створення кластерних контролерів або розподіленої файлової системи. Приклад розподіленої архітектури показаний на рис. 1.3;

3. Багаторівнева архітектура з розподіленими контролерами. Наприклад, Kandoo визначає два ієрархічні рівні контролерів: нижній рівень контролерів, які не пов'язані один з одним, і кожен з яких управляє власною групою маршрутизаторів без інформації про глобальний мережі, та верхній рівень – логічно централізований кореневий контролер, який відповідає за стан мережі.

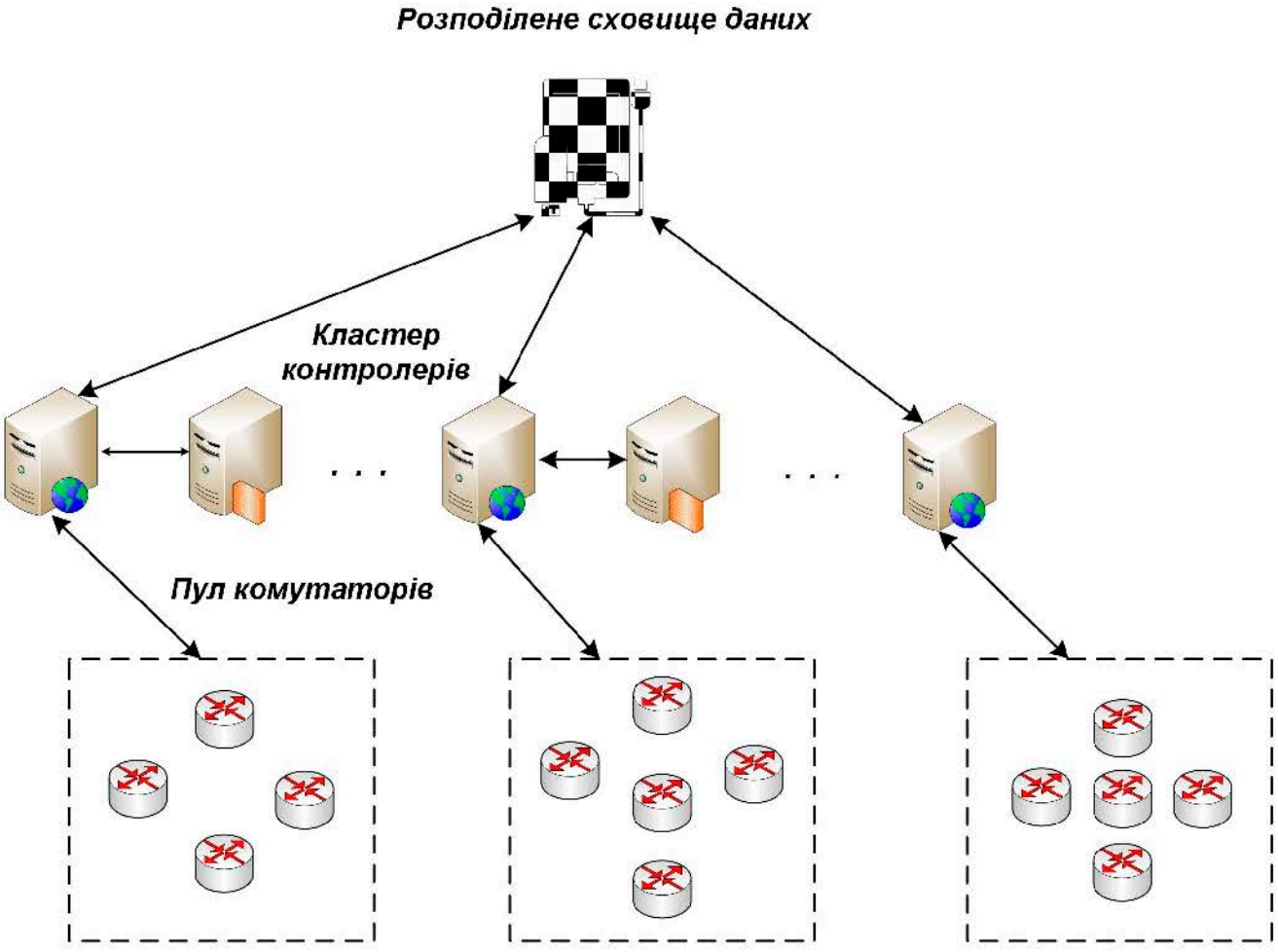


Рисунок 1.3. Структура мережі із розподіленими контролерами

Зокрема, основною задачею архітектури HyperFlow є спрощення масштабу мережі з забезпеченням логічної централізації управління мережею. Всі контролери реалізують однаковий глобальний вид мережі та обслуговують запити

локально без використання активних дзвінків. Таким чином, час налаштування потоку зводиться до мінімуму [3].

HyperFlow та подібні архітектури прозорі з точки зору стандарту OpenFlow, тобто не потребують змін. Архітектура HyperFlow складається з ефективних маршрутизаторів OpenFlow, контролерів, кожен з яких виконує екземпляр HyperFlow, та систем розподілу подій для зв'язку між контролерами. Усі контролери сприймають однаковий глобальний вид всієї мережі та діють так, ніби вони керують усією мережею. Усі вони мають однакову програму. Кожен маршрутизатор призначається певному контролеру. Якщо контролер виходить з ладу, маршрутизатори пошкодженого контролеру повинні бути відновлені і має бути призначений активний контролер поряд. Кожен контролер безпосередньо керує призначеними маршрутизаторами і впливає на роботу інших, спілкуючись з іншими контролерами. Організація структури HyperFlow, де зображена система взаємодії між контролерами, наведена на рис.1.4.

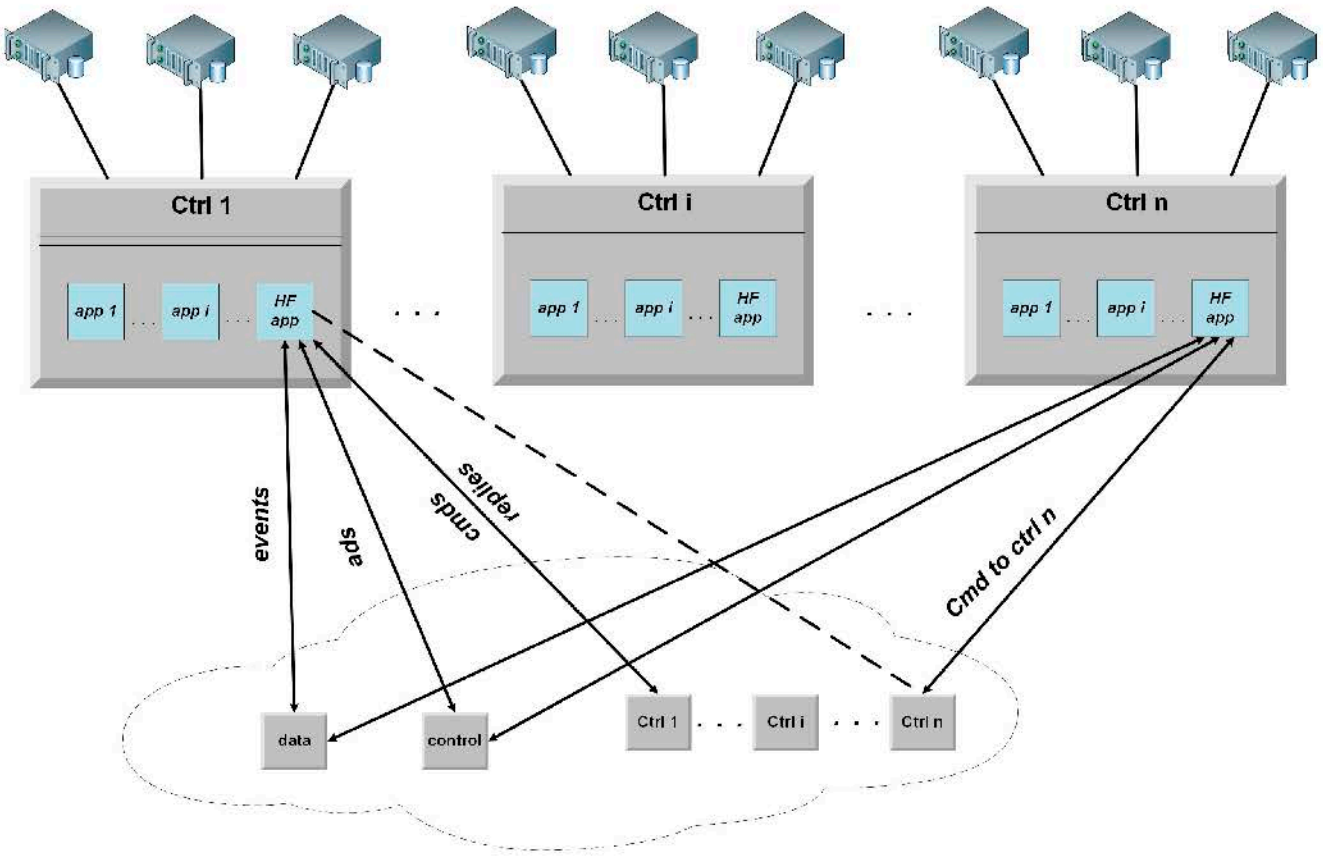


Рисунок 1.4. Архітектура технології HyperFlow

Система HyperFlow посилає події, що змінюють стан мережі на систему зв'язку “публікація-підписка” на кожному контролері для досягнення цілісності представлення мережі серед усіх контролерів. Інші контролери відтворюють трансляцію подій на мережевих моделях для відновлення поточного стану мережі.

Видалення з маршрутизаторів функції управління програмно-визначуваними мережами дозволяє цим пристроям перенаправляти всі ресурси для прискорення трафіку, що значно підвищує продуктивність. Віртуалізація управління мережею зменшує витрати на їх створення та утримання.

Програмно-визначувані мережі дозволяють адміністраторам додавати нові функції до існуючої мережевої архітектури. Однак такі програмні рішення є універсальними, оскільки вони не мають залежності від прошивки маршрутизатора кожного постачальника.

1.2 Методи маршрутизації у програмно-визначуваних мережах

Питання маршрутизації та розташування контролерів виникає у програмно-визначуваних мережах через відокремлення рівня управління та рівня даних. Мережа має бути розділеною і один контролер має бути розміщеним у кожній підмережі. Ключовим фактором у цьому випадку є показник затримки мережі.

1.2.1 Задача оптимізації розташування об'єктів

Задачі з оптимізації існують у багатьох контекстах, у тому числі за межами світу комп’ютерних мереж. Проблема Вебера – це проблема розташування об'єкта для мінімізації вимірюваної суми відстаней до певної сукупності точки. Більш складні задачі цієї дисципліни виникають, коли існують обмеження щодо розташування об’єктів та використання більш складних критеріїв оптимізації.

Задача розташування об'єктів у базовому формулюванні складається з потенційних точок розташування, що складаються з L точок, які можуть відкривати об'єкти, і D точок, які необхідно розмістити. Мета полягає у виборі меншої частини F точок розташування об'єктів для мінімізації величини відстаней від кожної точки обслуговування до найближчого місця об'єкта, а також суми

витрат на розташування об'єкта [4].

Оптимальне рішення задачі розташування таблиць у загальних графах характеризується як NP-повне і може бути отримане шляхом його зменшення у випадку кратності. Було розроблено декілька алгоритмів для задач розташування об'єктів у великій кількості варіацій. Без припущень щодо властивостей відстаней між клієнтами та місцями розташування (зокрема, не припускаючи, що відстань задовольняє нерівності трикутника), проблема відома як неметрична задача розташування об'єкта і може бути апроксимована фактором *O (log n)*. Якщо припустити, що відстані між клієнтами та місцями розташування не орієнтовані і трикутник нерівний, то мова йде про проблему метричного розташування об'єктів.

Задача щодо мінімального розташування об'єктів пов'язана з пошуком мінімальної відстані від точки до найближчого місця розташування. Якщо задано безліч точок $P \in Rd$, то потрібно знайти безліч точок $S \in Rd$, $| S | = K$ таку, що значення $max\ p \in P\ (min\ q \in S\ (d\ (p, q)))$ буде мінімальним. Усі такі задачі є NP-повними і всі вони мають вагомі типи, які вирішують задачі з урахуванням різного значення вузлів.

1.2.2 Задача компонування контролерів та маршрутизації у мережі

При компонуванні контролера є кілька взаємопов'язаних задач визначення оптимальної кількості контролерів у мережі, розділення мережі на кластери, вибору алгоритму маршрутизації, вибору місця розташування контролера у кожній підмережі за результатом кластеризації. Приклад кластеризації мережі та розташування контролерів у ній показаний на рис. 1.5 (пунктирна лінія вказує розділення мережі на підмережі). Рішення проблеми компонування контролерів впливає на всі аспекти рівня управління у програмно-конфігурованій мережі: від можливостей поширення стану до допущення помилок та продуктивності. Рішення визначає доступність та терміни обрання вузлів мережі у WAN з великою затримкою. Це має практичне значення для розробки програмного забезпечення, оскільки впливає на те, чи можуть контролери реагувати на події у режимі реального часу та чи передають вони інструкції маршрутизаторам заздалегідь. У

більшості документів ця проблема визначається як проблема багатоцільової комбінаторної оптимізації і є NP-повною, а для пошуку оптимального рішення використовуються евристичні алгоритми. Однак ці евристичні алгоритми мають недоліки: знаходять лише локальні оптимальні рішення та працюють повільно. З іншого боку, існують підходи до розташування контролерів у мережевих точках, які відповідають визначеним критеріям вибору.

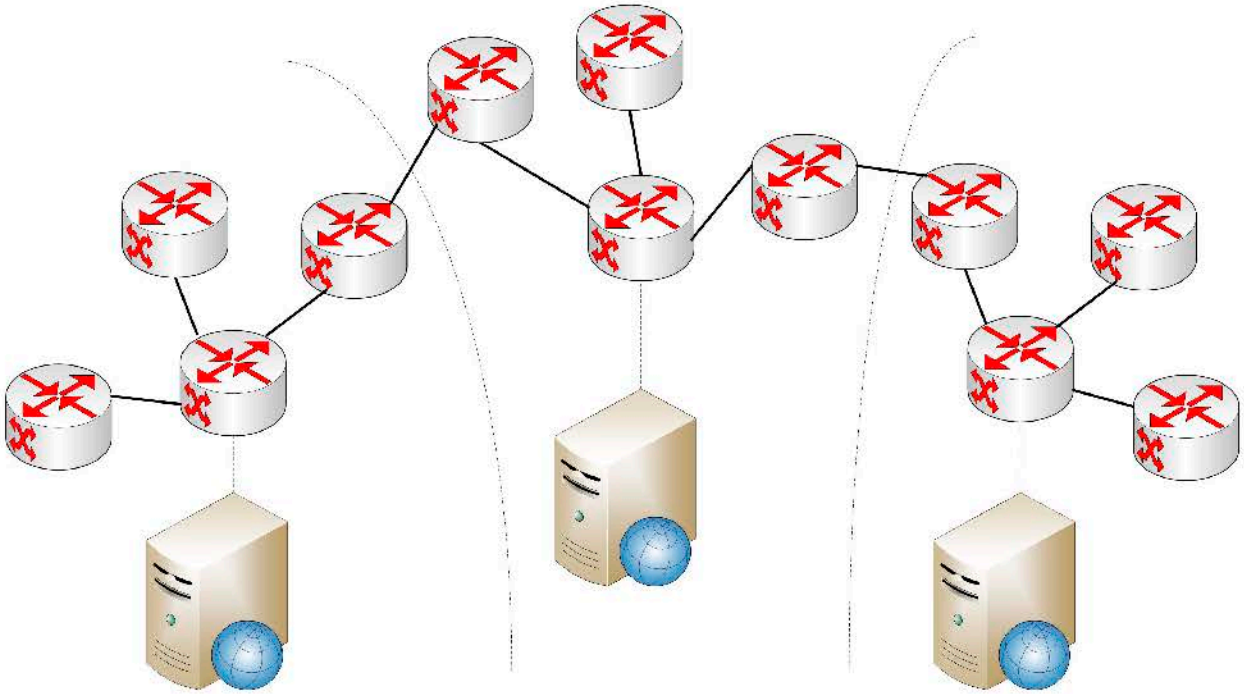


Рисунок 1.5. Приклад кластеризації мережі

Початкові рішення проблеми розгортання контролера полягали в тому, щоб спробувати зосередити увагу на затримці передачі сигналу в мережі незалежно від навантаження на контролери, але це ключове питання для реальних мереж. Тому було запропоновано модифіковану постановку задачі: проблема розташування контролера ємності. Хоча головна мета успішного розташування контролерів – мінімізувати затримки сигналу між вузлами мережі та контролерами, але враховуються не тільки затримки. Розташування контролерів також має відповідати певним вимогам щодо надійності та цілісності. Наприклад, розглянемо оптимальне регулювання $k = 5$ контролерів щодо максимальної затримки в графі мережі та проаналізуємо проблеми надійності, які можуть при цьому виникати.

1.2.3 Врахування відмов контролерів у мережі

Збільшення кількості контролерів, рівномірно розподілених по мережі, зменшує максимальну затримку між вузлами та контролерами в мережі. Це також збільшує відмовостійкість при закритті контролерів. У разі відмови контролера всі вузли, якими він керує, можуть бути делеговані іншим контролерам, тобто, у випадку кожного конкретного вузла – іншому, найближчому до цього вузла контролеру. Це робиться за допомогою таблиці призначення або звичайного алгоритму маршрутизації для знаходження найкоротшого маршруту.

Таким чином, якщо є хоча б один контролер – всі вузли можуть продовжувати працювати. Однак затримки з вузлів до нових контролерів можуть значно збільшитися порівняно з початковою ситуацією [5].

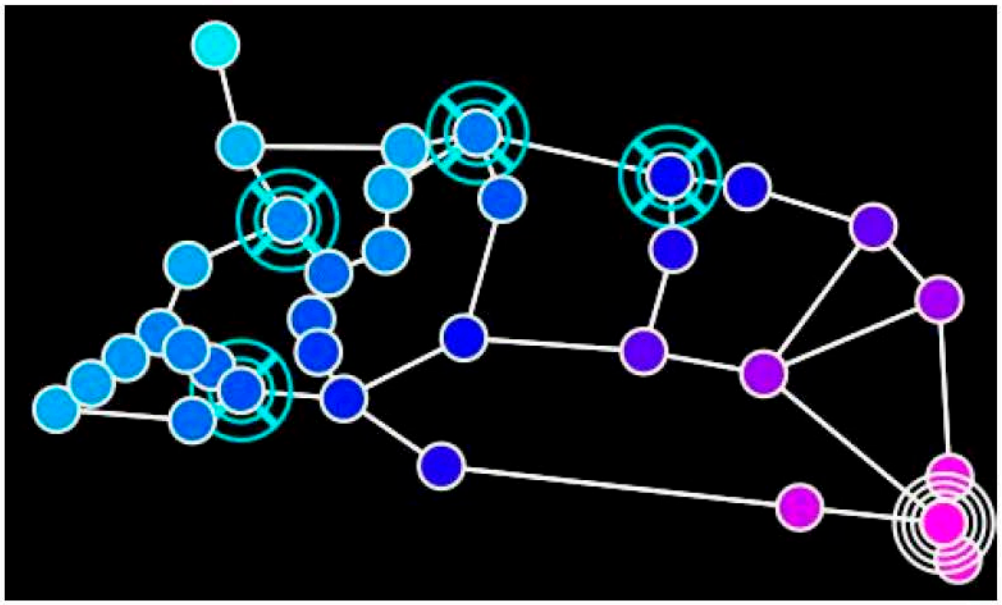


Рисунок 1.6. Вихід з ладу деяких мережевих контролерів

На рис. 1.6 показано збій чотирьох із п’яти контролерів у мережі (зупинені контролери позначені блакитними хрестиками, а один контролер, що працює – колом). Затримки від вузлів до останнього функціонуючого контролера забарвлені: рожевий колір означає затримку, що дорівнює нулю, синій колір означає затримку 50% від діаметра сітки, а блакитний – затримку 100% діаметра сітки. Найгірший сценарій цієї відмови контролерів полягає в тому, що контролер, який залишився, знаходиться найдалі від центру мережі. Це означає, що запити від деяких вузлів мають пройти майже через всю мережу, щоб дістатися до

контролера. Розташування контролерів має враховувати не тільки затримку, досягнуту без відмов мережі, але і найгіршу затримку відмов контролерів для підвищення стійкості мережі до цього явища.

1.2.4 Врахування пошкодження цілісності топології мережі

Видалення фізичних частин мережі, таких як вузли та з'єднання, на відміну від відмов контролера, має більший вплив на стабільність мережі, оскільки це змінює саму топологію. Найкоротші шляхи між деякими вузлами змінюються, викликаючи зміну затримок та, можливо, перепризначення деяких вузлів іншим контролерам. Крім того, серйозна проблема полягає в тому, що всі частини мережі мають ризик переривання (відключення від решти мережі) через збій вузла або комутатора. У найгіршому випадку деякі вузли більше не будуть підключені до жодного контролера, оскільки всі контролери не можуть отримати доступ до них. Такі вузли все ще активні і здатні передавати дані, але більше не можуть звертатися до контролера за інструкціями.

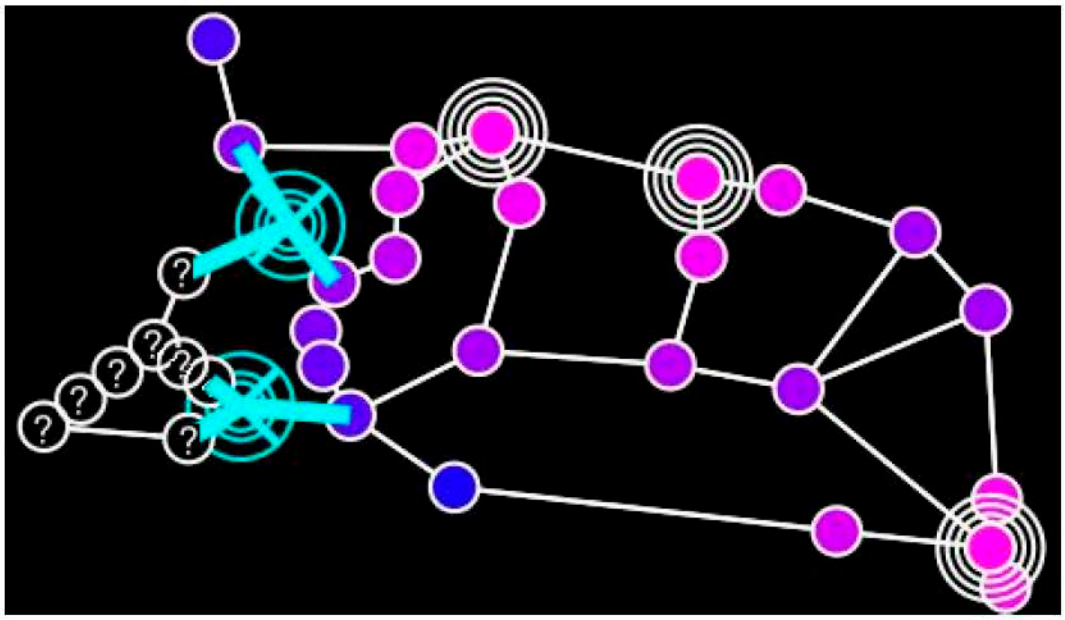


Рисунок 1.7. Вихід з ладу деяких вузлів та з'єднань у мережі

На рис. 1.7 показаний найгірший сценарій. Частини мережі, залишені без нагляду, позначаються знаками запитання. Для всієї підмережі, що складається з позбавлених від контролера вузлів, будь-яка функція, застосована у контролері, недоступна, навіть якщо самі вузли взаємопов'язані та функціональні.

1.2.5 Врахування перевантаження контролерів

У якості покажчика затримки можна використовувати найкоротший шлях між вузлом та контролером. Його аналіз дозволяє вибрати контролер, який керує вузлом. Розділ мережі на підмережі, кожна з яких управляється відповідним контролером, показаний на рис. 1.8. Кількість вузлів, присвоєних кожному контролеру, не є однаковою: від 4 до 10 вузлів на контролер.

Чим більше вузлів, контрольованих контролером, тим більше навантаження на контролер. Це особливо важливо під час частого доступу вузлами до часто використовуваних контролерів, особливо за допомогою віртуалізації мережевих функцій (NFV). Зі збільшенням кількості запитів, які приймає контролер від вузла, зростає й ризик додаткових затримок через збір запитів з боку контролера.

Щоб протистояти перевантаженню контролерів, призначення вузлів контролерам має бути збалансованим.

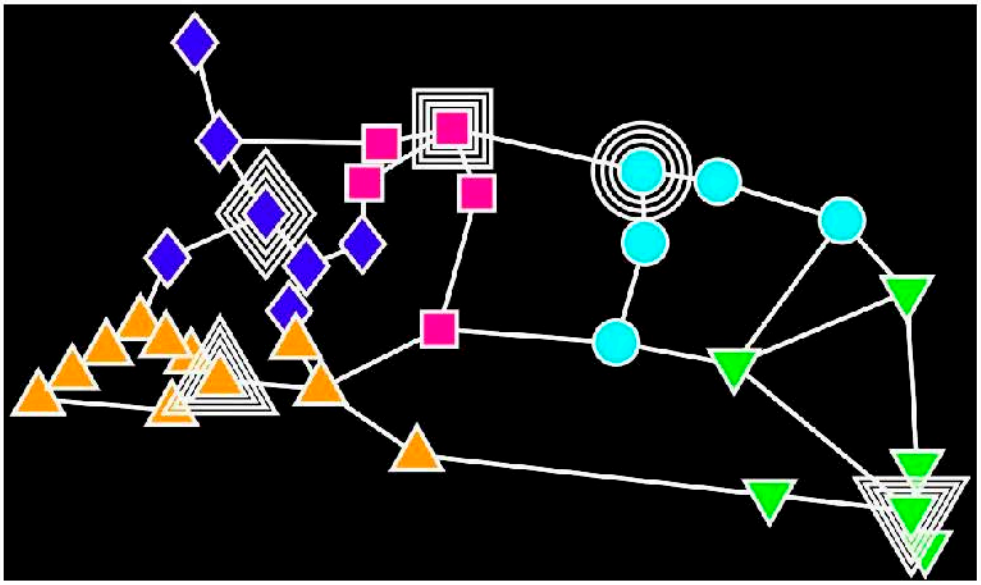


Рисунок 1.8. Кластеризація без урахування навантаження контролерів мережі

1.2.6 Врахування затримки передачі між мережевими контролерами

Ситуація, коли в мережі є контролер, не може відповідати вимогам надійності, але коли у мережі є декілька контролерів, виникає нова проблема.

Коли логіка управління мережею розподіляється між декількома контролерами, ці контролери повинні бути синхронізовані, щоб забезпечити узгодженість в усій мережі. Залежно від частоти синхронізації контролерів,

затримка передачі між контролерами може відігравати важливу роль. Відстані між мережевими контролерами, показаними на рис. 1.9, складають не менше 50% від діаметра мережі.

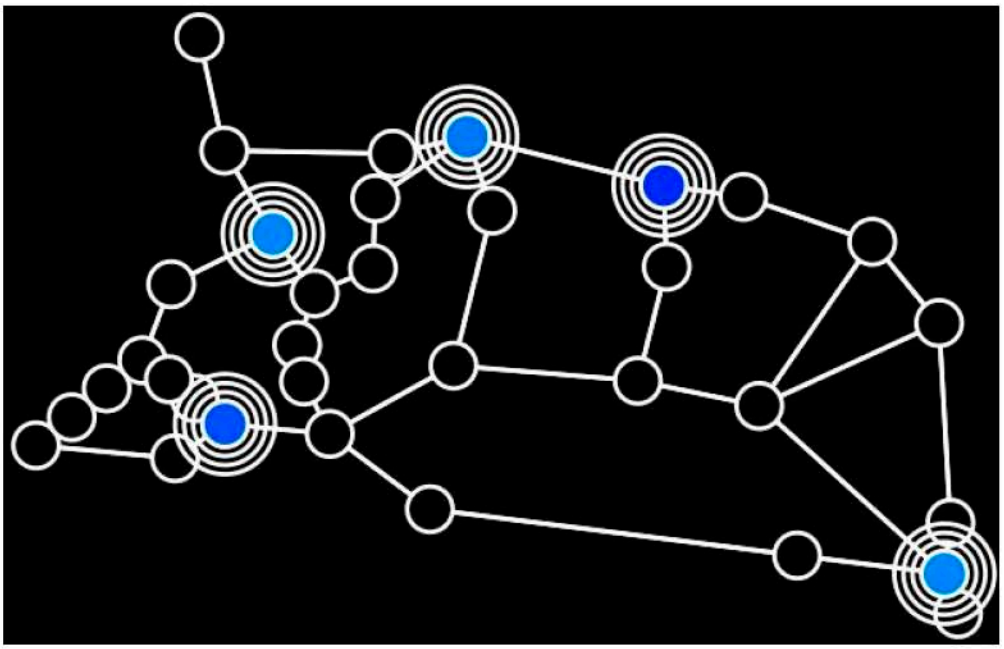


Рисунок 1.9. Затримка передачі між мережевими контролерами

1.3 Аналіз способів вирішення проблем маршрутизації

Таким чином, вирішення проблеми маршрутизації та розташування контролерів є важливим і актуальним, що враховує аспект балансу навантаження і демонструє комплексний підхід.

1.3.1 Метод k-середніх у кластерному рішенні

Метод k-середніх зосереджується на швидкості захоплення мережі. Поломка мережі та перевантаження контролерів не враховуються при вирішенні проблеми розгортання контролерів. Фактично, цей метод передбачає застосування k-середніх у кластерному рішенні із середньою затримкою (мінімально k-медіанна задача) або із затримкою у гіршому випадку (мінімум k-центрів). Результати розв'язання задачі розташування контролерів за цими двома підходами показані на рис. 1.10, де знаком "o" позначено розташування з метою зменшення середньої затримки, а знаком "x" – з метою зменшення максимальної затримки.

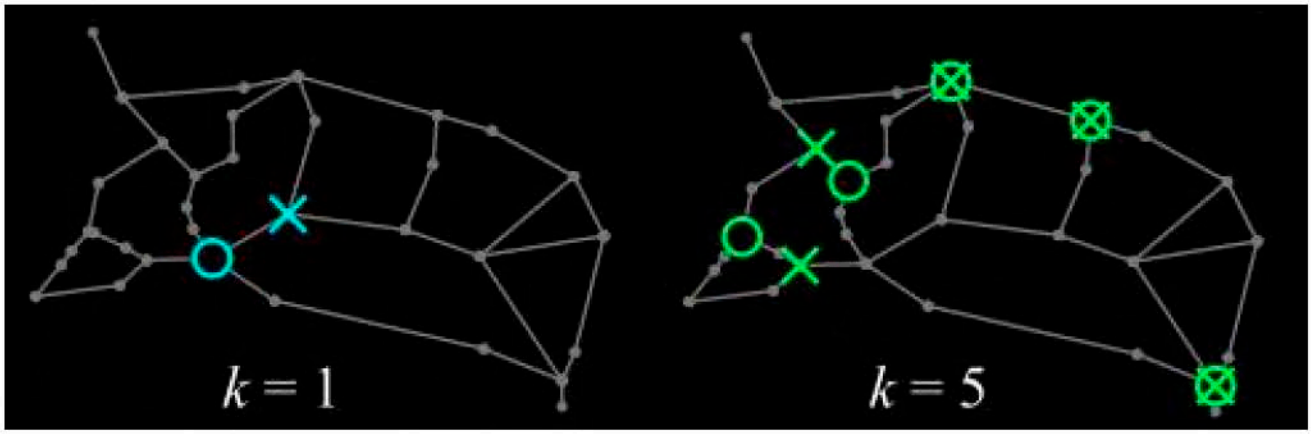


Рисунок 1.10. Розташування одного і п’яти контролерів
для оптимальної середньої та максимальної затримок

Також існує варіант, названий “проблемою максимального покриття”. При цьому максимальна кількість вузлів у межах заданої затримки є критерієм оптимального розташування контролерів.

1.3.2 Метод маршрутизації на основі щільності

Метод маршрутизації і розташування контролерів за щільністю (DBCP) використовує алгоритм кластеризації на основі щільності для поділу мережі на кілька підмереж. Вузли тісно пов'язані всередині підмережі і мають менш щільне підключення до вузлів поза підмережою, кожна підмережа має контролер [5].

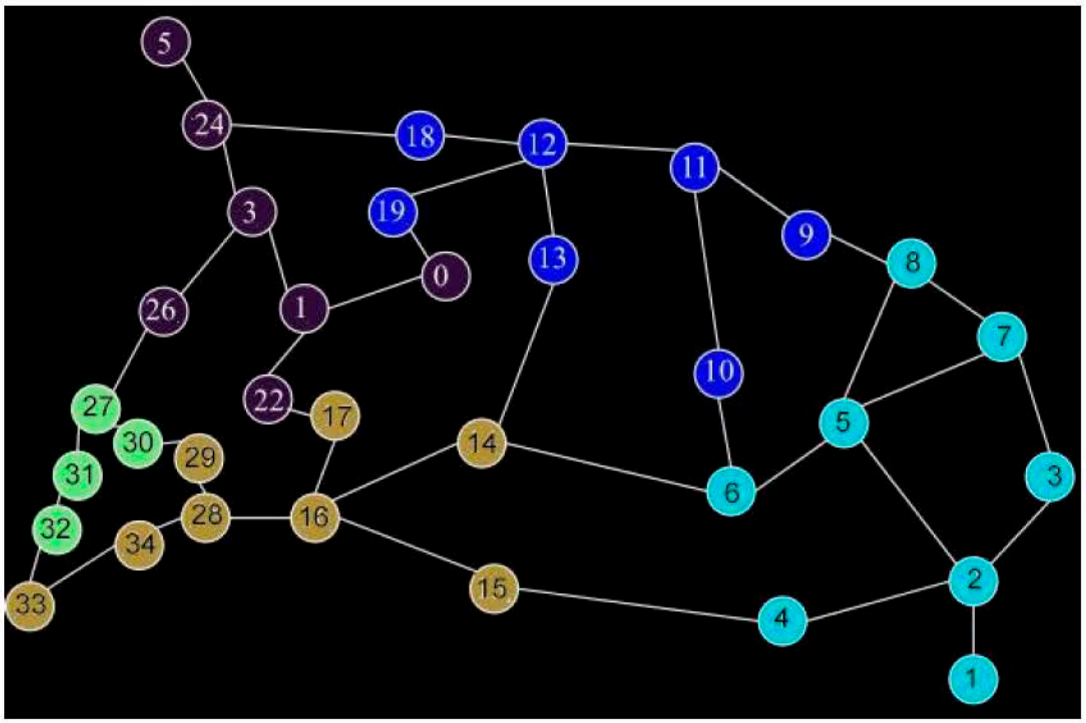


Рисунок 1.11. Кластеризація мережі за щільністю

Розмір кожної підмережі можна визначити за потужністю контролера. Крім того, оптимальна кількість контролерів визначається у процесі кластеризації на основі щільності.

У DBCP мережа поділена на підмережі з декількома згрупованими вузлами. Для кожного вузла s_i обчислюються дві властивості: локальна щільність ρ_i та відстань до вузлів δ_i більшої щільності. Певна щільність вузлів ρ_i залежить від кількості вузлів у d_c. Для масштабних мереж d_c, як правило, визначається як 30% діаметра графу. δ_i вимірюється шляхом обчислення мінімальної відстані між вузлом s_i та будь-яким вузлом з більш високою локальною щільністю. Для вузла з найбільшою щільністю $\delta_i = \max_j (d_{ij})$. Таким чином, в центрах множин занадто багато вузлів δ.

Після отримання центрів кластерів кожен з інших вузлів привласнюється тому ж кластеру, що і найближчий сусід, що має більшу щільність. На рис. 1.11 наведено приклад кластеризації мережі з використанням зазначеного вище алгоритму: мережа поділена на п'ять груп (позначено різними кольорами).

У кожній підмережі має бути встановлений контролер. Оптимальне місце розташування контролерів для кожної підмережі визначається відповідно до різних цільових функцій. Основними розглянутими параметрами є надійність та затримка сигналу.

При аналізі затримки від контролера до робочого вузла для графа підмережі обчислюється середня затримка передачі сигналу збитості $\pi^{avglatency}(S(\theta))$ між контролером та вузлами S(n), керованими контролером у положенні v. Також обчислюється $\pi^{maxlatency}(S(\theta))$ – максимальна затримка підмережі [6].

Для зменшення витрат на обмін інформацією між контролерами вони мають розміщуватися якомога ближче один до одного. Затримка сигналу між контролерами має бути мінімізована, але розташування кожного обирається незалежно. Тому заздалегідь розрахувати відстань між контролерами неможливо. Для мінімізації затримки від одного контролера до іншого необхідно розмістити його якомога ближче до інших підмереж. При цьому можливо розрахувати затримку від контролера до вузлів інших підмереж.

1.3.3 Метод маршрутизації за критерієм стійкості до відмов

Для оптимальної маршрутизації та розташування контролерів програмно-визначуваних мереж застосовують також метод стійкості до відмов POCO, який дозволяє врахувати критерії відмовостійкості мережі.

У багатьох топологіях, де одного контролера достатньо з точки зору пропускної здатності, вимоги стійкості вимагають їх більшої кількості. Цей алгоритм також враховує такі показники, як затримка передачі між контролерами та залишки навантаження різних контролерів.

У алгоритмі POCO основною метрикою э затримка π^{max}, тобто максимальна затримка від управління в мережі до вузла. Така затримка обчислюється як у разі, коли відмови контролера в мережі не відбувається, так і у разі відмови контролерів. Якщо контролер перестає працювати, вузли, якими він керує, делегуються іншим контролерам, тим самим збільшуючи затримку для вузла контролера. Для відмов від 0 до k-1 побудована серія C-сценаріїв для розташування k контролерів, включаючи всі можливі комбінації. Максимальна затримка в цьому наборі визначається затримкою π_C^{max} або затримкою π_0^{max} за відсутності відмов. Для контролерів має сенс бути максимально рівно розташованими, коли вони дотримуються принципу мінімізації затримки π_0^{max} у мережі. Якщо метою є мінімізація затримки π_C^{max}, то контролери розташовують у центрі мережі. Алгоритм POCO використовується для розташування контролерів, де ці два параметри, затримка π_0^{max} та затримка π_C^{max} – оптимальні за принципом Парето. Враховуючи масштаб, функції та доступні технічні засоби, алгоритм передбачає багато можливих рішень, найбільш підходящих для певної мережі.

1.4 Розробка та обґрунтування математичної моделі алгоритму маршрутизації

При наявності неперевіреного графу $G = (V, E, W, C, P)$, де V не є порожнім набором (відповідає набору вузлів обчислювальної мережі, у якій розташовані маршрутизатори); $E = \{(i, j) \in V \times V\}$ – набір ребер (відповідає набору з'єднань між вузлами обчислювальної мережі); $W: V \to R$ – вагова функція, яка присвоює

кожній вершині певну кількість ($w_i > 0$ – вага вершини $i \in V$, яка відповідає часу затримки маршрутизатора); $C: E \to R$ – функція, яка присвоює кожному ребру дійсне число ($c_{ij} > 0$ – вага ребра $(i,j) \in E$ відповідає проходу каналу); $P: E \to$ – функція, яка дає кожній стороні реальне число з діапазону ймовірності $0 < c_{ij} \leq 1$).

Поділ графа на частини (підграфи) означає, що початковий граф G подається у вигляді набору частин (блоків) $V_1, V_2, .., V_r (r > 1)$, таких, що $V_1 \cup V_2 \cup .. \cup V_r = V$; $V_k \cap V_l = \emptyset$; $k, l = 1..r$; $k \neq l$.

З'єднаний з $j \in V_l$ ребром, кінець $i \in V_k$, називається вершиною. Частина, яка з'єднує кінці різних блоків, називається краєм розділу. Набори E_{kl} називаються підрозділами. Відповідно граф з сукупністю підграфів V_k і набором ребер $E_k = \{(i,j) \in V \vee i \in V_k, j \in V_k\}$ підграфом G_k, тобто G_k=(V_k, E_k). Якість поділу графу можна оцінити за одним або декількома критеріями, які є параметрами підрозділів $E_{kl} (\sum_{k<1} f(E_{kl}) \to \min(max))$ вершин $V_1, V_2, .., V_r (\Sigma_{k=1}^{r} g(V_k) \to \min(max))$ та підрозділів $G_1, G_2, .., G_r (\Sigma_{k=1}^{r} h(G_k) \to min(max))$.

Наведене рішення також може застосовувати деякі обмеження (граничні умови), які необхідно враховувати при пошуку розділу. До них відносяться обмеження щодо наступних значень (верхнє та/або нижнє): кількість підграфів; кількість кінців, що входять до підграфів; обмеження балансу (тому всі підмножини $V_1, V_2, .., V_r$ повинні бути приблизно однакового розміру); максимальний дисбаланс між кінцями частин (підграфів); загальна вага кінців підграфів; ймовірність виходу з ладу всіх країв зрізів; середня кількість ребер, що входять до перетинів; середня вага ребер, що входять до перетинів; характеристики граничних кінців підграфів [7].

При поділі мережі на зони маршрутизації за необхідності мінімізації загальної пропускної здатності зони з'єднання, коли розмір домену повинен відповідати заданим вимогам балансу, модель буде виглядати так:

$$\sum_{kl} f(E_{kl}) = \sum_{kl} \sum_{(i,j) \in E_{kl}} c_{ij} \to min, \qquad (1.1)$$

$$|V_k| \geq (1 - \varepsilon_V)\lfloor \frac{|V|}{r} \rfloor, k = 1..r. \quad (1.2)$$

де параметр дисбалансу $0 < \varepsilon_v < 1$, який обмежує відхилення порядку графа (домену) менше середнього рівня.

Якщо мережа розділена на зони, ймовірність того, що перебоїв не буде, є мінімальною, а діаметр підграфів не повинен перевищувати задане значення Δ_c. Тоді модель буде виглядати так:

$$\sum_{kl/|E_{kl}| \neq 0} f(E_{kl}) = \sum_{kl/|E_{kl}| \neq 0} \prod_{(i,j) \in E_{kl}} (1 - p_{ij}) \to min, \quad (1.3)$$

$$d(G_k) \leq \Delta_c, k = 1..r. \quad (1.4)$$

У відповідній моделі цільова функція полягає у зведенні загального часу затримки на граничних кінцях до мінімуму:

$$\sum_{kl/|E_{kl}| \neq 0} \sum_{i \in V_{kl}} \omega_i \to min \quad (1.5)$$

Оптимальний поділ мережі на зони маршрутизації дозволяє збалансувати навантаження між зонами та мінімізувати ймовірність виходу з ладу маршрутизаторів на кордонах розподілу зон мережі.

1.5 Вимоги до модифікованого алгоритму маршрутизації

Відповідно до технічного завдання, топологія мережі $G(S, L)$, буде складатися з багатьох маршрутизаторів S і багатонаправлених зв'язків між ними L. Якщо між вузлами мережі є прямий зв'язок – довжина з'єднання однакова, а якщо немає – дорівнює нулю.

Модифікований у даній роботі алгоритм, на відміну від інших методів, аналізує топологію мережі, що ділиться на підмережі. Загальними є вузли, що мають найпоширеніші з'єднання, тобто вузли, які мають більше з'єднань всередині підмережі, і такі, що мають менше з'єднань в інших підмережах. Тому ці підмережі мають бути розглядатися як окремі мережі. Тоді кожна підмережа буде

мати менший показник відмов і затримка підмережі також зменшиться через тонкозернисту кластеризацію [8].

Алгоритм розташування контролерів програмно-визначуваних мереж буде складатися з трьох основних кроків:

- проаналізувати мережеву топологію, щоб розподілити щільність з'єднання між маршрутизаторами;
- відповідно до виявлених значень відстані від вузла з більшою щільністю та поточного вузла розподілити мережеві маршрутизатори;
- вирішити задачу щодо вибору місця розташування контролера у кожній підмережі відповідно до визначених критеріїв.

Модифікований таким чином алгоритм маршрутизації та розташування контролерів програмно-визначуваних мереж буде мати наступні переваги:

1. Можливість вибрати оптимальну кількість контролерів для певної мережі;
2. Побудова мережі топологічних з'єднань на основі щільності знижує ймовірність розгортання контролерів у вузлах з високим ризиком виходу з ладу;
3. Проблема великої кількості контролерів у мережі зводиться до проблеми розташування в мережі контролера, що зменшує складність обчислення.

1.6 Розробка модифікованого алгоритму маршрутизації

Реальна топологія мережі складається з багатьох маршрутизаторів. У пропонованому алгоритмі поділ мережі відбувається за рахунок поділу множини. Для кожного маршрутизатора s_i обчислюються два значення: локальна щільність ρ_i і відстань до вузла з більш високим значенням локальної щільності δ_i (пріоритет вузла для завдання кластера). Ці значення залежать лише від зв'язків між маршрутизаторами. Локальна щільність ρ_i для кожного маршрутизатора обчислюється за формулою, яка використовує коефіцієнт зв'язування k_{cl} та відстань між маршрутизаторами d_{ij}.

$$\rho_i = \sum_j k_{cl}(d_{ij} - d_c) \tag{1.6}$$

де d_c – максимальне обмежене значення відстані до маршрутизатора, яке вузли мережі можуть "наблизити" до поточного маршрутизатора;

k_{cl} – коефіцієнт підключення, який дорівнює 1, якщо є позитивна різниця між опорною межею відстані та відстані між маршрутизаторами.

Для великих розподілених мереж (з кількістю вузлів більше п'ятдесяти) ефективно використовувати максимальне значення відстані маршрутизатора поблизу як 0,3 діаметра мережі. При цьому відстань до вузла з найвищим значенням локальної щільності δ_i обчислюється як мінімальна відстань до вузла з найвищим значенням локальної щільності:

$$\delta_i = \min_{j:\rho_j > \rho_i} (d_{ij}) \tag{1.7}$$

При аналізі вертикальних точок з найбільшим значенням локальної густини максимальним приймається значення відстані до вузла з найвищим значенням локальної щільності. Такі кінці обираються напочатку як центри заданих підмереж.

Алгоритм аналізу розподілу щільності зв'язків у мережевій топології у вигляді псевдокоду є таким:

```
int analyzeDensity(G = (S, L), dc) {
    int k = 0;
    for (s: S) {
        ro[s] = getNumberOfNodesWithinDistance(s, dc, G);
    }
    for (s: S) {
        delta[s] = minDistanceToHigherDensityNode(s, ro, G);
        delta_Avg = sum(delta[0], ..., delta[S]) / S;
        if delta[s] > delta_Avg {
            k++;
        }
    }
    return k;
}
```

Представлений вище алгоритм також обирає рекомендовану кількість вузлів, які слід вибрати як окремі задані центри. Це створює чергу

маршрутизаторів, з яких можна вибрати вузли. До переваг цього методу можна віднести можливість вибору будь-якої кількості маршрутизаторів підряд. Якщо кількість наборів, що підлягають вибору, менше рекомендованого значення для кількості наборів – для даного набору будуть обрані вузли з більшим шляхом до вузла з більш високою локальною щільністю [9]. Якщо кількість наборів, що вибираються, перевищує рекомендоване значення заданого числа – спочатку вибираються вузли з більшим пріоритетом, а потім – вузли з більш високою локальною щільністю між вершинами одного пріоритету. Потім кожна вершина підключається до того набору, у якому центральний вузол максимально наближений до даного. Цей алгоритм не враховує навантаження трафіку та кількість вузлів.

Алгоритм підключення вузлів до кластерів мережі у вигляді псевдокоду є таким:

```
void clustering(Graph G = (S, L), ro, delta, k) {
    clusterCenters = findClusterCenters(k);
    for (s: S) {
        if clusterCenters.contains(s) {
            cluster = new Cluster();
            cluster.add(s);
        }
        else {
            cluster = findNearestHigherDensityNode(s);
            cluster.add(s);
        }
    }
}
```

1.7 Вирішення проблеми маршрутизації у кластерах

При прийнятті рішення про розділення програмно-визначуваної мері на підмережі останнім кроком є вибір місця для контролера кластерів. Кожен кластер має лише один контролер. Завдяки підходу, описаному вище, проблема багатофакторного розташування декількох контролерів може бути замінена проблемою оптимального розташування контролерів у кожній підмережі [10].

Для пошуку контролера може використовуватися декілька заходів, включаючи багаторазові затримки трафіку, затримки трафіку між контролерами та створення помилки підмережі (коли одне або більше з'єднань не вдається). Розроблений алгоритм буде використовувати вимірювання затримки трафіку у межах заданої підмережі, особливо із середнім та найгіршим значенням. Середня затримка трафіку для контролера на вузлі підмережі для графу підмережі може бути обчислена за такою формулою:

$$\pi^{avglatency}(S(\theta)) = \min_{v \in S(\theta)} \frac{1}{|S(\theta)|} \sum_{s \in S(\theta)} d(v,s) \quad (1.8)$$

Обчислене за формулою 1.8 значення залежить від довжини з'єднання між контролером і поточним вузлом кластера та тривалості з'єднання між ними. Цільову функцію для найгіршого значення затримки трафіку для контролера, що він знаходиться у вузлі підмережі, можна обчислити за такою формулою:

$$\pi^{avglatency}(S(\theta)) = \min_{v \in S(\theta)} \frac{1}{|S(\theta)|} \sum_{s \in S(0)} d(v,s) \quad (1.9)$$

1.8 Визначення обчислювальної складності алгоритму маршрутизації

Для оцінки обчислювальної складності розробленого алгоритму необхідно виконати чотири дії:

- обчислення щільності для кожного вузла;
- знаходження найближчого маршрутизатора;
- кластеризація;
- знаходження розташування контролерів.

При цьому складність часу обчислення для кожного вузла залежить від значення d_c та загальної кількості маршрутизаторів n у мережі. Високе значення d_c призводить до високої середньої щільності ρ та великого часу пошуку, що можна оцінити як часову складність фази $O(\rho n)$. Часову складність пошуку маршрутизатора з найменшою відстанню з більшою щільністю можна приблизно оцінити як $O(\rho)$, адже більшість маршрутизаторів близькі до маршрутизаторів

вищої щільності і лише невелика їх частина є віддаленими [11].

При визначенні складності часу для фази розроблення кластерів слід враховувати, що алгоритм проходить через усі мережеві вузли і вузол призначається тому ж кластеру, якому належить сусідній маршрутизатор з найбільшою щільністю. Отже, часову складність заданої фази можна оцінити $O(n)$. Вибір найкращого місця для контролера вимагає від алгоритму пройти всі можливі позиції, які слід враховувати всередині набору. У гіршому випадку часова складність процесу оцінюється як $O(n^2)$. Загальну часову складність вдосконаленого алгоритму можна оцінити як $O(\rho n + n + n + n^2)$. Алгоритм визначення обчислювальної складності алгоритму розташування контролерів з урахуванням ліміту завантаження кластера у вигляді псевдокоду є таким:

```
void clusteringWithLoad(Graph G = (S, L), ro, delta, controllerCapacity)
{
    for (s: S) {
        delta[s] = getBorderline(s, G);
    }
    S.sortBy(delta);
    for (s: S) {
        n[s] = cluster(s);
    }
    for (s: S) {
        ul = getNeighbors(s, (ro > ro[s]));
        ul.descendingSortBy(ro);
        for (s1: ul) {
            si = union(n[s], n[s1]); //si – всі ребра, що належать до n[s]
або n[s1]
            if (controllerCapacity >= sum(l(si))) {
                n[s1].add(s);
                break;
            }
        }
    }
}
```

Після моделювання та програмної реалізації розглянутого алгоритму розташування контролерів програмно-визначуваних мереж необхідно провести експеримент для порівняння трудомісткості вдосконаленого алгоритму з іншими. Результати експерименту будуть наведені нижче.

1.9 Застосування модифікованого алгоритму маршрутизації у програмно-визначуваних мережах

У даному підрозділі наведено приклад застосування розробленого модифікованого алгоритму маршрутизації і розташування контролерів для графу з 34 вузлів (рис. 1.12).

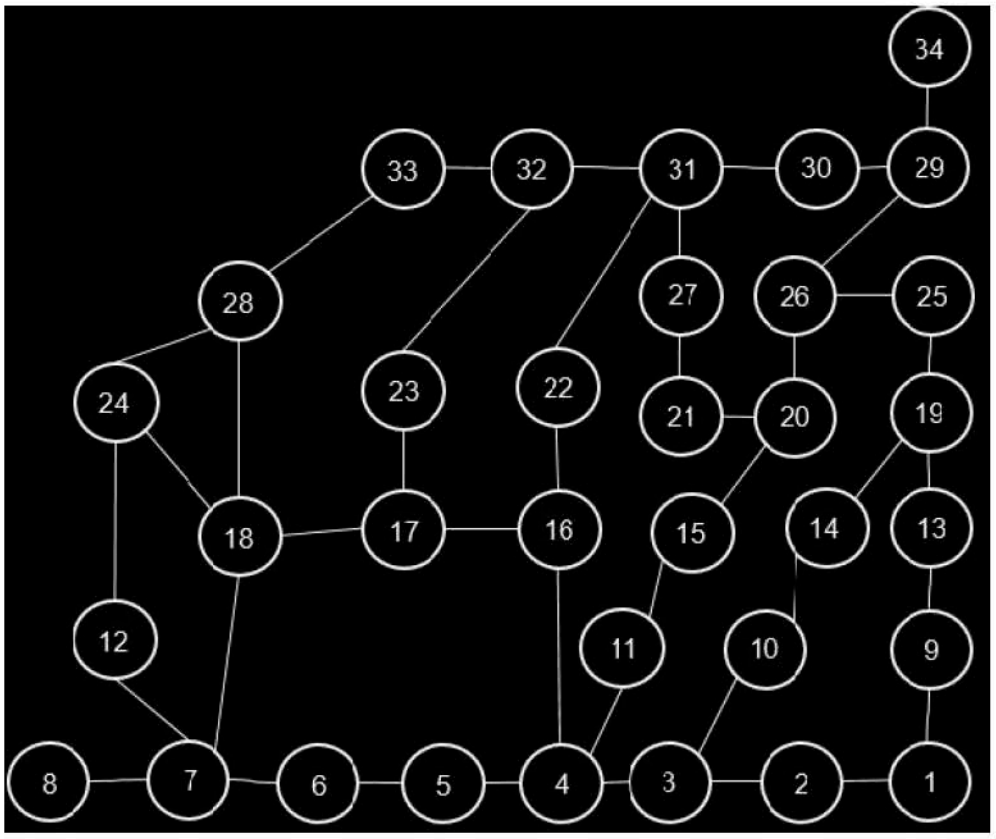


Рисунок 1.12. Приклад графу для моделювання кластеризації мережі

Наведений на рис. 1.12 граф складається з 34 вузлів і 41 ребра між ними (кожен вузол – це маршрутизатор, а кожне з'єднання являє собою фізичну лінію зв'язку). У цьому випадку припускається, що відстані між маршрутизаторами однакові і дорівнюють 1, тому їх можна не враховувати у прикладі. У результаті найменша відстань між маршрутизаторами може бути обчислена із загальної кількості з'єднань на шляху між маршрутизаторами. Можна враховувати дві одиничні відстані як максимальне значення відстані маршрутизатора d_c, що знаходиться поблизу, тобто приймаються маршрутизатори на відстані не більше двох. Для обчислення використовуємо формули 1.6 та 1.7.

Між мережевими маршрутизаторами аналіз розподілу щільності показує, що маршрутизатори 18, 31, 4 і 26 мають найбільше значення відстані для вузла з

більш високим значенням локальної щільності δ та відносно велике значення густини ρ. Середнє значення відстані δ до вузла з більш високим значенням локальної щільності становить 1,26. Значення відстані δ до вузла з більш високим значенням локальної щільності та більш високим значенням локальної щільності ρ для маршрутизатора 19 вище середнього, тому він може бути обраний [12].

Дану мережу, таким чином, можна розділити на 5 груп, що вимагає розгортання 5 контролерів. Деякі маршрутизатори, особливо маршрутизатори 32 і 26, мають схожі значення локальної щільності ρ, але різні значення відстані δ для вузла з більш високим значенням локальної щільності. Тому маршрутизатори, локальна щільність яких є меншою, відносяться до тієї ж групи, що і сусідній маршрутизатор з більш високим значенням локальної щільності ρ. Більша відстань δ від вузла з більш високою локальною щільністю вказує на те, що ці маршрутизатори можуть бути призначені концентратором для нового кластера. Після призначення вузлів 18, 31, 4, 26 і 19 центрами кластерів, інші маршрутизатори можуть бути підключені до того ж кластеру з сусіднім маршрутизатором з більш високим значенням локальної щільності ρ. На рис. 1.13 наведені результати роботи алгоритму поділу мережі на кластери.

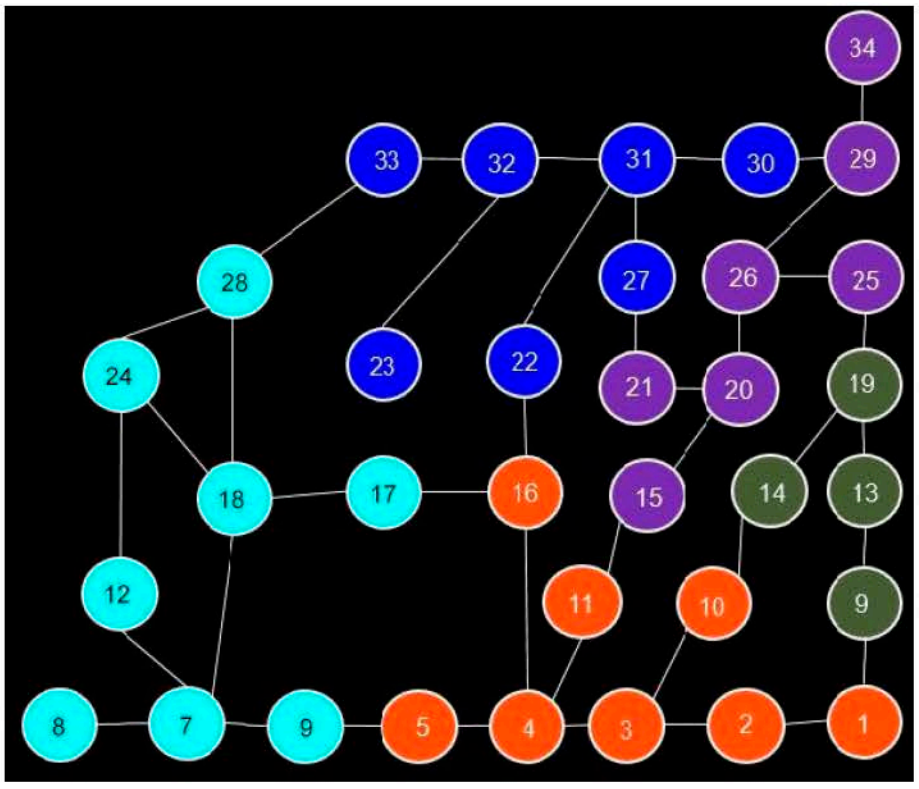


Рисунок 1.13. Результати роботи алгоритму поділу мережі на кластери

При поділі мережі отримано 5 кластерів (маршрутизатори показані відповідними кольорами). Результати моделювання з використанням розробленого алгоритму показані у табл. 1.1. Візуальне зображення локальної щільності та розподілу відстані вузла з більшою щільністю у вузлах зображено на рис. 1.14.

Таблиця 1.1. Результати моделювання з використанням розробленого алгоритму

id вузла	ρ	δ	id вузла	ρ	δ
1	4	1	18	9	3
2	8	1	19	6	2
3	8	1	20	7	1
4	10	3	21	5	1
5	6	1	22	7	1
6	6	1	23	6	1
7	8	1	24	6	1
8	4	1	25	6	1
9	4	1	26	8	3
10	5	1	27	6	1
11	6	1	28	7	1
12	6	1	29	6	1
13	5	1	30	7	1
14	5	1	31	9	3
15	5	1	32	8	1
16	9	1	33	6	1
17	8	1	34	3	1

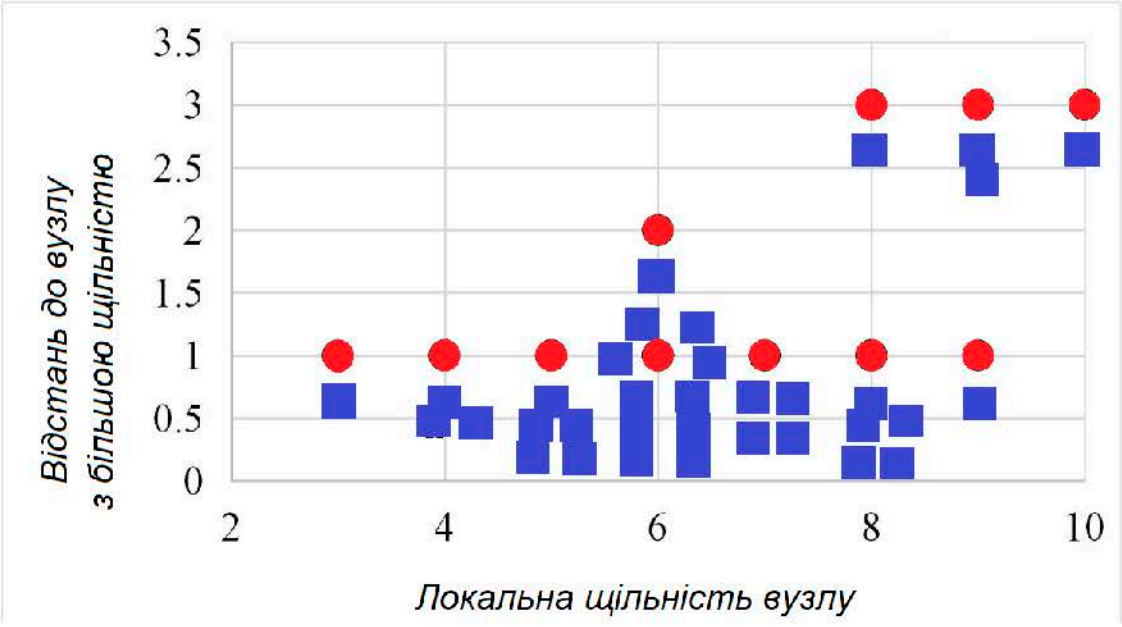


Рисунок 1.14. Розподіл щільності по відстані до вузлів із більшою щільністю

1.10 Вибір програмних засобів для розробки додатку

Для реалізації додатку, що моделює роботу мережі за наведеним вище алгоритмом, була обрана об'єктно-орієнтована мова програмування Java. Бібліотека JavaFX використовується для візуалізації графу мережі. На рис.1.15 наведена функціональна схема компіляції програми мова Java з використанням віртуальної машини.

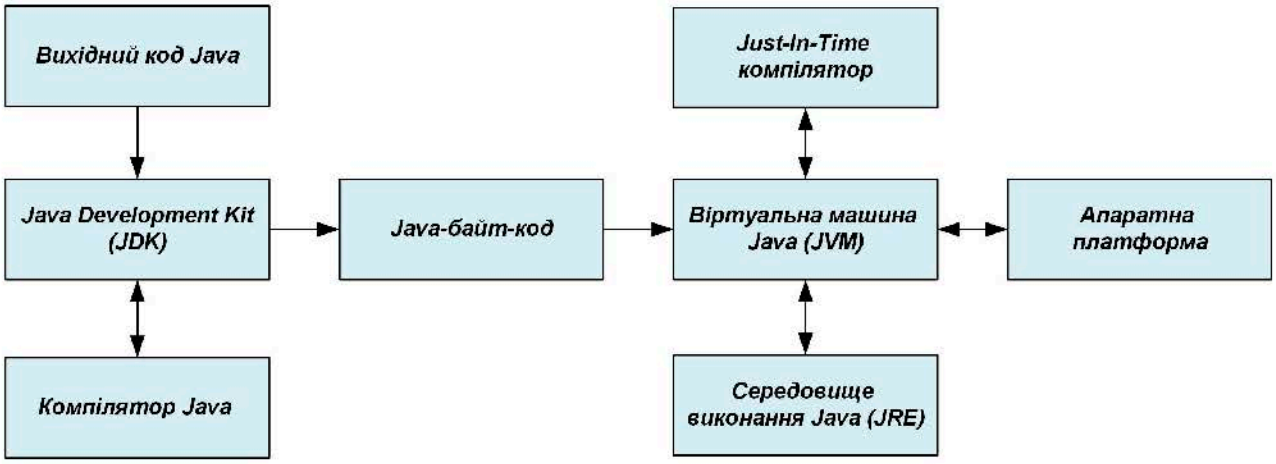


Рисунок 1.15. Схема компіляції програми на мові Java

1.11 Розробка БСА додатку та його ОО-моделі

Розроблюваний додаток має реалізовувати наступні функції:

- моделювати обраний алгоритм маршрутизації;
- створювати, зберігати, редагувати графи мережі;
- оцінювати та аналізувати результати моделювання;
- розширювати можливості програми шляхом додавання інших алгоритмів маршрутизації мережі.

У відповідності до наведених вище вимог програмну модель доцільно розбити на такі шари:

- моделювання алгоритму маршрутизації мережі;
- візуалізація моделі;
- модель зберігання графу системи.

Далі докладно розглянуто структуру та принципи роботи програми.

Класи Model, Node і Link відповідають за рівень моделі, клас View

відповідає за візуалізацію моделі, а клас Algorithm відповідає за моделювання алгоритму. Клас вузлів відповідає за один вузол мережевого графіка. Вузол містить поля id, координати x і y центру вузла зображення вузла при візуалізації, поточне значення кольору вузла та методи, які читають і змінюють значення цих полів. Клас з'єднання відповідає за окреме з'єднання між двома вузлами графу. Модель графу системи зберігається у класі Model. Модель, що представляє собою сукупність класів прикладів Node та Link, має поля Node та ModelLinks відповідно. Значення наступного ідентифікатора доданого вузла базується на збільшенні поля maxIndex, забезпечуючи таким чином ідентичність ідентифікатора. Клас моделі відповідає за додавання вузла, видалення вузла, переміщення вузла, додавання з'єднання між двома вузлами у графі, видалення з'єднання між двома вузлами у графі та зміну ваги з'єднання між двома вузлами в мережевій моделі графу. Цей клас містить методи changeLinkWeight, addNode, removeNode, moveNode, addLink, removeLink відповідно. Клас моделі включає також наступні методи:

- getNode – повертає вузол, спираючись на ідентифікатор;
- getModelLinks – повертає набір посилань моделі;
- getModelNodes – повертає набір модельних вузлів;
- getNodeSize – повертає розмір вузла у графі колекції посилань;
- findLink –повертає з'єднання між двома вузлами графу, якщо таке є;
- getLinks – повертає набір посилань, спираючись на ідентифікатор вузла, який може бути початком або кінцем цих з'єднань;
- generatorLinkBounds – повертає координати початкової та кінцевої точок контактів на екрані відображення графу.

У класі файлів передбачені методи для створення, збереження та відкриття файлів, що дозволяють створити новий файл, зберегти модельний файл та зчитати модель з файлу відповідно. Клас моделі є спадкоємцем класу файлів.

Для зберігання ідентифікації вузлів використовуються змінні startNodeId і endNodeId, оскільки потрібно по черзі натискати на початкові та кінцеві вузли, щоб додати та видалити з'єднання між двома вузлами у графі. Для виконання

операції необхідно натиснути відповідну кнопку ліворуч у вікні та на область налаштування графу один чи два рази (залежно від операції: додавання та видалення вузлу графа вимагає одного натискання на область, а переміщення вузла та робота із посиланнями – двох). Після натискання на поле редагування остаточні координати, які використовуються для надсилання даних до класу TaskModel, фіксуються у масиві clickData. Метод ClickHandler записує обведення у області виправлення, має кнопку запуску, що перемикає програму на певний режим введення та скидає послідовний ключ ClickHandler: 0 – режим додавання вузлу; 1 – режим видалення папки з вузлом; 2 – режим зміни ваги вузлу; 3 – режим переміщення вузлу; 4 – режим додавання посилання; 5 – режим видалення з'єднання; 6 – режим зміни ваги переходу. Операції, що змінюють вагу (додавання/видалення) вузла/посилання при введенні ваги, використовують значення, отримане у текстовому полі textArea.

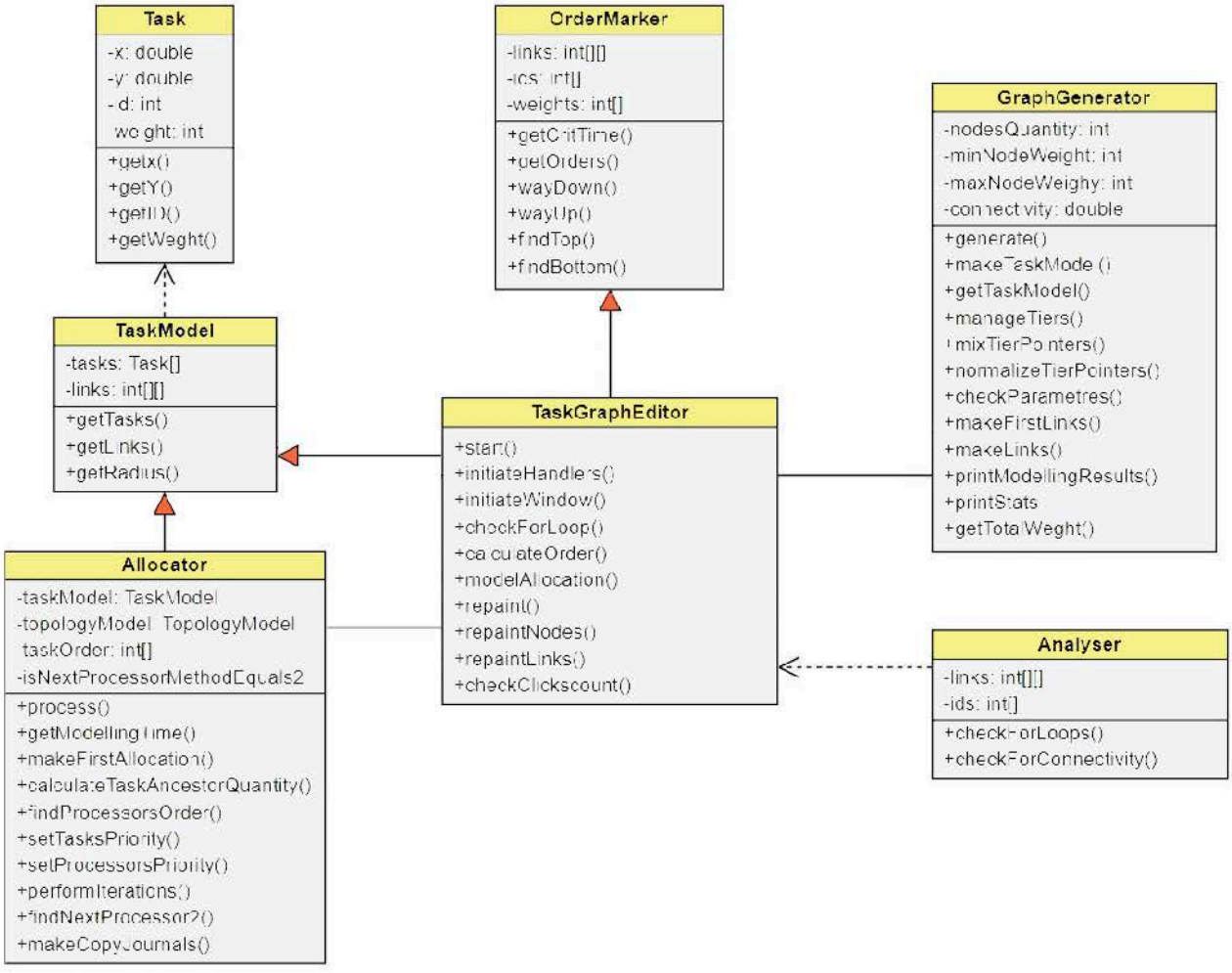


Рисунок 1.16. Діаграма класів додатку додатку для моделювання алгоритму оптимального вибору маршруту

Якщо операція входу була успішною, граф видаляється з моделі за допомогою методу перефарбовування (вузли repaintNodes та з'єднання repaintLinks), зчитуються колекції ModelNode та ModelLinks класу Model, а також встановлюються вузли та з'єднання у графі.

Додаток складається з візуальної частини та фактичної моделі графу. Вся логіка, пов'язана з графом, відбувається у моделі, а граф відображається за допомогою методу перемальовування класу View.

GraphGenerator представляє собою клас, що складає граф. Параметри, що відповідають за кількість вузлів, мінімальну та максимальну вагу торців та з'єднання системи, показані у конструкторі класу. Параметр MaxClusterNode, відповідальний за найвищу кількість точок у наборі, встановлюється окремо.

Кількість груп графу формується визначеними кінцевими точками, які приблизно вдвічі перевищують кількість вузлів. Початковий поділ вузлів здійснюється за допомогою рівномірного розподілу. "Змішування" наступних наборів – перенесення певного кінця з однієї групи у іншу (метод mixClusterPointers). Отримані вказівники зберігаються у серії записів на етапі кластерних покажчиків.

CheckParametres – метод, що генерує ваги, знаходить загальну вагу торців та з'єднань, які потрібно будувати (поля “Загальна вага” та “загальне посилання”). Метод makeClusterLinks визначає з'єднання між кінцями, що належать різним групам, з урахуванням максимальної межі з'єднання. Для кожного вузла виконується випадкове моделювання з'єднань, кількість яких не перевищує 75% від максимально можливих з'єднань вузлу.

MakeInterClusterLinks – метод, що визначає з'єднання між кінцями, що належать різним групам, з урахуванням максимальної межі з'єднання.

GenerateTrafficQuantity – метод, що генерує випадкові, рівномірно розподілені значення трафіку для з'єднань за умови, що мінімальні та максимальні значення minTraffic та maxTraffic відповідно обмежені.

DensityAlgorithm – клас, що відповідає за моделювання самого алгоритму. При цьому вводяться параметри топології графу, параметри навантаження

контролера (з урахуванням цього обмеження у алгоритмі) maxControllerLoad та значення кількості кластерних груп, на які мережа має бути поділена.

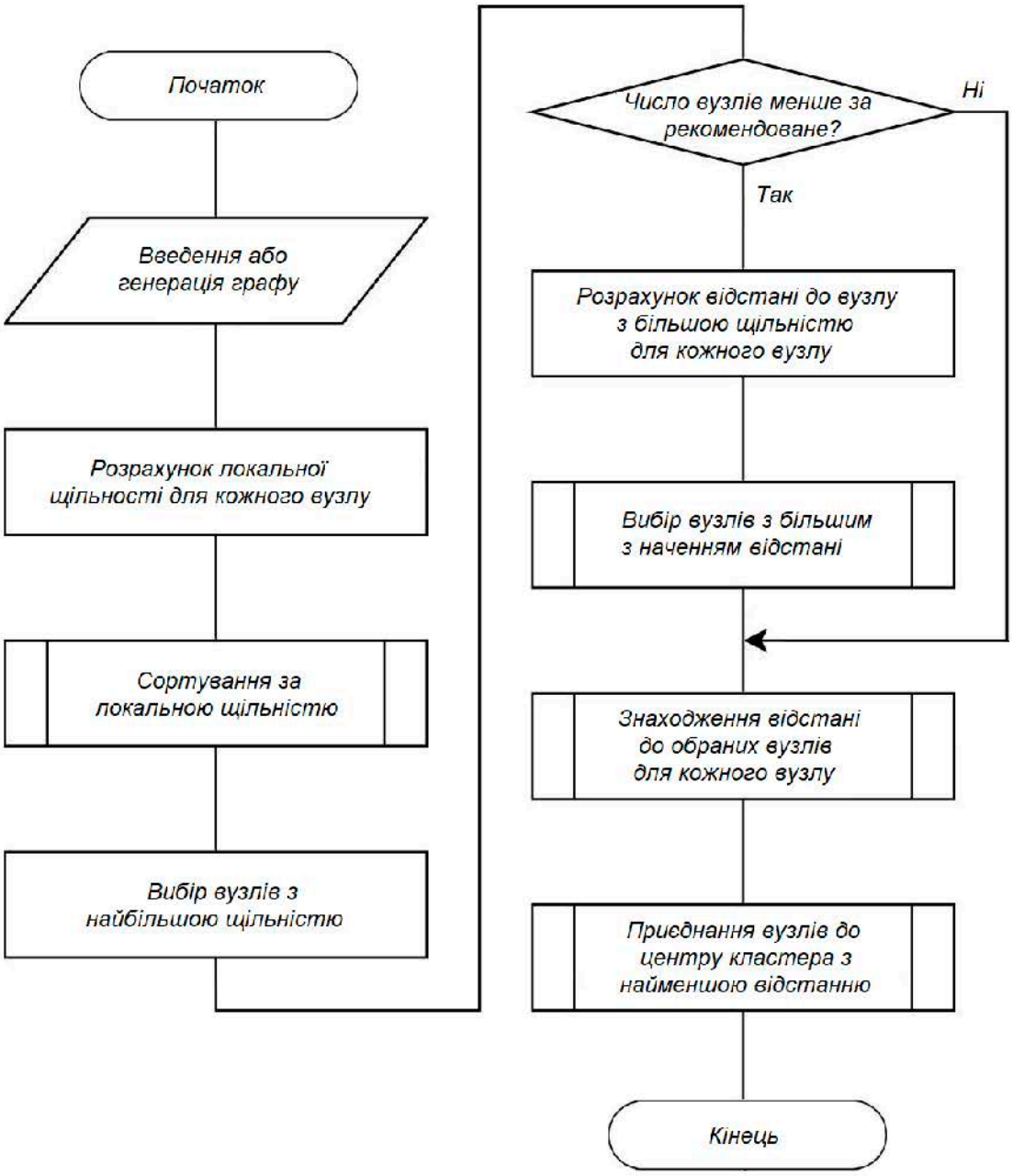


Рисунок 1.17. БСА додатку для моделювання алгоритму оптимального вибору маршруту

1.12 Розробка та опис інтерфейсу додатку для моделювання алгоритму оптимального вибору маршруту

Розроблене програмне забезпечення дозволяє моделювати алгоритм оптимального вибору маршруту та розташування контролерів. Створений додаток дозволяє також редагувати граф мережі. При створенні інтерфейсу додатку була врахована наступна функціональність:

– робота з файлами графів (завантажувати та зберігати файл);
– можливість роботи з вузлами (додавання, виймання та перенесення ваги вузла);
– вміння працювати з посиланнями (додавати вагу, видаляти та змінювати).

Далі описано особливості інтерфейсу розробленого під ОС Windows 10 додатку. Меню розробленої програми розташоване у верхній частині головного вікна програми і складається з трьох елементів: “Файл”, “Редагувати” та “Моделювати”.

Меню “Файл” використовується для загального управління програмою і містить такі елементи: “Створити новий”, “Зберегти”, “Відкрити”, “Згенерувати граф” та “Вийти”. Для зручності доступні комбінації клавіш: Ctrl + N, Ctrl + S, Ctrl + O, Ctrl + G, Ctrl + E відповідно. Елементи “Створити новий”, “Зберегти”, “Відкрити” та “Вийти” використовуються для створення нового графу, збереження поточного стану графу у файл, зчитування графа із збереженого документа та виходу з програми. Елемент “Згенерувати граф” використовується для створення графа, який відповідає обраним параметрам у новому вікні. Кнопки “Додати вузол”, “Видалити вузол”, “Пересунути вузол”, “Додати зв’язок”, “Видалити зв’язок” та “Змінити вагу зв’язку” розташовані у лівій частині вікна програми.

Меню “Редагувати” використовується при створенні та редагуванні мережевого графу і включає такі елементи: “Додати вузол”, “Видалити вузол”, “Пересунути вузол”, “Додати зв’язок”, “Видалити зв’язок” та “Змінити вагу зв’язку”. Ці елементи для зручності мають наступні гарячі клавіші: “1”, “2”, “3”, “4”, “5” і “6”.

Редагування графу мережі відбувається наступним чином. Для розміщення нового вузлу треба натиснути на пункт “Додати вузол” у меню “Редагувати” або на відповідну кнопку з лівого боку вікна програми, а потім торкнутися області мережевого графу. Після цього середньоцентрований вузол з’явиться у регіоні. Ідентифікатор вузлу автоматично генерується з відповідним кроком.

Для видалення вузлу необхідно натиснути на пункт “Видалити вузол” у меню “Редагувати” або на кнопці зліва вікна програми торкнутися папки у області графу мережі. При з’єднанні вузлу всі з'єднання з цим вузлом автоматично зникають.

Для переміщення вузлу треба натиснути на пункт “Пересунути вузол” у меню “Редагувати” або на кнопці зліва у вікні програми, а потім торкнутися вузлу графу та перенести його на нове місце.

Для додавання нового з'єднання між двома вузлами у графі треба натиснути на пункт “Додати зв’язок” у меню “Редагувати” та торкнутися двох вузлів у графі або відповідної кнопки. Якщо між вузлами немає зв’язку, він буде доданий до моделі та відображений у області графу.

Для видалення нового з'єднання між двома вузлами у графі треба натиснути на пункт “Видалити зв’язок” у меню “Редагувати” або на відповідну кнопку зліва у вікні програми, а потім по черзі торкнутися двох вузлів графу. Якщо у цей момент між вузлами був зв’язок, він буде видалений з моделі і зникне з графової області.

Для регулювання потоку трафіку між двома вузлами графу треба натиснути на пункт меню “Змінити вагу зв’язку” або на відповідну кнопку з лівого боку вікна програми, послідовно торкнутися двох вузлів графу та записати нову вагу зв'язку, після чого на графі буде показане нове значення для трафіку між вузлами.

Елемент, вибраний у меню “Редагувати” або за допомогою відповідної кнопки з лівого боку вікна програми активний, поки не буде обрано новий елемент або не виконано перемикання на інше меню. Редагуючи граф таким чином можна додати більше одного вузлу, посилання чи перемістити більше ніж один вузол, не змінюючи режим.

Меню “Моделювати” використовується для управління імітацією і містить такі елементи: “Перевірити граф”, “Перейти до моделювання без обмеження”, “Переключитися на обмежене моделювання”, “Визначити алгоритм моделювання”, “Показати області кластерів”, “Показати параметри кластера”, “Розташування контролерів у наборах”, для зручності є такі гарячі клавіші: Shift

+ C, Shift + L, Shift + E, Shift + A, Shift + R, Shift + P, Shift + K. Використовується загальна клавіша для цього меню: Shift.

Елемент меню “Перевірити граф” використовується для переходу у режим моделювання. Програма перевіряє наявність вершин без зв’язків та інших помилок налаштування графу. Елементи “Перейти до моделювання без обмежень”, “Перейти до моделювання з обмеженням” відповідають за вибір режиму моделювання, тобто можливість врахувати розподіл навантаження у розробленому алгоритмі. Після натискання на пункт “Задати алгоритм моделювання” у меню можна вибрати алгоритм моделювання, зокрема той, з яким треба порівняти створений алгоритм. Елементи “Вивести регіони кластеризації”, “Вивести параметри кластеризації”, “Розмістити контролери у кластерах” відповідають за відображення результатів моделювання на екрані.

На рис. 1.18 показані результати створення графу мережі, на рис. 1.19 – результати кластеризації, на рис. 1.20 – результати розбиття вершин графу на кластери у вигляді тексту, на рис. 1.21 – результати обчислення параметрів моделювання, на рис. 1.22 – результати розташування контролерів у імітованій мережі.

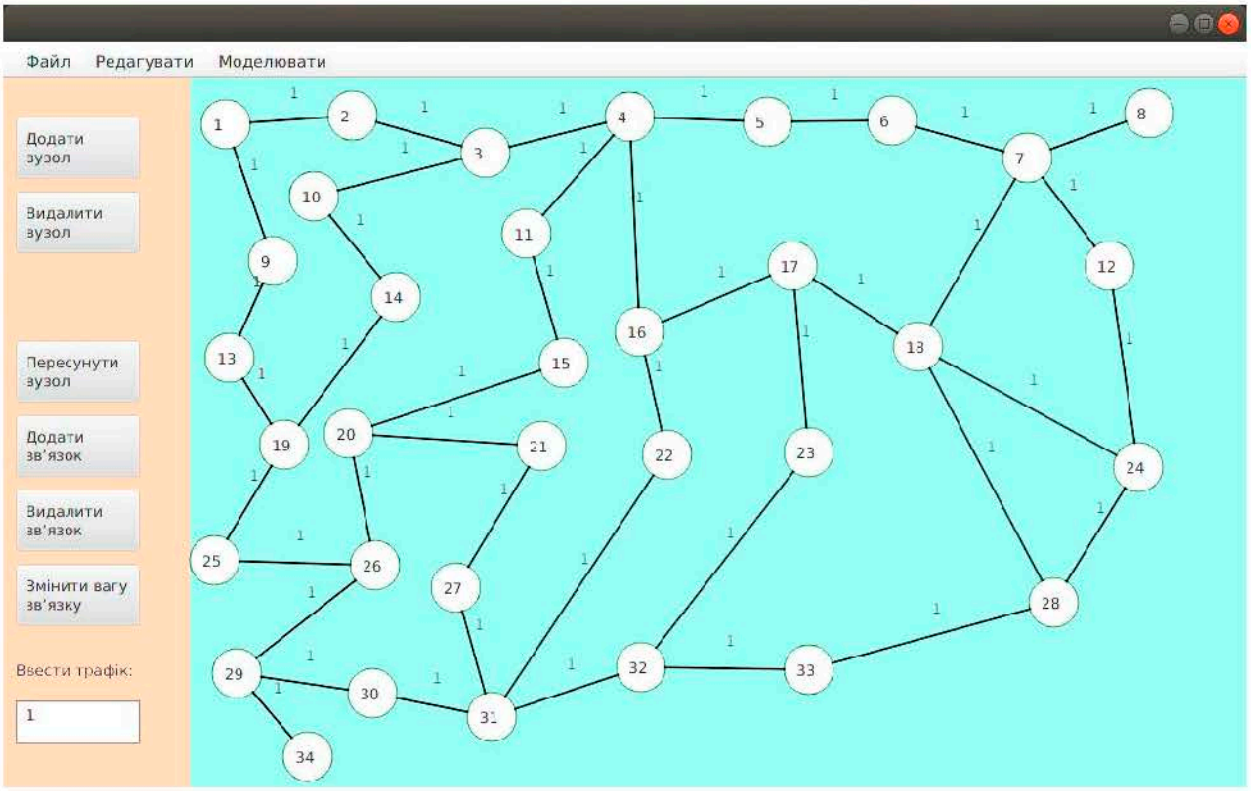


Рисунок 1.18. Результати створення графу мережі

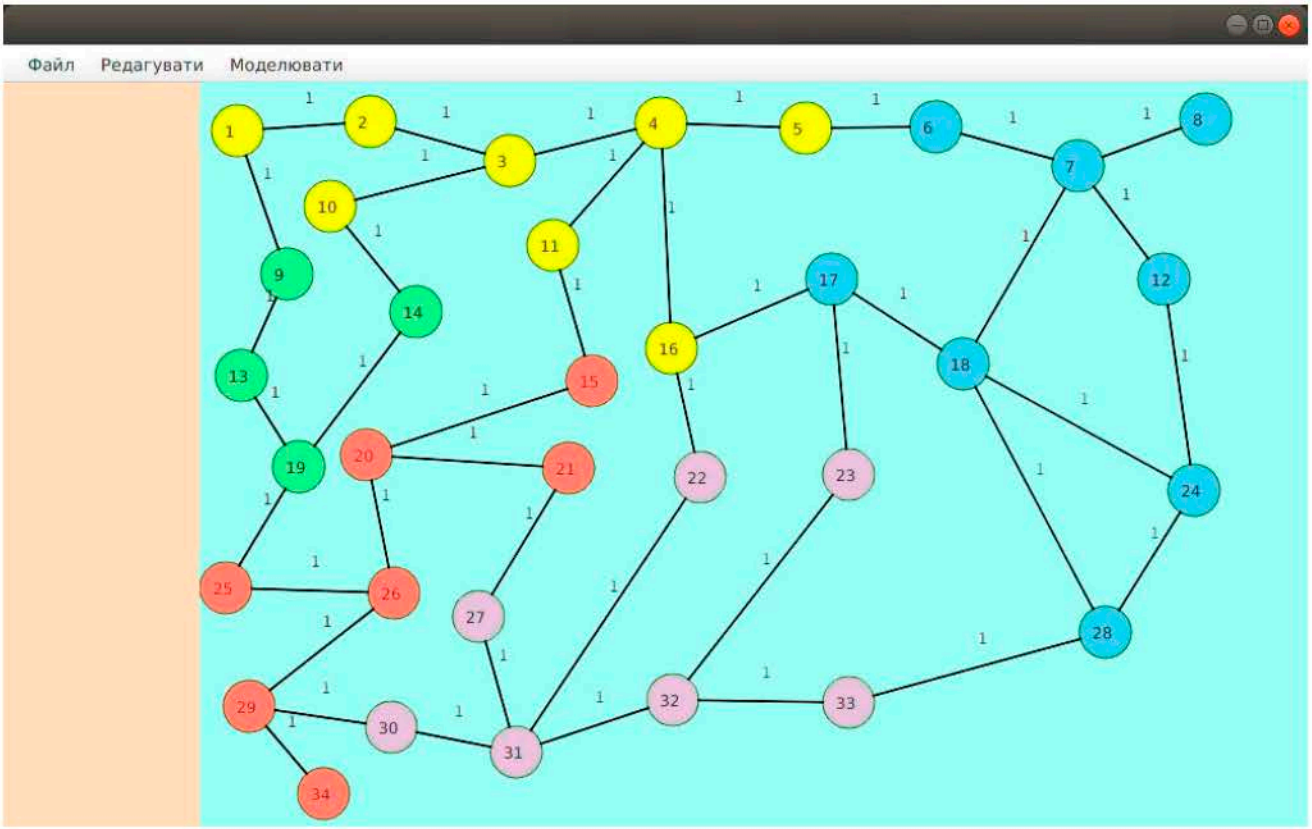
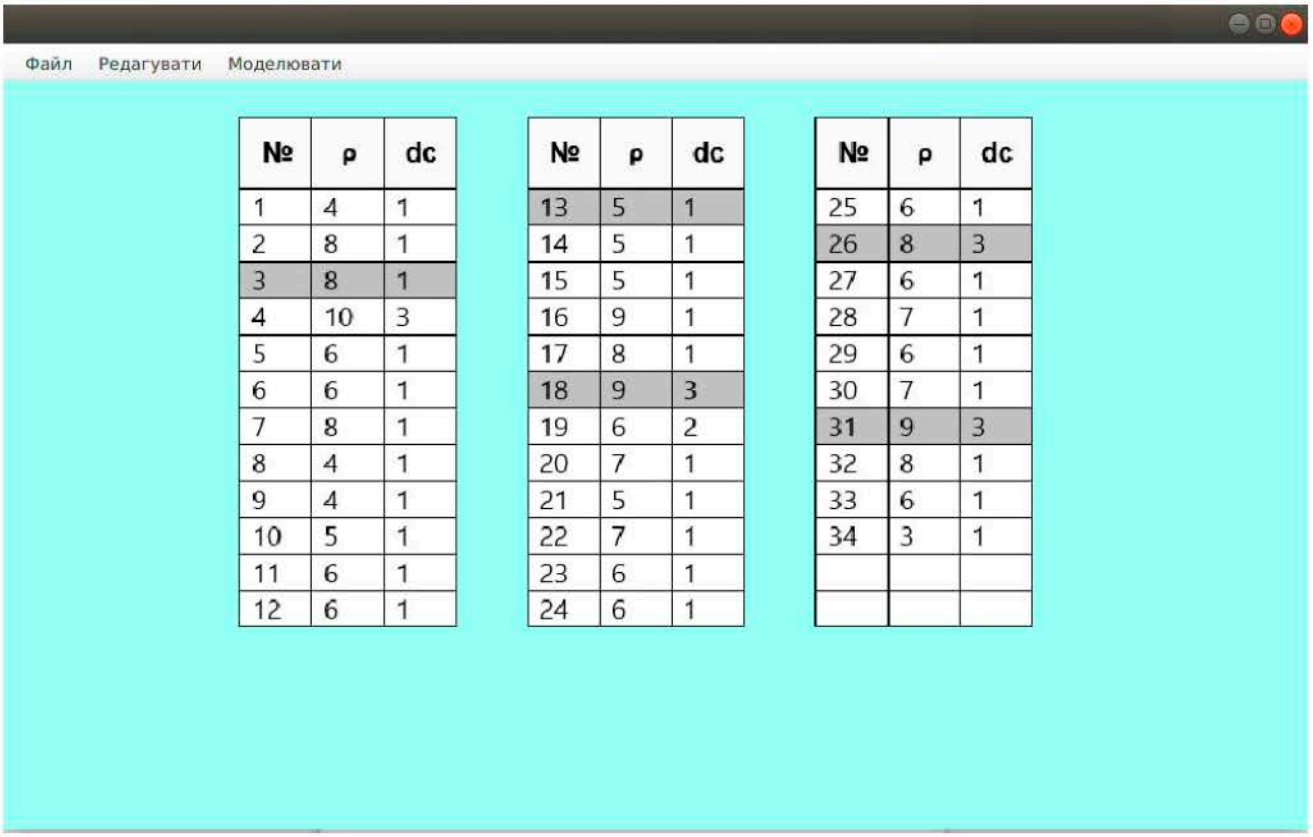


Рисунок 1.19. Результати маршрутизації мережі

```
Результати виконання кластеризації:
----------
----------
Кластер S1 складається з вершин {V1, V2, V3, V4, V5, V5, V10, V11, V16}.
----------
Кластер S2 складається з вершин {V6, V7, V8, V12, V17, V18, V24, V28}.
----------
Кластер S3 складається з вершин {V9, V13, V14, V19}.
----------
Кластер S4 складається з вершин {V15, V20, V21, V25, V26, V29, V34}.
----------
Кластер S5 складається з вершин {V22, V23, V27, V30, V31, V32, V33}.
----------
```

Рисунок 1.20. Результати розбиття вершин графу на кластери у вигляді тексту



№	ρ	dc
1	4	1
2	8	1
3	8	1
4	10	3
5	6	1
6	6	1
7	8	1
8	4	1
9	4	1
10	5	1
11	6	1
12	6	1

№	ρ	dc
13	5	1
14	5	1
15	5	1
16	9	1
17	8	1
18	9	3
19	6	2
20	7	1
21	5	1
22	7	1
23	6	1
24	6	1

№	ρ	dc
25	6	1
26	8	3
27	6	1
28	7	1
29	6	1
30	7	1
31	9	3
32	8	1
33	6	1
34	3	1

Рисунок 1.21. Результати обчислення параметрів моделювання

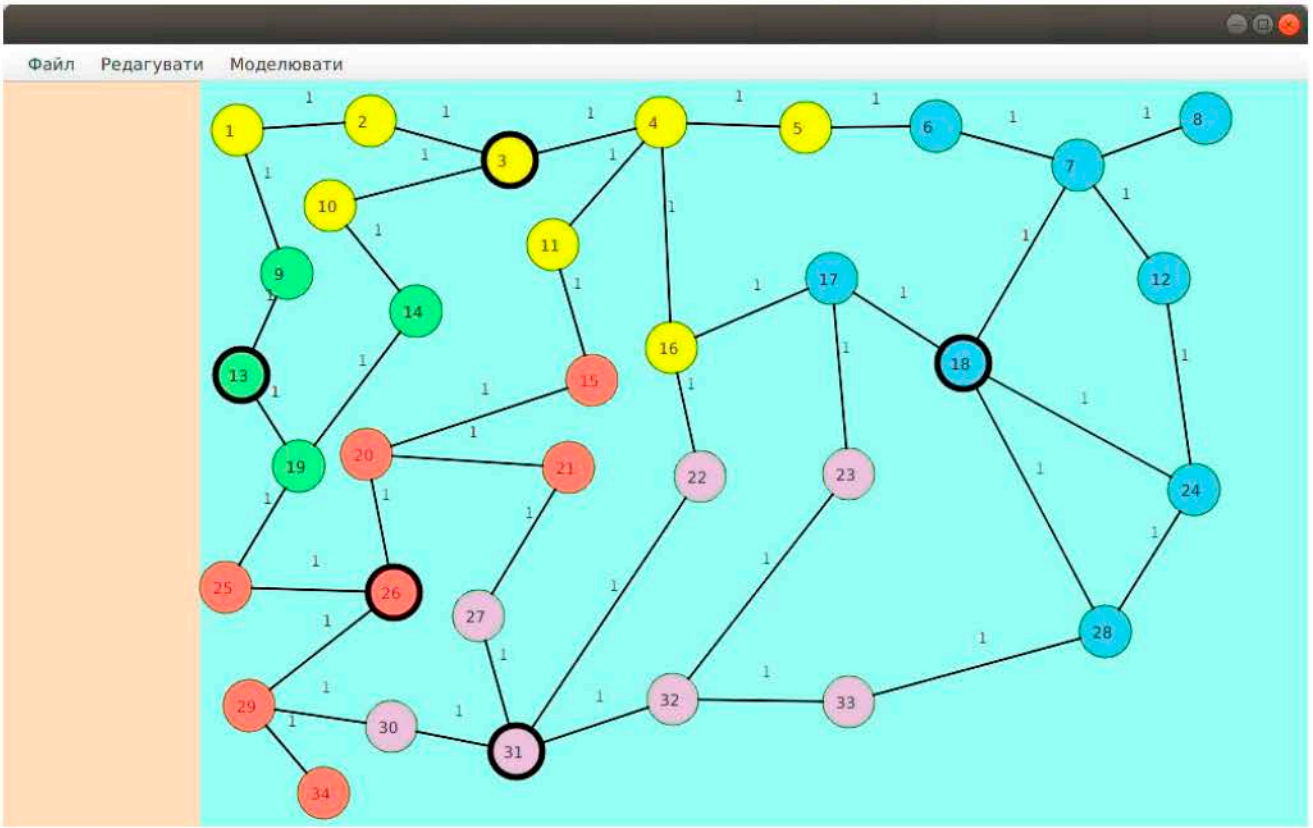


Рисунок 1.22. Результати розташування контролерів у імітованій мережі

1.13 Порівняння з іншими алгоритмами та аналіз результатів

При виконанні тестування розробленого алгоритму маршрутизації та кластеризації програмно-визначуваної мережі було здійснено декілька вимірювань для аналізу роботи моделі. При цьому було виконане порівняння розробленого методу з методами кластеризації POCO та k-середніх.

Для реалізації тестування використано розроблений у розділі 3 генератор випадкових мережевих графів.

Результати аналізу швидкості процесу вирішення проблеми розгортання контролерів у мережі представлені у табл. 1.2 та на рис. 1.23. Для аналізу використовувався графічний генератор з інкрементом по 25 вузлів, всі трафікові з'єднання приймалися за одиницю.

Таблиця 1.2. Швидкість різних алгоритмів залежно від кількості вузлів мережі

	Алгоритм моделювання		
Кількість вершин	**Модифікований**	**k-середніх**	**POCO**
25	0,145	0,121	0,213
50	0,345	0,249	0,346
75	0,687	0,784	1,234
100	1,237	1,456	1,983
125	1,751	1,876	4,305
150	2,221	2,687	6,217
175	2,987	3,456	8,329
200	3,415	4,324	9,299
225	3,769	5,607	10,845
250	4,681	6,809	13,657

Виконані вимірювання дозволили отримати наступні результати. Різниця у швидкості між відносно невеликими (до 100) мережами у алгоритмів невелика, але різниця починає зростати між розробленим алгоритмом, та алгоритмом K-середніх та алгоритмом POCO з поліноміальною складністю. Варто відмітити, що порядок поліноміальної складності різниці з алгоритмом K-середніх вище. У табл. 1.3 та на рис. 1.23 представлені результати аналізу затримки розподілу сервісного трафіку залежно від кількості вузлів. Для аналізу використовувався графічний генератор з інкрементом по 25 вузлів, всі трафікові з'єднання приймалися за одиницю.

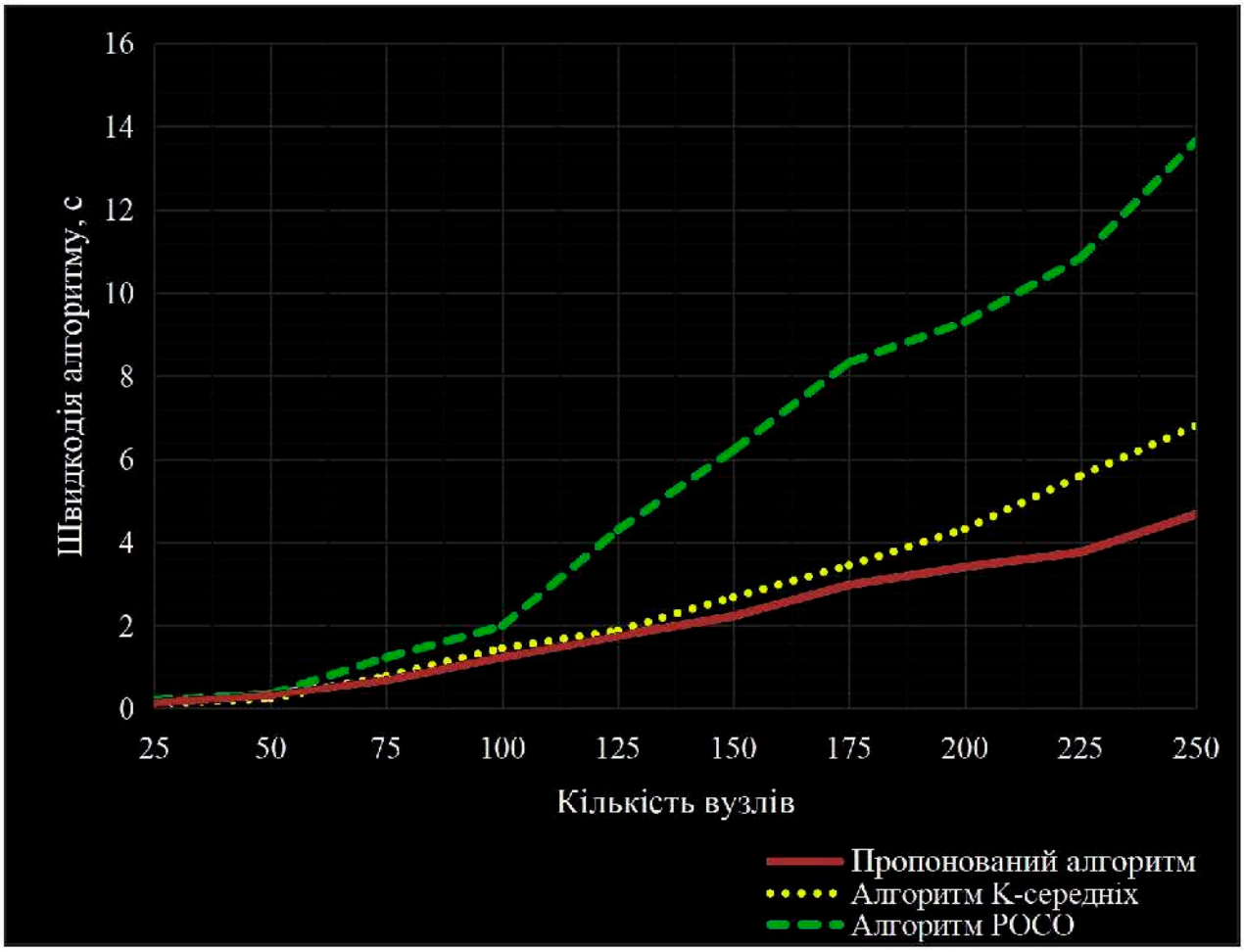


Рисунок 1.23. Залежність продуктивності різних алгоритмів кластеризації від кількості вузлів мережі

Таблиця 1.3. Затримка розподілу сервісного трафіку залежно від кількості вузлів мережі

	Алгоритм моделювання		
Кількість вершин	**Модифікований**	**k-середніх**	**POCO**
25	6	7	8
50	8	7	7
75	5	6	9
100	8	12	14
125	9	8	9
150	10	9	11
175	11	9	10
200	10	11	12
225	15	18	17
250	21	20	23

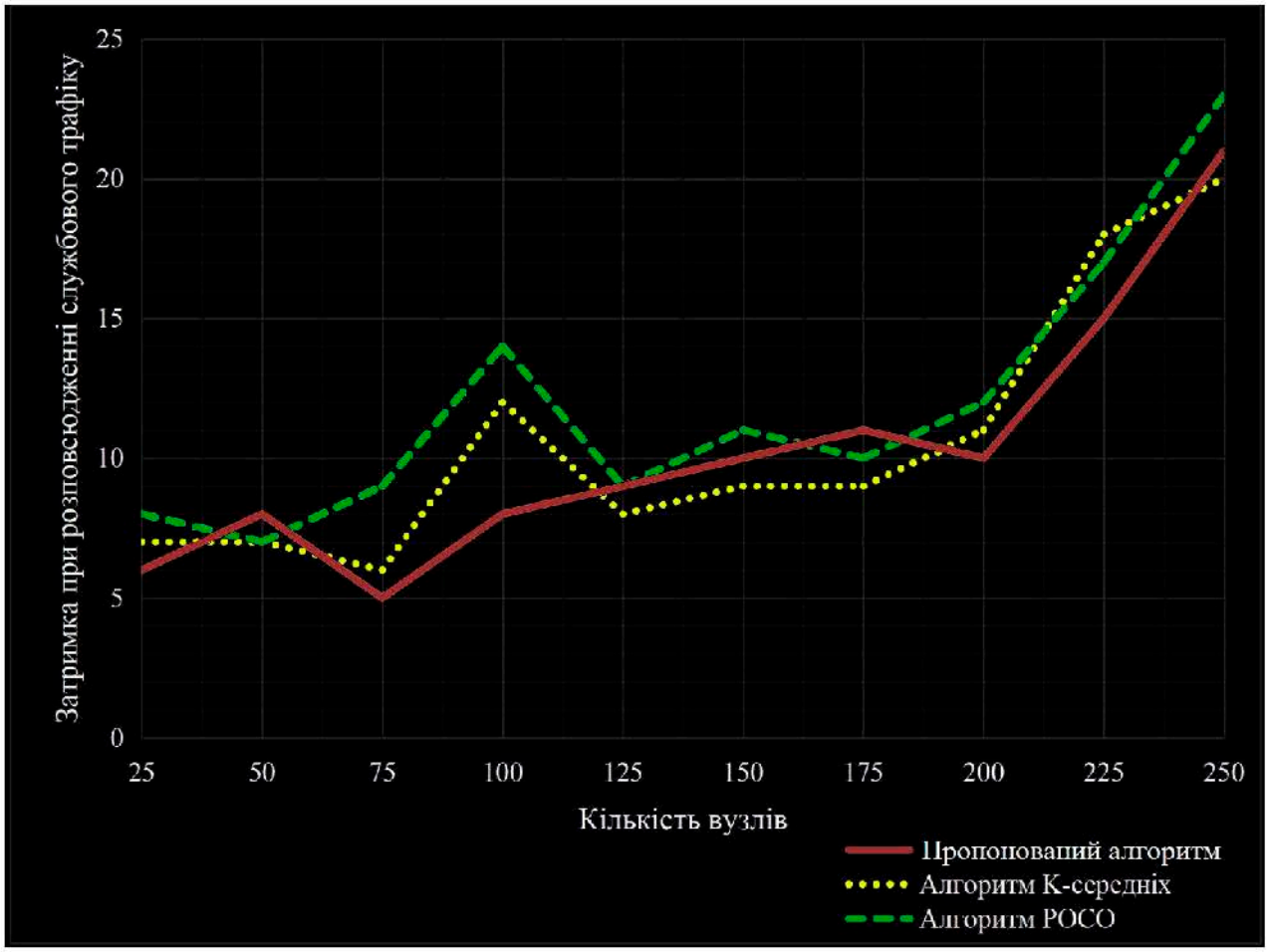


Рисунок 1.24. Залежність затримки розподілу сервісного трафіку від кількості вузлів мережі

Виконані вимірювання дозволили отримати наступні результати. Запропоновані алгоритми мають різні значення затримки для різних кількісних значень вузлів, але кращі показники має саме розроблений алгоритм.

У табл. 1.4 та на рис. 1.25 представлені результати аналізу насичення топології контролерами, коли загальний трафік маршрутизаторів обмежений. Для аналізу використовувався великий граф (250 вершин) і розгорнута відповідна кількість контролерів.

Виконані вимірювання дозволили отримати наступні результати. Розроблений алгоритм дозволяє охопити створену топологію більш високим показником покриття, що свідчить про те, що алгоритм працює більш ефективно при вирішенні проблеми розташування контролерів з обмеженим трафіком підлеглих маршрутизаторів.

Таблиця 1.4. Насичення топології контролерами у випадках обмеження трафіку маршрутизаторів мережі

	Алгоритм моделювання		
Кількість контролерів	**Розроблений**	**k-середніх**	**POCO**
5	0,19	0,16	0,17
10	0,24	0,19	0,22
15	0,34	0,27	0,31
20	0,43	0,37	0,39
25	0,54	0,49	0,48
30	0,69	0,595	0,58
35	0,75	0,7	0,67
40	0,91	0,84	0,82
45	0,95	0,92	0,89
50	0,98	0,94	0,95

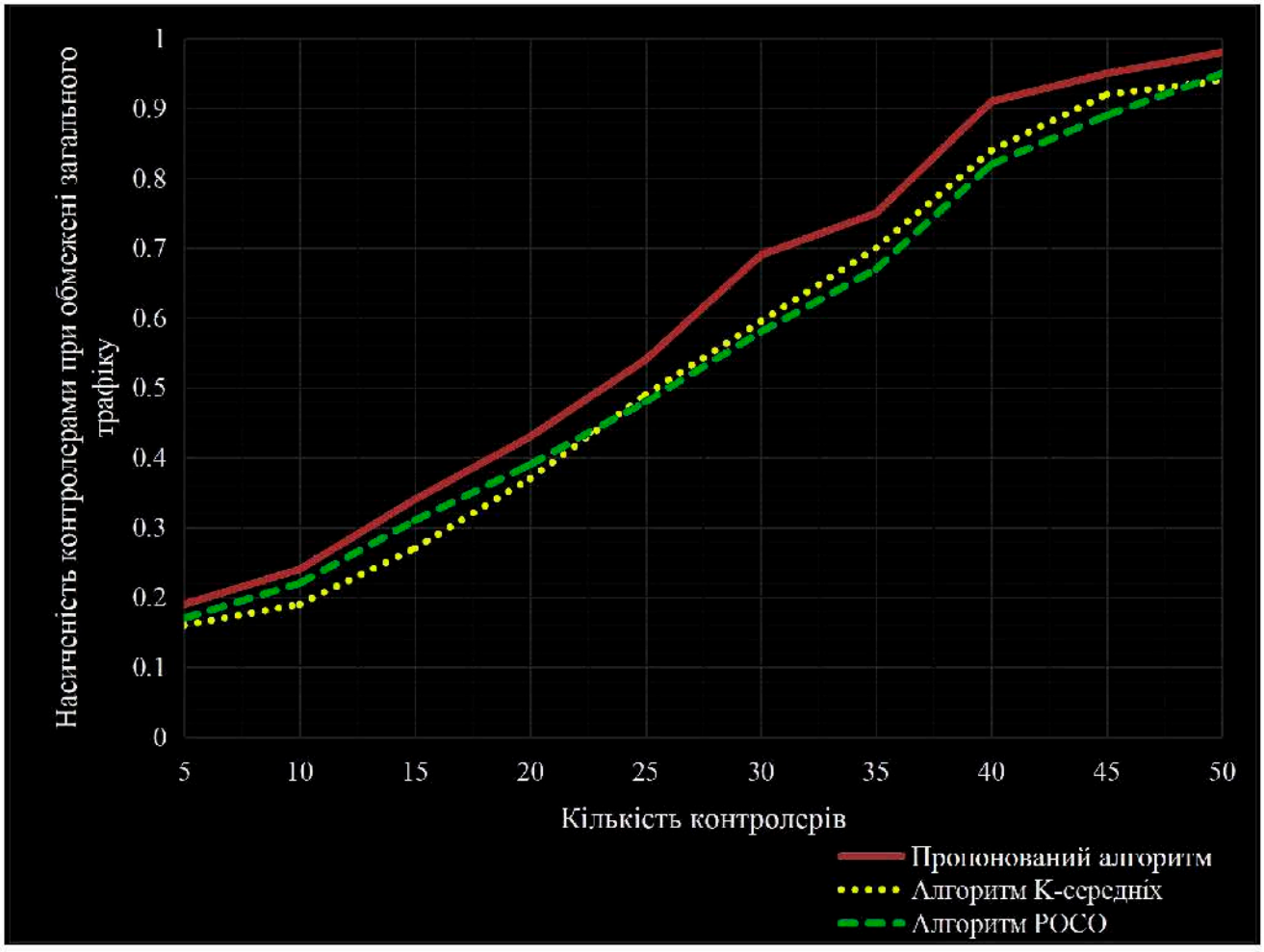


Рисунок 1.25. Насичення топології контролерами у випадках обмеження трафіку маршрутизаторів мережі

У табл. 1.5 та на рис. 1.26 наведені результати аналізу насичення топологічними контролерами, у випадку коли обмежили загальну кількість

маршрутизаторів. Для аналізу використовувався великий граф (250 вершин) і розгорнута відповідна кількість контролерів.

Таблиця 1.5. Насичення топології контролерами з обмеженням на загальну кількість табличних маршрутизаторів мережі

	Алгоритм моделювання		
Кількість вершин	**Розроблений**	**k-середніх**	**POCO**
5	0,19	0,15	0,16
10	0,34	0,29	0,32
15	0,55	0,48	0,51
20	0,76	0,63	0,64
25	0,91	0,77	0,83
30	0,95	0,89	0,93
35	0,96	0,95	0,94
40	0,98	0,97	0,97
45	0,99	0,99	0,99
50	1	1	1

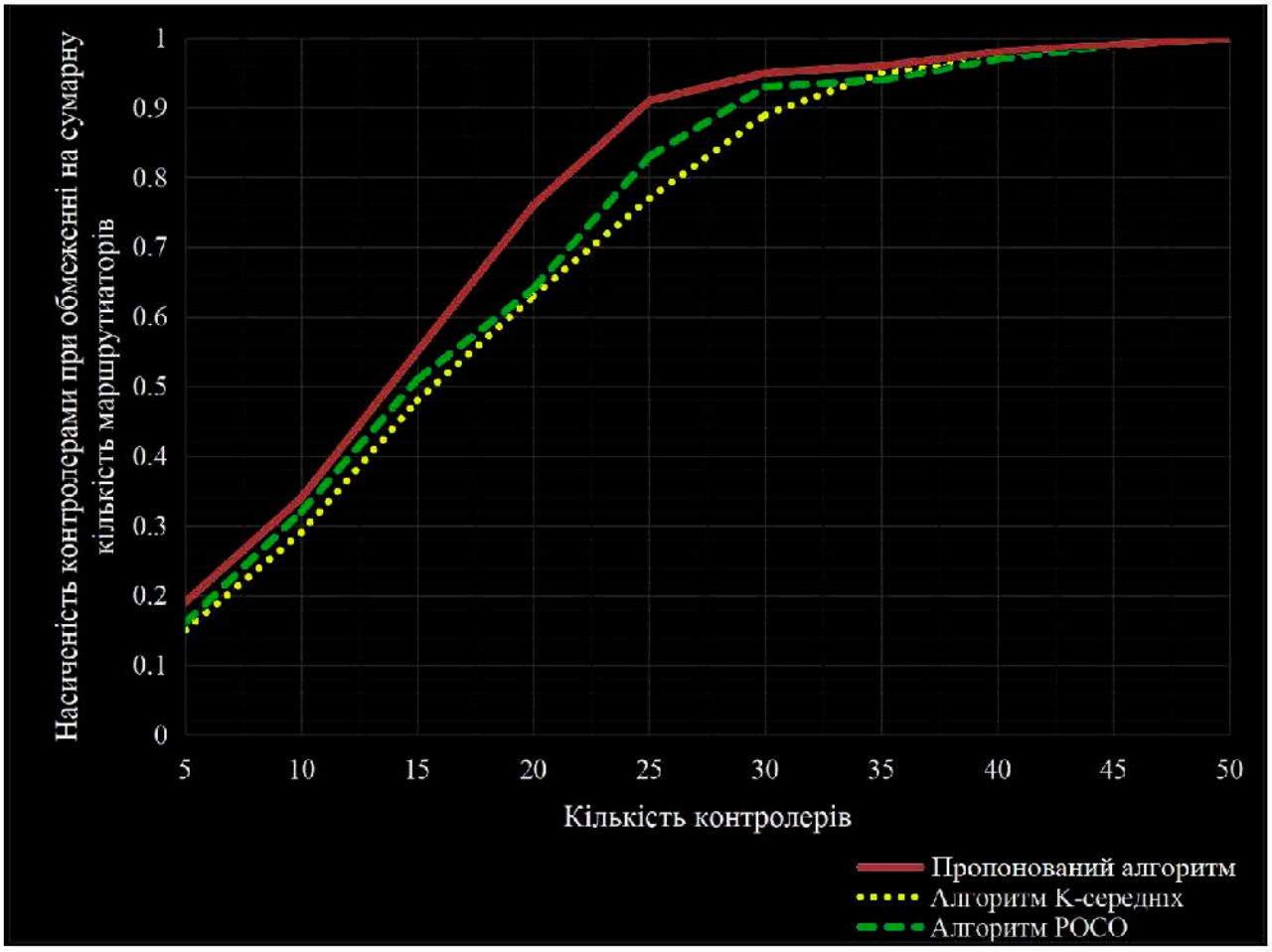


Рисунок 1.26. Насичення топології контролерами з обмеженням на загальну кількість табличних маршрутизаторів мережі

2 ОХОРОНА ПРАЦІ

Важливою умовою життєдіяльності людей є праця, тобто діяльність, спрямована на створення матеріальних цінностей. Праця становить єдність двох функцій: засобу до життя і сферу ствердження особи. Перша з цих функцій знаходить своє відображення в орієнтуванні працівника на зміст праці, її відповідність його внутрішнім запитам, моральне задоволення роботою.

Основним завданням ергономічного забезпечення є оптимізація взаємодії між людиною й машиною не тільки в період експлуатації людино-машинних систем, але й при виготовленні й навіть утилізації технічних компонентів. Це досягається в результаті проведення й виконання комплексу взаємопов'язаних за значенням, логіці й послідовності ергономічних процедур і заходів, здійснюваних у ході розробки системи людина-машина й при її експлуатації.

В даному розділі дипломного проекту розглядається питання охорони праці програміста.

2.1 Аналіз небезпечних і шкідливих факторів, що впливають на програміста

Інтенсивна і тривала робота з персональним комп'ютером є причиною виникнення низки хвороб. Постійні користувачі ПК частіше і більшою мірою піддаються психологічними стресами, функціональним порушенням центральної нервової системи і верхніх дихальних шляхів. Низькочастотні електромагнітні поля при взаємодії з іншими негативними факторами можуть ініціювати ракові захворювання і лейкемію. Пил, що притягається електростатичним полем монітора, як і будь-який пил, іноді стає причиною дерматитів обличчя, загострення астматичних симптомів, роздратування слизистих оболонок. Мікрокліматичні умови на комп'ютеризованих робочих місцях найчастіше не задовольняють встановленим нормам, що призводить до перерахованих вище фізичних відхилень організму. Програміст як і користувач персонального комп'ютера випробовує значне навантаження, як фізичне (сидяче положення, навантаження на очі), так і розумове, що приводить до зниження його

працездатності до кінця робочого дня. Тому необхідно розробити засоби захисту від цих шкідливих факторів.

2.2 Гігієнічні вимоги до виробничого середовища.

До даних засобів захисту відносять: вентиляція, штучне освітлення, звукоізоляція. Існують нормативи, що визначають комфортні умови й гранично припустимі норми запиленості, температури повітря, шуму, освітленості. На робочому місці програміста повинні бути створені умови для високопродуктивної праці.

2.2.1 Вимоги до приміщення

Обємно-планувальні рішення будівель та приміщень для роботи з ВДТ мають відповідати вимогам ДСанПІН 3.3.2.007-98. Розміщення робочих місць з ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено. Площа на одне робоче місце становить не менше 6,0 м^2, а об'єм – не менше ніж 20,0 м^3. У приміщеннях слід щоденно робити вологе прибирання. Вони повинні бути оснащені аптечками першої медичної допомоги. При приміщеннях мають бути обладнанні побутові приміщення для відпочинку.

2.2.2 Освітлення

Робочі кімнати і кабінети повинні мати природне освітлення. В інших приміщеннях допускається штучне освітлення. У тих випадках, коли одного природного освітлення не вистачає, установлюється сполучене освітлення. При цьому додаткове штучне освітлення застосовується не тільки в темне, але та у світлий час доби. Раціональне колірне оформлення приміщення впливає на нервову систему людини, його настрій і в кінцевому рахунку на продуктивність праці. Норма для необхідної освітленості робочого місця становить 300-500 лк.

2.2.3 Шум

Зниження шуму в джерелі випромінювання можна забезпечити застосуванням пружних прокладок між підставою машини, приладу та опорною поверхнею. Як прокладки використовуються гума, повсть, пробка, різної конструкції амортизатори. Під настільні шумливі апарати можна підкладати м'які

килимки із синтетичних матеріалів, а під ніжки столів, на яких вони встановлені, – прокладки з м'якої гуми, повсті, товщиною 6...8 мм. При розумовій праці, яка вимагає зосередженості припустимий рівень шуму становить 50дБ.

2.2.4 Мікроклімат

В процесі трудової діяльності людина знаходиться в постійній тепловій взаємодії з виробничим середовищем. Посилення енерговитрат і обміну речовин, при виконанні роботи викликає в організмі працівника збільшення теплотворення, яке відображається на його терморегуляції.

Одним з важливих складових мікроклімату є концентрація іонів в повітрі робочої зони. Дослідження показали, що в процесі роботи ВДТ протягом зміни концентрація іонів в повітрі робочої зони користувачів зазнає значні зміни.

Для забезпечення оптимальних мікрокліматичних умов в будь-який період року для приміщень в яких розташовані комп'ютеризовані робочі місця повинно бути виконано:

- раціональне розміщення технологічного обладнання (обладнання яке є джерелом тепла, бажано розміщувати безпосередньо біля зовнішніх стін будівлі і в одну низку на такій відстані один від одного, щоб теплові потоки від них не перехрещувалися на робочих місцях);
- опалювання і кондиціонування повітря (найпоширеніші способи нормалізації мікроклімату у виробничих приміщеннях, забезпечують нормальні теплові умови в холодний період року у великогабаритних і полегшених промислових будівлях);
- раціоналізація режимів праці і відпочинку (досягається скороченням тривалості робочого часу за рахунок додаткових перерв, створенням умов для ефективного відпочинку в приміщеннях з нормальними метеорологічними умовами); теплоізоляція обладнання і захисних екранів (як теплоізоляційні матеріали широко використовують:
- азбест, азбоцемент, мінеральну вату, склотканина , керамзит, пінопласт); для підтримки допустимих значень мікроклімату і концентрації позитивних

і негативних іонів необхідно передбачити установки або прилади зволоження та / або штучної іонізації, кондиціонування повітря.

2.2.5 Вимоги до організації робочого місця працівника

На робочому місці програміста повинні бути створені умови для безпечної та високопродуктивної праці.

Робоче місце і взаємне розташовує всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи..

Ергономічними аспектами проектування подібних робочих місць, зокрема, є: висота робочої поверхні, розміри простору для ніг, вимоги до того, що розташовує документів на робочому місці (наявність і розміри підставки для документів, можливість різного розміщення документів, відстань від очей користувача до екрану, документа, клавіатури і т.д.), характеристики робочого крісла, вимоги до поверхні робочого столу тощо.

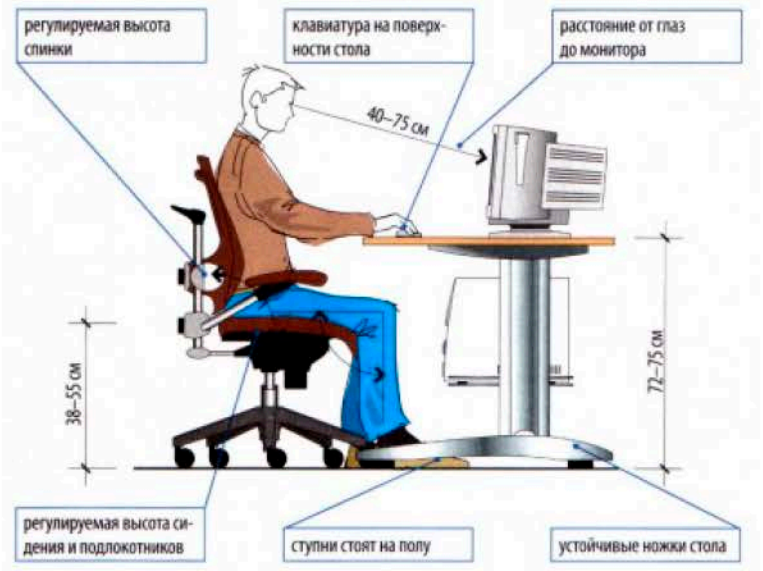


Рисунок 2.1. Організація робочого місця оператора

2.6 Електробезпека

Устаткування ЕОМ, що відноситься до електричних установок, представляє для людини велику потенційну небезпеку, тому що в процесі експлуатації або проведенні профілактичних робіт людина може торкнутися частин, що знаходяться під напругою. Специфічна небезпека електроустановок:

струмоведучі провідники, корпуси стійок ЕОМ і іншого устаткування, що опинилося під напругою в результаті ушкодження (пробою) ізоляції, не подають яких-небудь сигналів, що попереджають людини про небезпеці. Реакція людини на електричний струм виникає лише при протіканні останнього через тіло людини.

Винятково важливе значення для запобігання електротравматизму має правильна організація обслуговування діючих електроустановок , проведення ремонтних, монтажних і профілактичних робіт. При цьому під правильною організацією розуміється строге виконання ряду організаційних і технічних заходів і засобів, установлених діючими Правилами технічної експлуатації електроустановок споживачів(ПТЕ) і правила техніки безпеки при експлуатації електроустановок споживачів (ПТБ споживачів) і Правила устрою електроустановок (ПУЕ). У залежності від категорії приміщення необхідно прийняти визначені міри, що забезпечують достатню електробезпечність при експлуатації і ремонті електроустаткування.

2.7 Пожежна безпека

Пожежна безпека -- це такий стан об'єкта, при якому з регламентованою ймовірністю виключається можливість виникнення й розвитку пожежі та впливу на людей небезпечних факторів пожежі, а також забезпечується захист матеріальних цінностей.

Пожежна безпека приміщень, що мають електричні мережі, регламентується ГОСТ 12.1.033-81, ГОСТ 12.1.004-85. Робота оператора ЕОМ повинна вестися в приміщенні, що відповідає категорії Д пожежної безпеки (негорючі речовини й матеріали в холодному стані).

Приміщення оснащені вуглекислотними або порошковими вогнегасниками. У випадку виникнення пожежі необхідно відключити електроживлення, викликати по телефону 101 пожежну команду, евакуювати людей із приміщення відповідно до плану евакуації і приступити до ліквідації пожежі.

ВИСНОВКИ

У кваліфікаційні роботі виконано аналіз алгоритмів маршрутизації у програмно-визначуваних мережах, визначені оптимальні шляхи розташування контролерів та запропоновано удосконалений метод маршрутизації, який забезпечує рівномірний розподіл навантаження у мережі.

При виконанні роботи було виконано дослідження існуючих алгоритмів маршрутизації та кластеризації програмно-визначуваних мереж, сформовано та описано вдосконалену математичну модель, яку можна використовувати для маршрутизації та кластеризації програмно-визначуваних мереж. При вивченні проблеми маршрутизації, кластеризації та розташування контролерів у підмережі, описані технічні рішення, що дозволяють вдосконалити алгоритм маршрутизації. Цей алгоритм дозволяє вирішити не тільки проблему маршрутизації, але і проблему розташування контролерів у два етапи: мережева кластеризація та вибір місця керування у кожній підмережі програмно-визначуваного типу.

У роботі також розглянуто приклад використання вдосконаленого алгоритму маршрутизації та кластеризації. Проаналізовано вплив на ефективність розробленого алгоритму параметра максимального значення найближчої дистанції до маршрутизатора.

У роботі доведено, що запропонований метод демонструє більш високу ефективність на різних мережевих топологіях у порівнянні з існуючими алгоритмами POCO та k-середніх за показниками затримки і швидкості руху службового трафіку. Модифікований вдосконалений алгоритм має переваги з точки зору параметрів продуктивності, показників топологічної насиченості при розміщенні контролерів та розподілу сервісного трафіку по групах. Наявність великої кількості маршрутів дозволяє практично виключити затримку або втрату пакетів в процесі маршрутизації трафіку. Чим більше шляхів буде сформовано в контролері програмно-визначуваної мережі, тим меншою буде ймовірність затримки або втрати пакетів.

Результати, отримані у роботі, дозволяють підвищити ефективність керування програмно-визначуваними мережами великого розміру.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Комп'ютерне моделювання систем та процесів. Методи обчислень: навчальний посібник / Квєтний Р. Н., Богач І. В., Бойко О. Р., Софина О. Ю.,
2. Гніденко М.П. Побудова SDN мереж. Навчальний посібник / М.П. Гніденко, В.В. Вишнівський, О.О. Ільїн. – Київ: ДУТ, 2019. – 190 с.
3. Шушура О.М.; за заг. ред. Р.Н. Квєтного. – Вінниця: ВНТУ, 2012. – 193 с.
4. Кулаков Ю.О. Комп'ютерні мережі / Ю.О. Кулаков – Юніор, 2005. – 397 с.
5. Вишневський В. М. Теоретичні основи проектування комп'ютерних мереж / В. М. Вишневський – Техносфера, 2004. – 512 с.
6. Cisco Systems Руководство по технологиям объединенных сетей / Cisco Systems – 3-е издание. СПб: "Вильямс", 2002. – 1040 с.
7. Дебра Литтлджон Шиндер Основы компьютерных сетей / Дебра Литтлджон Шиндер – СПб: "Вильямс", 2002. – 656 с.
8. Коротыгин С. Стандарт IEEE 802.11 и его расширения / С. Коротыгин, А. Нежуренко – Сети и телекоммуникации, вып. 6(25), 2002 г.
9. Стеклов В. К. Проектування телекомунікаційних мереж / В. К. Стеклов, Л. Н. Беркман. ; під ред. В. К. Стеклова – Київ : Техніка, 2002. – 792 с.
10. Стеклов В. К. Сучасні системи управління в телекомунікація / В. К. Стеклов, Б. Я. Костік, Л. Н. Беркман – Київ : Техніка, 2005. – 395 с.
11. Орлов Є. В. Програмно-конфігуровані мережі (SDN): архітектура, міжнародна стандартизація / Є. В. Орлов // Наукові записки УНДІЗ. – 2014. – №4(32) – С. 85-91.
12. Смелянский Р. В. Программно-конфигурируемые сети [Електронний ресурс] / Р. В. Смелянский // Открытые системы. – 2012. – № 9. – Режим доступу: http://www.osp.ru/os/2012/09/13032491.

ДОДАТОК А

Фрагмент коду додатку для моделювання вдосконаленого алгоритму маршрутизації та кластеризвації (мова Java, клас OrderMaker)

```
package pzks.model;
import java.util.Arrays; import java.util.LinkedList;
public class OrderMaker { int[][] links;
int[] ids;
int[] weights; int[][] parametres;
boolean[] isMarked;
int[][] results; int critTime;
public OrderMaker(int[][] links, int[] ids, int[] weights) { this.links = new
int[links.length][links.length];
this.ids = new int[links.length]; this.weights = new int[links.length]; for (int i = 0; i < links.length;
i++) {
this.ids[i] = ids[i]; this.weights[i] = weights[i];
for (int j = 0; j < links.length; j++) { this.links[i][j] = links[i][j];
}
}
parametres = new int[7][links.length]; wayDown();
wayUp();
int tMax = 0;
for (int i = 0; i < links.length; i++) {
if (tMax < parametres[0][i] + weights[i]) tMax = parametres[0][i] + weights[i];
}
for (int i = 0; i < links.length; i++) { parametres[2][i] = tMax - parametres[1][i];
parametres[3][i] = parametres[2][i] - parametres[0][i];
}
System.arraycopy(weights, 0, parametres[5], 0, links.length);
System.arraycopy(weights, 0, parametres[6], 0, links.length);
critTime = 0;
for(int i=0; i<parametres[1].length; i++) if(critTime<parametres[1][i])
critTime=parametres[1][i];
int[] generatedArray = new int[links.length]; for (int i = 0; i < generatedArray.length; i++) {
generatedArray[i] = i;
}
int[][] temp1 = new int[2][links.length]; System.arraycopy(generatedArray, 0, temp1[0], 0,
links.length);
System.arraycopy(parametres[3], 0, temp1[1], 0, links.length); sort1(temp1[0], temp1[1]);
int[][] temp2 = new int[3][links.length]; System.arraycopy(generatedArray, 0, temp2[0], 0,
links.length);
System.arraycopy(parametres[4], 0, temp2[1], 0, links.length);
System.arraycopy(parametres[5], 0, temp2[2], 0, links.length); sort2(temp2[0], temp2[1],
temp2[2]);
int[][] temp3 = new int[2][links.length]; System.arraycopy(generatedArray, 0, temp3[0], 0,
links.length);
System.arraycopy(parametres[6], 0, temp3[1], 0, links.length); sort3(temp3[0], temp3[1]);
results = new int[3][links.length]; System.arraycopy(temp1[0], 0, results[0], 0, links.length);
System.arraycopy(temp2[0], 0, results[1], 0, links.length);
System.arraycopy(temp3[0], 0, results[2], 0, links.length);
}
public int getCritTime() { return critTime;
}
void printTranslatedArray(int[] array) { if (array.length > 0) {
String result = "[";
```

```
for (int i = 0; i < array.length - 1; i++) { result += ids[array[i]] + ", ";
}
result += ids[array[array.length - 1]] + "]"; System.out.println(result);
} else {
System.out.println("[]");
}
}
int[][] getOrders() { return results;
}
int[] sort1(int[] ids, int[] a) {
for (int i = 0; i < links.length; i++) {
for (int j = i + 1; j < links.length; j++) { if (a[j] < a[i]) {
int bufId = ids[i]; int bufA = a[i]; ids[i] = ids[j];
a[i] = a[j]; ids[j] = bufId; a[j] = bufA;
}
}
}
return ids;
}
int[] sort2(int[] ids, int[] a, int[] b) { for (int i = 0; i < links.length; i++) {
for (int j = i + 1; j < links.length; j++) {
if (a[j] < a[i] || (a[j] == a[i] && b[j] > b[i])) { int bufId = ids[i];
int bufA = a[i]; int bufB = b[i];
ids[i] = ids[j];
a[i] = a[j];
b[i] = b[j];
ids[j] = bufId; a[j] = bufA; b[j] = bufB;
}
}
}
return ids;
}
int[] sort3(int[] ids, int[] a) {
for (int i = 0; i < links.length; i++) {
for (int j = i + 1; j < links.length; j++) { if (a[j] < a[i]) {
int bufId = ids[i]; int bufA = a[i]; ids[i] = ids[j];
a[i] = a[j]; ids[j] = bufId; a[j] = bufA;
}
}
}
return ids;
}
void wayDown() { LinkedList<Integer> top = findTop();
isMarked = new boolean[links.length]; LinkedList<Integer> newWave = new LinkedList<>(); for
(int currentNode : top) {
isMarked[currentNode] = true;
for (int i = 0; i < links.length; i++) { if (links[currentNode][i] != 0) {
boolean flag = true;
for (int j = 0; j < links.length && flag; j++) { if (links[j][i] != 0 && !isMarked[j]) {
flag = false;
}
}
if (flag) { newWave.add(i);
}
}
}
```

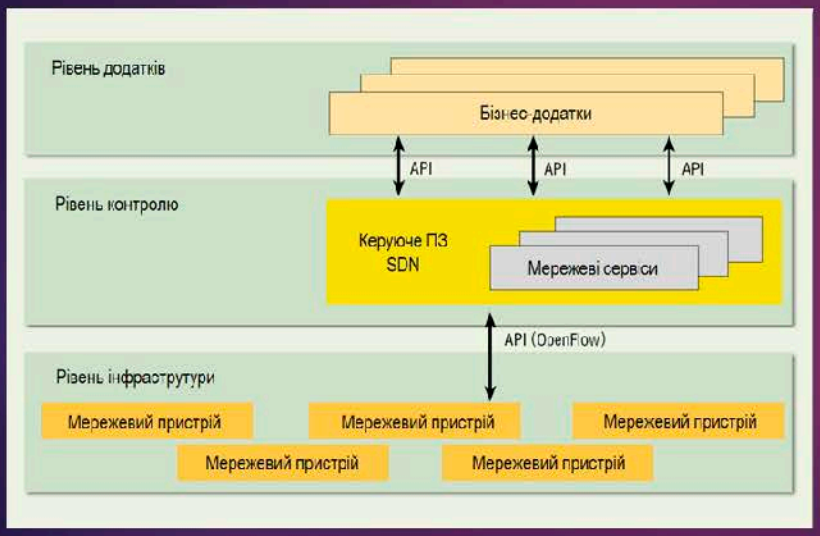
```
while (!newWave.isEmpty()) {
int currentId = newWave.getFirst();
for (int i = 0; i < links.length; i++) { if (links[i][currentId] != 0) {
if (parametres[0][i] + weights[i] > parametres[0][currentId]) { parametres[0][currentId] =
parametres[0][i] + weights[i];
}
if (parametres[4][i] + 1 > parametres[4][currentId]) { parametres[4][currentId] = parametres[4][i] +
1;
}
}
}
isMarked[currentId] = true;
newWave.removeFirst();
for (int i = 0; i < links.length; i++) { if (links[currentId][i] != 0) {
boolean flag = true;
for (int j = 0; j < links.length && flag; j++) { if (links[j][i] != 0 && !isMarked[j]) {
flag = false;
}
}
if (flag) { newWave.add(i);
}
}
}
}
}
}
void wayUp() {
LinkedList<Integer> bottom = findBottom(); isMarked = new boolean[links.length];
LinkedList<Integer> newWave = new LinkedList<>(); for (int currentNode : bottom) {
isMarked[currentNode] = true; parametres[1][currentNode] = weights[currentNode]; for (int i = 0; i
< links.length; i++) {
if (links[i][currentNode] != 0) { boolean flag = true;
for (int j = 0; j < links.length && flag; j++) { if (links[i][j] != 0 && !isMarked[j]) {
flag = false;
}
}
if (flag) { newWave.add(i);
}
}
}
while (!newWave.isEmpty()) {
int currentId = newWave.getFirst();
for (int i = 0; i < links.length; i++) { if (links[currentId][i] != 0) {
if (parametres[1][i] + weights[currentId] > parametres[1][currentId])
{ parametres[1][currentId] = parametres[1][i] + weights[currentId];
}
}
}
isMarked[currentId] = true; newWave.removeFirst();
for (int i = 0; i < links.length; i++) { if (links[i][currentId] != 0) {
boolean flag = true;
for (int j = 0; j < links.length && flag; j++) { if (links[i][j] != 0 && !isMarked[j]) {
flag = false;
}
}
if (flag) { newWave.add(i);
```

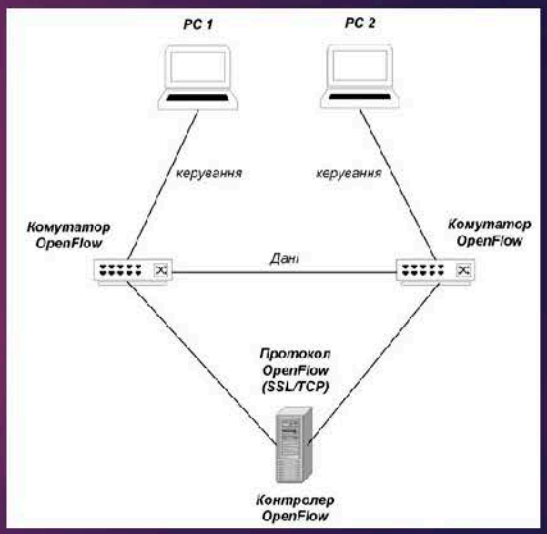
```
}
}
}
}
}
}

LinkedList<Integer> findTop() { LinkedList<Integer> top = new LinkedList<>(); for (int i = 0; i <
links.length; i++) {
boolean flag = true;
for (int j = 0; j < links.length && flag; j++) { if (links[j][i] != 0) {
flag = false;
}
}
if (flag)
top.add(i);
}
return top;
}
LinkedList<Integer> findBottom() { LinkedList<Integer> bottom = new LinkedList<>(); for (int i
= 0; i < links.length; i++) {
boolean flag = true;
for (int j = 0; j < links.length && flag; j++) { if (links[i][j] != 0) {
flag = false;
}
}
if (flag)
bottom.add(i);
}
return bottom;
}
}
```

ДОДАТОК Б

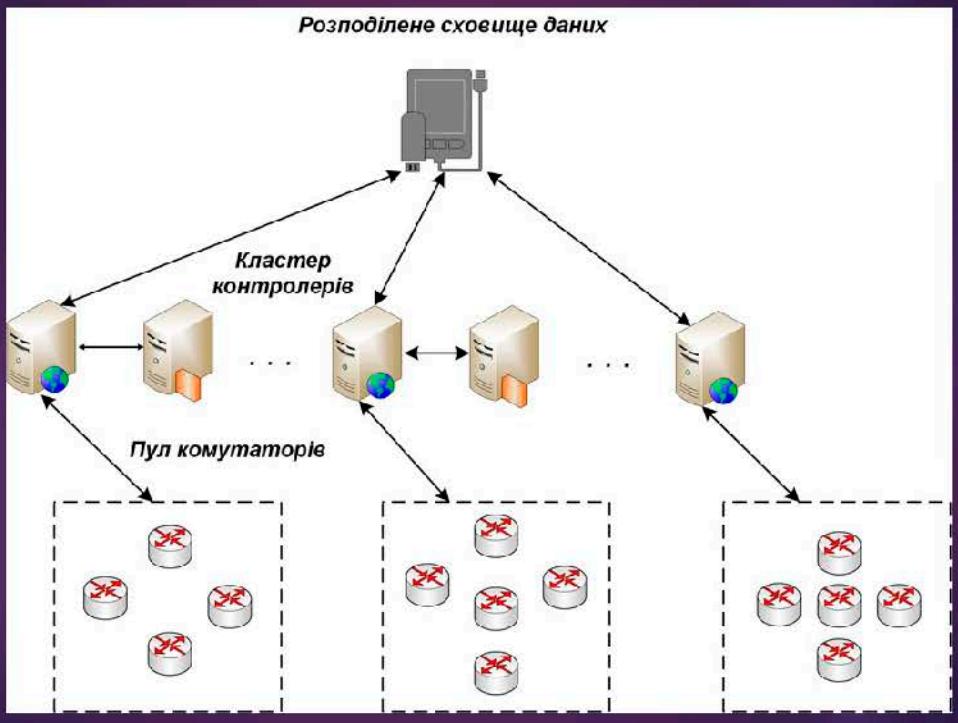
Слайди мультимедійної презентації

Структура програмно-визначуваної мережі за протоколом OpenFlow

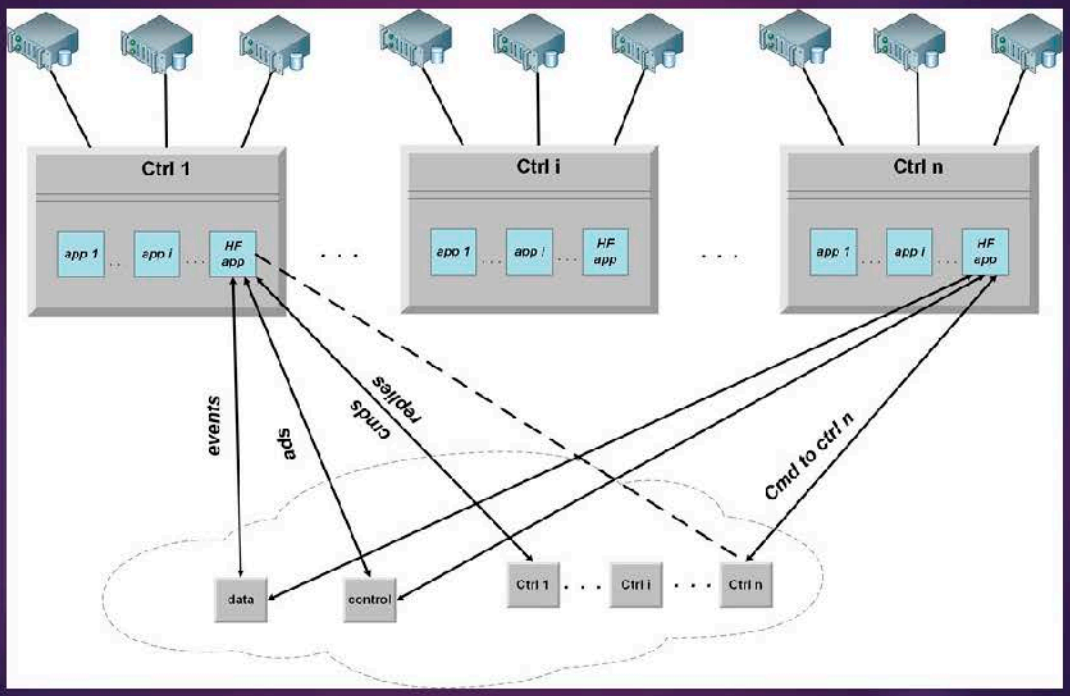




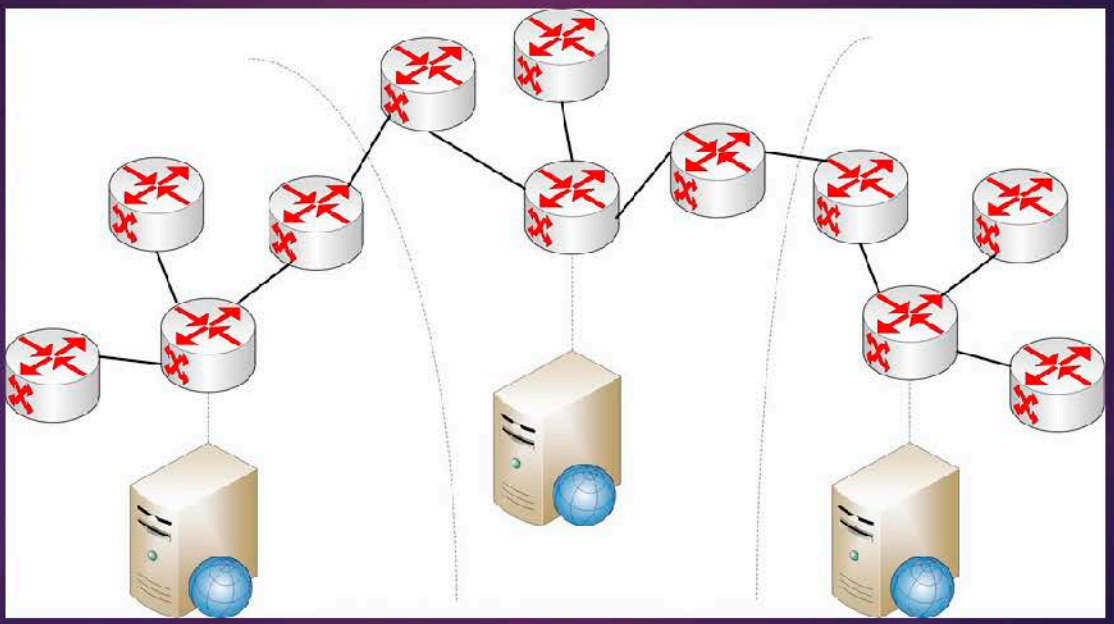
Мережа із розподіленими контролерами



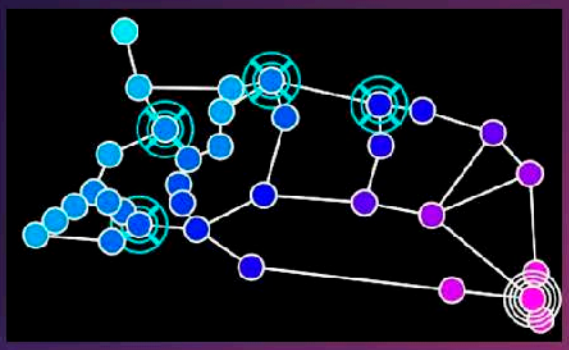
Архітектура технології HyperFlow



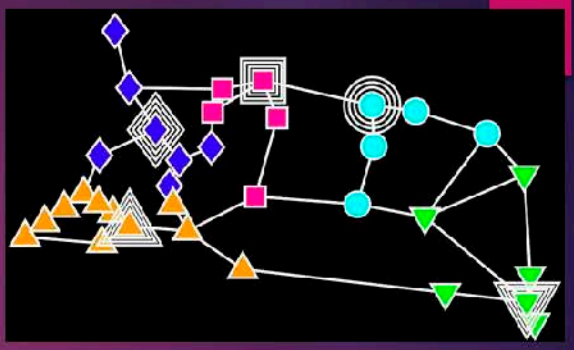
Кластеризація у мережі



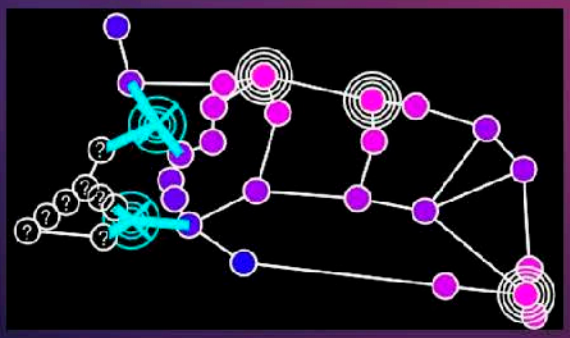
Вихід з ладу деяких мережевих контролерів



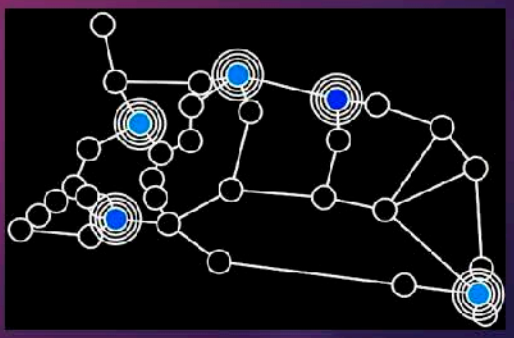
Маршрутизація без урахування навантаження контролерів



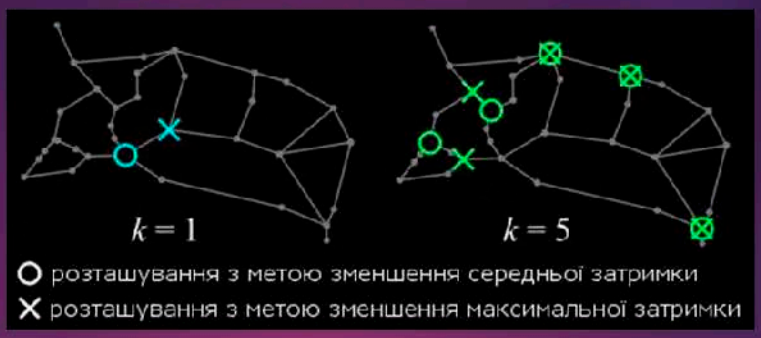
Вихід з ладу деяких вузлів та з'єднань у мережі



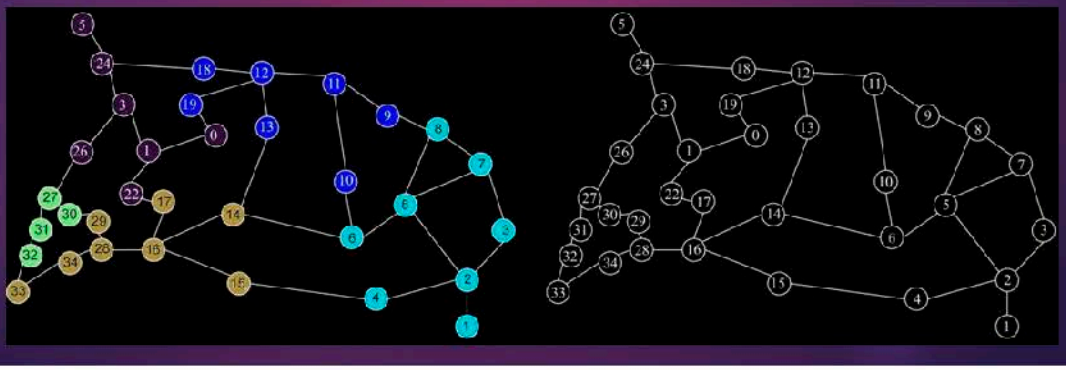
Затримка передачі між мережевими контролерами



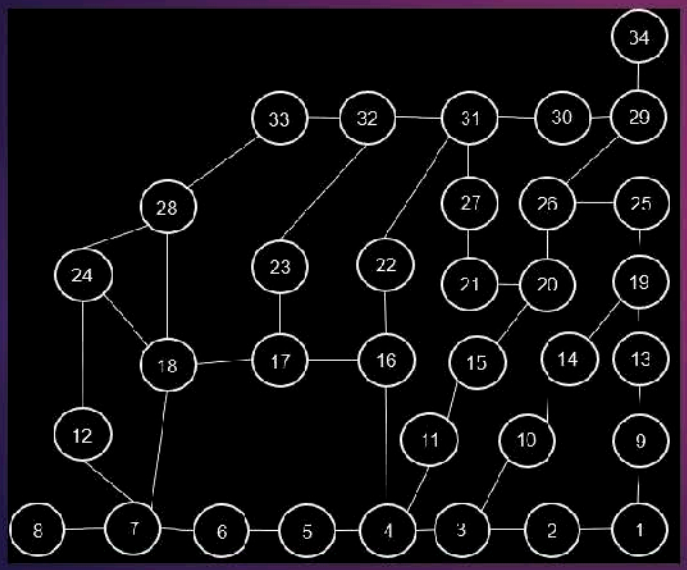
Маршрутизація за методом k-середніх



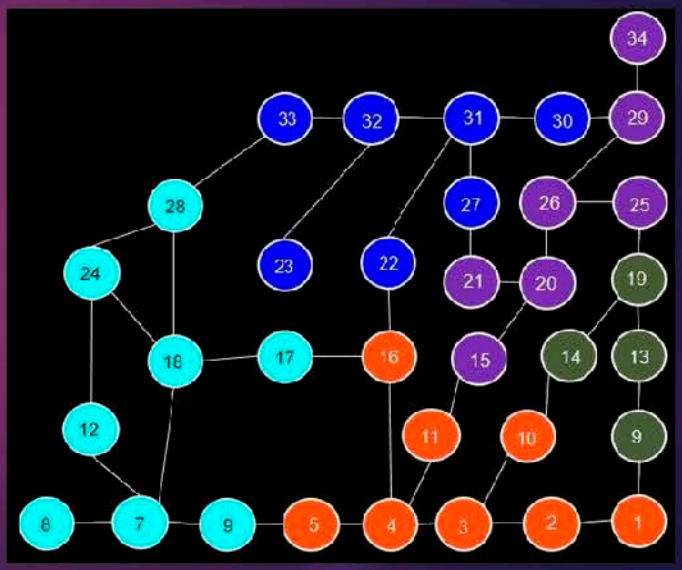
Маршрутизація мережі за щільністю



Приклад графу для моделювання маршрутизації у мережі



Результати роботи удосконаленого алгоритму маршрутизації



Результати моделювання з використанням удосконаленого алгоритму

id вузла	*p*	*δ*	*id вузла*	*p*	*δ*
1	*4*	*1*	*18*	*9*	*3*
2	*8*	*1*	*19*	*6*	*2*
3	*8*	*1*	*20*	*7*	*1*
4	*10*	*3*	*21*	*5*	*1*
5	*6*	*1*	*22*	*7*	*1*
6	*6*	*1*	*23*	*6*	*1*
7	*8*	*1*	*24*	*6*	*1*
8	*4*	*1*	*25*	*6*	*1*
9	*4*	*1*	*26*	*8*	*3*
10	*5*	*1*	*27*	*6*	*1*
11	*6*	*1*	*28*	*7*	*1*
12	*6*	*1*	*29*	*6*	*1*
13	*5*	*1*	*30*	*7*	*1*
14	*5*	*1*	*31*	*9*	*3*
15	*5*	*1*	*32*	*8*	*1*
16	*9*	*1*	*33*	*6*	*1*
17	*8*	*1*	*34*	*3*	*1*

Розподіл локальної щільності по відстані до вузлів із більшою щільністю

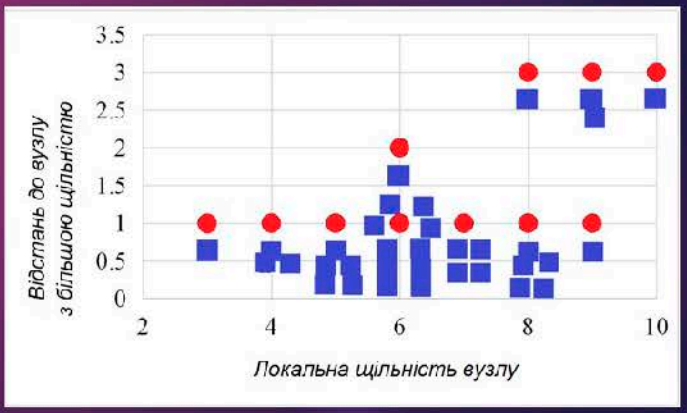
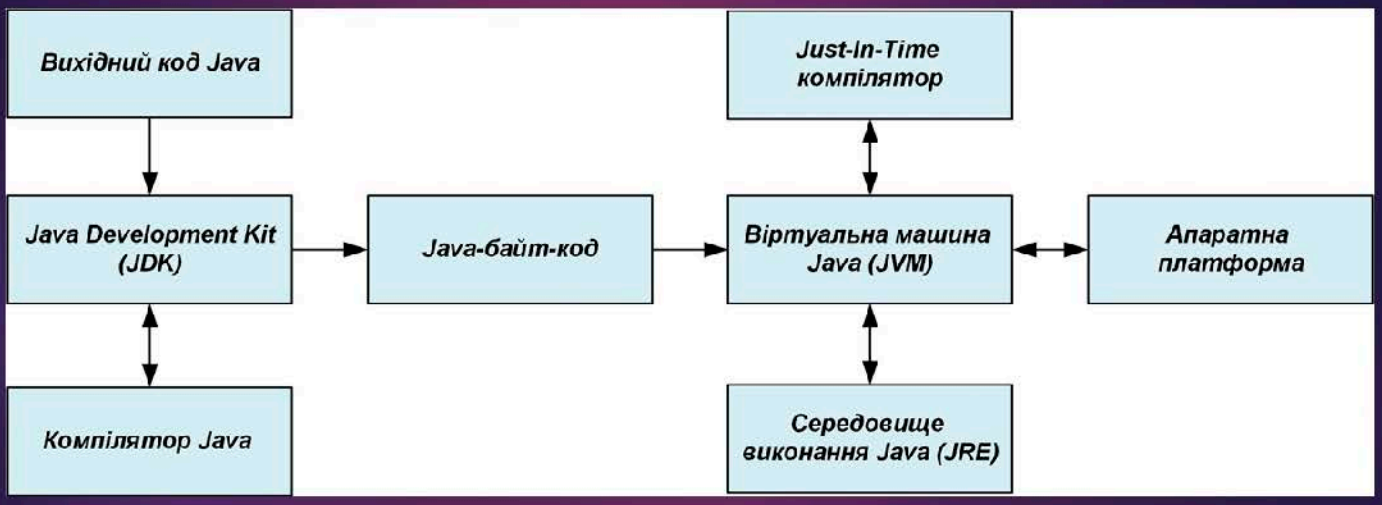
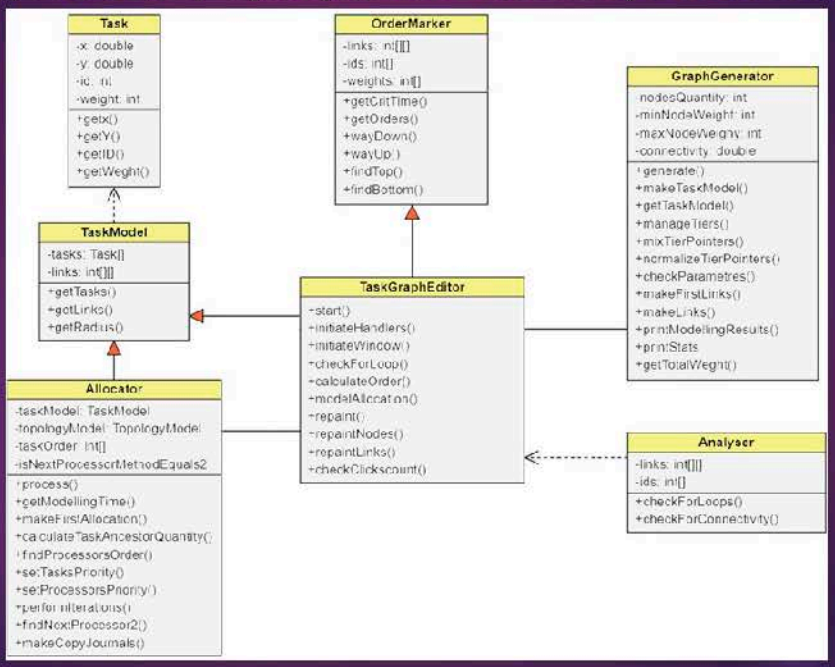


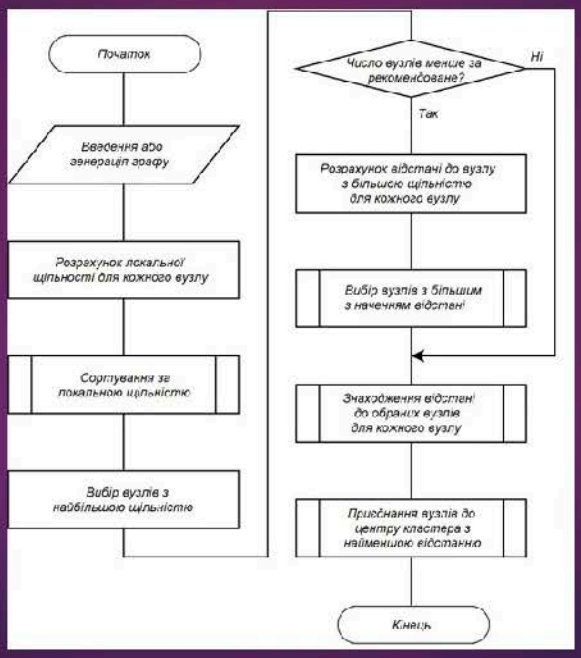
Схема компіляції програми (мова Java)



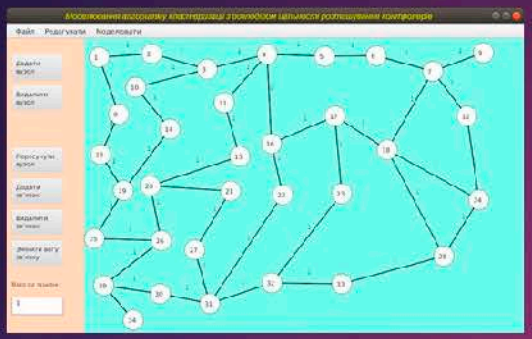
Діаграма класів додатку для моделювання маршрутизації у програмно-визначуваній мережі



БСА програми для моделювання маршрутизації у програмно-визначуваній мережі



Результати створення графу мережі



Результати розбиття вершин графу на кластери у вигляді тексту

Результати виконання кластеризації:

Кластер S1 складається з вершин {V1, V2, V3, V4, V5, V5, V10, V11, V16}.

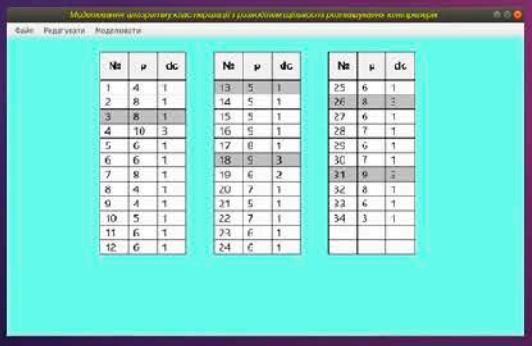
Кластер S2 складається з вершин {V6, V7, V8, V12, V17, V18, V24, V28}.

Кластер S3 складається з вершин {V9, V13, V14, V19}.

Кластер S4 складається з вершин {V15, V20, V21, V25, V26, V29, V34}.

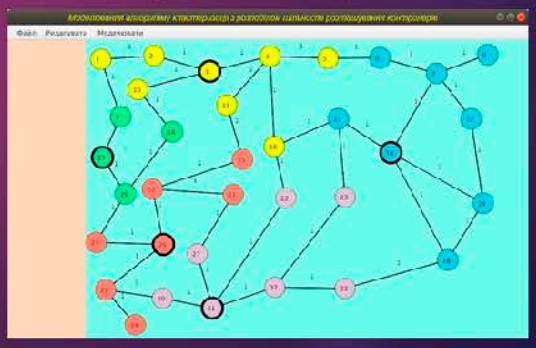
Кластер S5 складається з вершин {V22, V23, V27, V30, V31, V32, V33}.

Результати обчислення параметрів моделювання



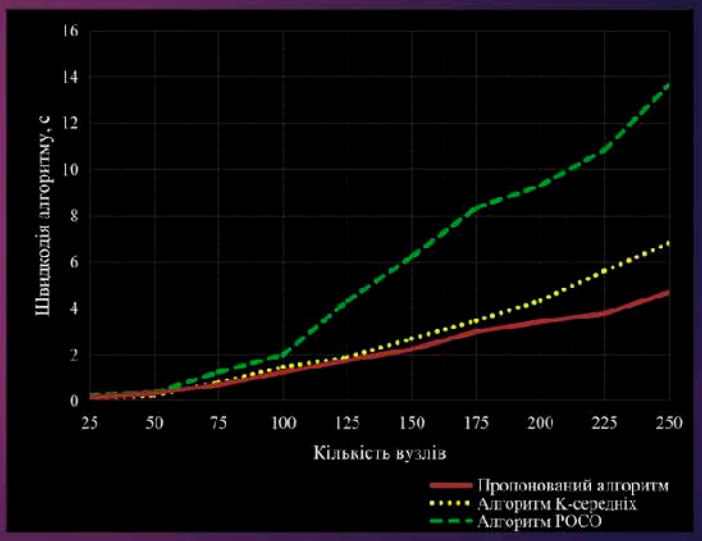
№	p	dc
1	4	1
2	8	1
3	8	1
4	10	3
5	6	1
6	6	1
7	8	1
8	4	1
9	4	1
10	5	1
11	6	1
12	6	1
13	5	1
14	5	1
15	5	1
16	9	1
17	8	1
18	9	3
19	6	2
20	7	1
21	5	1
22	7	1
23	6	1
24	6	1
25	6	1
26	8	3
27	6	1
28	7	1
29	6	1
30	7	1
31	9	3
32	8	1
33	6	1
34	3	1

Результати маршрутизації у імітованій мережі



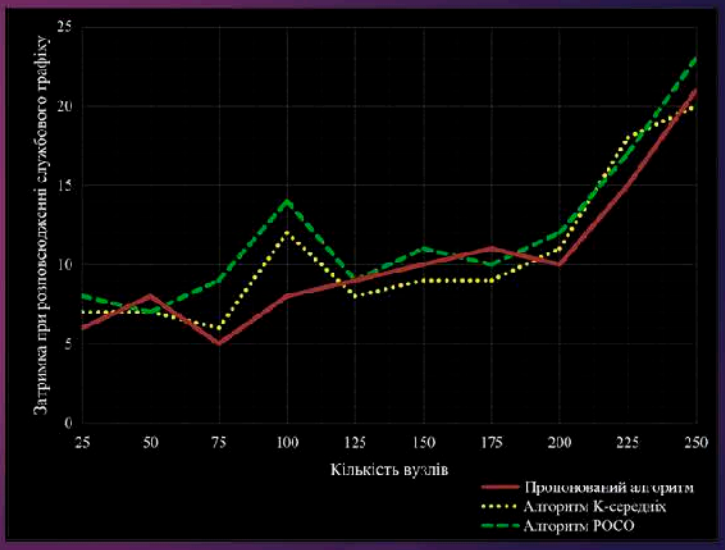
Залежність продуктивності різних алгоритмів маршрутизації від кількості вузлів мережі

	Алгоритм моделювання		
Кількість вершин	***Розроблений***	***k-середніх***	***POCO***
25	0,145	0,121	0,213
50	0,345	0,249	0,346
75	0,687	0,784	1,234
100	1,237	1,456	1,983
125	1,751	1,876	4,305
150	2,221	2,687	6,217
175	2,987	3,456	8,329
200	3,415	4,324	9,299
225	3,769	5,607	10,845
250	4,681	6,809	13,657



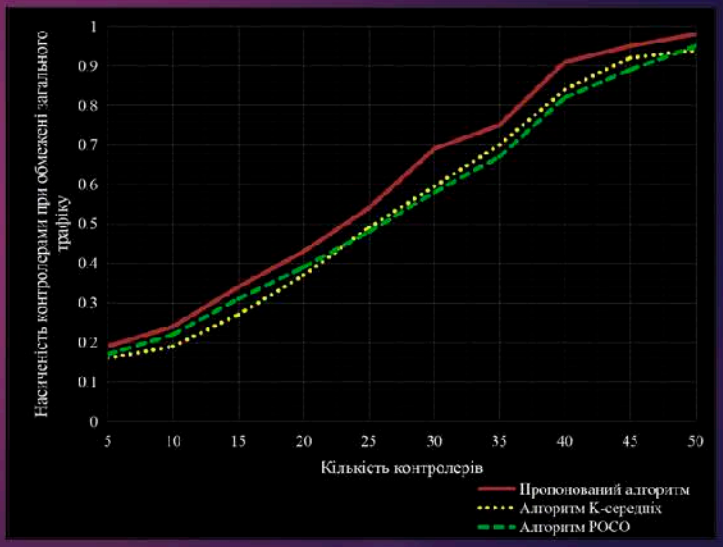
Залежність затримки розподілу сервісного трафіку від кількості вузлів мережі

	Алгоритм моделювання		
Кількість вершин	***Розроблений***	***k-середніх***	***POCO***
25	6	7	8
50	8	7	7
75	5	6	9
100	8	12	14
125	9	8	9
150	10	9	11
175	11	9	10
200	10	11	12
225	15	18	17
250	21	20	23



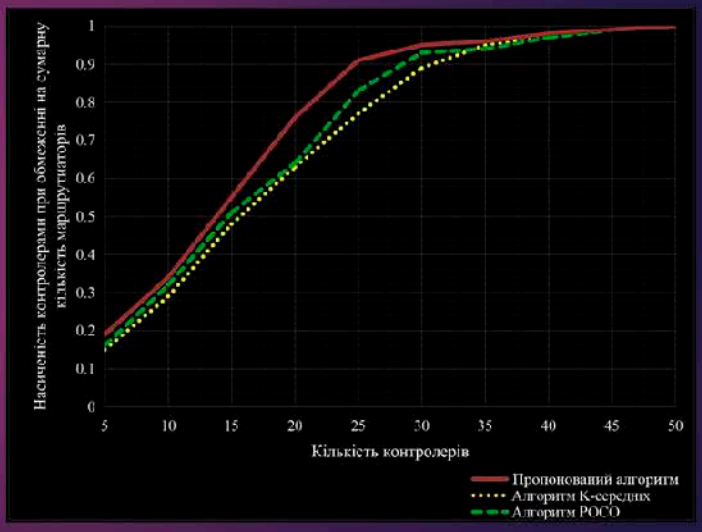
Насичення мережі контролерами при обмеженні трафіку маршрутизаторів мережі

	Алгоритм моделювання		
Кількість контролерів	*Розроблений*	*k-середніх*	*POCO*
5	0,19	0,16	0,17
10	0,24	0,19	0,22
15	0,34	0,27	0,31
20	0,43	0,37	0,39
25	0,54	0,49	0,48
30	0,69	0,595	0,58
35	0,75	0,7	0,67
40	0,91	0,84	0,82
45	0,95	0,92	0,89
50	0,98	0,94	0,95



Насичення топології контролерами при обмеженні на сумарну кількість маршрутизаторів

	Алгоритм моделювання		
Кількість вершин	*Розроблений*	*k-середніх*	*POCO*
5	0,19	0,15	0,16
10	0,34	0,29	0,32
15	0,55	0,48	0,51
20	0,76	0,63	0,64
25	0,91	0,77	0,83
30	0,95	0,89	0,93
35	0,96	0,95	0,94
40	0,98	0,97	0,97
45	0,99	0,99	0,99
50	1	1	1



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на кваліфікаційну роботу здобувача (здобувачки) освіти

відділення комп'ютерних систем

Мілєва Микити Івановича

(прізвище, ім'я та по батькові)

Спеціальність 123 Комп'ютерна інженерія

Освітня програма Комп'ютерна інженерія

Керівник кваліфікаційної роботи Кривченко Юрій Вікторович

(прізвище, ім'я та по батькові)

Тема кваліфікаційної роботи *Дослідження алгоритмів маршрутизації у програмно-визначуваних мережах*

Обсяг розрахунково-пояснювальної записки 68 сторінок

Обсяг графічної (презентаційної) частини 17 аркушів (слайдів)

ХАРАКТЕРИСТИКА КВАЛІФІКАЦІЙНОЇ РОБОТИ

а) заключення про ступінь відповідності виконаного кваліфікаційної роботи завданню

Представлена на рецензію випускна кваліфікаційна робота відповідає затвердженій темі та виконаний відповідно до технічного завдання. Випускна робота має актуальну тематику проблеми маршрутизації, кластеризації та розташування контролерів програмно-визначуваних мереж.

б) характеристика виконання кожного розділу кваліфікаційної роботи

Пояснювальна записка складається з технологічного розділу, розділу охорони праці та додатку. Технологічний розділ пояснювальної записки містить підрозділи, що поетапно охоплюють аналітичну частину, реалізацію суті роботи, дослідження ефективності прийнятих рішень. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора ЕОТ

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

Графічна частина складається з 17 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять креслення та ілюстративні схеми, таблиці, графіки, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів. Якість виконання графічної частини роботи та пояснювальної записки висока, розробку виконано у повному обсязі

г) перелік позитивних якостей кваліфікаційної роботи

Запропонований метод демонструє більш високу ефективність на різних мережевих топологіях у порівнянні з існуючими алгоритмами POCO та k-середніх за показниками затримки і швидкості руху службового трафіку.

д) основні недоліки кваліфікаційної роботи

У розділі охорони праці наведені відомі нормативні вимоги загального плану замість конкретних розрахунків освітлення приміщення, вентиляції, рівня шуму.

Немає обгрунтування чому було обрано саме інтерфейс – OpenFlow.

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента Стайкуца Сергій Володимирович

Місце роботи і посада рецензента Державний університет інтелектуальних технологій і зв'язку, к.ф.н., доцент кафедри КБ та ТЗІ, пом.декану факультету інформаційних технологій та кібербезпеки

Підпис:

« 16 » червня 2023 р.

ПІДПИС ПОСВІДЧУЮ
НАЧАЛЬНИК ВІДДІЛУ
КАДРІВ ДУІТЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ * ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ'ЯЗКУ * 43997335 * ВІДДІЛ КАДРІВ

ВСП «Одеський технічний фаховий коледж ОНТУ»

ВІДГУК

керівника про випускну роботу бакалавра

Мілєва Микити Івановича

(прізвище, ім`я та по батькові)

Спеціальність 123 "Комп'ютерна інженерія"

Тема випускної роботи Дослідження алгоритмів маршрутизації у програмно-визначуваних мережах

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (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 68 сторінок. У пояснювальній записці розглянуто проблему маршрутизації, кластеризації та розташування контролерів програмно-визначуваних мереж. Графічна частина складається з 17 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 Мілєв М.І. 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

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

Вважаю, що теоретична підготовка здобувача освіти добра 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 Мілєв 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 Java JRE 8 та Java SDK 8, Microsoft PowerPoint, Visio, Corel Draw та ін.

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Відмінно

Загальна оцінка Відмінно

Прізвище, ім`я, по батькові Кривченко Юрій Вікторович

Місце роботи і посада керівника роботи ВСП "Одеський технічний фаховий коледж ОНТУ", викладач кафедри комп'ютерної інженерії

Підпис

«12» червня 2023 р.

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

Ми, що нижче підписалися,

Мілєв Микита Іванович,
здобувач освіти гр. 2БКС-27, та

Кривченко Юрій Вікторович,
керівник дипломного проекту,

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

«Дослідження алгоритмів маршрутизації у програмно-визначуваних мережах» (автор роботи – Мілєв М.І., керівник роботи – Кривченко Ю.В.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2023 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 _____ / Мілєв М.І. /

Керівник _____ / Кривченко Ю.В./

«15» червня 2023 р.

UNICHECK

Ім'я користувача:
Наталія Вікторівна Копусь

ID перевірки:
1015241984

Дата перевірки:
25.05.2023 10:41:16 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
25.05.2023 10:42:19 EEST

ID користувача:
100011688

Назва документа: 2БКС-27 Мілєв М.І.

Кількість сторінок: 56 Кількість слів: 8831 Кількість символів: 67240 Розмір файлу: 7.93 MB ID файлу: 1014917967

19.6%
Схожість

Найбільша схожість: 11.3% з Інтернет-джерелом (https://ela.kpi.ua/bitstream/123456789/34430/3/Vasylenko_bakalavr.p..

19.6% Джерела з Інтернету | 406 ..... Сторінка 58

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 58