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

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____ Віктор ГНАТЮК
“_____”_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР

Тема: «Ефективний алгоритм оптимізації трафіку в програмно-конфігурованих мережах»

Виконавець: _____ Сергій ОТРОШКО
(підпис)

Керівник: _____ Олексій ЗУЄВ
(підпис)

Консультанти з окремих розділів пояснювальної записки:

Консультант розділу «Охорона праці» _____ Батир ХАЛМУРАДОВ
(підпис)

Консультант розділу «Охорона навколишнього середовища»
_____ Андріан ЯВНЮК
(підпис)

Нормоконтролер: _____ Денис БАХТІЯРОВ
(підпис)

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

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

ЗАТВЕРДЖУЮ
Завідувач кафедри

Віктор ГНАТЮК
“_____”_____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи

Отрошка Сергія Віталійовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема кваліфікаційної роботи: «Ефективний алгоритм оптимізації трафіку в програмно-конфігурованих мережах»
затверджена наказом ректора від «28» вересня 2023 р. №1965/ст
2. Термін виконання роботи: з 02.10.2023 р. по 31.12.2023 р.
3. Вихідні дані до роботи: програмно-конфігурована мережа
4. Зміст пояснювальної записки: аналіз проблем побудови маршрутів у програмно-конфігурованих мережах; вибір інструментальних засобів та реалізація алгоритму, емуляції складного сегмента програмно-конфігурованої мережі; тестування та апробація алгоритму оптимізації трафіку в програмно-конфігурованих мережах
5. Перелік обов'язкового графічного (ілюстративного) матеріалу: лістинг програмного коду алгоритму оптимізації трафіку в програмно-конфігурованих мережах

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Розробити деталізований зміст розділів кваліфікаційної роботи	02.10.2023-04.10.2023	Виконано
2	Вступ	05.10.2023-08.10.2023	Виконано
3	Аналіз проблем побудови маршрутів у програмно-конфігурованих мережах	09.10.2023-22.10.2023	Виконано
4	Вибір інструментальних засобів та реалізація алгоритму, емуляції складного сегмента програмно-конфігурованої мережі	23.10.2023-05.11.2023	Виконано
5	Тестування та апробація алгоритму оптимізації трафіку в програмно-конфігурованих мережах	06.11.2023-30.11.2023	Виконано
6	Охорона праці	01.12.2023-06.12.2023	Виконано
7	Охорона навколишнього середовища	07.12.2023-17.12.2023	Виконано
8	Усунення недоліків та захист кваліфікаційної роботи	18.12.2023-31.12.2023	Виконано

7. Консультанти з окремих розділів

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв
Охорона праці	к.м.н., професор Батир ХАЛМУРАДОВ		
Охорона навколишнього середовища	к.б.н., доц. Андріан ЯВНЮК		

8. Дата видачі завдання: “29" вересня 2023 р.

Керівник кваліфікаційної роботи _____ Олексій ЗУЄВ
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Сергій ОТРОШКО
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Кваліфікаційна робота «Ефективний алгоритм оптимізації трафіку в програмно-конфігурованих мережах» містить 82 сторінки, 32 рисунки, 2 таблиці, 60 використаних джерел.

ПРОГРАМНО-КОНФІГУРОВАНІ МЕРЕЖІ (SDN). ОПТИМІЗАЦІЯ ТРАФІКУ. QOS (QUALITY OF SERVICE). ТРАФІК ІНЖЕНЕРІЯ. ТРАФІК-ІНЖЕНЕРНИЙ АЛГОРИТМ. SD-WAN (SOFTWARE-DEFINED WIDE AREA NETWORK). МАРШРУТИЗАЦІЯ. МЕРЕЖЕВИЙ АНАЛІЗ. ПРОТОКОЛИ МАРШРУТИЗАЦІЇ (НАПРИКЛАД, OSPF, BGP). ТУНЕЛЮВАННЯ МЕРЕЖЕВОГО ТРАФІКУ. ВІРТУАЛЬНІ МЕРЕЖІ (VPN). МЕРЕЖЕВИЙ ТРАФІК. ВІРТУАЛЬНІ МЕРЕЖЕВІ ФУНКЦІЇ (VNF). ІНФРАСТРУКТУРА ЯК КОД. ТРАФІКОВА ІНТЕЛІГЕНЦІЯ. АНАЛІЗ ПРОПУСКНОЇ ЗДАТНОСТІ. УПРАВЛІННЯ МЕРЕЖЕЮ. ВІДМІНЕНІ СЛУЖБИ (DIFFERENTIATED SERVICES). АРХІТЕКТУРА МЕРЕЖІ.

Метою кваліфікаційної роботи є дослідження алгоритмів побудови маршрутів, розробка та оптимізація їхньої швидкості виконання для програмно-конфігурованих мереж.

Об’єктом дослідження є самі програмно-конфігуровані мережі (SDN) та їхні складові частини, які включають апаратне забезпечення, програмне забезпечення, маршрутизатори, комутатори, контролери, а також весь мережевий трафік, що протікає через ці мережі.

Предметом дослідження є конкретні аспекти оптимізації трафіку в програмно-конфігурованих мережах.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

SDN - Software-Defined Networking (програмно-керовані мережі).

SD-WAN - Software-Defined Wide Area Network (програмно-керована широкосмугова мережа).

QoS - Quality of Service (якість обслуговування).

IoT - Internet of Things (інтернет речей).

MPLS - Multiprotocol Label Switching (багатопротокольне маркування пакетів).

DPI - Deep Packet Inspection (глибока інспекція пакетів).

BGP - Border Gateway Protocol (протокол маршрутизації за кордонами мережі).

SLA - Service Level Agreement (угода про рівень обслуговування).

WAN - Wide Area Network (широкосмугова мережа).

LAN - Local Area Network (локальна мережа).

OSPF - Open Shortest Path First (протокол відкритого найкоротшого шляху).

VPN - Virtual Private Network (віртуальна приватна мережа).

NMS - Network Management System (система управління мережею).

PCE - Path Computation Element (елемент обчислення маршруту).

VNF - Virtual Network Function (віртуальна функція мережі).

SFC - Service Function Chaining (ланцюжок функцій обслуговування).

GRE - Generic Routing Encapsulation (загальне упаковування маршрутизації).

DSCP - Differentiated Services Code Point (кодова точка відмінених служб).

OPEX - Operating Expenses (операційні витрати).

CAPEX - Capital Expenses (капітальні витрати).

ACL - Access Control List (список керування доступом).

BFD - Bidirectional Forwarding Detection (виявлення двостороннього пересилання).

CAC - Call Admission Control (контроль допуску до викликів).

VRRP - Virtual Router Redundancy Protocol (протокол віртуальної резервності маршрутизатора).

DDoS - Distributed Denial of Service (розподілене заперечення обслуговування).

VM - Virtual Machine (віртуальна машина).

NFV - Network Functions Virtualization (віртуалізація функцій мережі).

CDN - Content Delivery Network (мережа доставки контенту).

NAT - Network Address Translation (переклад мережевих адрес).

PaaS - Platform as a Service (платформа як сервіс).

SaaS - Software as a Service (програмне забезпечення як сервіс).

VMWare - Virtual Machine Software (програмне забезпечення віртуальної машини).

API - Application Programming Interface (інтерфейс програмування додатків).

GUI - Graphical User Interface (графічний інтерфейс користувача).

ROI - Return on Investment (рентабельність інвестицій).

SDR - Software-Defined Radio (програмно-керований радіо).

VSDN - Virtual Software-Defined Network (віртуальна програмно-керована мережа).

VNF Orchestration - Virtual Network Function Orchestration (оркестрація віртуальних функцій мережі).

NaaS - Network as a Service (мережа як сервіс).

SLA Management - Service Level Agreement Management (управління угодами про рівень обслуговування).

ВСТУП

У XXI столітті постійне зростання підключених пристроїв до мережі «Інтернет» спричинив експоненціальне зростання загальносвітового трафіку. Цей різкий стрибок стимулював розвиток нових швидших та інноваційних комп'ютерних мереж під назвою "Програмно-конфігуровані мережі". Нове покоління комунікаційних мереж мало важливу перевагу у вигляді збільшеної пропускної спроможності кожного передавального пристрою, проте головним мінусом даних мереж став час створення нового з'єднання, він збільшився в більш ніж два рази.

Проблеми викликані складністю побудови маршрутів на великих сегментах підконтрольних мереж, у яких кількість підключених передавальних пристроїв понад 300 штук.

Теорія побудови найкоротших шляхів на графах має величезний вплив на сучасний світ. Завдяки своєму широкому спектру можливостей, що надаються, вона останнім часом інтенсивно розвивається: разом із поліпшенням уже розроблених методів винаходять принципово нові.

Пошук шляхів є важливим завданням, яке використовують у різних галузях і сферах. Правильно розрахований маршрут може заощадити кошти на одиничну або групову доставку, великий обсяг даних буде швидше доставлений користувачеві, штучний інтелект у комп'ютерних іграх покаже більш цікаву поведінку, а маршрут для вашого автомобіля буде побудований ефективно з точки зору витраченого часу на шлях.

Проблемою пошуку шляхів почали займатися ще в XIX столітті. Тоді з'явилася перша задача з пошуку шляхів - задача Комівояжера. Сьогодні ця задача є класичною для комбінаторики та пошуку маршрутів.

Починаючи з 1969 року комп'ютерні мережі почали розвиватися. Спочатку в найпершій мережі під назвою ARPANET було з'єднано 4 суперкомп'ютери того часу.

Завдяки цій мережі було випробувано маршрутизацію пакетів за допомогою протоколу IP, який використовується і донині. Починаючи з 1990 року з'явилася всесвітня павутина, яка з'єднала між собою найвіддаленіші куточки світу.

Починаючи з 1990 року складність комп'ютерних мереж зростає в геометричній прогресії, стаючи дедалі об'ємнішими та заплутанішими. У сучасних комп'ютерних мережах пошук шляху здійснюється на два кроки вперед, тому що повний пошук шляху може зайняти величезну кількість часу, однак такий "швидкий" пошук не дає змоги завжди побудувати найоптимальніший і найшвидший маршрут. Через збільшену складність сучасних мереж їм на зміну розробляють програмно-конфігуровані мережі, суть, яких полягає в тому, щоб винести логіку побудови маршрутів з передавальних пристроїв і залишити пристроям тільки саму передачу даних. Для того, щоб це реалізувати, необхідно використовувати швидкий і ефективний алгоритм побудови маршруту, який буде враховувати навантаження на кожен окремий пристрій, щоб домогтися максимальної продуктивності мережі.

В сучасному цифровому світі, який характеризується безмежним обсягом даних і зростаючими вимогами до якості обслуговування, програмно-конфігуровані мережі (SDN) стали ключовим елементом для управління і оптимізації мережевого трафіку. Стрімка еволюція технологій зв'язку та зростаючі вимоги користувачів змусили організації вдосконалювати способи керування мережами, щоб забезпечити ефективну і надійну передачу даних.

Трафік в мережах розмаїтий і динамічний, і вирішення проблеми ефективного керування ним є завданням важливим для підтримання якості обслуговування та ефективності мережевих інфраструктур. Оптимізація трафіку в програмно-конфігурованих мережах вимагає розробки та впровадження спеціалізованих алгоритмів та інструментів, які дозволять підтримувати стабільність мережі, забезпечувати високу пропускну здатність та гарантувати якість обслуговування в умовах змінних потреб користувачів [1-45].

Актуальність розробки ефективного алгоритму оптимізації трафіку для програмно-конфігурованих мереж зумовлена небажанням частини гравців IT-ринку переходити на нове покоління мереж у зв'язку з великими затримками при створенні

з'єднань. Тому зараз дуже важливо розробити швидкий і ефективний алгоритм для побудови маршруту, за допомогою якого затримки при створенні нового з'єднання істотно знизяться.

Ця кваліфікаційна робота спрямована на дослідження та розробку ефективного алгоритму оптимізації трафіку в програмно-конфігурованих мережах з метою забезпечення оптимальної роботи мережі та вдосконалення якості обслуговування. Ми будемо досліджувати ключові концепції, методи та інструменти, пов'язані з оптимізацією трафіку, і будемо розглядати їх застосування в реальних сценаріях.

Метою кваліфікаційної роботи є дослідження алгоритмів побудови маршрутів, розробка та оптимізація їхньої швидкості виконання для програмно-конфігурованих мереж.

Під час роботи було поставлено такі завдання:

- Аналіз проблем побудови маршрутів у програмно-конфігурованих мережах;
- Вибір інструментальних засобів для реалізації алгоритму, емуляції та тестування реалізованого алгоритму на базі програмно-конфігурованих мереж;
- Розробка і реалізація алгоритму, тестування на емульованому сегменті програмно-конфігурованої мережі;
- Апробація алгоритму;

Об'єктом дослідження є самі програмно-конфігуровані мережі (SDN) та їхні складові частини, які включають апаратне забезпечення, програмне забезпечення, маршрутизатори, комутатори, контролери, а також весь мережевий трафік, що протікає через ці мережі.

Предметом дослідження є конкретні аспекти оптимізації трафіку в програмно-конфігурованих мережах, що включають в себе такі ключові елементи:

- Алгоритми оптимізації трафіку. Дослідження алгоритмів, які дозволяють вибирати оптимальні шляхи для передачі даних в програмно-конфігурованих мережах з метою забезпечення ефективності і якості обслуговування.

- Керування ресурсами мережі. Вивчення способів ефективного керування ресурсами, такими як пропускна здатність та мережеві вузли, для оптимізації трафіку.
- Якість обслуговування (QoS). Аналіз методів і механізмів, які дозволяють гарантувати високу якість обслуговування користувачам при передачі даних.
- Методи балансування навантаження. Розгляд можливостей балансування навантаження між мережевими шляхами та вузлами для підвищення ефективності мережі.
- Аналіз даних трафіку. Дослідження методів аналізу реальних даних трафіку для виявлення паттернів та вдосконалення стратегій оптимізації.
- Засоби моніторингу та керування мережею. Розробка і впровадження інструментів для моніторингу та керування мережею з метою реагування на зміни в трафіку.

Дослідження ефективного алгоритму оптимізації трафіку в програмно-конфігурованих мережах може використовувати різноманітні **методи та підходи** для аналізу, моделювання і розробки алгоритмів:

- Аналіз літератури. Перший етап дослідження полягає в аналізі наявної літератури. Дослідники вивчають публікації, наукові статті, книги та інші джерела, щоб отримати розуміння сучасного стану галузі, ідентифікувати проблеми та визначити можливі напрями дослідження.
- Емпіричні дослідження. Включають в себе збір даних з реальних програмно-конфігурованих мереж або симуляційних середовищ для аналізу трафіку та вимірювання результатів застосування алгоритмів оптимізації. Цей підхід дозволяє оцінити ефективність розроблених алгоритмів на практиці.
- Математичне моделювання. Розробка математичних моделей для опису програмно-конфігурованих мереж і трафіку в них. Моделювання може включати рівняння, графи, стохастичні моделі і інші математичні інструменти для аналізу і оптимізації мережі.

- Симуляції. Використання програмних інструментів для створення симуляційних середовищ, де можна вивчати поведінку мережі під різними умовами і тестувати алгоритми оптимізації.
- Експерименти. Проведення практичних експериментів у реальних мережах або лабораторних умовах для тестування та вдосконалення розроблених алгоритмів.
- Моделювання пропускної здатності. Розрахунки і аналіз пропускної здатності мережі для визначення оптимального розміру та параметрів ресурсів.
- Аналіз даних трафіку. Використання методів статистики та аналізу даних для ідентифікації паттернів трафіку та виявлення можливостей оптимізації.
- Розробка алгоритмів і програмного забезпечення. Розробка нових алгоритмів оптимізації та програмного забезпечення для їхньої реалізації.

Наукова новизна та практичне значення отриманих результатів.

Наукова новизна отриманих результатів:

- Набув подальшого розвитку алгоритм для оптимізації трафіку в програмно-конфігурованих мережах, який покращує якість обслуговування або зменшує ресурси, необхідні для обробки трафіку.

Практичне значення одержаних результатів дослідження полягає в наступному:

- Покращення ефективності мереж. Результати дослідження сприяють покращенню ефективності програмно-конфігурованих мереж шляхом зменшення затримок, покращення використання пропускної здатності та ефективного розподілу ресурсів мережі.
- Підвищення якості обслуговування (QoS). Алгоритми оптимізації трафіку забезпечують вищий рівень якості обслуговування для користувачів мережі, що особливо важливо для послуг, які вимагають мінімальних затримок і стабільності.

- Зниження витрат. Покращення оптимізації трафіку призводить до зниження витрат на ресурси, такі як електроенергія, обладнання та бандвідшвидкість, що може бути важливим для бізнесу та організацій.

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

- Науково-практична конференція «Проблеми експлуатації та захисту інформаційно-комунікаційних систем», м. Київ, 2023 р.

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМ ПОБУДОВИ МАРШРУТІВ У ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖАХ

1.1. Актуальність проблеми побудови маршрутів у програмно-конфігурованих мережах

Проблема затримки перед створенням з'єднання для програмно-конфігурованих мереж є найголовнішим фактором, який зупиняє їхнє розповсюдження по всьому світу. Пов'язано це з тим, що сервер, який називається програмно-конфігуруємим контролером, здійснює прокладення маршруту в складних сегментах мережі, в яких кількість вузлових пристроїв на поточний момент може доходити до 1000 штук. Ціна таких прокладок набагато вища, ніж у класичних мережах, у них середній час прокладання маршруту в складному сегменті мережі займатиме приблизно 14-20 мікросекунд, а в програмно-конфігурованих 200 і більше мікросекунд. Для розв'язання цієї проблеми необхідне розроблення нового алгоритму [1].

1.2. Загальні відомості про програмно-конфігуровані мережі

Програмно-конфігуровані мережі - це нове покоління комп'ютерних мереж, у яких ключовою відмінністю є винесення логіки маршрутизації за межі пристрою на окремий виділений сервер. Поява цього покоління пов'язана з експоненціальним зростанням трафіку і падінням ефективності поточних комп'ютерних мереж [2].

У ПКМ рівні управління мережею і передавання даних розділяють за допомогою винесення функцій управління (комутаторами, маршрутизаторами тощо) у додатки, які працюють на виділеному сервері, так званому контролері. Перші концепції таких мереж були сформульовані фахівцями університетів Стенфорда і Берклі ще в 2006 році, а проведені ними дослідження отримали схвалення не тільки в наукових

колах, а й у сфері бізнесу. Ці ідеї тепло зустріли такі виробники мережевого обладнання, як, як Cisco, HP та інші. [20] У березні 2011 року було засновано консорціум Open Networking Foundation (ONF). Засновниками цієї спільноти є низка великих і впливових компаній у сфері IT. Цими компаніями були: Verizon, Yahoo, Microsoft, Google, Deutsche Telekom, і Facebook. Склад ONF швидко поповнився й іншими не менш відомими компаніями, як-от Marvell, Citrix, IBM, NEC, Brocade, Oracle, HP, Dell, Ericsson, і низка інших. Найперша практична реалізація ПКМ була запропонована компанією Nicira, яка нещодавно стала частиною VMware [3].

Інтерес IT-компаній до ПКМ викликаний тим, що така технологія дає змогу підвищити ефективність мережевого обладнання на 25-30%, знизити на 30% витрати на експлуатацію мереж, дає змогу перетворити управління мережами з творчого проєктування на інженерію, підвищити захищеність мережі та дати змогу користувачам самостійно за допомогою програм створювати нові сервіси, оперативно завантажувати й використовувати їх на мережевому обладнанні.

У більшості напрацювань і досліджень ключові моменти в ПКМ пов'язані з розроблюваною в США програмою Global Environment for Network Innovations (GENI) дослідження майбутнього Інтернету, що включає в себе близько 40 провідних університетів США. Діяльність об'єднаного центру Стенфорда і Берклі, що здійснює різні вивчення, дослідження, експерименти і напрацювання в галузі Internet2; а також із Сьомою рамковою програмою досліджень Європейського Союзу Ofelia і проєктом FEDERICA [4].

Основні концепції ПКМ:

- Винесення процесу управління даних із передавального пристрою на виділений сервер, а процеси передавання даних залишити на передавальних пристроях;
- Уніфікований і незалежний інтерфейс між рівнем керування та рівнем передавання даних;
- логічно централізоване управління мережею, здійснюване за допомогою контролера зі встановленою мережевою операційною системою та реалізованими поверх мережевими додатками;

- віртуалізація фізичних ресурсів мережі;

Головна проблема поточних комп'ютерних мереж полягає в тому, що приблизно 20-25% передавальний пристрій витрачає на прокладання маршруту сегментом мережі. Концепція програмно-конфігурованих мереж полягає в тому, щоб прибрати логіку побудови маршрутів із пристрою передавання даних і залишити йому тільки передавання даних. Новий тип мереж пропонує розміщувати логіку передавання даних на спеціальних серверах, званих програмно-конфігурованими контролерами, на яких відбувається обчислення маршруту в межах сегмента мережі. Ці мережі насамперед націлені на центри обробки даних, адже саме вони відповідають за великий медіаконтент, що надається на різних ресурсах [5].

За рахунок централізованості управління сегментом мережі ця мережа пропонує низку серйозних переваг:

- Зниження вартості обладнання, за рахунок спрощення обладнання;
- Більш детальне управління мережею;
- Дуже низька ймовірність несанкціонованого доступу до ресурсів мережі;
- Можливість розробки або доопрацювання наявних інструментів для програмно-конфігурованих мереж для управління ресурсами мереж і потоками даних.

У рамках розвитку концепції цієї технології було розроблено спеціальний протокол передачі даних OpenFlow. Цей протокол базується на концепції управління обробки потоків даних. Принцип роботи протоколу простий, якщо комутатор отримує дані для певного потоку, він дивиться в спеціальні таблиці потоків, іменовані як flow tables, в яких зазначено дії, які необхідно вчинити в разі отримання цього потоку, якщо він є новим, то тоді комутатор запитує інформацію в контролера, надалі контролер створює новий маршрут і заносить його в таблиці потоків пристроїв, які беруть участь у процесі передачі даних цього потоку [6].

Головною проблемою цього підходу є величезна кількість інформації про просування пакетів, що називається forwarding state explosion. Через це навантаження на контролер є величезним і величезні сегменти мережі, що включають в себе більше

1000 передавальних пристроїв, є непосильною ношею з точки зору затримок і стабільності передавальної інфраструктури [7].

Архітектура програмно-конфігурованих мереж. В архітектурі ПКМ (software-defined network) є три рівні управління мережею (Рис. 1.1.):

1. інфраструктурний рівень, що надає набір мережевих пристроїв, як-от комутаторів і каналів передавання даних;

2. рівень управління, що містить мережеву операційну систему, яка забезпечує додаткам мережеві сервіси та програмний інтерфейс для управління мережевими пристроями і мережею;

3. рівень мережевих додатків для гнучкого та ефективного управління мережею;

Перший пункт дає змогу швидко змінювати та модифікувати інфраструктуру мережі без повної перебудови всього сегмента мережі.

Другий пункт вирішує проблему недостатньо ефективного використання ресурсів передавальних пристроїв для передавання даних.

Третій пункт дає широкі можливості для управління доступом до мережі, управління передачею даних, модифікації генерації маршрутів та інших інших можливостей [8].

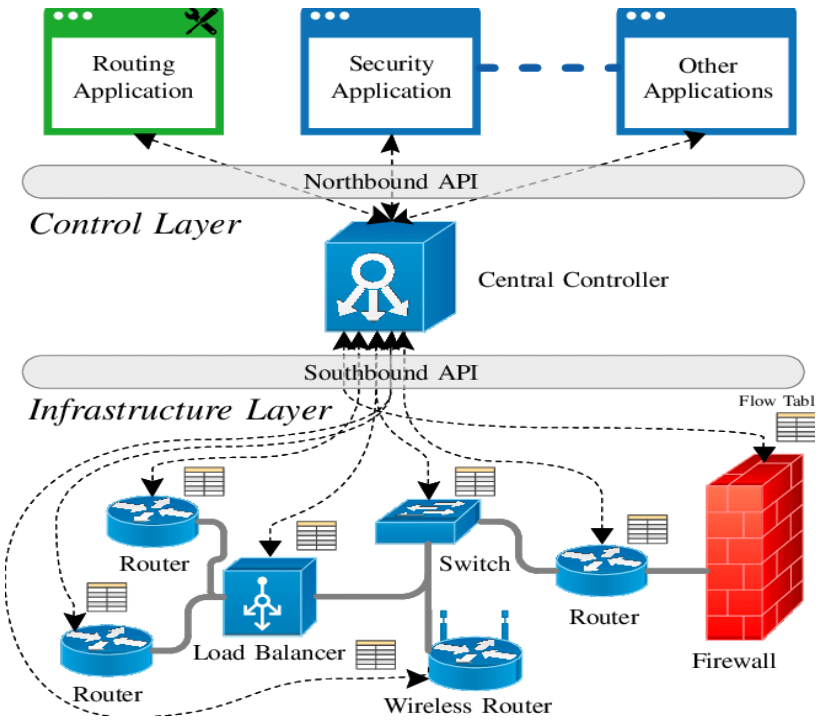

Рис. 1.1. Архітектура програмно-конфігурованих мереж

Найперспективнішим еталоном для мереж з програмно-конфігурованою структурою та еталоном, що швидко розвивається, є ПКМ є протокол OpenFlow. Це відкритий протокол взаємодії, розроблений спеціально для ПКМ, у якому описуються вимоги, що висуваються до комутатора, який підтримує протокол OpenFlow для віддаленого управління [9].

За допомогою сучасних маршрутизаторів повсюдно виконуються дві головні задачі: передача даних - передача пакета з вхідного порту на певний вихідний порт і управління даними - обробка пакета й ухвалення рішення про те, куди його передавати далі, на основі поточної завантаженості маршрутизатора та інших чинників [10]. Це відображає рівень передавання даних, на якому об'єднано засоби передавання, різноманітні лінії зв'язку, каналоутворювальне обладнання, маршрутизатори, комутатори, та рівень управління станами засобів передавання даних (Рис. 1.2.). Розвиток маршрутизаторів по сьогоднішній день рухався в напрямі об'єднання цих рівнів, однак зі ставкою на передачу, включаючи в себе різні апаратні прискорення, удосконалення ПЗ і впровадження нових функціональних можливостей для збільшення швидкості обробки кожного пакета, водночас рівень управління залишався досить примітивним і спирався на непрості розподілені алгоритми маршрутизації та складні інструкції щодо налаштування і конфігурації мережі. Через складність програмного забезпечення маршрутизаторів, у межах яких розроблено рівень управління, компанії розробники пристроїв закривали вихідний код [11].

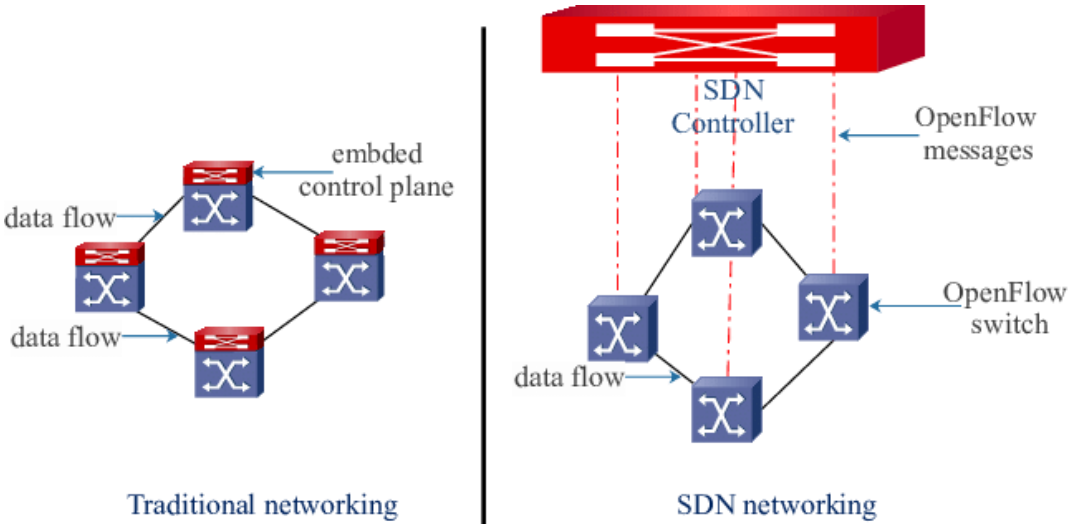

Рис. 1.2. Традиційні мережі та ПКМ

Згідно зі специфікацією 1.3 стандарту OpenFlow, спілкування програмно-конфігурованого контролера з комутатором здійснюється засобами протоколу OpenFlow, в якому будь-який комутатор зобов'язаний містити щонайменше одну таблицю потоків і групову таблицю, а також підтримувати канал OpenFlow для зв'язку з віддаленим контролером, який є центральним сервером. Специфікація не описує внутрішній устрій і архітектуру контролера, а також API для його внутрішніх додатків. Будь-яка таблиця потоків усередині комутатора має містити набір записів з інформацією про потоки або правила маршрутизації. Будь-який із записів складається з полів-ознак, лічильників і набору інструкцій [12-15].

Механізм роботи комутатора OpenFlow є дуже простим. У кожного нового вхідного пакета забирається заголовок, який є бітовим рядком певної довжини. Далі проводиться пошук правила в таблицях потоків, для даної бітової маски. У разі знаходження збігу, над пакетом і його заголовком виконуються різні перетворення, що визначаються деяким набором інструкцій, визначених у знайденому правилі. Інструкції, асоційовані з кожним записом таблиці, описують дії, пов'язані з надсиланням пакета, модифікацією його заголовка, обробкою в таблиці груп, обробкою в конвеєрі та надсиланням пакета на певний порт комутатора. Інструкції конвеєра обробки дають змогу надсилати пакети в подальші таблиці для подальшого опрацювання і у вигляді метаданих передавати інформацію між таблицями. Інструкції також визначають правила модифікації лічильників, які можуть бути використані для збору різноманітної статистики [16].

Якщо потрібне правило не знайдено, то пакет інкапсулюється і надсилається контролеру, той зі свого боку формує необхідне для обробки правило. Сформоване правило для певного типу пакетів завантажується в комутатор або групу підконтрольних комутаторів.

Також невідомий пакет може бути скинутий, при встановленні відповідного налаштування на комутаторі [17].

Запис про потік може повідомляти про відправлення пакета в певний фізичний, зарезервований або віртуальний порт [18]. Зарезервовані віртуальні порти можуть ви-

значати загальні дії пересилання, як-от: надсилання контролеру, широкомовне розсилання, або ж пересилання без використання протоколу OpenFlow. Віртуальні порти, визначені комутатором, можуть точно визначати групи об'єднання каналів, тунелі або інтерфейси зі зворотним зв'язком [19].

Записи про потоки можуть також зберігати інформацію про групи, в яких є додаткова обробка. Групи є набором дій для широкомовної розсилки. Також набори дій можуть являти собою пересилання зі складною семантикою, наприклад, агрегування каналів або зміна маршруту. Механізм груп дає змогу ефективно змінювати однакові вихідні дії для потоків. Таблиця груп зберігає в собі записи про групи, які, своєю чергою, зберігають список контейнерів дій з особливою семантикою, що залежить від типу групи. Дії в одному або декількох контейнерах дій застосовуються до пакетів, що надсилаються в групу [20].

Розробники комутаторів можуть вільно реалізовувати будь-яку внутрішню начинку, однак процедура перегляду пакетів і семантика інструкцій мають бути спільними для всіх. Наприклад, у той момент як потік може використовувати всі групи для пересилання в деяку безліч портів, розробник комутатора може вибрати для реалізації цього функціоналу єдину бітову маску всередині великої апаратної таблиці маршрутизації. Ще одним прикладом є процедура перегляду таблиць: конвеєр фізично може бути реалізований із використанням деякої кількості апаратних таблиць. Встановлення, оновлення та видалення правил у таблицях потоків комутатора здійснюються тільки контролером. Правила можуть встановлюватися реактивно (у відповідь на пакети, що надійшли) або проактивно (заздалегідь, до приходу пакетів) [21]. Управління даними в OpenFlow здійснюється на рівні їхніх потоків, а не на рівні окремих пакетів. Правило в комутаторі OpenFlow прописується за участю контролера тільки для першого пакета, а всі інші пакети потоку його використовують надалі [22]. Наявні на сьогоднішній день фізичні комутатори ПКМ відповідають поки що специфікації OpenFlow 1.0 і містять тільки одну таблицю потоків.

Протокол OpenFlow. Ідея розробки ПКМ полягає в розробці узагальненого і незалежного мережевого обладнання. Що володіє єдиним інтерфейсом взаємодії між

передавальним середовищем мережі та контролером. Ця концепція є основоположною в протоколі OpenFlow. Він дає змогу користувачам самостійно контролювати і визначати, за яких умов, які вузли і з якою якістю можуть взаємодіяти в Мережі між собою. Протокол підтримує три типи повідомлень: контролер-комутатор, асинхронні та симетричні [23].

Повідомлення типу контролер-комутатор ініціюються контролером і використовуються для керування та відстеження стану комутатора. Повідомлення цього типу можуть використовуватися контролером для завантаження конфігурації комутатора, для вивантаження статистики, для додавання, видалення і модифікації записів у таблицях потоків.

Асинхронні повідомлення ініціюються комутатором для повідомлення контролера про різні мережеві події, наприклад, отримання пакетів, видалення запису з таблиці потоків у зв'язку із закінченням терміну дії тайм-ауту. Також вони можуть повідомляти про різні зміни стану комутатора і помилки в результаті роботи комутатора [24].

Симетричні повідомлення можуть ініціюватися комутатором або контролером без запиту і використовуються під час встановлення з'єднання. Також використовуються для вимірювання пропускної здатності з'єднання контролер-комутатор, затримок або для перевірки стану з'єднання.

Мережеві операційні системи в ПКМ. Логічно-централізоване управління даними в мережі передбачає винесення функцій управління мережею з передавальних пристроїв на окремий сервер, званий програмно-конфігурованим контролером (ПКК), який перебуває під контролем адміністратора сегмента мережі. ПКК може керувати одним або кількома OpenFlow-комутаторами. Також містить у собі спеціальну мережеву операційну систему, що надає мережеві сервіси та уніфіковані інтерфейси з низькорівневого управління мережею, сегментами мережі та моніторингу за станом мережевих елементів, а також додатки, що здійснюють високорівневе управління мережею та потоками даних [25].

Мережева ОС (СОС) є спеціальною обгорткою над низькорівневим протоколом. За допомогою вбудованого API надає додаткам інструменти для доступу до управління мережею, а також виконує моніторинг конфігурації засобів мережі. На відміну від традиційного тлумачення терміна ОС, під СОС розуміють програмну систему, що забезпечує моніторинг, доступ і управління ресурсами всієї мережі, а не її конкретного вузла [26].

Подібно до класичної ОС, СОС забезпечує програмний інтерфейс для додатків управління мережею і реалізує механізми управління таблицями комутаторів: додавання, видалення, модифікацію правил і збір різноманітної статистики [27]. Таким чином, фактично вирішення завдань управління мережею виконується за допомогою програмного забезпечення, розробленого на базі API мережевої операційної системи. Цей інструментарій дає змогу створювати додатки на рівні високорівневих абстракцій, приміром, ім'я хоста та ім'я користувача, а не низькорівневих параметрів конфігурації, приміром, MAC- та IP-адрес. Це дає змогу виконувати керівні команди, незважаючи на базову топологію мережі, однак необхідно, щоб СОС підтримувала відображення між високорівневими абстракціями і низькорівневими конфігураціями.

У кожному контролері зберігається щонайменше один застосунок, для керування під'єднаними комутаторами. Також програмне забезпечення формує модель підконтрольної топології фізичної мережі, завдяки цьому централізується управління [28]. Уявлення топології мережі містить у собі топологію комутаторів, розташування хостів і користувачів та інших елементів і різних сервісів мережі. Подання також містить у собі зв'язок між іменами та адресами, тому одним із найважливіших завдань, які розв'язує СОС, є щосекундний моніторинг мережі. Це дає змогу СОС створювати додатки у вигляді централізованого ПЗ, з використанням високорівневих імен, на базі різних алгоритмів, наприклад, алгоритму Дейкстри пошуку найкоротшого шляху в графі, замість складних розподілених алгоритмів, на кшталт алгоритму Беллмана-Форда, в термінах низькорівневих адрес, які використовуються в сучасних маршрутизаторах [29].

На даний момент є велика кількість реалізацій мережевих ОС для програмно-визначуваних мереж. Одними з найвідоміших на сьогоднішній день є: RUNOS, BigSwitch, Beacon, POX, Maestro, FloodLight, NOX і Trema.

Для контролерів у ПКМ важливою вимогою є те, щоб усі додатки одного контролера в будь-який момент часу повинні бути синхронізовані та мати однакове уявлення про топологію мережі. Однак перехід від розподіленого управління мережею до централізованого таїть у собі низку недоліків. Наприклад, зниження надійності, відмовостійкості та масштабованості [30].

Сьогодні активно розвиваються 3 підходи для побудови розподіленого масштабованого контролера. Вони називаються Kandoo, HyperFlow і Onix. У рамках досліджень ЦПІКС було виявлено, що найперспективнішим є альтернативний підхід, представлений на малюнку Рис. 1.3. З огляду на те, що будь-який контролер може бути з'єднаний із кількома комутаторами, а кожен комутатор може бути з'єднаний кількома контролерами, то виходить, що є можливість об'єднати контролери в груповий контролер (ГК). Група контролерів, об'єднаних у ГК, повинні мати синхронізоване подання підконтрольної топології сегмента мережі. Як видно на рис. 1.3, C1 - C3 - контролери, S1 - S4 - комутатори, а V1 - V3 - фрагменти мережі, до яких забезпечує доступ комутатор S1, S2, S3 відповідно. Тоді ГК1 утворюють контролери C1 і C2, ГК2 - C2 і C3, а всі застосунки в ГК1 повинні мати узгоджене уявлення про топологію V1 і V2, всі застосунки в ГК2 - про топологію V2 і V3. У разі виходу з ладу, наприклад, контролера C1 його може замінити C2, взявши на себе управління V1. Подання про стан відповідної частини мережі контролери можуть узгоджувати або через комутатор S4, або через S1, S2 і S3 [31].

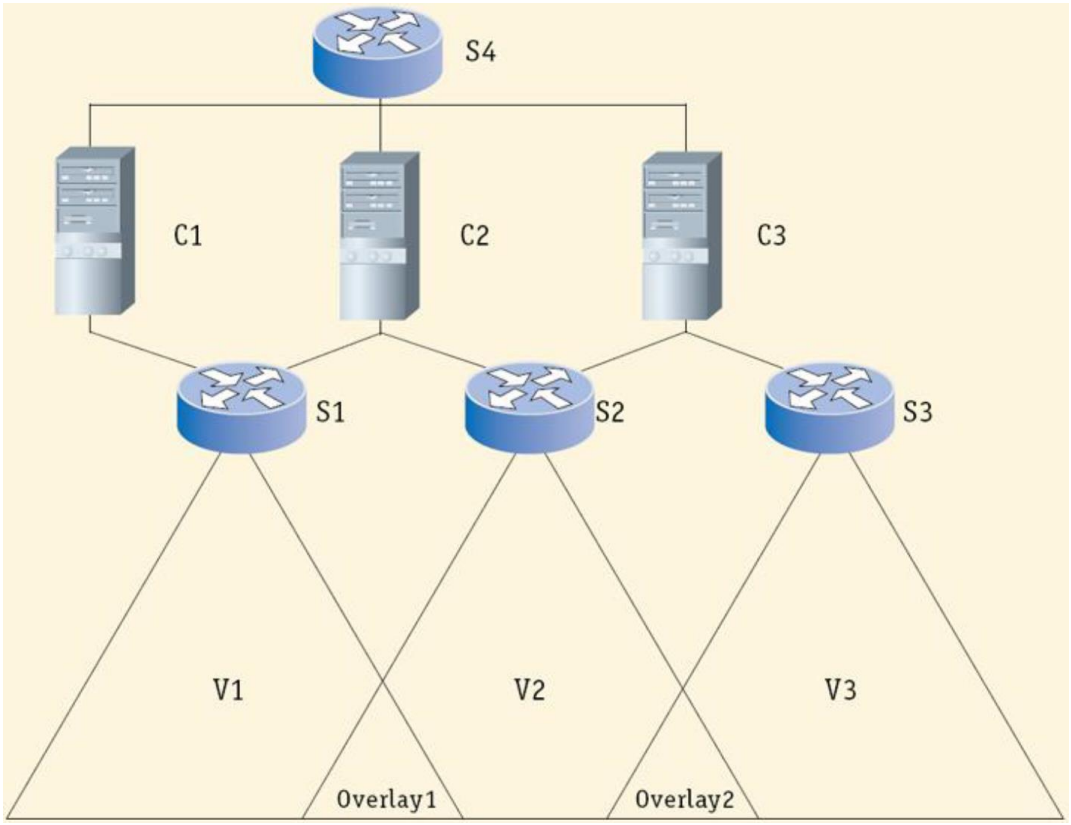

Рис. 1.3. Альтернативний підхід до побудови розподіленого масштабованого контролера

Такий принцип побудови розподіленого контролера вирішує проблему масштабованості та підвищує відмовостійкість ПКМ.

1.3. Загальні відомості про побудову шляхів

Найчастіше найбільше найкоротший шлях розглядається за допомогою математичного об'єкта, званого графом.

Існують три найбільш ефективних алгоритми знаходження найкоротшого шляху [32]:

1. алгоритм Дейкстри (використовується для знаходження оптимального маршруту між двома вершинами);
2. алгоритм Флойда (для знаходження оптимального маршруту між усіма парами вершин);

3. алгоритм Єна (для знаходження k-оптимальних маршрутів між двома вершинами).

Основним завданням даної науково-дослідної практики є програмна реалізація алгоритму пошуку найкоротшого шляху між двома будь-якими вершинами графа [33].

Програма має працювати так, щоб користувач вводив кількість вершин і довжини ребер графа, а після опрацювання цих даних на екран виводився найкоротший шлях між двома заданими вершинами та його довжина. Необхідно передбачити різні результати пошуку, щоб програма не видавала помилок і працювала правильно.

Ця програма може використовуватися в дискретній математиці для дослідження графів або як наочний посібник, що демонструє застосування алгоритму Флойда на практиці.

Граф - це система, яку інтуїтивно можна розглядати як множину вузлів і множину ліній, що їх з'єднують, геометричний спосіб задання графа представлено на рис. 1.4. Вузли називаються вершинами графа, лінії зі стрілками - дугами, без стрілок - ребрами. Граф, у якому напрямок ліній не виокремлюють, тобто всі лінії є ребрами, називається неорієнтованим; граф, у якому напрямок ліній є принциповим, тобто лінії є дугами, називається орієнтованим.

Теорію графів можна також розглядати як розділ дискретної математики, а якщо точніше, то теорії множин, і формальне визначення графа є таким: якщо задано скінченну множину X, яка складається з n елементів (X = {1, 2,..., n}), які називаються вершинами графа, і підмножину

V декартового добутку, що називається множиною дуг, тоді орієнтованим графом G називається сукупність (X , V), неорієнтованим графом називається сукупність множини X і множини невпорядкованих пар елементів, кожен з яких належить множині X. Дугу між вершинами i і j позначатимемо (i , j). Число дуг графа будемо позначати m (V = (v1, v2,..., vm)). Приклад графа подано на Рис. 1.4.

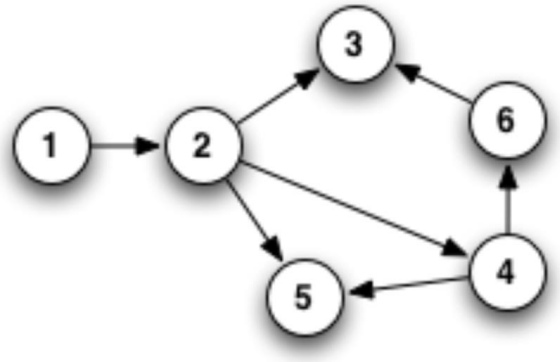

Рис. 1.4. Приклад графа

Мова графів є зручною для опису багатьох фізичних, технічних, економічних, біологічних, соціальних, економічних, економічних, економічних, соціальних та інших систем [34].

У межах теорії графів зазвичай розв'язують транспортні та технологічні задачі.

- "Транспортними" завданнями є величезний спектр логістичних задач, у яких вершинами графа є пункти, а ребрами дороги або інші транспортні маршрути. Іншим прикладом є мережі постачання. У них вершинами є пункти виробництва і споживання, а ребрами - можливі маршрути переміщення. Відповідний клас задач оптимізації потоків вантажів, розміщення пунктів виробництва і споживання тощо, іноді називається задачами забезпечення або задачами про розміщення. Їхнім підкласом є задачі про вантажоперевезення;
- "Технологічні задачі", у яких вершини відображають виробничі елементи заводів, верстатів тощо, а дугами є - потоки сировини, матеріалів і продукції між ними. Суть цих завдань полягає у визначенні оптимального завантаження всіх виробничих елементів.

РОЗДІЛ 2
ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ТА РЕАЛІЗАЦІЯ АЛГОРИТМУ, ЕМУЛЯЦІЇ СКЛАДНОГО СЕГМЕНТА ПРОГРАМНО-КОНФІГУРОВАНОЇ МЕРЕЖІ

2.1. Огляд мережевих ОС

З моменту розвитку програмно-конфігурованих мереж, з'явилося багато контролерів, які іноді називають Мережевими Операційними системами, однак, не всі ці контролери є досить хорошими. Одними з тих, що зарекомендували себе, є контролери: RUNOS, HP Virtual Application Networks ПКМ Controller і POX. Нижче буде проведено детальний огляд контролерів з метою виявити слабкі та сильні сторони кожного [35].

Огляд RUNOS. Починаючи з 2013 року, Центр Прикладних Досліджень Комп'ютерних Мереж, почав розробку контролера RUNOS. За заявами ЦПІКС це найшвидший на сьогоднішній момент контролер порівняно з наявними аналогами. Цей контролер дуже ефективно використовує обчислювальні ресурси для управління мережею. Пропонує величезний набір мережевих додатків, можливість фільтрації трафіку, робота з мережевими протоколами та інше.

Станом на 2023 рік RUNOS є однією з найшвидших і найефективніших мережевих ОС, через малу середню швидкість генерації нового з'єднання - 45 мкс і підтримку 1 тисячі комутаторів одноразово. У RUNOS реалізовано балансування навантаження, узгоджене бачення всієї мережі, робота з розподіленими мережевими додатками, безпека і протидія зовнішнім навантаженням.

Головними характеристиками є [36]:

- Обробка 30 мільйонів потоків на секунду;
- Час на встановлення нового з'єднання - 45 мкс;
- Підтримка 1000 комутаторів;
- Можливість управління з графічного інтерфейсу;

Розробники запевняють, що на осінь 2023 року - це найшвидший ПКМ-контролер у світі. Його продуктивність досягається за рахунок використання можливостей багатоядерних і багатопроцесорних систем. У ньому задіяно набір мережевих застосунків - як зі світу традиційних мереж, так і нових: L2/L3-маршрутизація з урахуванням якості обслуговування, багатопотокова маршрутизація, фільтрація трафіку, робота з мережевими протоколами (ARP, DNS, DHCP, BGP), трансляція адрес (NAT), балансування навантаження, віртуалізація мереж, анти-DDoS, верифікація мережі, інтеграція з системою управління ЦОД.

RUNOS насамперед орієнтований на корпоративний сегмент. Його споживчою аудиторією в ЦПІКС бачать мережевих адміністраторів та інженерів ЦОД, телекомоператорів, сервіс-провайдерів, а також учнів за напрямом "мережеві технології", дослідників у сфері комп'ютерних мереж і розробників перспективних мережевих технологій [37].

Огляд контролера HP Virtual Application Networks SDN Controller. Цей контролер для програмно-конфігурованих мереж розробляється компанією HP. Є платним, проте має можливість роботи з кількома протоколами програмно-конфігурованих мереж.

Компанія дотримується своєї політики відсутності прив'язки програмного забезпечення до апаратних засобів і з цієї причини їхній контролер забезпечує можливість взаємодії з будь-яким мережевим обладнанням, яке підтримує протоколи програмно-конфігурованих мереж.

Головними характеристиками є:

- Максимальна кількість підтримуваних пристроїв - 4 000;
- Максимальна кількість підключених комп'ютерів - 30 000;
- Максимальна кількість потоків на секунду - 2.3 мільйона;
- Час на встановлення з'єднання - 50-60 мкс;

Як видно, цей контролер є потужним інструментом, однак, час на встановлення з'єднання та максимальна кількість потоків залишає бажати кращого [38].

Огляд NOX. NOX був розроблений інженерами Nicira Networks паралельно з протоколом OpenFlow. Спочатку був розроблений з підтримкою двох мов: C++ і

Python. У 2008 р. NOX опублікували під ліцензією GPL, і відтоді цей контролер є базовим для багатьох науково-дослідницьких груп, які тільки починають вивчати ПКМ. NOX орієнтований на дистрибутиви Linux (зокрема Ubuntu 21.2 і 22.04, але також можливе використання на Debian і RHEL 6). Містить сервіси для побудови топології мережі та L2-L3 комутації.

У процесі тестування з'ясувалося, що підтримка двох мов сильно позначається на продуктивності, тому частину, яка відповідає за Python, витягли в окремий проєкт, який пізніше назвали POX [39].

Огляд POX. Контролер POX є "молодшим братом" NOX. Якщо під час розроблення NOX основною метою була висока продуктивність, то POX насамперед спрямований на навчання та дослідження. За своєю суттю POX - це платформа для швидкого розроблення та прототипування ПЗ керування мережею. Цей контролер написаний на Python, його легко запустити на Window, Linux і Mac OS. Наприклад, дослідницька група в Стенфорді використовує POX для дослідження ключових проблем ПКМ. Є досить продуктивним контролером, поширюється безкоштовно і підтримує останню версію протоколу OpenFlow. POX перебуває у стадії активного розвитку: всі вдалі ідеї постійно переміщуються з лабораторних експериментів в офіційні релізи контролера POX (принаймні так стверджують його розробники зі Стенфорда).

Головними характеристиками є:

- Максимальна кількість пристроїв - 3000;
- Максимальна кількість потоків на секунду - 1 мільйон;
- Час на встановлення з'єднання - 90 мкс;

POX підтримує ті самі компоненти, графічний інтерфейс, засоби візуалізації, як і NOX [40].

Огляд Beacon. Це досить швидкий, кросплатформний, модульний OpenFlow контролер на Java. Цей контролер розробляється вже понад два роки. Beacon використовується в багатьох науково-дослідних проєктах і тестових впровадженнях / розгортаннях. Beacon застосовується в експериментальному ЦОДі Стенфорда, в якому він керує 100 віртуальними і 20 фізичними комутаторами Beacon написаний на Java і працює на багатьох платформах, починаючи від високопродуктивних багатоядерних

Linux-серверів до смартфонів на Android. Розробник контролера - Девід Еріксон, учень Ніка Маккеона, Стенфорд.

Огляд Maestro. Maestro - це операційна система, розроблена в Rice University. Maestro надає інтерфейси для реалізації модульних додатків управління мережею для доступу та зміни стану мережі, а також координації їхньої взаємодії. Незважаючи на те, що цей проєкт спрямований на створення OpenFlow-контролера, Maestro не обмежується лише OpenFlow-мережами. Середовище програмування Maestro надає інтерфейси для додавання нових користувацьких компонентів з управління мережею.

Крім того, Maestro намагається використовувати паралелізм у межах однієї машини для поліпшення пропускної здатності системи. Розробники заявляють основними властивостями Maestro портативність і масштабованість. Maestro розроблено на Java (і саму платформу, і її компоненти), вона є універсальною для різноманітних ОС та архітектур [41].

У результаті аналізу мережевих ОС було виявлено, що для реалізації цієї магістерської роботи найкраще підходить RUNOS завдяки швидкості роботи, істотному обсягу одночасно підтримуваних потоків і швидкості відгуку.

2.2. Огляд засобів для емуляції програмно-конфігурованої мережі

На поточний момент існує велика кількість засобів для емуляції програмно-конфігурованих мереж. Найвідомішими і визнаними фахівцями є: ns-3 Network Simulator, OPNET, NetSim і Mininet. Під час вибору засобу для емуляції варто звернути увагу на функціональні можливості, підтримку необхідних протоколів, переваги та недоліки. Також, варто врахувати значущість недоліків емулятора для поставлених завдань.

Огляд ns-3 Network Simulator. ns-3 Network Simulator є проєктом з відкритим кодом, що поширюється за ліцензією GNU GPL. Мета, яку поставили перед собою розробники, - це створити безкоштовний симулятор побудови комп'ютерних мереж, за допомогою якого можна виконувати різноманітні дослідження щодо роботи комп'ютерних мереж, без необхідності будувати реальні.

Перша версія ns з'явилася в далекому 97 році, тоді симулятор був доволі слабким, незважаючи на доволі швидке ядро, написане на C++, через сценарії, засновані на Tcl, сильно гальмувалася робота.

Трохи пізніше, DARPA, Xerox і низка інших зацікавлених компаній та організацій почали допомагати в розробці такого програмного забезпечення, за кілька років з'явився ns-2, який став продуктивнішим і просунутішим з точки зору налаштувань.

Починаючи з 2006 року розпочалося розроблення третьої версії програми, ця версія має суттєву перевагу порівняно зі старими версіями, змінився скриптовий рушій, тепер скрипти, що використовуються, пишуть на мові Python, яка проста в освоєнні. Крім цього, забезпечено зворотну сумісність із другою версією. Реліз третьої версії відбувся в червні 2008 року, після цього проєкт доопрацьовували до дуже стабільної версії, яка у 2010 році вийшла досить стабільною, щоб повністю замінити другу версію, отже, у 2010 році підтримка ns-2 припинилася [42].

Найголовнішими плюсами цього емулятора є: можливість створення топології, конфігурування вузлів і з'єднань, аналіз навантаження і візуалізація даних.

Найголовнішим мінусом є відсутність підтримки протоколів програмно-конфігурованих мереж.

Огляд OPNET. OPNET це пропрієтарне програмне забезпечення, розроблене компанією OPNET Technologies. Перша версія програми була представлена публіці 2000 року.

Основна аудиторія цього продукту - комерційні компанії, яким необхідно розробити дата-центр, центр обробки даних або значну локальну мережу.

Їхній інструмент надає зручний інтерфейс під час роботи з емульованим середовищем, за допомогою такої програми дуже просто і швидко будується мережа будь-якої складності.

Opnet містить бібліотеки, завдяки яким здійснюється формування телекомунікаційних мереж, і полегшує вивчення моделі шляхом підключення різних типів вузлів, з використанням різних видів зв'язку тощо [43].

Редактор вузлів являє собою редактор, який використовується для створення моделей вузлів і зазначення їхньої внутрішньої структури. Ці моделі використовуються для створення вузлів усередині мережі в редакторі проєкту.

Внутрішні вузли моделі мають модульну структуру, яка визначається як вузол з'єднання декількох модулів з пакетом потоків і кабелів. Це з'єднання дає змогу обмінюватися інформацією та пакетами між ними. Кожен модуль має певну функцію у вузлі, як-от: створення пакетів, склеювання, процес або передача та приймання.

У цьому редакторі елементи доступні як чорні скриньки, корпусу атрибутів, які можуть бути налаштовані. Кожен із них представляє функцію у вузлі.

Також має потужний редактор моделі з'єднань. Редактор дає можливість створювати нові типи об'єктів з'єднання. Кожен новий тип з'єднання може мати різні атрибути та подання.

Типи підтримуваних редактором з'єднань: усі з'єднання, які ми можемо підтримувати, одне або всі чотири, допускаються симулятором. Цими з'єднаннями є: точка-точка, дуплекс точка-точка, шина і болт [44].

Об'єкти представляють у цьому редакторі процесори. Їхню поведінку визначають у процесі редактора. Є попередньо сконфігуровані моделі, такі як джерела даних, поглиначі тощо.

Найголовнішими плюсами цього емулятора є: простота використання, підтримка всіх наявних протоколів.

Найголовнішим мінусом є вартість цього продукту.

Огляд NetSim. Програмне забезпечення, розроблене компанією TETCOS. Розробку було розпочато в червні 2002 року. Це програмне забезпечення підтримує сучасні протоколи, забезпечує можливість емуляції бездротових мереж [44].

Проєкт створено мовою C, що робить симулятор дуже швидким. Цей симулятор є чудовим інструментом для розробки архітектури складних мереж.

З огляду на свою ефективність і гнучкість, цей інструмент є складним в освоєнні і використовується тільки професіоналами.

Цей продукт має величезну вартість, що є його головним мінусом.

Огляд Mininet. Симулятор мережі Mininet є безкоштовним програмним забезпеченням, що розробляється спільнотою програмістів, які є прихильниками безкоштовного програмного забезпечення.

Mininet спроєктований таким чином, щоб можна було легко і просто створювати віртуальні програмно-конфігуровані мережі. Крім цього було розроблено спосіб підключення віддаленого контролера, який дає змогу тестувати будь-які програмно-конфігуровані контролери.

Цей емулятор з'явився насамперед через швидко зростаючий інтерес до програмно-конфігурованих мереж. Володіє функціями генерації даних, дає змогу тестувати отриману мережу на предмет продуктивності та відразу виявляти слабкі місця спроєктованої топології [45].

Вся емуляція відбувається на рівні ядра. За допомогою зручного API дає змогу швидко будувати складні топології, які можуть зберігати не одну тисячу пристроїв.

Під час аналізів було виявлено, що Mininet є лідером серед емуляторів завдяки наданим можливостям і безоплатному поширенню.

2.3. Огляд середовищ розробки

З моменту розвитку програмування для спрощення та прискорення розроблення було розроблено різноманітні середовища розроблення, які надають широкий спектр можливостей та інструментів для прискорення проєктування та написання коду.

За недовгу історію еру програмування було розроблено багато середовищ розробки під різні мови програмування. Оскільки в рамках виконання проєкту нам знадобиться мова програмування C++, то ми розглянули найкращі середовища розробки для мови програмування C++. Найкращими на даний момент є: Microsoft Visual Studio, Eclipse CDT, NetBeans, Code Lite.

Огляд Microsoft Visual Studio. Microsoft Visual Studio - це продукт компанії Microsoft розроблений насамперед для розробки програмного забезпечення під опе-

раційну систему Windows. Завдяки модульності має підтримку декількох мов програмування такі як C++, C#, JavaScript тощо. Він містить у собі прекрасний редактор вихідного коду з підтримкою технології IntelliSense.

IntelliSense спеціальна технологія автодоповнення тексту. Як і інші системи автодоповнення є зручним інструментом для перегляду опису функцій і списків аргументів. За допомогою неї проводиться прискорення розробки програмного забезпечення, за рахунок зменшення навантаження на програміста. Крім того, вона дає змогу зменшити кількість звернень до документації, завдяки виведенню частини документації у вигляді допоміжних вікон, що спливають, у редакторі коду. Під час роботи IntelliSense здійснює сканування метаданих класів, змінних та інших конструкцій підключених бібліотек і зберігає ці дані у внутрішню базу даних. "Класична" реалізація IntelliSense здійснює пошук спеціальних маркерів у коді, наприклад, символ крапки. Як тільки виявляється маркер, то програмісту одразу демонструються доступні методи класу, згідно з його рівнем доступу, а також параметри, необхідні для цього методу.

Microsoft Visual Studio є дорогим і якісним середовищем розробки, проте для студентів і викладачів може надаватися безкоштовно за спеціальною програмою DreamSpark.

Мінуса у цього середовища розробки два: сильно орієнтоване на розробку під Windows і відсутність крос платформності [37, 39-41].

Огляд Eclipse CDT. Eclipse - це вільне модульне інтегроване середовище розробки, написане мовою Java. Історія розробки починається з компанії IBM, як приймач IBM VisualAge, як корпоративного стандарту середовища розробки, різними мовами під різні платформи IBM. Згідно з даними IBM початкова розробка коштувала 40 мільйонів доларів. Вихідний код був повністю відкритий і зроблений доступним після того, як Eclipse було передано для подальшого розвитку незалежній від IBM спільноті.

Однією з найважливіших переваг цього середовища розробки є можливість розробки різних розширень, які можуть допомогти програмісту в розробці. Уже зараз є потужний інструмент для мови Java під назвою Java Development Tools, для C/C++

під назвою C/C++ Development Tools. Ці та інші розширення під різні мови були розроблені спільнотою спільно з розробниками IBM.

Завдяки розширенням Eclipse має хороші розширення для роботи з різноманітними системами контролю версій, таких як CVS та GIT. Також має чудові засоби для зв'язку із системами керування помилок на кшталт Youtrack або Jira.

Завдяки безоплатному розповсюдженню та високій якості програмного забезпечення, що не сильно поступається такому важковаговику, як Microsoft Visual Studio, здобула велику популярність серед програмістів-одинаків і величезної кількості організацій [27].

Завдяки тому, що сам продукт написаний на Java, середовище розроблення є кросплатформеним, що дає йому змогу працювати на багатьох платформах, починаючи з Windows-подібних систем і закінчуючи широким спектром Unix-подібних систем і MacOS.

Архітектура складається з таких компонентів:

- Ядро платформи, відповідає за завантаження Eclipse і запуск модулів, що підключаються, наприклад, CDT або JDT;
- OSGi;
- SWT;
- JFace;
- Робоче середовище Eclipse Представляє собою різні понелі, редактори тощо;

OSGi (Open Services Gateway Initiative) - це бібліотека для динамічної модульної системи та сервісної платформи для Java-додатків. Вона розробляється консорціумом OSGi Alliance. Специфікації дають модель для побудови програми з компонентів, пов'язаних разом за допомогою сервісів. Суть полягає в можливості динамічного перевстановлення різних компонентів і складових частин додатка без необхідності в перезапуску.

Коло застосувань цього стандарту досить широке. Спочатку розробка велася для створення вбудованих систем, але зараз на базі OSGi будують багатофункціональні автономні настільні додатки та корпоративні рішення.

OSGi Alliance, раніше відома як Open Services Gateway initiative - організація відкритих стандартів (open Standards Development Organization - SDO). Протягом останніх кількох років вона розробляла засновану на Java сервісну платформу OSGi (також відома як The Dynamic Module System for Java), якою можна було керувати віддалено. Основна частина цієї розробки - фреймворк (каркас), який визначає модель життєвого циклу застосунку та службового реєстру.

SWT (Standard Widget Toolkit) - спеціальна відкрита бібліотека для швидкого розроблення графічних інтерфейсів користувача мовою Java.

Розробка велася фондом Eclipse. Ліцензується під Eclipse Public License, однією з ліцензій відкритого програмного забезпечення.

SWT не є самостійною графічною бібліотекою, а лише являє собою крос-платформну оболонку для графічних бібліотек конкретних платформ, наприклад, під Linux SWT використовує бібліотеку GTK+. SWT написана на стандартній Java і отримує доступ до OS-специфічних бібліотек через Java Native Interface, який розглядається як сильний засіб, незважаючи на те, що це не є чистою Java.

SWT є гідною альтернативою для розробників замість AWT і Swing (Sun Microsystems). Він необхідний для тих розробників, які бажають отримати звичний зовнішній вигляд програми в даній операційній системі. Використання SWT робить Java-додаток ефективнішим і гнучкішим, але знижує незалежність від операційної системи й устаткування, вимагає ручного звільнення ресурсів і певною мірою порушує Sun-концепцію платформи Java.

JFace це великий набір допоміжних Java-класів, що реалізує найзагальніші завдання побудови GUI. У рамках проєкту Eclipse бібліотека JFace має такий опис: "Елементи користувацького інтерфейсу, реалізація яких може бути стомлюючою". JFace являє собою додатковий програмний шар над SWT, що реалізує патерн програмування Model-View-Controller. JFace надає такі можливості:

1. Надає "Viewer" класи, що відповідають за відображення і реалізують трудомісткі завдання щодо заповнення, сортування, фільтрації, а також оновлення віджетів;

2. Надає "Action" класи, які дають змогу розробнику визначати специфічну поведінку для окремих елементів користувацького інтерфейсу, таких як пункти меню, кнопки тощо;

3. Надає регістри, що містять шрифти та зображення;

4. Надає набір стандартних діалогових вікон і віджетів, а також надає фреймворк для створення складного графічного інтерфейсу для взаємодії з користувачем;

Основна мета JFace полягає в наданні розробнику великої кількості інструментів для розробки призначеного для користувача інтерфейсу, даючи йому змогу насамперед зосередитися на бізнес-логіці програми.

Основним завданням групи розробників Eclipse було приховування реалізації компонентів графічного інтерфейсу, побудованих на основі бібліотеки SWT, і по можливості максимальне використання бібліотеки JFace як більш високорівневої та простої у використанні. Бібліотека JFace використовує SWT, але SWT є не залежною від JFace. Однак, робоче середовище Eclipse побудовано з використанням обох бібліотек і в деяких місцях SWT використовується безпосередньо в обхід JFace.

Огляд NetBeans. NetBeans IDE - вільне інтегроване середовище розроблення застосунків (IDE) на мовах програмування Java, Python, PHP, JavaScript, C, C++, Ада та низки інших.

Проєкт NetBeans IDE підтримується і спонсорується компанією Oracle, однак розробка NetBeans ведеться незалежним спільнотою розробників-ентузіастів (NetBeans Community) і компанією NetBeans Org. Останні версії NetBeans IDE підтримують рефакторинг, профілювання, виділення синтаксичних конструкцій кольором, автодоповнення конструкцій, що набираються, на льоту і безліч зумовлених шаблонів коду.

Для розробки програм у середовищі NetBeans і для успішної інсталяції та роботи самого середовища NetBeans має бути попередньо встановлено Sun JDK або J2EE SDK відповідної версії. Середовище розробки NetBeans за замовчуванням підтримувало розробку для платформ J2SE і J2EE. Починаючи з версії 6.0 NetBeans підтримує розробку для мобільних платформ J2ME, C++ (тільки g++) і PHP без встановлення додаткових компонентів.

У вересні 2016 року Oracle передала інтегроване середовище розробки NetBeans у руки фонду Apache.

NetBeans IDE підтримує плагіни, даючи змогу розробникам розширювати можливості середовища. Одним із найпопулярніших плагінів є потужний дизайнер звітів iReport (заснований на бібліотеці JasperReports).

На ідеях, технологіях і значною мірою на вихідному коді NetBeans IDE базуються пропоновані фірмою Sun комерційні інтегровані середовища розроблення для Java - Sun Java Studio Creator, Sun Java Studio Enterprise і Oracle Solaris Studio (для ведення розроблення на C, C++ або Фортран). Порівняно недавно Sun стала пропонувати ці середовища розробки безкоштовно для розробників, які зареєструвалися в Sun Developer Network (SDN), а сама реєстрація на сайті безкоштовна і не вимагає ніяких попередніх умов, крім згоди з ліцензією CDDL.

NetBeans IDE доступна у вигляді готових дистрибутивів (прекомпільованих бінарних файлів) для платформ Microsoft Windows, Linux, FreeBSD, Mac OS X, OpenSolaris і Solaris (як для SPARC, так і для x86- Intel і AMD). Для всіх інших платформ доступна можливість скомпілювати NetBeans самостійно з вихідних текстів.

Огляд Code Lite. CodeLite - вільне кросплатформне середовище розроблення програмного забезпечення для мови C/C++ з відкритим вихідним кодом.

Є простим і нехитрим середовищем розробки з вбудованою підтримкою інтеграції системи контролю версій Subcersion і Git. Має автодоповнення ctags + clang, зручний рефакторинг коду.

CodeLite поширюється за ліцензією GNU General Public License v2 або пізнішої версії. Є вільним програмним забезпеченням. CodeLite наразі, будучи розробленим і налагодженим, використовує себе як платформу розробки.

Серед представлених вище середовищ розроблення, для розроблення ефективного алгоритму оптимізації трафіку підходить Eclipse CDT, завдяки своїй кросплатформеності та відсутності прив'язки до будь-якої операційної системи [24].

2.3. Проєктування структури ефективного алгоритму оптимізації трафіку ефективного алгоритму

Суть ефективного алгоритму оптимізації трафіку полягає в тому, щоб знизити затримки на створення з'єднання. Насамперед варто звернути увагу на те, що таке потоки в рамках програмно-конфігурованих мереж. Потік даних - це певний маршрут, який прокладається всередині сегмента мережі між комутатором A і комутатором B, для того, щоб n кількість пакетів досягла деякого вузла мережі.

У програмно-конфігурованому контролері використовується стандартний алгоритм Дейкстри для побудови маршруту на графі, що не є зовсім швидким рішенням, оскільки використовується універсальна версія алгоритму з використанням складних структур даних. У разі його модифікації можна зробити поліпшення продуктивності в межах цього завдання.

Алгоритм Дейкстри не гарантує знаходження шляху на графі, для розв'язання цієї проблеми необхідно додатково розробити механізм відновлення маршруту з цільової вершини, що не потрапила в маршрут.

Таким чином ми отримуємо, що наш алгоритм буде розбитий на дві підзадачі: перша - це розробка модифікації алгоритму Дейкстри на графах для прискорення створення нових маршрутів у межах цього завдання; друга - це розробка швидкого алгоритму відновлення маршруту від кінцевої вершини до стартової в межах побудованого алгоритмом Дейкстри шляху. Алгоритм генерації маршрутів завжди має будувати оптимальний маршрут, щоб унеможливити ситуацію необґрунтованого перевантаження одного комутатора щодо інших, якщо це можливо.

Проєктування та реалізація алгоритму побудови маршрутів на базі простих структур даних. RUNOS є контролером із відкритим вихідним кодом, а отже ми можемо вільно модифікувати вихідні файли. Насамперед варто зазначити, що спочатку використовувалася бібліотека boost для розрахунку маршруту за алгоритмом Дейкстри, але є низка мінусів, таких як надмірно складна структура зберігання даних і повністю універсальна реалізація алгоритму [37]. Це викликано тим, що бібліотека boost

є вкрай універсальною у своїх реалізаціях, у зв'язку з цим з її допомогою не завжди можна отримати максимальну продуктивність [34].

Перше на що варто звернути увагу - це структура зберігання даних, вона є складною з не найшвидшою швидкістю доступу. Класичною структурою даних для роботи з графами є двовимірний масив (Рис. 2.1.). Головною відмінною особливістю є константний час доступу до будь-якої комірки даних, і він мізерно малий, чого не скажеш про складні структури даних. Завдяки цій перевазі масиви дають змогу швидко модифікувати зв'язки між окремими вузлами мережі. Під час розрахунку маршруту є низка складнощів, і найголовніша полягає в тому, що для повного обходу всіх можливих ребер одного вузла нам знадобиться здійснити O(n) операцій, де n - це кількість комутаторів, а в найгіршому разі це буде O(n).2

У більшості мов програмування масиви простіше ініціалізувати і використовувати, ніж складні структури даних.

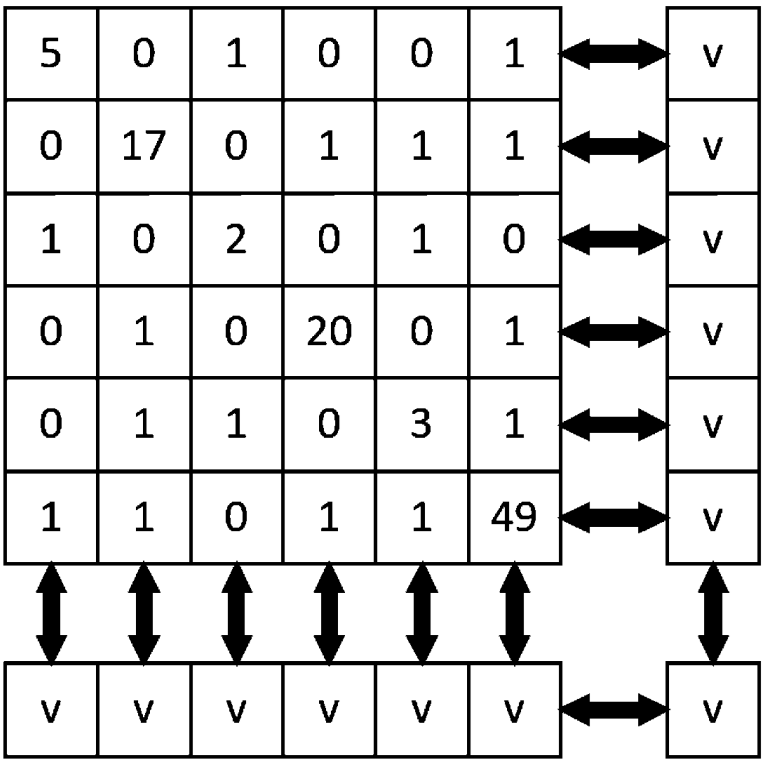

Рис. 2.1. Подання двовимірного масиву

Проблему обходу можна буде розв'язати за допомогою розроблення спеціальної, простої структури даних, яка дасть змогу швидко обходити всі ребра певної точки, завдяки тому, що в ній зберігатимуться тільки реальні зв'язки. Крім переваги у швидкості розрахунків, можна мати ще одну важливу перевагу, це займана пам'ять, у більшості випадків вона буде меншою, ніж у масиву, винятком буде ситуація, коли в кожного комутатора буде понад 50 з'єднань з іншими пристроями. Важливим недоліком такої структури буде швидкість модифікації такої структури.

На рис 2.2. представлено лістинг-реалізацію двовимірного масиву для зберігання даних про топологію мережі.

```
graphData = new *int32_t[count];
        for (int32_t i = 0; i < count; i++) {
                graphData[i] = new int32_t[count];
                for (int j = 0; j < count; j++) {
                        graphData[i][j] = 0;
                }
        }
```

Рис. 2.2. Реалізація двовимірного масиву для зберігання даних про топологію мережі

Для простої взаємодії з масивом необхідно розробити обгортку з методами, що дають змогу виконувати різні дії з масивом, як-от ініціалізація масиву, додавання або видалення зв'язку між вузлами, збільшення або зменшення лічильника потоків, що проходять. Також усередині обгортки має бути реалізація пошуку маршруту у вигляді data_link_route. Цю логіку реалізують такі методи: addLink, removeLink, incrementFlows, decrimentFlows і computeRoutes.

Метод addLink виконує генерацію зв'язку між двома вузлами. На рис. 2.3 представлено пошук початкового вузла в масиві даних.

```
map::iterator it = dpidToGraphId.find(from.dpid);
if (it != dpidToGraphId.end()) {
        fromId = it->second;
}
else {
        dpidToGraphId[from.dpid] = switchIdCounter++;
        fromId = dpidToGraphId[from.dpid];
}
```

Рис. 2.3. Пошук початкового вузла в масиві даних

На рис. 2.4. представлено пошук кінцевого вузла в масиві даних.

```
map::iterator it = dpidToGraphId.find(to.dpid);
if (it != dpidToGraphId.end()) {
        toId= it->second;
}
else {
        dpidToGraphId[to.dpid] = switchIdCounter++;
        toId = dpidToGraphId[to.dpid];
}
```

Рис. 2.4. Пошук кінцевого вузла в масиві даних

На рис. 2.5. представлено створення зв'язку між двома комутаторами.

```
graphData[fromId][toId] = 0;
graphData[toId][fromId] = 0;
```

Рис. 2.5. Створення зв'язку між комутаторами

Метод додавання виявився таким великим насамперед через те, що з ПКК ми отримуємо дані у вигляді структури switch_and_port, через це нам необхідно спочатку знайти

відповідність між ідентифікатором та ідентифікатором у межах протоколу OpenFlow, а після цього здійснити встановлення самого з'єднання.

Метод removeLink виконує знищення зв'язку між двома вузлами перед зміною. На рис. 2.6 представлено видалення зв'язку між комутаторами.

```
graphData[fromId][toId] = -1;
graphData[toId][fromId] = -1;
```

Рис. 2.6. Видалення зв'язку між комутаторами

Метод знищення зв'язку виявився таким великим через те, що з ПКК ми отримуємо дані у вигляді структури switch_and_port, у якій зберігається ідентифікатор у межах протоколу OpenFlow, після знаходження ідентифікатора в масиві відбувається знищення зв'язку.

Метод incrementFlows виконує збільшення лічильника потоків, що проходять через дане з'єднання комутатор. На рис. 2.7 представлено реалізацію методу.

```
int fromId, toId;
map::iterator fromIt = dpidToGraphId.find(fromSwitch.dpid);
map::iterator toIt = dpidToGraphId.find(toSwitch.dpid);
if (fromIt != dpidToGraphId.end() && toIt != dpidToGraphId.end()) {
        fromId = fromIt->second;
        toId = toIt->second;
        graphData[fromId][toId]++;
        graphData[toId][fromId]++;
}
```

Рис. 2.7. Реалізація методу incrementFlows

Метод decrimentFlows виконує зменшення лічильника потоків, що проходять через цей комутатор. На рис. 2.8 представлено реалізацію методу.

```
int fromId, toId;
map::iterator fromIt = dpidToGraphId.find(fromSwitch.dpid);
map::iterator toIt = dpidToGraphId.find(toSwitch.dpid);
if (fromIt != dpidToGraphId.end() && toIt != dpidToGraphId.end()) {
        fromId = fromIt->second;
        toId = toIt->second;
        graphData[fromId][toId]--;
        graphData[toId][fromId]--;
}
```

Рис. 2.8. Реалізація методу decrimentFlows.

Безпосередньо сам пошук шляху розбитий на кілька невеликих підзадач. Перше підзавдання - це побудова самого маршруту за допомогою алгоритму Дейкстри, у разі знаходження мінімального маршруту до комутатора, що цікавить, завершуємо пошук. Другий етап - це відновлення шляху з кінцевої точки, до стартової.

Для поліпшення читабельності коду й об'єднання повторюваних операцій було розроблено кілька допоміжних функцій, перша - це обрахунок мінімальних відстаней до певних вузлів, що має назву setMinVals та findNextMinId для пошуку наступної точки, з якої потрібно здійснювати наступний пошук.

Метод setMinVals приймає мінімальні значення і здійснює сканування ваг ребер і визначає те, наскільки далеко розташована наступна точка. Реалізація методу представлена на рис. 2.9.

```
int32_t startValue = graphData[fromId][fromId];
for (int i = 0; i < count; i++) {
    int32_t flowsCount = graphData[fromId][i];
    if (flowsCount == -1) {
        continue;
    }
    int32_t currentValue = graphData[i][i];
    if (currentValue == -1 || currentValue > startValue + flowsCount)
{
        graphData[i][i] = startValue + flowsCount;
    }
}
```

Рис. 2.9. Реалізація методу setMinVals

Метод findNextMinId здійснює пошук наступного не відвіданого комутатора, у якого мінімальна відстань від стартової точки. Реалізація цього методу представлена на рис. 2.10.

```
int32_t minVal = -1;
int32_t minId = -1;
for (int32_t i = 0; i < count; i++) {
    if (!visited[i]) {
        currentVal = graphData[i][i];
        if (minVal == -1 || minVal > currentVal) {
            minVal = currentVal;
            minId = i;
        }
    }
}
return minId;
```

Рис. 2.10. Реалізація методу findNextMinId

Для відновлення маршруту використовується спеціальний алгоритм, реалізований у методі restoreRoute, якому повідомляється ідентифікатор кінцевого вузла і з нього алгоритм будує маршрут. Реалізація алгоритму представлена на рис. 2.11.

```
data_link_route result;
int32_t prevId = endId;
do {
        result.insert(0, graphIdToSwitch[prevId]);
        prevId = findPrevId();
} while (prevId != -1);
return result;
```

Рис. 2.11. Реалізація алгоритму відновлення маршруту

Під час вивчення особливостей роботи алгоритму Дейкстри було виявлено важливу закономірність, що з будь-якої кінцевої точки можна знайти маршрут до початкової. Оскільки в ході алгоритму кожній точці проводиться призначення мінімальної відстані від початкової точки, яка є сумою ваг усіх з'єднань, що беруть участь у цьому, то ми можемо побачити, що якщо ми братимемо кожне ребро графа, відніматимемо його вагу та перевірятимемо значення мінімальної відстані, то ми побачимо, що ця мінімальна відстань збігається лише у тому разі, коли ми рухаємося в бік нашої початкової позиції. Пошук попередньої точки реалізовано у функції findPrevId і представлено на рис. 2.12.

```
        if (graphData[fromId][fromId] == 0) {
            return -1;
        }
        for (int32_t i = 0; i < count; i++) {
            if (graphData[fromId][i] < 0) {
                continue;
            }
            if (graphData[i][i] == graphData[fromId][fromId] -
graphData[fromId][i]) {
                return i;
            }
        }
        return -1;
```

Рис. 2.12. Реалізація findPrevId

Були проведені наступні оптимізації алгоритму Дейкстри, реалізованого в ПКК RUNOS:

1. Завершення алгоритму відбувається відразу після знаходження мінімального маршруту до кінцевої точки. Під час генерації випадкових з'єднань оптимізація дає змогу зберегти величезну кількість часу, у середньому 10 мс від початкового часу;

2. Перехід на просту структуру двовимірного масиву для зберігання даних. Під час операцій побудови маршрутів відбувається приріст продуктивності за рахунок гарантованої швидкості доступу до кожного ребра і вузла у вигляді O(1). За рахунок цього вдається виграти від 10 до 5 мс залежно від розміру сегмента;

3. Також було оптимізовано зберігання вже відвіданих вершин у вигляді масиву bool значень. Цей масив створюється тільки один раз під час старту програми. Таке рішення було зроблено для прискорення роботи, адже операція генерації та видалення масиву є дорогим задоволенням. Спочатку цей масив генерувався всередині роботи алгоритму, після аналізу продуктивності за допомогою профайлера було виявлено, що 2 мс витрачається на створення і видалення цього масиву, тому було ухвалено рішення створювати його тільки один раз;

Оптимізований алгоритм Дейкстри розбитий на кілька етапів: ініціалізація даних, пошук мінімальних маршрутів зі стартової позиції, відновлення маршруту. Алгоритм реалізовано у функції computeRoute. На рис. 2.13 представлено ініціалізацію даних перед пошуком.

```
for (int i = 0; i < count; i++) {
        visited = false;
        graphData[i][i] = -1;
}
int fromId, toId;
fromId = dpidToGraphId[from.dpid];
toId = dpidToGraphId[to.dpid];
```

Рис. 2.13. Ініціалізація даних перед пошуком

Після успішної ініціалізації відбувається виконання першого кроку циклу. Під час першого кроку здійснюється пошук мінімальних відстаней з початкової точки до всіх доступних. Реалізація цього етапу роботи алгоритму представлена на рис. 2.14.

```
if (fromId == toId) {
        return route;
}
visited[fromId] = true;
graphData[fromId][fromId] = 0;
setMinVals(fromId);
int32_t nextMinId = findNextMinId(visited);
```

Рис. 2.14. Реалізація першого кроку алгоритму

Далі основний цикл повторює один і той самий крок, доти, доки не виявить побудований маршрут до кінцевої точки. Щойно це відбувається, відбувається вихід з алгоритму. Цикл представлений на рис. 2.15.

```
while (nextMinId != toId) {
        setMinVals(nextMinId);
        visited[nextMinId] = true;
        nextMinId = findNextMinId(nextMinId);
}
```

Рис. 2.15. Основний цикл алгоритму

Після побудови маршруту виконується відновлення маршруту у форматі зрозумілим для ПКК Runos. Виклик методу представлено на рис. 2.16.

```
return restoreRoute(nextMinId);
```

Рис. 2.16. Виклик методу відновлення маршруту

2.4. Оптимізація роботи алгоритму побудови маршрутів на базі простих структур даних за допомогою OpenMP

На жаль, алгоритм Дейкстри є рекурентним алгоритмом і не дає змоги обчислити маршрути паралельно, однак під час детального розгляду реалізації ми побачимо, що цей алгоритм має одне місце, яке можливо розпаралелити. Це місце знаходиться в методі setMinVals. У ньому відбувається простий нерекурентний перебір ребер і розрахунок нових значень мінімального маршруту. Реалізація паралельного методу представлена на рис. 2.17.

```
            int32_t startValue = graphData[fromId][fromId];
        #pragma omp parallel for num_threads(4)
            for (int i = omp_get_thread_num(); i < count; i += omp_get_max_threads()) {
                int32_t flowsCount = graphData[fromId][i];
                if (flowsCount == -1) {
                    continue;
                }
                int32_t currentValue = graphData[i][i];
                if (currentValue == -1 || currentValue > startValue + flowsCount) {
                    graphData[i][i] = startValue + flowsCount;
                }
            }
```

Рис. 2.17. Паралельна версія setMinVals.

Паралелізм залежно від сегмента може як прискорити, так і сповільнити роботу. Для маленьких сегментів мережі, у яких менше 100 комутаторів, робота сповільнюється на 5 мс. Під час роботи з великими сегментами, від 500 до 1000 вузлів, спостерігається прискорення продуктивності роботи алгоритму на 3-5 мс.

2.5. Емуляція тестових сегментів мережі

У межах першого розділу було розглянуто різні інструменти для емуляції сегментів комп'ютерних мереж, під час аналізу було виявлено, що інструмент Mininet є чудовим варіантом для тестування алгоритму побудови маршрутів.

Mininet дає змогу конфігурувати топології за допомогою скрипта мовою Python. Це дає нам змогу генерувати сегменти з випадковими зв'язками будь-якої складності та розмірності.

Насамперед для коректної роботи необхідно провести імпорт бібліотек для коректної роботи засобу емуляції. На рис. 2.18 продемонстровано імпорт необхідних бібліотек.

```
from mininet.net import Mininet
from mininet.node import NOX, OVSSwitch, Controller, RemoteController
from mininet.topo import Topo
from mininet.log import setLogLevel
from mininet.cli import CLI
```

Рис. 2.18. Імпорт бібліотек

Клас із назвою Mininet, що знаходиться в просторі імен mininet.net, використовується для емуляції мережі з логічними комп'ютерами.

Бібліотека mininet.node зберігає в собі опис класів, які відповідають за контролери та комутатори.

Бібліотека mininet.topo використовується необхідна для коректної роботи топології сегмента мережі. До завдань цієї бібліотеки входить: зберігання інформації про вузли, створення/видалення/модифікація підключень між вузлами.

Бібліотека mininet.log є допоміжною бібліотекою, за допомогою, якої проводиться логування налагоджувальної інформації, виведення тих чи інших помилок. Це дуже корисна бібліотека на етапі налагодження і тестування, тому що допомагає виявляти проблеми практично відразу.

Бібліотека mininet.cli необхідна для можливості конфігурування і виконання різних команд вже після запуску засобу для емулювання, за допомогою командного рядка.

Створення підключення до віддаленого ПКК RUNOS продемонстровано на рис. 2.19.

```
runos = RemoteController('c0', ip='127.0.0.1')
```

Рис. 2.19. Створення підключення до ПКК

Для створення власної топології необхідно успадкувати клас Topo і після цього можна буде модифікувати його роботу. Реалізацію користувацької топології продемонстровано на рис. 2.20.

```
class DeikstraTopo ( Topo ):
        def __init__ ( self ):
                Topo.__init__( self )
```

Рис. 2.20. Створення користувацької топології

Після створення топології необхідно створити і запустити мережу, для цього необхідно додати всі логічні комутатори в мережу, а після з'єднати їх у випадковому порядку, для коректної роботи мережі. На рис. 2.21 продемонстровано генерацію логічних комутаторів.

```
        RemoteController = self.addSwitch('s0')
        i = 0
        while i < topoSize:
                NewSwitch = self.addSwitch('deikstraSwitch' + str(i))
                i = i + 1
                self.addLink( RemoteController, NewSwitch )
        i = 0
        while i < topoSize:
                Switch = self.getSwitch('deikstraSwitch' + str(i))
                j = 0
                while j < 20:
                ForConnect = self.getSwitch('deikstraSwitch' + str(random.randInt(0,
topoSize))
                self.addLink( Switch, ForConnect)
                j = j + 1
```

Рис. 2.21 Генерація логічних комутаторів

Після генерації топології необхідно провести створення і запуску екземпляра Mininet. Для запуску мережі спершу необхідно створити екземпляр топології

DeikstraTopo. Наступним кроком буде передача необхідних параметрів для створення екземпляра мережі. Наприкінці необхідно буде провести побудову і слідом запуск мережі. Після запуску необхідно викликати консольну оболонку CLI, яка прийматиме різні команди від користувача, за допомогою яких під час виконання можна вносити різні зміни в роботу створеної мережі.

Створення і запуск представлені на рис. 2.22.

```
topo = DeikstraTopo()
net = Mininet( topo=topo, switch=OVSSwitch, build=False )
net.build()
net.start()
CLI( net )
net.stop()
```

Рис. 2.22. Створення та запуск екземпляра мережі

Увімкнення Mininet здійснюється за допомогою спеціальної консольної команди, представленої на рис. 2.23. У вхідних параметрах ми вказуємо, де знаходиться файл скрипта для запуску і яку топологію використовувати.

```
sudo mn --custom ~/mininet/custom/otroshko_kvalif_rob.py
--topo otroshko_kvalif_rob
```

Рис. 2.23. Команда для запуску Mininet

……

РОЗДІЛ 3
ТЕСТУВАННЯ ТА АПРОБАЦІЯ АЛГОРИТМУ ОПТИМІЗАЦІЇ ТРАФІКУ В ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖАХ

Після розроблення алгоритму для програмно-конфігурованої мережі на базі мережевої операційної системи RUNOS, була проведена емуляція різних складних сегментів мережі, для визначення ступеня досягнення поставлених цілей і завдань.

Загалом було проведено 2 групи тестів. На основі тестів було отримано дані про продуктивність наявного, паралельно реалізованого алгоритму та однопоточного алгоритму. Було складено діаграми продуктивності для більш зрозумілого відображення результату виконаної роботи. На цих діаграмах можна побачити відмінність за швидкістю виконання побудови маршруту на різних розмірах сегментів програмно-конфігурованих мереж.

Метою тестів першого етапу було виявити наскільки ефективними є різні реалізації алгоритму Дейкстри, на базі різних структур даних та оптимізацій. Отримані дані відображають сильні та слабкі сторони того чи іншого підходу. Схема проведення тестів була різною, перша група тестів була націлена суто на дослідження часу генерації з'єднань. Для дослідження часу розрахунку маршруту на тестовому сегменті кожні 100 мс створювався запит на передачу даних з одного випадкового комутатора до іншого. Кожен тест цієї групи проводився протягом 24 годин. Між тестами змінювався тільки розмір сегмента і реалізація алгоритму. Результати тестування представлено на рис. 3.1.

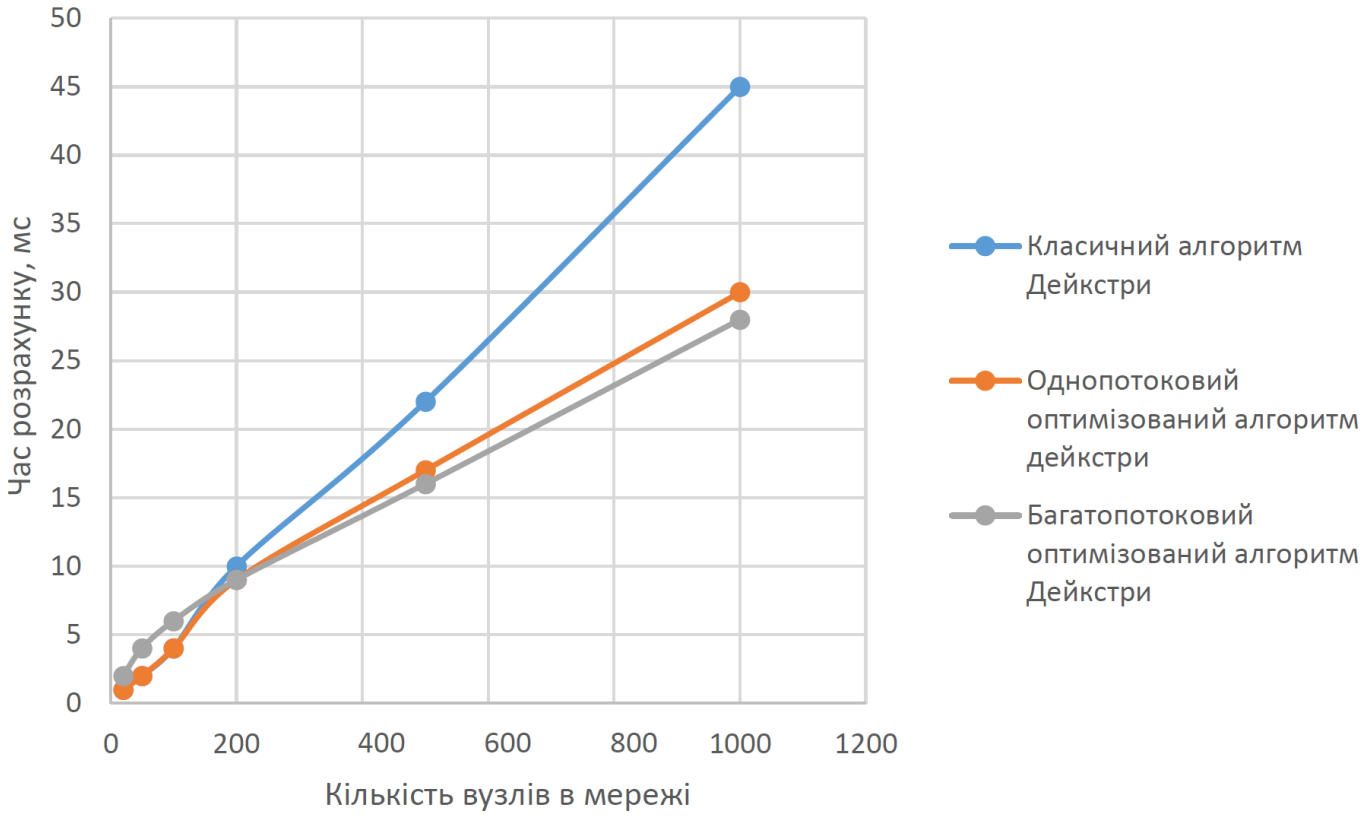

Рис. 3.1. Результат першого етапу тестування

Як видно в рамках результату тестування, при збільшенні кількості вузлів у мережі графік зростає не лінійно для класичної реалізації та однопотокової оптимізованої реалізації. Однак реалізація багатопоточного оптимізованого оптимізованого алгоритму Дейкстри за часом виконання трохи програє на невеликих сегментах мережі, через збільшені витрати на створення потоків для роботи в багатопотоковому режимі. Як можна бачити, на сегменті розміром у 100 вузлів оптимізований алгоритм Дейкстри працює трохи швидше, ніж класична реалізація, використана в мережевій ОС RUNOS. Починаючи з 200 вузлів, багатопотокова версія оптимізованої реалізації починає працювати швидше, тому що кількість оброблюваних даних збільшується, і разом із цим зростає й ефективність роботи паралельної версії алгоритму. Другий етап тестів був націлений на тестування продуктивності програмно-конфігурованої мережі в умовах передачі маленьких файлів розміром 1024 байта. Результати другого етапу тестування представлені на рис.3.2.

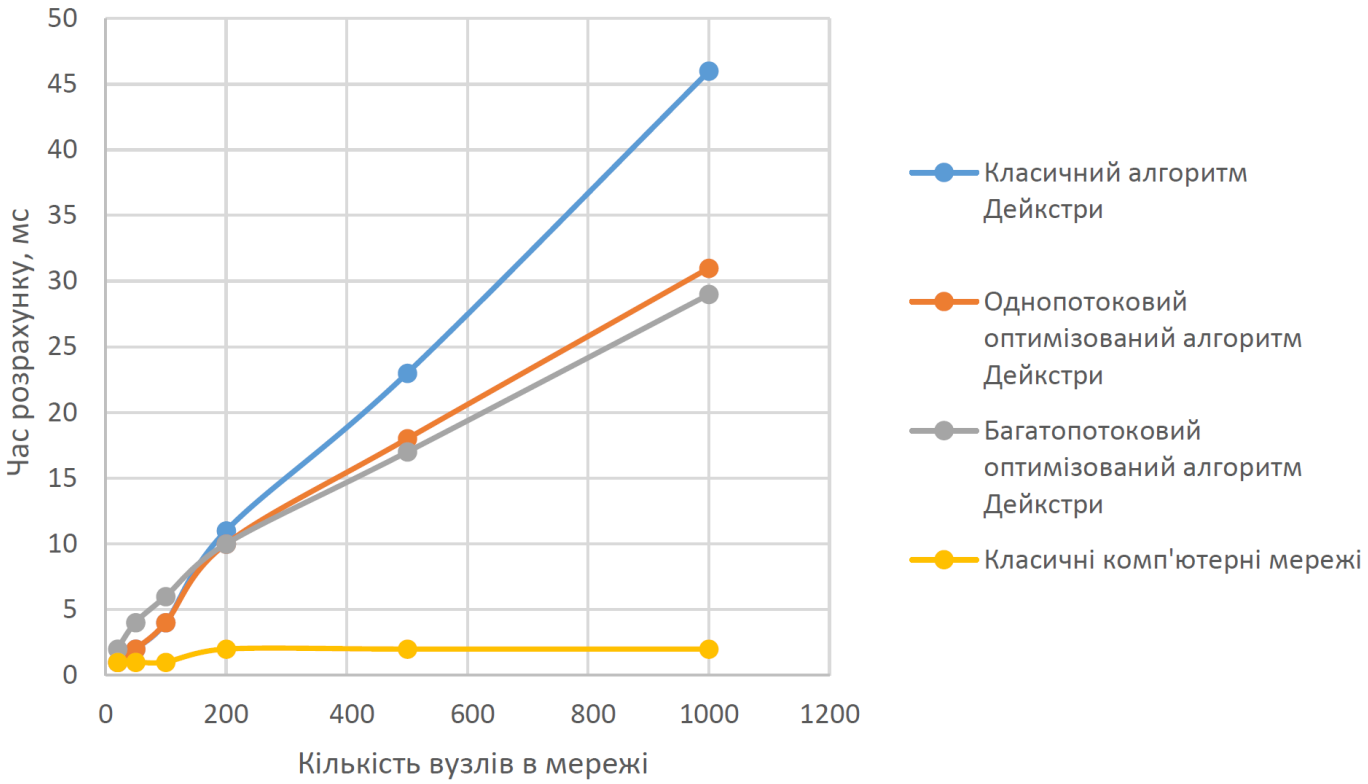

Рис. 3.2. Результати другого етапу тестування

Варто звернути увагу на те, що в рамках цього тесту порівнювали ефективність роботи не тільки між різними реалізаціями алгоритмів, але ще й між класичними мережами. Як можна бачити, на маленьких обсягах даних програмно-конфігуровані мережі не є дуже ефективним рішенням. Також варто зауважити, що будь-який створений шлях у ПКМ зберігається певний час, і якщо відбувається повторний запит на відправлення в одне й те саме місце, то шлях не генерується повторно.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ

Магістерська робота присвячена темі: «Ефективний алгоритм оптимізації трафіку в програмно-конфігурованих мережах». Робота пов’язана із моделюванням, імітацією та вивченням інформаційних сигналів, та приймання кращого рішення, фактично не маючи діючого зразка.

У цьому розділі розглядаються умови праці інженера-програміста, які включають проведення та перевірку розрахунків, створення спеціального програмного забезпечення та документацію, а також складання звітів. Він працює в відділі науково-дослідного інституту [46].

4.1. Аналіз умов праці інженера-програміста

Організація робочого місця інженера-програміста. Інженер-програміст працює в відділі науково-дослідної роботи інституту.

Кабінет призначений для чотирьох інженерів-програмістів і відповідає ДБН В.2.2-28:2010 «Будинки адміністративного та побутового призначення» [47], маючи лінійні розміри 7 на 6 метрів і висоту стелі 3 метри. Стіни світлого блакитного кольору, світлий паркет на підлозі. На рис. 4.1 показана схема робочого залу.

На робочому місці інженера-програміста повинно бути не менше 6,0 м^2 і 20,0 м^3. В цьому випадку на одну особу припадає 8 м^2 площі та 31,5 м^3 об’єму, що є допустимим.

У приміщенні є штучне та природне освітлення.

Сидіння є основною діяльністю інженера-програміста. На робочому місці інженера-дослідника є комп’ютерні науки та оргтехніка.

На робочому місці інженера-дослідника є пристрої, робота яких обумовлена наявністю шкідливих і небезпечних виробничих факторів, як-от персональний комп’ютер і багатофункційний пристрій.

Рис. 4.1. Схема робочого відділу

Аналіз шкідливих та небезпечних виробничих чинників До переліку шкідливих чинників, що негативно впливають на інженера-програміста відносяться:

- освітлення (недостатня освітленість робочої зони);

- мікроклімат (підвищена або знижена температура повітря робочої зони, підвищена або знижена вологість повітря);

- нервово-психічні перевантаження (розумове перенапруження, монотонність праці, емоційні перевантаження).

- електробезпека [48].

Аналіз освітлення. Ступінь штучного освітлення лабораторії становить 270 Лк, що нижче стандартів. Рівень штучного освітлення повинен становити від 300 до 500 Лк, згідно з ДБН В.2.5-28-2018 «Природне і штучне освітлення» [49]. Таким чином, слід збільшити кількість світильників і використовувати більш ефективні світлодіодні лампи, щоб підвищити рівень штучного освітлення.

Аналіз параметрів мікроклімату. Інженер-програміст відноситься до категорії Іа (легкі роботи, що не потребують фізичної напруги) за складністю виконуваних робіт відповідно до норм і стандартів ДСН 3.3.6.042-99 «Канітарні норми мікроклімату виробничих приміщень» [49].

Таблиця 4.1

Оптимальні величини температури, відносної вологості та швидкості руху повітря в робочій зоні виробничих приміщень [50]

Період року	Температура повітря	Відносна вологість	Швидкість руху, м/сек.
Холодний період року	22-24	60-40	0,1
Теплий період року	23-25	60-40	0,1

Микроклімат – це сукупна дія температури, відносної вологості та швидкості руху повітря, яка може зростати або падати.

Перш за все, щоб покращити мікроклімат у приміщенні, необхідно поставити сучасні склопакети, які будуть краще захищати від зміни температури на вулиці. Регулярне провітрювання приміщення також необхідно. Зволожувач повітря та кондиціонер з функцією іонізації повітря також будуть корисними додатками. Вони допоможуть встановити необхідні рівні вологості та температури повітря в будь-який час року.

Аналіз психоемоційного навантаження. Для інженерів-програмістів, які працюють з великою кількістю інформації, а також через велике навантаження на очі, наднормова робота за комп'ютером є дуже стомлюючою. У графіку роботи інженера-програміста повинні бути перерви, щоб зменшити негативний вплив на здоров'я людини. У регламенті робочого часу мають бути:

- перерви для відпочинку та вживання їжі;
- перерви для відпочинку і особистих потреб (згідно з трудовими нормами);
- додаткові перерви з урахуванням особливостей трудової діяльності.

Для того щоб зменшити вплив монотонності праці, необхідно чергувати процес усвідомлення тексту та числових даних з їх введенням в систему.

4.2. Розробка заходів з охорони праці

Заходи з поліпшення умов зорової роботи. У приміщенні можна знайти штучне та природне освітлення. У умовах недостатньої кількості природного світла або в темному часі доби використовується штучне світло. У вікнах достатньо природного світла. Тим не менш, штучне освітлення потребує покращення шляхом розумного розташування робочих місць по відношенню до віконних прорізів і світильників штучного освітлення, а також зменшення стомлення зору через зменшення пульсації світлового потоку; використання екранів захисту, фільтрів з покриттям антивідблиску; використання окулярів для користувачів ПК; і розумне використання режимів праці та відпочинку.

Коефіцієнт використання світлового потоку використовується для визначення загальної кількості світильників у приміщенні.

Необхідна освітленість робочого місця Е = 300-500 Лк для даних робіт відповідно до ДБН В.2.5-28-2018 «Природне і штучне освітлення».

Основне розрахункове рівняння методу світлового потоку, за яким можливо визначити загальний світловий потік, має такий вигляд

$$\Phi_{\text{л}} = \frac{ESZK_{\text{з}}}{\eta},$$

де Е – нормована освітленість, лк (Е=400Лк);

S – площа приміщення, що освітлюється, м^2;

$К_з$ – коефіцієнт запасу, що враховує зниження освітленості в результаті забруднення та старіння ламп (в приміщеннях громадських та житлових будівель $К_з \geq 1{,}5$ – для газорозрядних ламп і $К_з \geq 1{,}3$ для ламп розжарювання);

Z – коефіцієнт нерівномірності освітлення (Z=1,15 для ламп розжарювання та ДРЛ Z=1,1 для люмінесцентних ламп);

η – коефіцієнт використання світлового потоку.

Коефіцієнт η визначається за світлотехнічними таблицями залежно від коефіцієнтів відбиття стін ($\rho_{стін}$ = 50 %) і стелі ($\rho_{стелі}$ 70%) та від показника приміщення (індексу приміщення) і, який визначається за формулою:

$$i = \frac{AB}{h_p(A+B)}$$

де А,В – довжина і ширина приміщення, м;

h_p – висота світильника над робочою поверхнею, м.

Лінійні розміри приміщення: А = 7 м, В = 6 м.

Площа приміщення складає:

S = A*B = 7*6 = 42 м^2

Розрахуємо значення індексу приміщення:

$$i = \frac{AB}{h_p(A+B)} = \frac{7*6}{2{,}3(7+6)} = 1{,}4$$

Коефіцієнт використання світлового потоку становить 0,51.

Розрахуємо загальний світловий потік системи штучного освітлення:

$$\Phi_{заг} = \frac{ESZK_з}{\eta} = \frac{400*42*1{,}1*1{,}2}{0{,}51} = 43482 \text{ лм}$$

Для забезпечення загального штучного освітлення обираємо світильник з світлодіодною лампою SI-LED-30A-600x600-Alum.

Світловий потік однієї лампи в світильнику SI-LED-30A-600x600-Alum становить *Фл= 3800 Лм.*

Тепер визначимо кількість світильників, необхідну для освітлення приміщення:

$$N = \frac{\Phi_{заг}}{\Phi л} = \frac{43482}{3800} = 12 \text{ світильників}$$

Таким чином, щоб забезпечити світловий потік $Ф_{заг}$ = 43482 лм необхідно використовувати 12 світильників. Розміщуємо світильники у 3 ряди.

Електрична потужність однієї лампи світильника SI-LED-30A-600x600-Alum дорівнює 29 Вт.

Потужність усієї освітлювальної системи:

$$W_{заг} = W_n * N = 29 * 12 = 348 \text{ Вт.}$$

Світильники з решітками, що екрануються та відбивачами повинні використовуватися для загального освітлення, щоб забезпечити відсутність дзеркальних відображень поверхонь, що світяться на дисплеях і горизонтальних поверхнях, а також необхідний рівень контрасту для рукописних і машинописних текстів.

Над робочими поверхнями слід розташовувати світильники загального освітлення у рівномірно-прямокутному порядку.

Розташування джерел світла по відношенню до робочого місця має бути таким, щоб пряме світло не потрапляло в очі. Світильники повинні мати регулятори світла.

Оскільки освітлювальні пристрої використовують переважно відбитий або розсіяний світлорозподіл, вони не повинні створювати сліпучих відблисків на клавіатурі та екрані монітора в напрямку очей користувача.

4.3. Пожежна безпека

Пожежна безпека на підприємстві - це комплекс заходів, які використовуються для захисту майна та людей від пожежі та можливих наслідків.

Забезпечення пожежної безпеки є важливою частиною виробничої та іншої діяльності посадових осіб, працівників підприємств, установ, організацій і підприємців. Це повинно міститися в статутах підприємств, установ і організацій, а також у трудових договорах, також відомих як контракти.

Пожежна безпека повинна забезпечуватися шляхом організаційних, технічних та інших заходів, спрямованих на попередження пожеж, забезпечення безпеки людей,

зменшення можливих втрат майна та негативних екологічних наслідків у разі їх виникнення, створення умов для швидкого виклику пожежних підрозділів і ефективного гасіння пожеж.

Для забезпечення охорони праці необхідно створити план евакуації людей і матеріальних цінностей (рис. 4.2.). Для малих будівель його складають уповноважена особа або спеціальна комісія.

Згідно з НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні» [51], якщо планування або функціональне призначення будівель (приміщень, споруд), технології виробництва або штатний графік персоналу змінюються, адміністрація зобов'язана негайно переробити плани евакуації та інструкції. На підприємстві має бути система оповіщення про пожежі, з якою кожен працівник повинен бути знайомий.

В приміщенні класу «Д» (у відповідності до ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків і зовнішніх установок за вибухопожежною та пожежною небезпекою [52]) повинна бути встановлена система пожежної сигналізації «ППКП ТІРАС-8П.1» з вбудованим GSM-комунікатором. Цей пристрій призначений для цілодобової централізованої охорони об'єктів і будівель від пожеж, забезпечуючи постійний контроль восьми зон.

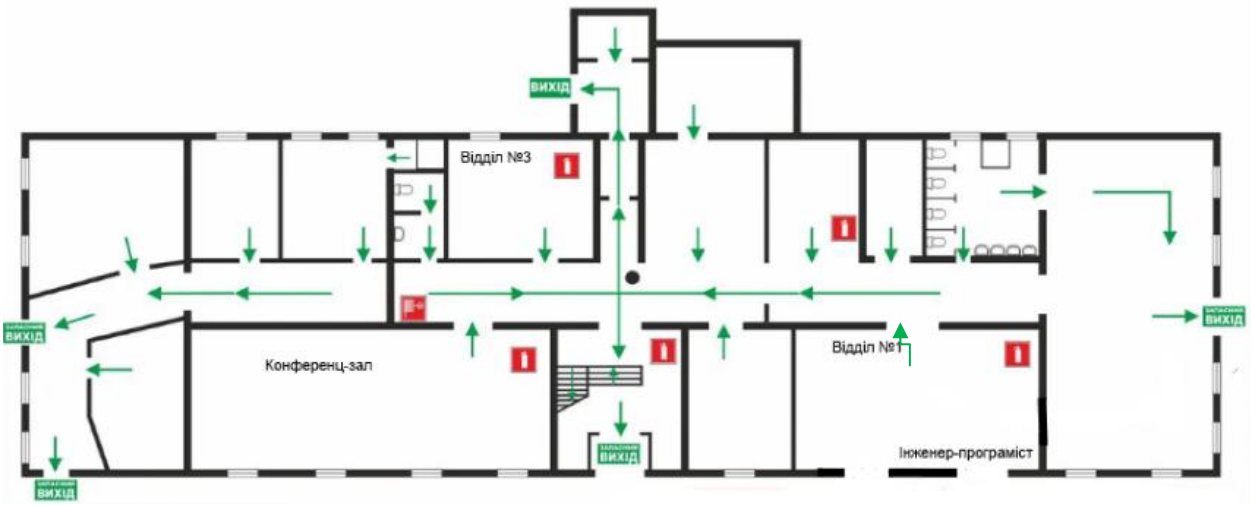

Рис. 4.2.План евакуації з приміщення

Пожежна безпека входить в комплекс заходів з охорони праці, і організаційна робота в цій сфері на об'єктах господарювання включає широкий спектр заходів, а саме:

- створення умов для безпечної праці,
- мінімізації ризику виникнення пожеж;
- своєчасне і повноцінне забезпечення технічними засобами для запобігання займання і усунення самих пожеж та їх наслідків;
- контроль дотримання протипожежних вимог і норм законодавства;
- розробка і впровадження регламентів по гасінню пожеж, евакуації і порятунку з місць пожежі і задимлення людей і майна (матеріальних цінностей);
- внутрішнє і зовнішнє навчання співробітників.

Висновки по розділу «Охорона праці»

У розділі Охорона праці розглядається робоче місце та умови праці інженера-програміста, а також список шкідливих факторів, які можуть негативно вплинути на роботу інженера-програміста. описав мікроклімат і методи покращення його. Були проведені розрахунки штучного освітлення на робочому місці та розглянуті проблеми пожежної безпеки.

Рекомендується встановити нові та більш ефективні світлодіодні лампи, щоб покращити освітлення. Вони не мають шкідливих речовин, споживають менше енергії та виділяють менше тепла, мають більший термін служби (від 30 000 до 100 000 годин) і не потребують багато електроенергії. Крім того, ці світильники хороші тим, що вони не перевантажують мережу.

РОЗДІЛ 5
ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

5.1. Електромагнітне випромінювання: джерела, види, енергетичні характеристики

Електромагнітне випромінювання — це як процес утворення вільного електромагнітного поля через нерівномірний рух і взаємодію електричних зарядів, так і взаємопов'язані коливання електричного (Е) і магнітного (В) полів, які утворюють електромагнітне поле. Електромагнітні хвилі передають випромінювання. Заряджені частинки, атоми, молекули, антени та інші випромінювальні системи створюють електромагнітні хвилі. Електромагнітне випромінювання — це потік фотонів, який можна розглядати як неперервний процес лише за великої кількості фотонів [53].

Ми зустрічаємося з електромагнітними полями (ЕМП) кожну хвилину нашого життя, але іноді ми не усвідомлюємо цього. ЕМП поділяються на поля штучного походження (системи зв'язку, радіолокаційні установки, радіо та телевізійні мовні станції) та природного походження (електричне та магнітне поле Землі, космічні джерела радіохвиль, атмосферна електрика: розряди блискавок, коливання зарядів в іоносфері).

Природні поля є постійно діючим екологічним фактором, який багато в чому визначає еволюцію біосфери Землі, в тому числі еволюцію людства. Наприклад, утворення стоячих електромагнітних хвиль низьких і наднизьких частот між поверхнею Землі та іоносферою, відомих як резонансні частоти Шумана, корелює з ритмами роботи мозку людини.

Штучні джерела спеціально розроблені для випромінювання енергії електромагнетизму (ЕМП). Створювачі паразитних електромагнітних випромінювань (ЕМВ) у зовнішньому просторі можуть бути джерелом штучного ЕМП. До них відносяться си-

стеми передачі та розподілу електроенергії, а також прилади, які її споживають, наприклад електроплити, холодильники, електронагрівачі, телевізори, освітлювальні прилади та багато іншого [54].

Електромагнітне випромінювання поділяється на діапазони:

- радіохвилі;
- видиме світло;
- терагерцове випромінювання;
- інфрачервоне випромінювання;
- ультрафіолетове випромінювання;
- рентгенівське випромінювання;
- жорстке (гамма-випромінювання).

Радиохвилі – це електромагнітні хвилі з довжиною 10-5 до 1010 м. Вони можуть виконувати багато завдань, включаючи радіомовлення, радіотелефонний зв'язок, телебачення, радіолокацію та радіометрологію, серед іншого. У всіх цих випадках інформацію можна передавати на відстань за допомогою радіохвиль.

Людське око безпосередньо сприймає область спектру електромагнітних хвиль, відому як видиме світло. Хвилі довжиною менше 380 нм називають ультрафіолетовими, а хвилі довжиною більше 750 нм називають інфрачервоними. Вплив людського ока на хвилі різної частоти в видимому діапазоні відрізняється. Вона найвища в середині діапазону (зелений колір) і найнижча в напрямку меж діапазону. Це означає, що зелене джерело буде яскравіше, ніж червоне або блакитне, коли джерело світла однакової інтенсивності.

Термагерцове випромінювання — це тип електромагнітного випромінювання, спектр частот якого знаходиться в діапазоні від інфрачервоного до надвисокочастотного. Наприклад, цей тип випромінювання використовується для сканування багажу та людей; на відміну від рентгенівського випромінювання, це випромінювання не завдає шкоди організму людини. Його можна використовувати для виявлення предметів, які приховані під одягом людини, таких як метал, кераміка та пластик, на відстані до десятків метрів. Його застосування в медицині є надзвичайно важливим, зокрема для проведення терагерцових томографів, яких за допомогою можна досліджувати

верхні шари тіла, включаючи шкіру, судини та м’язи, до глибини лише кількох сантиметрів.

Інфрачервоне випромінювання — це оптичне випромінювання з довжиною хвилі більшою за 750 нм, ніж у видимого випромінювання. Незважаючи на те, що людські очі не можуть бачити інфрачервоне випромінювання, органи чуття деяких інших тварин, таких як змії та кажани, сприймають інфрачервоне випромінювання, що допомагає їм орієнтуватися в темряві. Усі тіла з температурою вище абсолютного нуля випромінюють інфрачервоні промені з максимальною інтенсивністю.

Ультрафіолетове випромінювання — це електромагнітне випромінювання в діапазоні між видимим і рентгенівським випромінюванням. Його частота становить 380–10 нм і 7,9–1014–3×1016 Гц. Є важливим компонентом, який надає позитивний вплив на організм і знижує його чутливість до певних факторів. Якщо використовувати правильні дози променів, вони покращують роботу серця, покращують обмін речовин, підвищують активність ферментів дихання, покращують кровотворення та чинять бактерицидну дію. Ушкодження шкіри та очей можуть виникнути в результаті тривалої дії високих доз випромінювання.

Термін рентгенівське випромінювання відноситься до електромагнітного випромінювання, довжина хвилі якого коливається від 10 нм до 0,01 нм. Діапазон частот рентгенівського випромінювання знаходиться між гамма-променями та ультрафіолетом в електромагнітному спектрі. Невидиме випромінювання, такзване рентгенівське випромінювання, здатне проникати в будь-яку речовину. Цей його атрибут важливий для медицини, бізнесу та наукових досліджень. Рентгенівське випромінювання проходить крізь досліджуваний предмет і потрапляє на фотоплівку, щоб зобразити його внутрішню структуру. Через те, що різні матеріали мають різну проникаючу здатність, частини об’єкта, які менш прозорі для рентгенівського випромінювання, виглядають більш світлими на фотографіях, ніж ті, за якими випромінювання проникає добре.

Жорстке (гамма-випромінювання) — це електромагнітне випромінювання з довжиною хвилі менше одного ангстрема. Утворюється в реакціях за участю елементарних частинок і атомних ядер. У зв’язку з тим, що гамма-промені мають найвищу

проникність серед усіх видів радіації, їх важче захистити. Медицина використовує гамма-випромінювання для лікування пухлин, стерилізації приміщень, апаратури та лікарських препаратів [55].

До характеристик електромагнітного випромінювання відносять:

- довжину хвилі;
- частоту хвилі;
- поляризацію.

Поляризація — це термін, який використовується для опису поперечної анізотропії електромагнітних хвиль. Коли кожна хвиля коливається в одній площині, випромінювання вважається поляризованим.

Часота - це кількість гребенів хвилі, які проходять повз спостерігача (тут детектора) за одну секунду. Вимірюється за допомогою герца.

Найменша відстань між найближчими точками електромагнітного випромінювання, коливання яких відбуваються в одній фазі, називається довжиною хвилі [56].

5.2. Область випромінювання та діаграми спрямованості

4G зараз забезпечує основну передачу даних. Це дозволяє абонентам зі стаціонарним зв’язком передавати інформацію зі швидкістю 1 Гбіт/с для стаціонарних користувачів і 100 Мбіт/с для рухомих користувачів із високою мобільністю. Радіохвилі, які використовуються в поколінні 4G, мають ультракороткий діапазон частот, який варіюється від 30 МГц до 3 ТГц.

У просторі ультракороткі хвилі можуть рухатися лише в прямому світлі. На жаль, це скорочує тривалість їхніх дій. Крім того, структура та атмосфера УКХ впливають на її характер. Тому розповсюдження хвиль буде відрізнятися від розповсюдження сонячної погоди за умов туману або снігу.

Базові станції (БС) використовуються для прийому та передачі радіосигналу. Устаткування БС можна розташувати в нежилих приміщеннях будинку або, оскільки воно невеликого розміру, його можна встановити в контейнери та закріпити на стіні

чи стовпі. Часто радіомодуль можна встановити поруч з антенним блоком, щоб зменшити втрати та розсіювання потужності, що передається через антену.

Макро-, мікро-, піко- та фемтосоти — це типи БС. У них різні розміри, потужність і радіус радіо покриття. Зона покриття кожної БС залежить від багатьох факторів, включаючи висоту підвісу секції антени, рельєф місцевості та кількість перешкод на шляху до абонента. Радіус покриття не завжди є ключовим при установці базової станції.

Для стільникового зв'язку найчастіше використовують секторні панельні антени з діаграмою спрямованості (ДС) шириною 120, 90, 60 і 30 градусів. Три (ширина ДС 120 градусів) або шість (ширина ДС 60 градусів) антенні блоки можуть знадобитися для організації зв'язку в усіх напрямках (від 0 до 360). Типові діаграми спрямованості в логарифмічному масштабі показані на малюнку 5.1.

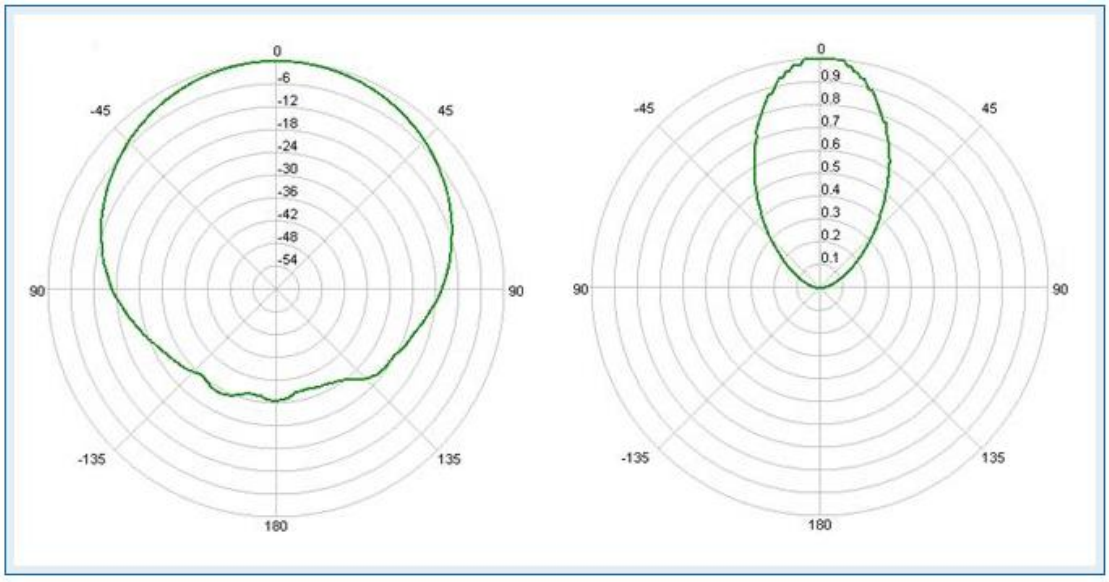

Рис. 5.1. Типові діаграми спрямованості БС в логарифмічному масштабі

Багато антен БС широкосмугові, що дозволяє працювати в діапазонах частот одного, двох або трьох. На відміну від GSM, мережі UMTS мають можливість змінювати площу радіопокриття відповідно до навантаження на мережу. Кут нахилу антени є однією з найефективніших стратегій регулювання випромінюваної потужності; це змінює площу опромінення на діаграмі спрямованості [57].

5.3. Вплив електромагнітного випромінювання на організм людини

Вчені виявили вплив ЕМВ на організм людини. Тим не менш, цей вплив залежить від інтенсивності випромінювання та тривалості впливу. Діапазон радіочастот і особливості організму впливають на швидкість реакції.

Накопичення біологічного ефекту відбувається під час впливу струмів високої та надвисокої частоти, що призводить до функціональних змін у нервовій і серцево-судинній системах і порушень в організмі в різних діапазонах. Крім того, радіохвилі малої інтенсивності мають різну спрямованість. Експерименти показали унікальну чутливість нервової системи, міокард, дистрофічні зміни в сім'яниках і відставання в розвитку тварин.

Таким чином, результати впливу мікрохвиль на організм включають прямий вплив на тканини, зміну основного функціонального стану центральної нервової системи з порушенням нейрогуморальної регуляції та рефлекторні зміни в ряді органів і систем, включаючи серцево-судинну систему [58].

Василь Литвин, лікар з гігієни праці Управління Держпраці у Рівненській області, пояснює, що наслідки впливу електромагнітних випромінювань в діапазоні 30 кГц–300 МГц на організм людини включають загальну слабкість, підвищену втому, порушення сну, головний біль та біль у ділянці серця. Роздратованість, невпевненість і сповільнені рухово-мовні реакції

Ряд симптомів вказує на порушення роботи певних органів, наприклад шлунку, печінки або підшлункової залози. Змінюються харчові та статеві рефлекси, функція серцево-судинної системи, метаболізм білків і вуглеводневих речовин, склад крові та рівень клітин. Підвищення кров'яного тиску та трофічні явища, такі як випадіння волосся та ламкість нігтів, виникають, коли ЕМВ високої та надвисокої частоти систематично впливають на організм людини. ЕМВ викликають зміну поляризації молекул і атомів, які є складовими частинами клітин, що призводить до небезпечного нагріву. Надмірне тепло шкідливо як для окремих органів, так і для всього організму людини [59-60].

Висновки до розділу охорона навколишнього середовища

У розділі Охорона навколишнього середовища розглядаються джерела електромагнітного випромінювання, його види та характеристики. Діаграми спрямованості та області випромінювання показані. Крім того, досліджувався вплив електромагнітного випромінювання на тіло людини. На жаль, електромагнітне забруднення навколишнього середовища є одним із найменш вивчених напрямків у сучасному світі. Причина полягає в тому, що дослідження вимагають тривалого часу та великого фінансування, а також участі великої кількості експертів з різних сфер, таких як біофізика, медицина, біологія та сангігієна, серед інших. З цієї причини досі немає чіткого якісного та кількісного опису шкідливого впливу електромагнітних полів на рослини, тварини та людей. Електромагнітний смог — це сукупність електромагнітних полів і різних радіовипромінювань, що виникають під час роботи складного електронного обладнання — був створений завдяки широкому поширенню електромагнітних випромінювань і їх стрімкому проникненню в усі сфери діяльності людини.

ВИСНОВКИ

Метою кваліфікаційної роботи є дослідження алгоритмів побудови маршрутів, розробка та оптимізація їхньої швидкості виконання для програмно-конфігурованих мереж. Для досягнення зазначеної мети перед роботою було поставлено низку завдань.

Практичне значення отриманих результатів полягає в можливості їх успішного впровадження в реальних програмно-конфігурованих мережах. Це може призвести до покращення продуктивності, зниження витрат та підвищення якості обслуговування. Організації та постачальники послуг зможуть відчути позитивний вплив цих рішень на їх бізнес.

Під час розв'язання задачі "Аналіз проблем побудови маршрутів у програмно-конфігурованих мережах" у роботі було вивчено проблему, важливість і актуальність цього напряму. Під час вивчення було встановлено, що таке завдання є проблемним і його необхідно вирішувати.

Під час розв'язання задачі "Вибір інструментальних засобів для реалізації алгоритму, емуляції та тестування реалізованого алгоритму на базі програмно-конфігурованих мереж" було здійснено детальний аналіз різних середовищ для розробки програмного забезпечення, мережевих операційних систем для ПКМ, різних систем емуляції. У результаті детального аналізу було виявлено, що для використання в роботі підходить середовище розроблення Eclipse CDT, мережева ОС RUNOS і емулятор Mininet.

Під час розв'язання задачі "Розроблення та реалізація алгоритму, тестування на емульованому сегменті програмно-конфігурованої мережі" було здійснено розроблення оптимізованого алгоритму Дейкстри, за допомогою спрощених структур зберігання та урізання зайвих кроків алгоритму в рамках ПКМ. Після цього було здійснено налаштування емулятора Mininet і налаштування RUNOS для взаємодії з емульованим сегментом мережі, для налагодження і тестування коректності роботи.

Під час розв'язання задачі "Апробація алгоритму" було проведено дослідження працездатності алгоритму на базі емульованих сегментів мережі різного розміру, для виявлення переваг і недоліків розробленого оптимізованого алгоритму.

Отримані результати вказують на можливості покращення функціонування мереж та зниження витрат, що є критичними для подальшого розвитку інформаційних технологій та забезпечення ефективної комунікації в сучасному світі. Моя робота відкриває двері для подальших досліджень та розробки нових підходів до оптимізації трафіку та поліпшення функціональності програмно-конфігурованих мереж.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. M. Gharbaoui et al., "Demonstration of Latency-Aware and Self-Adaptive Service Chaining in 5G/SDN/NFV infrastructures," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-2.
2. S. Woo, S. Lee, J. Kim and S. Shin, "RE-CHECKER: Towards Secure RESTful Service in Software-Defined Networking," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-5.
3. C. D. Martino, A. Walid and M. Thottan, "A Cloud-Based Platform Enabling Automation in Resiliency and Performance Testing of SDN," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-2.
4. M. Hajizadeh, T. V. Phan and T. Bauschert, "Probability Analysis of Successful Cyber Attacks in SDN-based Networks," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-6.
5. M. Gharbaoui et al., "Experimenting latency-aware and reliable service chaining in Next Generation Internet testbed facility," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-4.
6. "IEEE Draft Recommended Practice for Software Defined Networking (SDN) based Middleware for Control and Management of Wireless Networks," in P1930.1/D1, December 2021 , vol., no., pp.1-136.
7. A. Pacini, D. Scano, L. Valcarenghi, A. Sgambelluri and A. Giorgetti, "Enabling event-based hierarchical synchronization in SDN ONOS clusters," 2022 IEEE

Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Phoenix, AZ, USA, 2022, pp. 92-93.

8. F. Meneses, D. Corujo, A. Neto and R. L. Aguiar, "SDN-based End-to-End Flow Control in Mobile Slice Environments," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-5.
9. J. Narantuya et al., "SDN-Based IP Shuffling Moving Target Defense with Multiple SDN Controllers," 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Supplemental Volume (DSN-S), Portland, OR, USA, 2019, pp. 15-16.
10. M. B. Jimenez and D. Fernandez, "A Framework for SDN Forensic Readiness and Cybersecurity Incident Response," 2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Phoenix, AZ, USA, 2022, pp. 112-116.
11. P. Trúchly and J. Kubica, "Communication Networks with Multiple SDN Controllers," 2023 International Symposium ELMAR, Zadar, Croatia, 2023, pp. 29-32.
12. M. S. Jamal, A. Hirwe and K. Kataoka, "VIBHAJAN: A Lightweight and Scalable Control Plane Management for Multi-Controller SDN," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-7.
13. J. R. Dora and L. Hluchy, "Detection of Attacks in Software-Defined Networks (SDN)* : *How to conduct attacks in SDN environments," 2023 IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 2023, pp. 000623-000630.
14. Z. Yang and K. L. Yeung, "SDN Candidate Selection in Hybrid IP/SDN Networks for Single Link Failure Protection," in IEEE/ACM Transactions on Networking, vol. 28, no. 1, pp. 312-321.

15. R. S. Kalan, M. Sayit and A. C. Begen, "Implementation of SAND Architecture Using SDN," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-6.
16. L. Csikor, M. Szalay, G. Rétvári, G. Pongrácz, D. P. Pezaros and L. Toka, "Transition to SDN is HARMLESS: Hybrid Architecture for Migrating Legacy Ethernet Switches to SDN," in IEEE/ACM Transactions on Networking, vol. 28, no. 1, pp. 275-288.
17. D. Gedia and L. Perigo, "Performance Evaluation of SDN-VNF in Virtual Machine and Container," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-7.
18. N. Li, Y. Shi, Z. Zhang, J. Fernan Martinez and X. Yuan, "Search-tree Based SDN Candidate Selection in Hybrid IP/SDN Network," 2020 IEEE 28th International Conference on Network Protocols (ICNP), Madrid, Spain, 2020, pp. 1-6.
19. Y. Zhao and X. Zhang, "New media identity authentication and traffic optimization in 5G network," 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2017, pp. 1331-1334.
20. S. Troia, A. Rodriguez, R. Alvizu and G. Maier, "SENATUS: An Experimental SDN/NFV Orchestrator," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-5.
21. K. V. M. Mohan, S. Kodati and V. Krishna, "Securing SDN Enabled IoT Scenario Infrastructure of Fog Networks From Attacks," 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), Coimbatore, India, 2022, pp. 1239-1243.
22. L. Mamushiane, J. Mwangama and A. A. Lysko, "Given a SDN Topology, How Many Controllers are Needed and Where Should They Go?," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-6.

23. P. Amaral, P. F. Pinto, L. Bernardo and A. Mazandarani, "Application Aware SDN Architecture using Semi-supervised Traffic Classification," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-6.
24. G. Xilouris et al., "Towards Autonomic Policy-based Network Service Deployment with SLA and Monitoring," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-2.
25. J. Baranda, I. Pascual, M. Requena and J. Mangues-Bafalluy, "Deploying a containerized ns-3/LENA-based LTE mobile Network Service through the 5G-TRANSFORMER platform," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-2.
26. R. Huang, "A SDN-Based Traffic Information Acquirement Approach in Internet of Things," 2019 IEEE International Conference on Industrial Internet (ICII), Orlando, FL, USA, 2019, pp. 109-114.
27. A. Guo, C. Yuan, G. He and L. Xu, "Research on SDN/NFV Network Traffic Management and Optimization based on Big Data and Artificial Intelligence," 2018 18th International Symposium on Communications and Information Technologies (ISCIT), Bangkok, Thailand, 2018, pp. 377-382.
28. F. Faheem, Z. Zuraidah, A. Kayani and A. K. Aminuddin, "Optimization of vehicle actuation and multiplan algorithms for urban traffic control systems," 2017 IEEE Conference on Systems, Process and Control (ICSPC), Meleka, Malaysia, 2017, pp. 59-64.
29. G. Choudhury, G. Thakur and S. Tse, "Joint Optimization of Packet and Optical Layers of a Core Network using SDN Controller, CD ROADMs and Machine-Learning-Based Traffic Prediction," 2019 Optical Fiber Communications Conference and Exhibition (OFC), San Diego, CA, USA, 2019, pp. 1-3.

30. A. Kelkawi, A. Mohammed and A. Alyatama, "Incremental Deployment of Hybrid IP/SDN Network with Optimized Traffic Engineering," 2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Leganes, Spain, 2020, pp. 57-63.
31. M. Gharbaoui et al., "Demonstration of Latency-Aware and Self-Adaptive Service Chaining in 5G/SDN/NFV infrastructures," 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Verona, Italy, 2018, pp. 1-2.
32. S. Tomovic and I. Radusinovic, "A new traffic engineering approach for QoS provisioning and failure recovery in SDN-based ISP networks," 2018 23rd International Scientific-Professional Conference on Information Technology (IT), Zabljak, Montenegro, 2018, pp. 1-4.
33. S. Xu, X. Wang, G. Yang, J. Ren and S. Wang, "Routing optimization for cloud services in SDN-based Internet of Things with TCAM capacity constraint," in Journal of Communications and Networks, vol. 22, no. 2, pp. 145-158.
34. K. P. Kadiyala and J. A. Cobb, "Inter-AS traffic engineering with SDN," 2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Berlin, Germany, 2017, pp. 1-7.
35. X. Wang et al., "The Joint Optimization of Online Traffic Matrix Measurement and Traffic Engineering For Software-Defined Networks," in IEEE/ACM Transactions on Networking, vol. 28, no. 1, pp. 234-247.
36. C. J. Miguel, F. J. B. V. Neto, J. A. Santos and P. N. M. Sampaio, "Data collection in SDN networks with contextual analysis," 2019 14th Iberian Conference on Information Systems and Technologies (CISTI), Coimbra, Portugal, 2019, pp. 1-6.
37. M. S. Hossen, M. H. Rahman, M. Al-Mustanjid, M. A. Shakil Nobin and M. A. Habib, "Enhancing Quality of Service in SDN based on Multi-path Routing Optimization with DFS," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), Dhaka, Bangladesh, 2019, pp. 1-5.

38. S. Li and Z. Long, "Data Center Traffic Rescheduling Algorithm Based on Ant Colony Optimization Algorithm," 2023 12th International Conference of Information and Communication Technology (ICTech), Wuhan, China, 2023, pp. 98-102.
39. S. A. Kashinath et al., "Review of Data Fusion Methods for Real-Time and Multi-Sensor Traffic Flow Analysis," in IEEE Access, vol. 9, pp. 51258-51276.
40. S. Xiaowei and W. Jun, "Research on SDN Network Structure Optimization System Based on Computer 5G Technology," 2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA), Shenyang, China, 2023, pp. 1754-1758.
41. J. Dafda and M. Subhedar, "Dynamic Load balancing in SDN using Energy Aware Routing and Optimization Algorithm," 2022 IEEE Bombay Section Signature Conference (IBSSC), Mumbai, India, 2022, pp. 1-6.
42. A. K. Al Mhdawi, A. T. Azar, N. A. Kamal and C. Ben Njima, "Intelligent OpenFlow Switch for SDN Networks Based on COVID-19's High Network Traffic using Heuristic GA-Fuzzification Control Approach," 2022 International Conference on Control, Automation and Diagnosis (ICCAD), Lisbon, Portugal, 2022, pp. 1-6.
43. M. Tanha, D. Sajjadi, R. Ruby and J. Pan, "Traffic Engineering Enhancement by Progressive Migration to SDN," in IEEE Communications Letters, vol. 22, no. 3, pp. 438-441.
44. S. Tse and G. Choudhury, "Real-Time Traffic Management in AT&T's SDN-Enabled Core IP/Optical Network," 2018 Optical Fiber Communications Conference and Exposition (OFC), San Diego, CA, USA, 2018, pp. 1-3.
45. S. Miao, Y. Li and Q. Pan, "A Preliminary Study of UAV Cyber Traffic Playback Based on SDN," 2023 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), Beijing, China, 2023, pp. 1-5.
46. Методичні вказівки до виконання розділу «Охорона праці» в дипломних проектах і роботах. Для студентів всіх спеціальностей освітньо-кваліфікаційних

рівнів «спеціаліст» та «магістр». /Укладачі: О.І Запорожець, А. В. Русаловський. – К.: НАУ, 2011. –30с.

47. ДБН В.2.2-28:2010 «Будинки адміністративного та побутового призначення».
48. ДСТУ 12.0.003-74 Система стандартів безпеки праці (ССБТ). Небезпечні і шкідливі виробничі фактори. Класифікація
49. ДБН В.2.5-28-2018 «Природне і штучне освітлення».
50. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих приміщень
51. Наказ Міністерства соціальної політики України від 14.02.2018 № 207, зареєстрованого в Міністерстві юстиції України 25 квітня 2018 року за № 508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями»
52. НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні».
53. ДСТУ Б В.1.1-36:2016 Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою.
54. Електромагнітне випромінювання [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Електромагнітне_випромінювання.
55. Як виявити небезпечне випромінювання [Електронний ресурс] - Режим доступу до ресурсу: https://rostec.org/news/4517502.
56. Коротка характеристика видів електромагнітного випромінювання [Електронний ресурс] - Режим доступу до ресурсу: https://ekspertiza.com.ua/.
57. Електромагнітне випромінювання - визначення, різновиди, характеристики [Електронний ресурс] - Режим доступу до ресурсу: https://pue8.org/elektrotekhnik/.
58. Денисов Д. Базові станції стільникового зв'язку та їх антенна частина [Електронний ресурс] / Д. Денисов. - 2016. - Режим доступу до ресурсу: https://nag.org/articles/article/29957/.
59. Професійні хвороби: навчальний посібник для студ. вищ. навч. закладів, які навчаються за спеціальністю "Безпека життєдіяльності" / авт. укл. Т. Я. Біндюк, О. В. Бессчетнова. - Балашов: Миколаїв, 2007. - 128 с.

60. Негативний вплив електромагнітних полів на людину [Електронний ресурс] – Режим доступу до ресурсу: https://oppb.com.ua/articles/negatyvnyy-vplyv-elektromagnitnyh-poliv-na-lyudynu.