Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Метод забезпечення зелених обчислень на базі програмно-конфігурованих мереж
Назва теми

Рівень вищої освіти *другий (магістерський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 240125.24.01.25 ПЗ

Виконав здобувач II курсу, група КІ2м-24-1 _____ Максим СОЙКО
Підпис Ініціали, прізвище

Керівник д. техн. наук, професор _____ Сергій ЛИСЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер д. техн. наук, професор _____ Сергій ЛИСЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС _____ Ольга ПАВЛОВА
«01» травня 2026 р. підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ДРУГИЙ (МАГІСТЕРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІПС
_____ Ольга ПАВЛОВА

" 12 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сойку Максиму Олександровичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Метод забезпечення зелених обчислень на базі програмно-конфігурованих мереж

Керівник проекту (роботи) Лисенко Сергій Миколайович, д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 12.01.2026 р. № 6

2. Термін подання здобувачем роботи на кафедру 15.05.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз технологій забезпечення зелених обчислень на базі програмно-конфігурованих мереж та постановка задачі дослідження

Проєктування математичної моделі та методу динамічної консолідації трафіку для мінімізації енергоспоживання в інфраструктурі SDN

Розроблення алгоритмічного забезпечення інтелектуального модуля енергозберігаючої маршрутизації (EMEKT) із захистом від мережевих аномалій

Програмна реалізація розробленого модуля, проведення імітаційного моделювання та аналіз ефективності отриманих результатів

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання « 12 » 01 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проєкту (роботи)	Термін виконання етапів проєкту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	12.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	15.01.2026	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	01.02.2026	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	01.03.2026	виконано
5	Робота над науковою статтею	29.03.2026	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	03.04.2026	виконано
7	Робота над розділом 4 – проєктування та розробка програмного забезпечення для вирішення поставленої задачі, експериментальна частина	13.04.2026	виконано
8	Оформлення пояснювальної записки згідно вимог	20.04.2026	виконано
9	Попередній захист ВКР	27.04.2026	виконано
10	Захист ВКР на засіданні ЕК	До 20.05.2026	

Здобувач _____ Максим СОЙКО
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Сергій ЛИСЕНКО
Підпис Ім'я, ПРІЗВИЩЕ

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Метод забезпечення зелених обчислень на базі програмно-конфігурованих мереж

Автор роботи: Максим СОЙКО

Керівник роботи: д.т.н., професор Сергій ЛИСЕНКО

Пояснювальна записка: 82 с., 21 рис., 3 табл., 3 дод., 71 джерело.

ПРОГРАМНО-КОНФІГУРОВАНІ МЕРЕЖІ, ЗЕЛЕНІ ОБЧИСЛЕННЯ, ЕНЕРГОЕФЕКТИВНІСТЬ, МЕТОД КОНСОЛІДАЦІЇ ТРАФІКУ, МАРШРУТИЗАЦІЯ, ЯКІСТЬ ОБСЛУГОВУВАННЯ, ІЄРАРХІЧНА МЕРЕЖЕВА ТОПОЛОГІЯ

Об’єктом дослідження є процес управління ресурсами та споживанням електроенергії в сучасних комп’ютерних мережах центрів обробки даних.

Предметом дослідження є властивості та закономірності застосування технологій програмно-конфігурованих мереж для впровадження алгоритмів енергоефективної консолідації трафіку.

Метою кваліфікаційної роботи магістра є розробка методу та інтелектуальної системи динамічного управління мережевою інфраструктурою на базі SDN-контролера для зниження загальних енерговитрат при збереженні цільових показників якості обслуговування.

Для розв’язання поставлених задач використовувалися апарат теорії графів для моделювання топологій, методи математичного моделювання систем масового обслуговування та імітаційне мережеве моделювання у середовищі Mininet.

Наукова новизна отриманих результатів полягає у вдосконаленні евристичного методу енергоефективної консолідації трафіку через використання динамічного перерахунку ваг ребер графа з урахуванням штрафних коефіцієнтів за апаратне пробудження портів, а також інтеграцію механізму ізоляції широкомовних штормів на основі алгоритму мінімального

кістякового дерева.

Набула подальшого розвитку архітектура інтелектуальної системи управління мережею, яка реалізує замкнутий цикл моніторингу навантаження та превентивного перемикання станів мережевого обладнання.

Практична значимість отриманих результатів полягає у програмній реалізації модуля управління на базі SDN-контролера Ryu, що дозволяє знизити енергоспоживання інфраструктури на 50-55% при збереженні стабільної пропускної здатності та нульових втратах пакетів.

У першому розділі проведено аналіз предметної області, досліджено проблему зростання енергоспоживання у секторі ІКТ та обґрунтовано переваги використання SDN для реалізації концепції «зелених обчислень».

У другому розділі побудовано математичну модель енергоспоживання мережевої інфраструктури, формалізовано задачу мінімізації цільової функції та розроблено алгоритмічну логіку методу EMEKT.

У третьому розділі спроєктовано архітектуру програмно-технічного засобу, описано логічні блоки системи моніторингу та управління ресурсами, а також визначено формат взаємодії за протоколом OpenFlow.

У четвертому розділі на основі розробленого методу проведено серію експериментів на ієрархічній деревоподібній топології, в ході яких підтверджено ефективність системи у різних сценаріях навантаження та її здатність підтримувати стабільний рівень QoS.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЕМЕКТ – евристичний метод енергоефективної консолідації трафіку

ІКТ – інформаційно-комунікаційні технології

ЦОД – центр обробки даних

ALR (Adaptive Link Rate) – адаптивна швидкість передачі даних у каналі

API (Application Programming Interface) – програмний інтерфейс додатка

AQROM (Adaptive Quality-aware Resource and Offloading Management) – адаптивне управління ресурсами та розвантаженням з урахуванням якості обслуговування

ARP (Address Resolution Protocol) – протокол визначення адрес

BGP (Border Gateway Protocol) – протокол граничного шлюзу

CARPO (Correlation-Aware Power Optimization) – кореляційно-орієнтована оптимізація енергоспоживання

CCF (Cooling Capacity Factor) – коефіцієнт потужності охолодження

CUE (Carbon Usage Effectiveness) – ефективність використання вуглецю

DoS (Denial of Service) – атака типу «відмова в обслуговуванні»

DPID (Datapath Identifier) – унікальний ідентифікатор комутатора в SDN

DRL (Deep Reinforcement Learning) – глибоке навчання з підкріпленням

ECMP (Equal-Cost Multi-Path) – маршрутизація по декількох шляхах рівної вартості

IoE (Internet of Everything)– всеохопний інтернет

IoT (Internet of Things) – інтернет речей

IP (Internet Protocol) – міжмережевий протокол

LLDP (Link Layer Discovery Protocol) – протокол виявлення канального рівня

LPI (Low Power Idle) – режим очікування з низьким енергоспоживанням

MAC (Media Access Control) – унікальний ідентифікатор мережевого обладнання

PUE (Power Usage Effectiveness) – ефективність використання електроенергії

QoS (Quality of Service) – якість обслуговування

RAM (Random Access Memory) – оперативна пам'ять

SDN (Software-Defined Networking) – програмно-конфігуровані мережі

SLA (Service Level Agreement) – угода про рівень послуг

TCAM (Ternary Content-Addressable Memory) – тернарна контентно-адресована пам'ять

TCP (Transmission Control Protocol) – протокол керування передачею

UDP (User Datagram Protocol) – протокол користувацьких датаграм

ВСТУП

Сучасна тенденція розгортання мереж 5G/6G та систем IoE зумовлює стрімке зростання обсягів трафіку, що безпосередньо впливає на енергоспоживання ІКТ-сектору. Прогнозується, що до 2030 року частка галузі в загальносвітовому споживанні електроенергії може сягнути 20%. Класична мережева архітектура фундаментально негнучка. Принцип «надлишкового резервування» змушує залізо постійно працювати на межі можливостей. Як результат – обладнання функціонує на повну потужність навіть за нульового трафіку. Сучасні пристрої ігнорують принципи енергопропорційності. Типовий комутатор марнує до 90% енергії, просто очікуючи на пакет даних. В епоху Edge Computing така статичність стає фатальною для бюджетів та екології. Впровадження парадигми «зелених мереж» (Green Networking) – це більше не екологічний тренд. Це жорстка технічна вимога для стабільного розвитку систем.

Аналіз наукових праць Луки К'яравільйо [13, 52], Саймона Сінгха [18], Клейтона Крістенсена [33] та інших свідчить про наявність ефективних базових рішень для Ethernet-інфраструктур, зокрема стандарту IEEE 802.3az [13, 82]. Однак сучасні масштаби ЦОД вимагають динамічної адаптації всієї топології мережі в реальному часі. Технологія SDN дозволяє реалізувати такий підхід завдяки відокремленню площини керування від площини передачі даних. Це дає можливість SDN-контролеру централізовано консолідувати трафік на мінімальній кількості активних вузлів і переводити надлишкові ресурси в режими низького енергоспоживання. Головним технічним викликом залишається підтримка цільових показників QoS при зміні станів живлення обладнання.

Метою дослідження є оптимізація енергоспоживання мережевої інфраструктури центрів обробки даних при забезпеченні заданого QoS шляхом розробки методу та програмно-технічних засобів забезпечення «зелених обчислень» на базі програмно-конфігурованих мереж.

Об’єктом дослідження є процес управління ресурсами та споживанням електроенергії в сучасних комп’ютерних мережах, що передбачає детальний аналіз енергетичних станів вузлів залежно від профілю мережевого навантаження.

Предметом дослідження є метод та засоби забезпечення зелених обчислень на базі програмно-конфігурованих мереж.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1) проаналізувати та систематизувати сучасний стан проблеми енергоспоживання в мережах, а також існуючі підходи до реалізації «зелених» програмно-конфігурованих мереж;

2) обґрунтувати вибір архітектурних компонентів технології програмно-конфігурованих мереж як інструменту для оптимізації енергозбереження мережевої інфраструктури;

3) сформувати математичні та алгоритмічні моделі для забезпечення динамічної зміни топології та ефективної консолідації трафіку;

4) розробити програмне забезпечення для імітаційного моделювання запропонованих рішень у середовищі Mininet і провести оцінку їхньої результативності.

Практична цінність отриманих результатів виходить далеко за межі суто академічного інтересу, оскільки запропоновані алгоритми мають високий інтеграційний потенціал для модернізації сучасних високонавантажених систем. Перш за все, мова йде про центри обробки даних та масштабні хмарні платформи, де впровадження SDN-рішень дозволяє перетворити енергоспоживання з некерованої статті витрат на прогнозований і оптимізований ресурс. Шляхом інтелектуальної консолідації трафіку на обмеженій кількості активних вузлів стає можливим не лише пряме заощадження електрики, а й суттєве покращення показника PUE. Це відбувається завдяки каскадному ефекту: менша кількість активного «заліза» генерує менше тепла, що, своєю чергою, знижує навантаження на системи прецизійного кондиціонування та охолодження.

Для корпоративних мереж великих підприємств цей метод відкриває шлях до побудови адаптивної інфраструктури, яка фактично «підлаштовується» під робочий

графік організації. Замість того, щоб утримувати всю мережеву периферію в режимі пікової готовності цілодобово, програмне керування дозволяє автоматично переводити надлишкові комутаційні сегменти у глибокий режим очікування під час низької активності або в неробочі години. Такий підхід радикально трансформує структуру операційних витрат, перетворюючи мережу з пасивного споживача на гнучкий інструмент, що відповідає сучасним стандартам Green IT.

Зрештою, впровадження таких рішень забезпечує підприємствам вагому конкурентну перевагу в контексті глобальної декарбонізації. Мінімізація вуглецевого сліду цифрової інфраструктури сьогодні є не просто декларативним кроком до екологічності, а жорсткою економічною вимогою, що впливає на інвестиційну привабливість та відповідність міжнародним екологічним протоколам. Таким чином, розроблений метод стає фундаментом для створення технологічно стійких та фінансово виправданих систем майбутнього, де висока продуктивність не суперечить принципам розумного споживання ресурсів.

Виконана дипломна робота має взаємозв’язок з іншими науковими напрямками та роботами. Дослідження базується на перетині методів оптимізації маршрутизації, системного аналізу та технологій віртуалізації. На відміну від класичного підходу пошуку найкоротшого шляху, у роботі фокус зміщено на побудову енергоефективних траєкторій передачі даних, де системний аналіз параметрів QoS виступає інструментом контролю балансу між пропускною здатністю та енергозбереженням.

Технічним підґрунтям методу є концепції віртуалізації мережевих функцій та програмного абстрагування апаратних ресурсів, що є необхідною умовою для реалізації динамічного керування в SDN-мережах. Для розв'язання поставлених задач використано апарат теорії графів з метою моделювання топологій та методи математичного моделювання систем масового обслуговування для прогнозування поведінки мережі під навантаженням. Програмна реалізація алгоритмів здійснена мовою Python із використанням фреймворків Ryu/ONOS та середовища емуляції Mininet.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ

1.1 Аналіз предметної області і виявлення наявних проблем і завдань

Розвиток цифрової економіки тісно пов'язаний із впровадженням 5G/6G, IoT і хмарних обчислень. Ці технології створюють величезні обсяги даних. Для прикладу, у 2018 році було 33 зетабайти, а до 2030 року очікується 175 зетабайтів. Поширення розумних пристроїв, потокового відео високої якості, віртуальної реальності та автономних систем змушує постійно збільшувати потужності мереж для обробки даних майже миттєво, що зумовлює експоненціальний тренд накопичення інформації (рис. 1.1) [8, 9, 68].

Рисунок 1.1 – Зростання глобального обсягу даних (згенеровано за допомогою NotebookLM)

Через збільшення інфраструктури, зростає споживання енергії сектором ІКТ. Зараз глобальна ІТ-індустрія відповідає за близько 2% світових викидів парникових газів, як і вся авіація [18, 51]. Передбачається, що споживання енергії ІКТ буде рости приблизно на 5% щороку, і до 2030 року може споживати до 21% всієї

світової електроенергії [31, 60].

У структурі споживання енергії центрами обробки даних зазвичай звертають увагу на сервери, але мережеве обладнання також важливе. Згідно з дослідженнями науковців Дер'ї Чавдар [16] та Фатіха Алагоза [35], енергоспоживання типової мережі центру обробки даних становить від 10% до 30% від загального енергоспоживання ЦОД, тоді як решта енергії витрачається безпосередньо на сервери та охолодження [16, 35]. Типовий розподіл енерговитрат у сучасному дата-центрі з акцентом на ключові вузли споживання представлено на рисунку 1.2.

Рисунок 1.2 – Структура споживання електроенергії в сучасному центрі обробки даних (згенеровано за допомогою NotebookLM)

Інші наукові праці підтверджують цю статистику, вказуючи, що на

комунікаційні мережеві пристрої (комутатори, маршрутизатори та канали зв'язку) припадає від 10% до 20% (в середньому 15-20%) від загального обсягу електроенергії, спожитої дата-центрами [13, 54]. При цьому значна частина цієї енергії витрачається марно, оскільки традиційні мережі проєктуються з розрахунком на пікові навантаження, і мережеве обладнання працює на повну потужність навіть у періоди мінімального трафіку чи простою. Неможливість адаптації енергоспоживання до реального обсягу трафіку в традиційних архітектурах зумовлена жорсткою зв'язкою апаратного забезпечення та логіки управління, що вимагає переходу до парадигми програмно-конфігурованих мереж [29, 33, 57].

Програмно керовані мережі працюють ефективніше через окремий блок керування й потоку даних. Цей розподіл дає можливість тримати всю логіку в одному місці – у спеціальному контролері, який має глобальне бачення всієї топології та стану мережі в реальному часі [12, 38]. Аналізуючи загальну картину, такий контролер помічає, де обладнання ненавантажене або слабко навантажене. Тоді він переключає трафік, щоб менше споживати енергії. На відміну від старих систем, де кожен комутатор приймає рішення ізольовано, тут все координується. Через відкриті інтерфейси налаштування стають простішими, особливо коли необхідно економити живлення [29, 50].

З огляду на теперішні дослідження в галузі енергоефективних SDN стає зрозумілим – підходи до зменшення споживання електроенергії найчастіше об'єднують у трьох ключових напрямах.

Методи, які орієнтовані на трафік базуються на консолідації трафіку. Через те, що дані йдуть лише там, де справді треба, зайве обладнання можна вимкнути назавжди. Не всі пристрої працюють постійно: багато хто переходить у сон, коли немає завдань. Це дозволяє економити енергію без втрат у продуктивності. Найвідоміший приклад такого підходу – система під назвою ElasticTree [25, 54].

Методи, які орієнтовані на кінцеві системи фокусуються на віртуалізації та міграції віртуальних машин. Замість того щоб працювати постійно, деякі підходи враховують стан окремих машин. Вони переміщують віртуальні системи між

фізичними комп’ютерами. У періоди низького навантаження завдання консолідуються на одному пристрої, а решта обладнання разом із блоками живлення вимикається [11, 40].

Методи оптимізації розміщення правил спрямовані на зменшення використання енергоємної пам'яті TCAM у комутаторах SDN шляхом стиснення та оптимізації таблиць потоків [13, 33]. Такий підхід дозволяє мінімізувати звернення до швидкої, проте ресурсомісткої пам’яті. Результат – нижче споживання енергії через раціональніший порядок записів. Комутатори працюють ефективніше, бо навантаження на апаратну частину зменшується. Правила переставляються за певними схемами, які враховують їхню популярність і структуру. Часто використовувані записи стають доступнішими, рідкісні – не блокують цінний простір. Стиснення дає можливість тримати більше логіки в межах одного чіпа. Економія пам’яті виявляється критично важливою для масштабних мереж. Щільність розташування впливає на продуктивність у реальному часі.

У наведеній нижче таблиці представлено порівняльний аналіз стратегій енергоефективності у програмно-визначених мережах (табл. 1.1).

Таблиця 1.1 – Порівняння стратегій Green Networking в SDN

Категорія методу	Основний механізм	Переваги	Ризики
Методи, які орієнтовані на трафік	Консолідація потоків на активних шляхах	Висока економія енергії на рівні всієї топології	Збільшення затримок
Методи, які орієнтовані на кінцеві системи	Міграція віртуальних машин	Зменшення споживання серверної стійки	Висока складність синхронізації

Кінець таблиці 1.1

Категорія методу	Основний механізм	Переваги	Ризики
Методи оптимізації розміщення правил	Стиснення таблиць потоків	Оптимізація пам'яті комутаторів	Обмежений потенціал економії енергії

Хоча підходи, орієнтовані на трафік, мають великий резерв, їх впровадження стикається з критичною проблемою – коли економія струму шкодить якості з’єднань. Агресивне вимкнення комутаторів або переведення каналів у сплячий режим для максимізації економії енергії стає причиною виникнення технічних складнощів у забезпеченні безперебійного зв'язку.

По-перше, збільшення затримок через необхідність перемаршрутизації та "пробудження" обладнання (Wake on LAN). Іноді пристрої вмикаються повільніше – треба чекати, доки мережа знайде новий шлях. Прокидання за допомогою сигналу з мережі тягне за собою затримки. Коли маршрут міняється, час реакції падає. Сигнал до запуску проходить не одразу – спочатку йде переадресація.

По-друге, зниження пропускної здатності та підвищення ризику втрати пакетів у разі раптових сплесків трафіку через перевантаження активних, консолідованих каналів. Коли навантажені канали отримують несподіваний потік даних, пропускна здатність падає – це часто призводить до втрат пакетів. Через об’єднані лінії передачі такий стрибок створює переповнення. Замість стабільної трансляції трафіку виникають затримки та деградація пакетів, що призводить до раптового розриву мережевих сесій.

По третє, зниження надійності мережі, оскільки вимкнення надлишкових шляхів залишає мережу вразливою до апаратних збоїв. Якщо прибрати дублюючі з'єднання – система стає не такою міцною. Натомість один зламаний пристрій може порушити всю роботу. Загалом, використання альтернативних маршрутів дозволяє підтримувати безперервність роботи мережі у разі виходу обладнання з ладу.

Одночасно підвищити ефективність споживання електроенергії й зберегти якість передавання даних – складне завдання. Часто запропоновані способи просто заощаджують енергію, хоча при цьому не дають стабільного часу доставки важливих пакетів. А ще деякі методи навантажують контролер сильно, бо потребують багато потужностей для роботи, тож швидке реагування на зміни в мережі стає неможливим.

Аналіз предметної області свідчить про те, що навіть при багатьох наявних роботах, деякі важливі аспекти енергоефективності в програмно-керованих мережах досі не з’ясовані. Такі задачі вимагають уважного аналізу, адже сучасні мережі занадто складні для старих, жорстких методів. Хоча вже є певний прогрес, повна картина того, як системи економії енергії працюють із мінливими потоками даних, ще не побудована – тому дослідження в цьому напрямку продовжуються.

Цю задачу можна вирішити, якщо запустити SDN-мережі. Через роз'єднання керуючого шару і передавального каналу контролер отримує повну картину топології – ось чому стають реальними схеми для групування трафіку з урахуванням навантаження. Трафік спрямовується оптимізованими маршрутами через мінімальну кількість вузлів, що дозволяє вимкнути незадіяні порти [12, 38].

Головним викликом залишається компроміс: заходи з енергозбереження часто призводять до деградації показників якості обслуговування. Коли пристрої вимикають занадто часто, запускати їх доводиться довго, а це гальмує передачу даних. Іншим критичним недоліком є ризик втрати пакетів через різке зростання навантаження. У багатьох сучасних системах головна мета – економія, проте це не гарантує безперебійного функціонування критично важливих сервісів.

Через це важливо шукати нові способи для SDN, що дозволяють гнучко змінювати структуру мережі разом із розумним керуванням потоками даних. Натомість слід спиратися на точний баланс: енергоефективність (вимикання портів або приладів), проте без втрати якості обслуговування. Використання інтелектуальних алгоритмів замість фіксованих сценаріїв гарантує стійкість мережі до непередбачуваних сплесків трафіку. Цей підхід, урешті-решт, виявляється найближчим до практичних потреб сьогодення: висока продуктивність має йти

поряд із дбайливим ставленням до довкілля та раціональним використанням коштів.

1.2 Порівняльний аналіз переваг та недоліків існуючих рішень

Складнощі у створенні енергоефективних програмно-керованих мереж часто виникають через протиріччя – треба одночасно економити електроенергію й дотримуватись норм продуктивності: швидкості передачі, часу відгуку чи кількості загублених даних. Поглянувши на сучасні досягнення, можна помітити кілька ключових підходів до вирішення цього завдання. Разом із тим, кожен спосіб несе за собою особливі сильні сторони й серйозні обмеження.

Серед способів покращення енергоефективності у мережах поширеними є стратегії, спрямовані на аналіз трафіку для раціонального маршрутизаційного планування. В основі лежить принцип адаптивного використання ресурсів, де кількість активних вузлів мінімізується відповідно до зниження інтенсивності трафіку. Вони намагаються виявити найменший набір комутаторів і ліній, який здатний забезпечити стабільну передачу наявного обсягу трафіку без порушення встановлених вимог щодо пропускної здатності.

Серед популярних інструментів такого роду особливо вирізняється ElasticTree – фреймворк, створений спеціально під архітектуру дата-центрів. Замість простої передачі трафіку, він гнучко регулює потоки, одночасно вимикаючи неактивне обладнання для економії електроенергії. У мережах із ієрархічною деревоподібною топологією, що мають багаторівневу структуру й численні резервні шляхи, метод добре себе показує, скорочуючи споживання енергії при слабкому навантаженні майже наполовину [25]. Але практичне застосування ElasticTree в умовах великих мереж обмежене низькою масштабованістю: визначення найкращих активних каналів забирає занадто багато часу. Так само важливим недоліком є те, що взаємозв’язки між окремими потоками даних залишаються без уваги, що загрожує перевантаженнями під час раптових сплесків трафіку [54]. На рисунку 1.3, який знаходиться нижче, наведена схема, на

якій відображена архітектура системи ElasticTree, що складається з оптимізатора, модуля маршрутизації та модуля керування живленням.

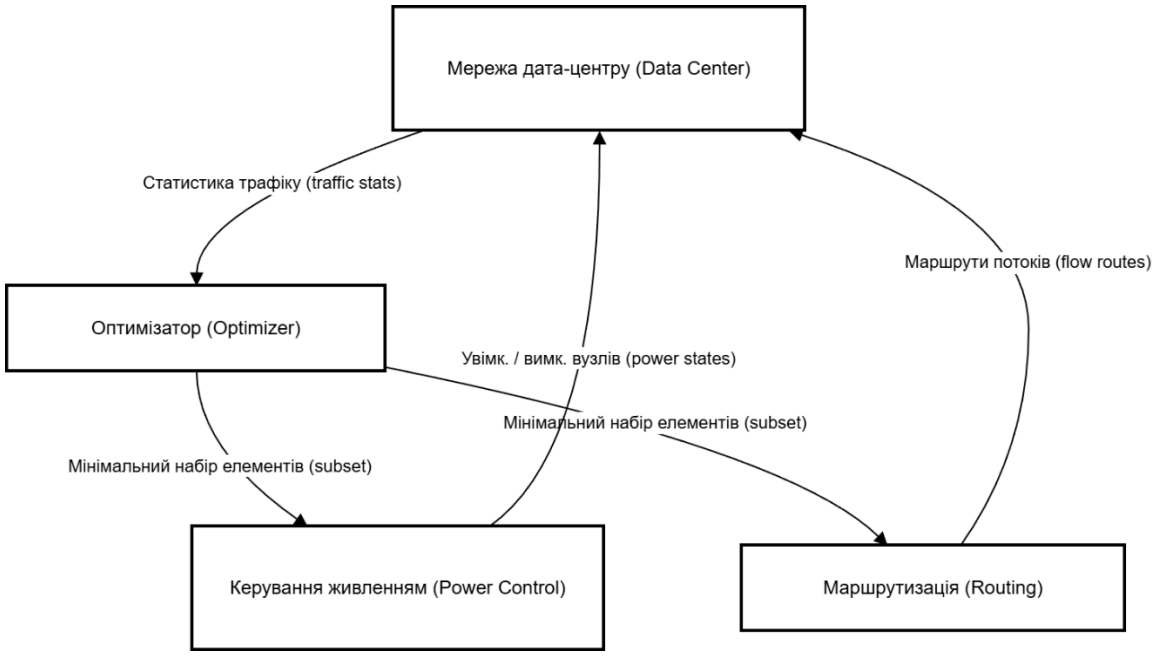


Рисунок 1.3 – Архітектура системи ElasticTree

Для усунення зазначених недоліків було запропоновано алгоритм CARPO – оптимізацію споживання енергії з урахуванням взаємозв’язків між потоками. Цей метод об'єднує трафік після вивчення статистики: максимальне навантаження на окремі частини мережі зазвичай не збігається за часом. Завдяки цьому можна здійснювати більш щільне компонування трафіку (досягаючи економії на рівні 46%) без критичного ризику переповнення каналів [52]. Але надто активне злиття потоків перевантажує ті канали, що лишились у роботі. Наслідком є накопичення пакетів у чергах маршрутизаторів, що підвищує ризик їх переповнення. Це робить використання CARPO малоефективним для сервісів, чутливих до часових показників, таких як передача мультимедійного контенту в реальному часі [12, 39].

Окрім методів, що фокусуються виключно на маршрутизації, існують комплексні багаторівневі та апаратні підходи, серед яких найбільш показовим є фреймворк GreenSDN [25, 52]. Його ключова особливість полягає в одночасній роботі на трьох ієрархічних рівнях: фізичному, системному та мережевому.

Управління мережею забезпечує система сталого управління мережею. Вона стежить за всією інфраструктурою, одночасно регулюючи енергоефективні параметри вузлів завдяки оцінці загальних моделей трафіку. Системний рівень (рівень вузла) реалізує механізм Synchronized Coalescing, суть якого полягає в інтелектуальній буферизації вхідних пакетів. Дані прибувають не безперервно, а групуються у черги для опрацювання порціями. Такий підхід дає можливість обладнанню довше залишатися в економному режимі між активними сеансами. На найнижчому рівні (рівні чіпа) застосовується протокол ALR або LPI, що дозволяє динамічно змінювати швидкість передачі даних у каналі (наприклад, перемикатися з 1 Гбіт/с на 100 Мбіт/с) залежно від поточної інтенсивності потоку [52, 60]. На рисунку 1.4, який знаходиться нижче, наведена схема, на якій відображена архітектура трирівневої системи управління енергоспоживанням GreenSDN.

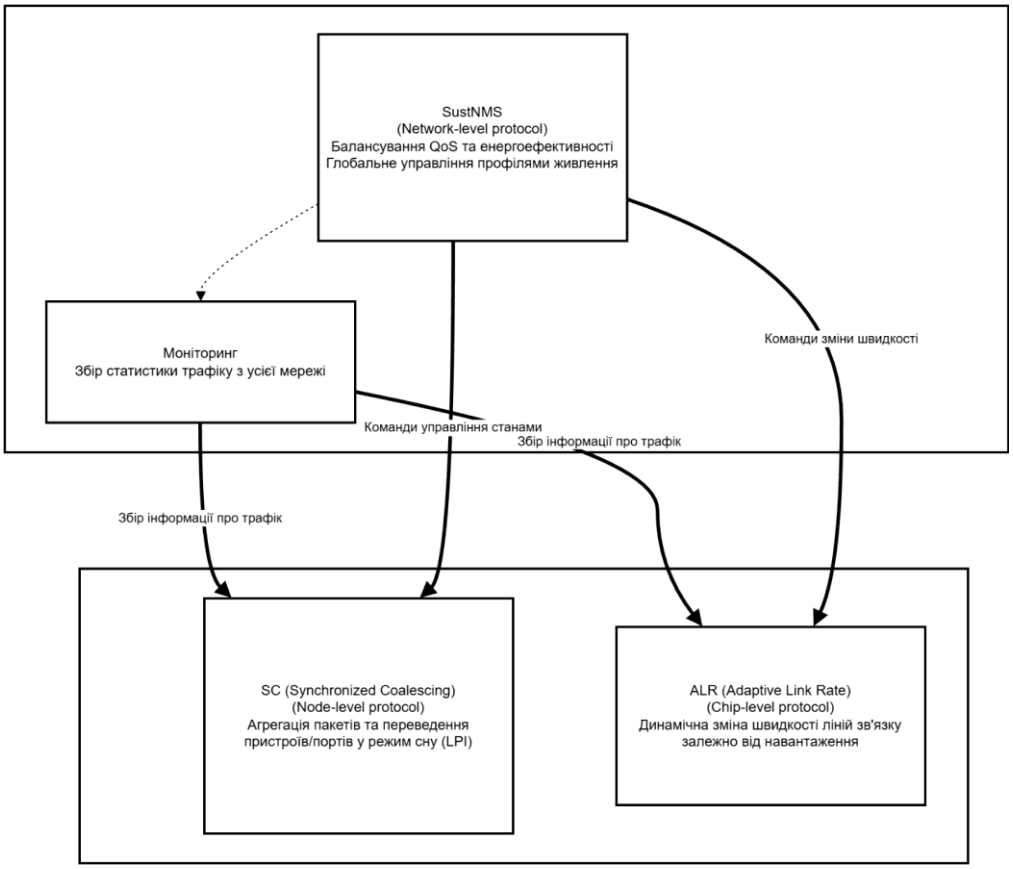


Рисунок 1.4 – Архітектура трирівневої системи управління енергоспоживанням GreenSDN

Точний баланс між енергозбереженням і якістю обслуговування – це ключова перевага такого інтегрованого підходу, адже система швидко пристосовується до коливань трафіку, не переналаштовуючи всю мережу. Хоча система управління енергоспоживанням GreenSDN має потенціал, її застосування стикається з реальними технологічними бар'єрами. Виконання функцій одразу на трьох рівнях сильно навантажує контролер та вимагає використання спеціального обладнання для інтерфейсів, що загострює проблему сумісності в гетерогенних середовищах [13, 38].

Останнім часом у науковому середовищі зростає кількість досліджень, спрямованих на впровадження розумних агентів всередину контролерів SDN. Замість фіксованих правил управління тепер застосовують підходи з галузі штучного інтелекту та машинного навчання, що дає можливість передбачати поведінку мережі в реальному часі.

Найбільш перспективним напрямом є застосування DRL для розв'язання задач динамічної маршрутизації [21, 22, 44]. У такій системі SDN-контролер стає своєрідним агентом, що постійно спостерігає за станом мережі. Кожен керуючий вплив тут подається як крок у Марковському процесі прийняття рішень. Агент аналізує дані: яка топологія, чи перевантажені канали, скільки пакетів у черзі – і вже тоді приймає рішення щодо вибору оптимального маршруту або зміни енергетичного стану вузлів [49, 72]. Рисунок 1.5, розташований нижче, показує, як інтелектуальний DRL-агент взаємодіє з мережею, що піддається програмним налаштуванням. Замість разової перевірки система безперервно спостерігає за станом середовища. Через контролер надходить інформація: дані про навантаження вузлів та параметри якості обслуговування. Опираючись на отримані дані, нейронна мережа всередині агента формує можливу поведінкову модель. Ця модель перетворюється на конкретні кроки – наприклад, прокладання нових маршрутів чи оновлення правил комутації. Результат кожної дії оцінюється числом – нагородою, що йде назад у процес навчання. Використовуючи цей зворотний зв’язок, алгоритм потрохи переналаштовує себе, прагнучи знайти компроміс: висока швидкість при меншому споживанні енергії.

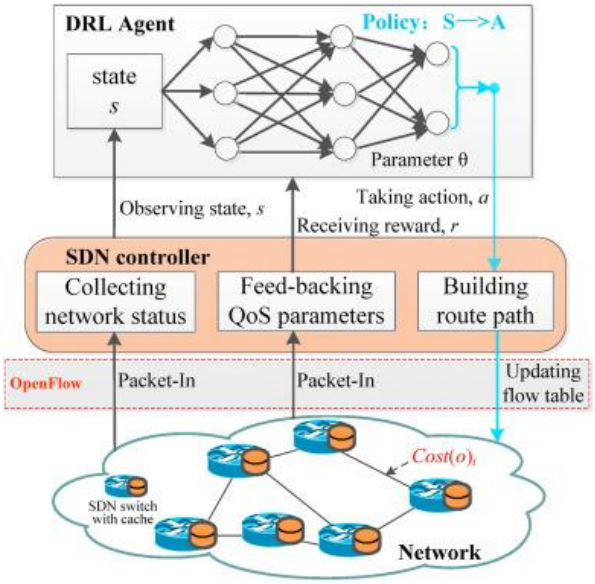


Рисунок 1.5 – Архітектура системи управління мережею на базі DRL

Одним із прикладів реалізації такого підходу є алгоритм AQROM. Він використовує архітектуру глибоких нейронних мереж для апроксимації функцій винагороди, що дозволяє одночасно оптимізувати метрики затримки, пропускної здатності та енергоспоживання [64, 70].

Застосування технологій глибинного навчання із підкріпленням у середовищі Green SDN дає чимало важливих переваг, насамперед завдяки високій адаптивності моделей. Складні нелінійні закономірності в патернах трафіку, які не піддаються аналізу звичайними евристичними правилами, успішно розпізнають DRL-агенти. Зокрема, алгоритми типу AQROM гарантують стабільну якість обслуговування, бо самостійно перерозподіляють потоки при стрибках навантаження, зберігаючи цілісність передачі. Після завершення етапу навчання керування мережею продовжується майже без участі адміністратора – такі рішення працюють автономно й ефективно.

Проте використання розумних агентів тягне за собою додаткові обчислювальні витрати. Навчання глибоких моделей потребує серйозних ресурсів – у багатьох випадках це означає задіяння графічного процесора, які інколи компенсують економію енергії, отриману через оптимізацію мережі. Також критичною є проблема «холодного старту», оскільки моделі потребують тривалого часу на навчання для досягнення стабільної точності, протягом якого система може

приймати неоптимальні рішення [26, 83]. Особливий виклик становить побудова функції нагороди – успіх системи залежить від того, наскільки точно враховано співвідношення між штрафами за затримки й винагородами за економію енергії, що значно ускладнює застосування таких методів у великих, неоднорідних мережах.

Схема на рисунку 1.6 показує, як працює прийняття рішень у трирівневій SDN-архітектурі. Рівень даних забезпечує постійний моніторинг стану комутаторів, передаючи результати вимірювань до рівня управління. Контролер формує цілісну картину стану мережі для інтелектуального агента AQROM, що функціонує в площині знань. На основі отриманих відомостей він визначає найефективнішу стратегію поведінки. Результат його аналізу перетворюється на набір правила маршрутизації. Правила інсталюються в таблиці потоків обладнання. Важливо те, що ресурсомісткі процеси штучного інтелекту не виконуються на рівні комутації, що є критичним для підтримки стабільного та прогнозованого рівня QoS.

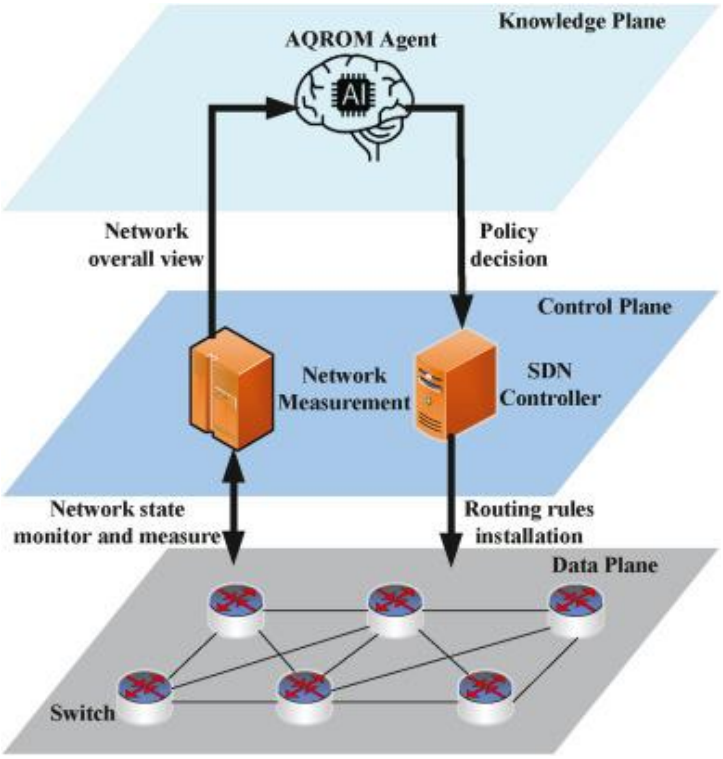


Рисунок 1.6 – Архітектура інтелектуального управління мережею з виділеною площиною знань

Окремим вагомим напрямом у сфері енергоефективних SDN є методи оптимізації використання пам'яті комутаторів, що зосереджуються на інтелектуальному розміщенні правил та стисненні таблиць потоків. Актуальність цього підходу зумовлена тим, що комутатори OpenFlow використовують пам'ять TCAM для здійснення швидкого пошуку за масками та префіксами. Хоча TCAM забезпечує високу продуктивність, вона є одним із найбільш енергоємних та дорогих компонентів мережевого пристрою, оскільки виконує порівняння вхідного пакета з усіма записами в таблиці паралельно. Через те, нові методи, подібні до Compact TCAM, зменшують навантаження на обладнання завдяки меншій кількості активних рядків і перебудові формату правил. Такий напрям дає можливість значно скоротити витрати живлення без втрати швидкості. Попри простоту ідеї, результати демонструють реальний ефект у практичному застосуванні.

Значно менші таблиці маршрутів – головна вигода цих підходів, разом із кращим контролем над масштабними потоками, навіть коли ресурси комутаторів у межах SDN обмежені. Без зайвого обладнання для пам’яті можна нарощувати мережеві функції, хоча техніки стискання й хитре розташування правил дають і серйозні проблеми. Динамічне присвоєння ідентифікаторів потокам, їх групування, а також часта необхідність пересилання поновлених правил між контролером та комутаторами напружують канали управління. Отже, затримки в мережі збільшуються під час активного переналаштування, що негативно позначається на загальній продуктивності.

Підсумовуючи огляд наведених вище підходів, можна констатувати, що вирішення проблеми енергоефективності в SDN розвивається на різних рівнях: від апаратного стиснення правил у пам'яті TCAM до складних систем консолідації трафіку та нейромережевих моделей. Проте кожне з цих рішень має свій специфічний компроміс між максимізацією заощадженої електроенергії, забезпеченням стабільної якості обслуговування та обчислювальним навантаженням на центральний контролер [54, 82].

Застосування евристичних рішень з низькою обчислювальною складністю

для подолання цих компромісів має першорядне значення в динамічних мережах. Для наочності, систематизації та зручності обґрунтування обраного методу дослідження, результати порівняльного аналізу основних переваг та недоліків існуючих рішень зведено у таблицю 1.2.

Таблиця 1.2 – Порівняльний аналіз існуючих методів забезпечення енергоефективності в SDN

Метод / Підхід	Переваги	Недоліки
ElasticTree (евристики консолідації)	Висока ефективність енергозбереження (до 50%), підтримка цільової продуктивності	Низька масштабованість, пошук оптимального набору ліній займає багато часу
CARPO (маршрутизація з урахуванням кореляції)	Щільна агрегація потоків, економія електроенергії за рахунок відключення зайвих ліній	Підвищення затримок через високу завантаженість залишених каналів, не підходить для трафіку реального часу
GreenSDN (багаторівневе управління)	Використання LPI та ALR, гнучкий моніторинг стану мережі	Висока складність архітектури, додаткові накладні витрати на контролер
DRL /AQROM (машинне навчання)	Відмінна адаптивність, зниження втрат пакетів та гарантування стабільного QoS.	Висока обчислювальна складність, тривалий час навчання моделі, складнощі з масштабуванням.

Кінець таблиці 1.2

Метод / Підхід	Переваги	Недоліки
TCAM Optimization (стиснення правил)	Пряме зниження апаратного енергоспоживання комутаторів, ефективне використання пам'яті	Ризик збільшення затримок через необхідність постійних запитів до контролера SDN.

1.3 Підходи до вирішення задачі за темою дослідження

На основі проведеного аналізу предметної області та недоліків існуючих рішень, для досягнення мети кваліфікаційної роботи обрано комплексний підхід, що поєднує методи математичного моделювання, теорії графів, розробку евристичних алгоритмів та імітаційне мережеве моделювання. Процес вирішення задачі енергозбереження в SDN пропонується розділити на три взаємопов'язані етапи: математичну формалізацію, алгоритмічну оптимізацію та експериментальну перевірку.

Мережеву топологію для чіткого опису задачі маршрутизації можна представити як неорієнтований зважений граф $G = (V, E)$, де V – множина мережевих вузлів, а E – множина ліній зв'язку між ними [30]. Кожен елемент мережі характеризується певним рівнем енергоспоживання. Робочий стан шасі забезпечується за рахунок статичної потужності. Навантаження ж, пов’язане із завантаженими портами й обміном даними, впливає на динамічну складову споживання. Загальний режим функціонування формується під впливом обох факторів одночасно.

У межах цього підходу стан кожного комутатора та каналу зв'язку описується бінарними змінними $x_v, y_e \in \{0, 1\}$, де значення «1» означає активний стан пристрою, а «0» – переведення у режим сну. Математична модель цільової функції, що відображає сумарні енерговитрати всіх активованих елементів мережі при дотриманні обмежень щодо пропускної здатності каналів та допустимих затримок

передачі пакетів, має наступний вигляд (1.1):

$$P_{total} = \sum_{v \epsilon V}(P^{v}_{static} * x_v + P^{v}_{dyn} * u_v) + \sum_{e \epsilon E} P^{e}_{link} * y_e \rightarrow min \quad (1.1)$$

де: P^{v}_{static} – статична потужність вузла v;

P^{v}_{dyn} – динамічна потужність вузла, що залежить від його завантаження u_v;

P^{e}_{link} – енергоспоживання активної лінії зв'язку e;

x_v, y_e – бінарні змінні стану вузлів та ліній відповідно.

При цьому накладаються жорсткі обмеження: пропускна здатність вибраних каналів не повинна перевищувати їх максимальної ємності, а затримки передачі пакетів мають залишатися в межах вимог якості обслуговування.

Хоча математична суворість забезпечує чіткість, класичні підходи до розв’язання таких задач оптимізації стикаються з критичною проблемою масштабованості. Так, формалізовані моделі дають найкращий результат – проте вартість обчислень росте разом із розміром мережі. Складність полягає в тому, що задача мінімізації енергоспоживання з урахуванням маршрутизації входить у категорію NP-задач, де перебір варіантів займає надто багато часу [13, 54]. Це зумовлює необхідність переходу від ідеальної математичної оптимізації до пошуку наближених, але швидких рішень.

Оскільки SDN-контролер повинен реагувати на зміни трафіку в реальному часі, використання ресурсоємних математичних вирішувачів є недоцільним. Алгоритм працюватиме за наступною логікою:

1) динамічне виявлення топології за допомогою протоколу LLDP для побудови актуального графа мережі;

2) циклічний збір статистики для визначення поточного завантаження u_v та u_e;

3) агрегація потоків даних на мінімально необхідній кількості ліній зв'язку, при розрахунку нових маршрутів алгоритм надає пріоритет уже активним

вузлам, щоб мінімізувати затримки, пов'язані з часом «пробудження» обладнання;

4) ініціація переведення незадіяних портів у режим LPI згідно зі стандартом IEEE 802.3az або повне вимкнення вузлів при нульовому навантаженні.

Оскільки розгортання та тестування нових алгоритмів маршрутизації на реальному апаратному обладнанні ЦОД є дороговартісним і не завжди можливим, основним підходом для валідації розробленого методу стане імітаційне моделювання.

Програмна реалізація логіки управління здійснюватиметься на мові Python з використанням SDN-контролера Ryu, який забезпечує повну підтримку протоколу OpenFlow 1.3+. В якості середовища емуляції обрано Mininet, що дозволяє створювати віртуальні топології з багаторівневою комутаційною архітектурою та високим ступенем достовірності мережевих процесів.

Оскільки Mininet не вимірює фізичне енергоспоживання, оцінка енергоефективності буде проводитися шляхом математичного розрахунку на основі кількості деактивованих портів та вузлів відносно загальної потужності системи. Перевірка результатів включатиме порівняльний аналіз отриманої економії енергії та динаміки показників QoS, зокрема затримки передачі та відсотка втрачених пакетів при різних сценаріях навантаження.

Завдяки запропонованому підходу, складна задача оптимізації енергоспоживання перетворюється на гнучкий алгоритм керування ресурсами в реальному часі. Теоретико-графове представлення топології у поєднанні з розробкою евристики дає можливість SDN-контролеру швидко приймати рішення, що особливо важливо в умовах непередбачуваних змін у мережі. Обраний інструментарій, що включає мову програмування Python, контролер Ryu та середовище емуляції Mininet, гарантує точне тестування кожного аспекту методу. Отримані результати покажуть не лише відсоток збереженої енергії, а й продемонструють стабільність основних метрик обслуговування, що є ключовою вимогою до сучасних «зелених» програмно-конфігурованих мереж.

1.4 Постановка задачі

З огляду на мету роботи – зниження енерговитрат в SDN-мережі за умови збереження якості обслуговування – загальна задача дослідження зводиться до розв'язання задачі оптимізації енергоспоживання при заданих обмеженнях на якість обслуговування.

Для математичної формалізації задачі топологію програмно-конфігурованої мережі задано у вигляді графа $G = (V, E)$, для якого відомі наступні вхідні параметри:

– множина комутаторів V та ліній зв'язку E;

– максимальна пропускна здатність C_e кожного каналу $e \in E$;

– матриця поточного трафіку T, що містить обсяги даних між парами хостів (s, d);

– моделі енергоспоживання обладнання $P\ (v, e)$, що враховують базову потужність та навантаження на порти.

Головною метою розв'язання задачі є мінімізація сумарного енергоспоживання інфраструктури (згідно з формулою 1.1) шляхом переведення максимально можливої кількості елементів $\{v, e\}$ у стан низького енергоспоживання.

Оптимізація енергоспоживання (вимкнення обладнання) не повинна призводити до деградації параметрів мережі, тому на систему накладаються наступні жорсткі обмеження:

– для кожного каналу e виконується умова $\sum f_e \leq C_e$, де f_e – консолідований потік (сумарний трафік, який алгоритм об'єднує на будь-якому залишеному активному каналі зв'язку, не повинен перевищувати його максимальної фізичної ємності);

– сумарна затримка на обраному шляху $L_{path} \leq L_{max}$, де L_{max} – поріг для конкретного класу сервісу (наскрізна затримка передачі пакетів по обраних консолідованих маршрутах не повинна перевищувати максимально допустимого

порогу для заданого типу трафіку);

– забезпечення неперервності передачі всіх потоків від джерела до отримувача.

Щоб задовольнити всі умови, потрібен евристичний підхід до роботи SDN-контролера: він має функціонувати без затримок, повторюючи серію взаємопов’язаних кроків. Спочатку система аналізує актуальну топологію мережі та чергу обслуговування потоків. На їхній основі розраховуються оптимальні шляхи пересилання, причому визначальним фактором є прогноз споживання електроенергії. Після цього трафік групується так, щоб навантаження зосередилося лише на критично важливих каналах. Це дозволить вивільнити решту обладнання, не задіяного безпосередньо. Кінцевий етап кожного циклу – коригування таблиць маршрутизації на активних комутаторах; паралельно надсилатимуться команди для вимкнення вільних портів або переведення їх у режим енергозбереження.

1.5 Висновки до першого розділу

Аналіз підтвердив, що впровадження 5G та IoT зумовлює зростання обсягів даних до 175 зетабайтів, що може підвищити частку ІКТ-сектору в глобальному енергоспоживанні до 20% до 2030 року. Традиційні мережі марнують до 90% енергії через негнучку архітектуру, що робить перехід до парадигми Green Networking жорсткою технічною вимогою для стабільного розвитку цифрових систем.

Технологія SDN дозволяє оптимізувати ресурси через централізовану консолідацію трафіку, проте існуючі моделі мають обмеження щодо підтримки стабільного рівня QoS. Дослідження фокусується на розробці методу на основі алгоритмів мінімального кістякового дерева для побудови енергоефективних топологій та захисту від DoS-атак. Реалізація методу за допомогою контролера Ryu у середовищі Mininet дозволить знизити енерговитрати інфраструктури на 50-55% при збереженні цільових показників продуктивності мережі.

2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ЗЕЛЕНИХ ОБЧИСЛЕНЬ В ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖАХ

2.1 Обґрунтування і вибір методів дослідження

Метою даної кваліфікаційної роботи було зменшення енергетичних витрат на програмно-визначені мережі при збереженні продуктивності. Цього досягнуто шляхом серії поглиблених покрокових аналізів та процесів оцінки. Існує низка доступних методів для досягнення цієї мети, і у роботі застосовано їх комбінацію: для задоволення вимог дослідження обрано як модельно-орієнтовані, так і тестово-орієнтовані методи оцінки.

Щодо конкретної методології системного аналізу та побудови, спочатку були проаналізовані загальні моделі, включаючи ElasticTree, CARPO або GreenSDN [25, 52, 54], щоб визначити їхній діапазон та обмеження. На цій основі розроблено метод, що поєднує низьке енергоспоживання з високою швидкістю роботи.

Графи можуть зображати комп'ютерну мережу, в якій кожне з'єднання позначає інтенсивність взаємозв'язку. Вершини графа моделюють мережеві комутатори, а ребра – фізичні канали зв’язку з відповідними вагами, де може змінюватися шлях інформаційного пакету. Такий підхід забезпечує наочність того, як поширюються пакети інформації, що спрощує аналіз мережі. Це також полегшує знаходження місць, де точки повинні бути з'єднані з точки зору оптимальної логіки дій.

Впровадження моделі енергетичного стану мережевого обладнання та гарантування обмежень якості обслуговування знижують енергоспоживання. Для цих завдань створюється цільова функція відповідно до правил математичної логіки. Працюючи в активному режимі або в режимі очікування, стан пристроїв регулюється бінарними змінними. Енергоспоживання в мережі має деякий постійний елемент, а також компонент, що залежить від навантаження. Модель також дозволяє знаходити випадки, коли загальний мережевий трафік менший за пропускну здатність каналів зв'язку. Затримка передачі пакетів не повинна перевищувати встановлених значень.

Як результат, задача енергоефективності, що полягає у виборі оптимального шляху та вимкненні всіх невикористаних пристроїв, є багатовимірною комбінаторною оптимізацією класу NP-складності.

Обчислювальна складність вирішення такої проблеми шляхом повного перерахування всіх шляхів або лінійного моделювання буде занадто високою, і в таких сценаріях призведе до неприпустимих часових витрат або вичерпання обчислювальних ресурсів керуючого вузла. На одному кінці діапазону можна вирішити цю проблему для статичної мережі, на іншому – для динамічної SDN-мережі, не можна відкласти велику кількість часу на реінжиніринг топології для задоволення максимальних мережевих навантажень через тривалий час очікування, необхідний для обчислення стану топології з вищими втратами інформації та затримками.

З огляду на це, одним з раціональних рішень для цієї проблеми в цілому є вибір евристичного методу з галузі алгоритмічної оптимізації. Евристичний алгоритм надає контролеру SDN задовільне рішення за поліноміальний час (хоча його точність нижча). Швидкий алгоритм також призводить до того, що трафік здебільшого спрямовується на вже активні канали, а не починається з нуля. Крім того, це дозволяє використовувати режим LPI для зайвих мережевих пристроїв, не перевантажуючи контролер SDN, що не було необхідно.

Тепер треба перш за все перевірити дійсність запропонованої процедури. Реальний експеримент – це дорога справа. Встановлення нових алгоритмів маршрутизації на реальному обладнанні – те, що було б неможливим у дата-центрі або компанії з великою кількістю машин. І мережеві адміністратори не будуть задоволені в такому випадку. Пряма імплементація в діючу інфраструктуру пов'язана з високими ризиками деградації якості обслуговування та потенційною нестабільністю системи в цілому. Зміни в робочих мережах вплинуть на якість обслуговування. Але навіть незначна несправність матиме серйозні наслідки. Коли ви намагаєтеся тестувати алгоритми на реальних серверах, зазвичай виникає багато простоїв.

Для цієї мети використовувалося моделювання як основна процедура під час

тестування встановлених алгоритмів. Імітаційне моделювання – це підхід, коли вводиться віртуальна симуляція справжньої системи, щоб перевірити різні можливі результати. Тому, після моделювання поведінки реальних комутаторів, каналів і протоколу OpenFlow, запропонований алгоритм був оцінений. Для верифікації запропонованого методу використано програмний інструментарій, що включає мову Python, SDN-контролер Ryu та середовище емуляції Mininet. Ця програмна бібліотека має ключ до створення реалістичного мережевого трафіку у випадку сучасної топології дата-центру та обчислення статистичних результатів.

Крім того, система симуляції дозволяє повторювати експерименти кілька разів. Енергозбереження вимірюється кількістю портів – включаючи віртуальні – і комутаторів, які вимкнені. Фактор затримки та втрати пакетів на різних мережевих операціях у різні часові інтервали є предметом уваги. Проводиться аналіз продуктивності за різних сценаріїв роботи системи.

2.2 Розробка базової математичної моделі мережевої інфраструктури

Щоб сформулювати завдання динамічного керування споживанням енергії в програмно налаштовуваних мережах, потрібно створити придатну математичну модель самої інфраструктури. Хоча пристрої можуть відрізнятися за конструкцією в залежності від бренду – необхідно забезпечити високий рівень абстракції від апаратних особливостей обладнання, але фіксувати стан живлення вузлів комутації, ліній передачі та опрацювання потоків даних у кожен конкретний момент. Тому загальний опис системи логічніше розділити на окремі частини, які одночасно працюють разом: представлення топології, модель витрат електроенергії й підхід до випадкових коливань навантаження.

За основу формалізації топології мережі обрано апарат теорії графів. Топологію SDN-мережі представлено у вигляді неорієнтованого зваженого графа $G = (V, E)$. У цій математичній моделі множина вершин $V = \{v_1, v_2, \dots, v_n\}$ відповідає множині комутаторів мережі, а множина ребер $E = \{e_1, e_2, \dots, e_m\}$ відображає фізичні лінії зв'язку між цими комутаторами.

Кожен вузол $v \in V$ характеризується власним набором обчислювальних та комутаційних ресурсів, які визначають його максимальну продуктивність μ_v. Крім того, кожен комутатор містить таблицю потоків, яка зберігається у високошвидкісній, але енергоємній пам'яті TCAM.

Кожне ребро $e \in E$ (канал зв'язку) характеризується максимальною пропускною здатністю C_e, яка визначає граничний обсяг трафіку, що може бути переданий цим каналом без виникнення втрат. Крім того, для кожного каналу визначається базова затримка розповсюдження сигналу D_{prop}^{e}.

Для опису динамічного стану мережевої інфраструктури у довільний момент часу t вводиться вектор бінарних змінних прийняття рішень:

– $x_v \in \{0,1\}$ – бінарна змінна стану комутатора v. Значення $x_v = 1$ означає, що комутатор перебуває в активному робочому стані. Значення $x_v = 0$ означає, що пристрій переведений у сплячий режим глибокого енергозбереження для економії енергії.

– $y_e \in \{0,1\}$ – бінарна змінна стану лінії зв'язку e. Значення $y_e = 1$ вказує на те, що канал активний і бере участь у передачі трафіку. Значення $y_e = 0$ означає, що канал переведено у режим LPI відповідно до стандарту IEEE 802.3az, або його повністю відключено програмним шляхом через нульове навантаження.

Енергоспоживання мережевої інфраструктури безпосередньо залежить від стану її елементів та інтенсивності мережевого трафіку. Дослідження показують, що навіть за умови нульового завантаження увімкнене мережеве обладнання споживає до 90% від своєї пікової потужності. Це зумовлює необхідність декомпозиції загального енергоспоживання кожного вузла v на статичну та динамічну складові:

$$P_v = P_{static}^{v} * x_v + P_{dyn}^{v}(u_v) * x_v, \tag{2.1}$$

де P_{static}^{v} – статична потужність, що споживається базовим шасі комутатора, блоками живлення, вентиляторами охолодження та центральним процесором для

підтримки операційної системи та зв'язку з SDN-контролером незалежно від обсягу трафіку;

$P_{dyn}^{v}(u_v)$ – динамічна складова енерговитрат, що нелінійно залежить від поточного навантаження комутатора $u_v \epsilon$ та кількості активних трансиверів;

x_v – бінарна змінна стану комутатора v.

Оскільки динамічна потужність $P_{dyn}^{v}(u_v)$ реального обладнання рідко є ідеально лінійною функцією від навантаження, для підвищення точності математичної моделі функція енергоспоживання визначається шляхом інтерполяції емпіричних даних поліномом ступеня n:

$$P_{dyn}^{v}(u_v) = a_1 * u_v + a_2 * u_v^2 + \cdots + a_n * u_v^n, \quad (2.2)$$

де $a_1, a_2, \ldots, a_n$ (або загалом a_i) – емпіричні коефіцієнти полінома. Це конкретні числові значення, які розраховуються для кожної моделі комутатора індивідуально на етапі його тестування, щоб формула точно відповідала фізичним вимірам його енергоспоживання при різних навантаженнях;

u_v – поточне навантаження комутатора v.

Енергоспоживання активної лінії зв'язку e визначається як:

$$P_e = P_{link}^{e} * y_e, \quad (2.3)$$

де P_e – загальне енергоспоживання конкретної лінії зв'язку e в даний момент часу;

P_{link}^{e} – потужність, що споживається портом для підтримки активного лінка;

y_e – бінарна змінна стану лінії зв'язку e.

Через нерівномірний і випадковий характер трафіку в сучасних промислових мережах звичайні потокові моделі не дають точної картини якості обслуговування.

Замість цього, у рамках моделі, будь-який комутатор SDN подається як багатоканальна система масового обслуговування, де є обмежена черга та можливі втрати запитів.

Вхідний потік даних розглядається як нестаціонарний пуассонівський потік пакетів з інтенсивністю $\lambda_v(t)$. Оскільки SDN-контролер має приймати рішення у реальному часі, безперервна функція $\lambda_v(t)$ дискретизується на інтервали Δt, протягом яких потік вважається стаціонарним з середньою інтенсивністю λ.

Обслуговуючими пристроями в цій системі масового обслуговування виступають процесорні ядра комутатора та активні вихідні порти $y_e = 1$. Середня інтенсивність обробки пакетів комутатором становить μ_v. Пакети, що надходять у момент зайнятості комутаційної матриці, поміщаються у буфер (чергу) обмеженої ємності Q_v.

Якщо інтенсивність вхідного навантаження перевищує пропускну здатність активних портів (особливо після переведення частини портів у режим LPI для економії енергії), виникає ризик переповнення буфера. Імовірність втрати пакета для комутатора v, згідно з формулами Ерланга для системи масового обслуговування з обмеженою чергою, визначається як:

$$p_{loss}^{v} = \frac{p^{k+Q_v}}{k^{Q} * k!} * p_0, \tag{2.4}$$

де p_{loss}^{v} – імовірність втрати пакета для конкретного комутатора v. Визначає математичну частку даних, які будуть примусово відкинуті комутатором через переповнення його буфера при раптових сплесках трафіку;

k – кількість активних вихідних портів комутатора;

Q_v (або Q) – максимальна ємність буфера комутатора v;

p_0 – імовірність абсолютно вільного стану комутатора.

Середній час очікування пакета в черзі (затримка буферизації D_{queue}) обчислюється за формулою Літтла:

$$D_{queue} = \frac{L_{queue}}{\lambda*(1-p_{loss}^{v})}, \quad (2.5)$$

де D_{queue} – середній час очікування пакета в черзі (затримка буферизації);

L_{queue} – середня довжина черги;

λ – загальна інтенсивність вхідного потоку.

Застосування апарату систем масового обслуговування у математичній моделі дозволяє SDN-контролеру не просто оцінювати статичну пропускну здатність, а й передбачати ймовірність втрати пакетів та зростання затримок при штучному зменшенні кількості активних ліній зв'язку.

Мінімізація енергоспоживання $P_{total} \rightarrow min$ має сенс виключно за умови дотримання жорстких обмежень на якість обслуговування. У протилежному випадку агресивне вимкнення портів призведе до деградації сервісів та розриву сесій передачі даних. Математична модель накладає на систему наступні обмеження:

1. Обмеження пропускної здатності каналів зв’язку встановлює, що сумарний обсяг даних, який проходить через будь-яку лінію в мережі, не може перевищувати її граничну фізичну спроможність. Крім контролю навантаження, цей механізм регулює можливість передачі трафіку залежно від енергетичного стану обладнання: якщо канал зв'язку переведено в режим сну для економії електроенергії, його доступна ємність стає нульовою. Таким чином, система гарантує, що інформаційні потоки будуть спрямовані виключно через активні шляхи, а передача пакетів через деактивовані сегменти мережі стає неможливою.

2. Для кожного інформаційного потоку, що прямує від вузла-джерела до вузла-призначення, повинна безумовно виконуватись умова неперервності. Згідно з цим принципом, у будь-якому транзитному вузлі мережі обсяг вхідного трафіку має бути ідентичним обсягу вихідного трафіку. Це свідчить про те, що комутатор не поглинає і не створює нові дані всередині себе, а лише перенаправляє їх згідно з актуальною таблицею маршрутизації, яку сформував SDN-контролер.

3. Сумарний показник затримки передачі даних уздовж обраного маршруту від джерела до отримувача формується як інтегральний час, що складається з декількох ключових часових компонентів. Сюди належать затримка безпосередньої обробки пакетів у комутаційних вузлах, тривалість очікування в чергах, що розраховується на основі моделі системи масового обслуговування, а також затримки передачі та розповсюдження сигналу безпосередньо у фізичному середовищі. Важливою особливістю математичної моделі є врахування часу апаратного пробудження пристроїв. Ця специфічна затримка додається до загального часу передачі лише у випадках, коли алгоритм керування обирає шлях через порти, які до моменту перенаправлення на них трафіку перебували в енергозберігаючому стані низького споживання.

Для забезпечення обчислювальної стійкості моделі та можливості її імплементації у вигляді програмного алгоритму для SDN-контролера прийнято низку обґрунтованих базових припущень:

– вважається, що SDN-контролер через протоколи LLDP та OpenFlow 1.3+ постійно та без суттєвих затримок отримує повну матрицю суміжності графа G та актуальну статистику завантаженості портів;

– існує жорстке ієрархічне обмеження між станом комутатора та його портів. Стан лінії зв'язку не може бути активним, якщо сам вузол вимкнений. Математично це виражається як $y_e \leq x_v$ для всіх ребер e, що є інцидентними вузлу v;

– вважається, що службовий трафік передається ізольованим керуючим каналом або має найвищий пріоритет у внутрішньоканальному з’єднанні, тому процеси переведення обладнання в режим сну не призводять до втрати зв'язку між рівнем управління та рівнем даних.

2.3 Формалізація цільової функції та обмежень якості обслуговування

Сформована базова математична модель мережевої інфраструктури створює необхідний фундамент для постановки головної оптимізаційної задачі

дослідження. Основною метою динамічного управління в програмно-конфігурованих мережах є досягнення концепції «зелених обчислень» – глибокої мінімізації загального енергоспоживання обладнання без порушення цілісності функціонування мережі. Завдяки централізованій логіці SDN-контролера, процес управління зводиться не до хаотичного вимкнення пристроїв, а до цілеспрямованого пошуку такого оптимального набору керуючих змінних x_v та y_e, при якому сумарні енерговитрати інфраструктури досягнуть свого мінімуму.

Основним інструментом або «серцем» запропонованого методу є оптимізаційна цільова функція. Вона математично описує прагнення системи до мінімізації енергетичних витрат на всіх рівнях топології. Загальна цільова функція енергоспоживання всієї програмно-конфігурованої мережі (P_{total}), яку SDN-контролер намагається мінімізувати, синтезується як сума статичних та динамічних витрат усіх активних елементів системи:

$$P_{total} = \sum_{v \epsilon V}(P^v_{static} * x_v + P^v_{dyn}(u_v) * x_v) + \sum_{e \epsilon E} P^e_{link} * y_e \rightarrow min, \qquad (2.6)$$

де P^v_{static} – статична потужність увімкнених шасі комутаторів. Це енергія, яка невідворотно витрачається на підтримку життєдіяльності базових апаратних компонентів (системи охолодження, центральний процесор, підтримка операційної системи та фоновий обмін службовими повідомленнями з SDN-контролером) за умови, що вузол перебуває в активному стані ($x_v = 1$). У разі переведення вузла у стан сну ($x_v = 0$), цей компонент обнуляється;

P^v_{dyn} – динамічна потужність завантажених вузлів. Це змінна складова енерговитрат, обсяг якої прямо корелює з поточним рівнем завантаженості обчислювальних та комутаційних ресурсів вузла u_v;

P^e_{link} – номінальна потужність активних мережевих портів, яка витрачається на підтримку фізичних ліній зв'язку $y_e = 1$ незалежно від того, передається по них корисний трафік у дану мікросекунду чи ні.

Дослідження показують, що динамічна потужність сучасного комутаційного обладнання зростає вкрай нелінійно при збільшенні навантаження. Використання спрощених лінійних моделей призводить до значних похибок в оцінці реальних енерговитрат, що робить алгоритм управління неефективним на практиці. Тому для забезпечення високої математичної точності та максимального наближення моделі до реальних фізичних вимірів апаратних засобів, динамічна складова $P_{dyn}^{v}(u_v)$ розраховується згідно з інтерполяційним поліномом n-го ступеня:

$$P_{dyn}^{v}(u_v) = \sum_{i=1}^{n} a_i * u_v^i, \tag{2.7}$$

де a_i – емпіричні коефіцієнти полінома. Це конкретні числові значення, які розраховуються для кожної моделі комутатора індивідуально на етапі його тестування, щоб формула точно відповідала фізичним вимірам його енергоспоживання при різних навантаженнях;

u_v^i – значення поточного навантаження комутатора v, піднесене до відповідного математичного ступеня.

Агресивна мінімізація цільової функції алгоритмічним шляхом природно тяжіє до масового зведення керуючих змінних x_v та y_e до нуля. Проте такий підхід неминуче призведе до перевантаження єдиних залишених активних шляхів, що спричинить відкидання пакетів, виникнення критичних затримок та повну деградацію мережевих сервісів. Мінімізація даної функції має науковий і практичний сенс виключно за умови дотримання жорстких математичних обмежень на якість обслуговування. Саме здатність SDN-контролера оптимізувати живлення без деградації мережі є індикатором життєздатності методу.

Консолідація (агрегація) трафіку на працюючих маршрутах завжди повинна враховувати фізичні межі середовища передачі. Для кожного ребра $e \in E$ сумарний потік даних f_e не може перевищувати його фізичну спроможність C_e, скориговану на поточний енергетичний стан каналу:

$$f_e \leq C_e * y_e, \forall_e \epsilon E, \quad (2.8)$$

де f_e – сумарний потік даних, що проходить через конкретний канал зв'язку e в даний момент часу;

C_e – максимальна фізична пропускна здатність каналу e;

y_e – бінарна змінна стану лінії зв'язку;

$\forall_e \epsilon E$ – математичний квантор, який вказує на те, що дана умова повинна виконуватися для всіх каналів e, які належать до загальної множини ліній зв'язку мережі E.

Ця математична умова виконує функцію жорсткого логічного вентиля. Вона гарантує, що сумарний інформаційний потік f_e лімітується значенням C_e. Крім того, якщо алгоритм приймає рішення перевести порт у сплячий режим для економії енергії, права частина рівняння стає нульовою. Відповідно, доступна ємність стає нульовою, що математично забороняє SDN-контролеру маршрутизувати будь-який трафік через вимкнений канал.

Перенаправлення трафіку з вимкнених вузлів на сусідні не повинно призводити до втрати інформації. Для забезпечення цілісності передачі даних у будь-якому транзитному вузлі мережі обсяг вхідного трафіку має бути ідентичним обсягу вихідного:

$$\sum_{j \epsilon V} f_{ij}^k - \sum_{j \epsilon V} f_{ji}^k = 0\,,\ \forall_i \neq S, D, \quad (2.9)$$

де $\sum_{j \epsilon V} f_{ij}^k$ – сума всіх вихідних потоків k з вузла i до всіх сусідніх вузлів j, які належать до множини вершин V;

$\sum_{j \epsilon V} f_{ji}^k = 0$ – сума всіх вхідних потоків k у вузол i з усіх сусідніх вузлів j;

f_{ij}^k – обсяг конкретного інформаційного потоку k на ребрі, що з'єднує вузол i та вузол j;

V – множина всіх комутаторів (вершин графа) у топології мережі;

k – індекс конкретного інформаційного потоку даних;

S (Source) – вузол-джерело, з якого починається рух потоку k;

D (Destination) – вузол-призначення (адресат), де потік k завершується;

$\forall_i \neq S, D$ – математична умова, яка вказує, що це рівняння має виконуватися для всіх транзитних вузлів мережі, крім джерела та отримувача.

Рівняння гарантує, що транзитний комутатор не створює і не губить пакети даних, а лише здійснює їх комутацію згідно з інструкціями контролера. Це базовий принцип стійкої передачі даних при динамічній зміні топології.

Для сучасних індустріальних додатків, систем реального часу та передачі потокового мультимедіа час доставки даних є критичним параметром. У процесі енергозбереження час доставки може суттєво зростати через переповнення черг на залишених каналах та апаратні затримки на ввімкнення обладнання, що спало. Інтегральна затримка вздовж обраного маршруту P не повинна перевищувати встановлений SLA поріг D_{max} для конкретного додатка:

$$D_{total} = \sum_{v \in P}(D^v_{proc} + D^v_{queue}) + \sum_{e \in P}\left(D^e_{prop} + \tau_{wake} * \left(1 - y_e^{prev}\right)\right) \leq D_{max} \quad (2.10)$$

де D^v_{proc} – час апаратної обробки пакетів та пошуку правил у таблиці потоків комутатора v;

D^v_{queue} – час очікування пакетів у буферних чергах, який розраховується відповідно до моделі систем масового обслуговування, обґрунтованої у попередньому підрозділі;

D^e_{prop} – фізична затримка розповсюдження сигналу в середовищі. Особливістю даного обмеження є врахування апаратного часу пробудження портів зі стану LPI – τ_{wake}. Змінна y_e^{prev} позначає енергетичний стан порту в попередній мікросекунді. Якщо порт вже був активним, множник стає нулем, і затримка на пробудження не додається. Якщо ж порт спав, система автоматично враховує штрафний час τ_{wake}.

Це змушує алгоритм контролера надавати пріоритет вже активним маршрутам і уникати ефекту постійного «пінг-понгу» (вмикання/вимикання) інтерфейсів.

Для забезпечення обчислювальної стабільності алгоритму динамічної маршрутизації та збереження безперервної керованості SDN-інфраструктурою, багатовимірна математична модель доповнюється двома фундаментальними технічними умовами:

1. Першою такою умовою є ієрархічне обмеження станів. Оскільки енергетичні стани пристроїв мають фізичну залежність, вводиться сувора структурна підпорядкованість між комутатором та його інтерфейсами. Лінія зв'язку не може перебувати в активному стані, якщо відповідний комутаційний вузол вимкнений або переведений у глибокий сон:

$$y_e \leq x_v, \forall_e \epsilon E_{incident}(v), \tag{2.11}$$

де $E_{incident}(v)$ – множина всіх ребер e, які є інцидентними до конкретного вузла v;

$\forall_e \epsilon E_{incident}(v)$ – математичний квантор, який означає, що ця умова повинна виконуватися для кожної лінії зв'язку, підключеної до даного комутатора.

Це обмеження діє як математичний запобіжник. Воно гарантує, що алгоритм не спробує активувати порт на тому комутаторі, який був повністю деактивований для економії базової статичної енергії, зберігаючи логічну консистентність моделі P_{total}.

2. Другою такою умовою є пріоритезація керуючого трафіку. Забезпечення життєздатності програмно-конфігурованих мереж неможливе без постійного та надійного зв'язку між площиною даних (комутаторами) та центральним контролером. У моделі приймається безапеляційне правило: усі службові пакети протоколу OpenFlow мають найвищий пріоритет. Вони передаються ізольовано. Це гарантує, що процес агресивного «засинання» мережі

для економії енергії не вплине на доставку службових пакетів і не призведе до ситуації, коли контролер втрачає зв'язок зі своїми комутаторами.

2.4 Висновки до другого розділу

У другому розділі розроблено та обґрунтовано комплексну математичну модель програмно-конфігурованої мережі, яка базується на апараті теорії графів та систем масового обслуговування. Запропонований підхід дозволяє формалізувати мережеву інфраструктуру через множину вершин і ребер із відповідними вагами, враховуючи при цьому як статичні, так і нелінійні динамічні складові енергоспоживання обладнання. Використання бінарних змінних стану вузлів та каналів у поєднанні з імовірнісними моделями черг створює фундамент для точного прогнозування затримок та втрат пакетів в умовах агресивної консолідації трафіку.

Сформована багатовимірна цільова функція, яка спрямована на мінімізацію сумарних енерговитрат системи при суворому дотриманні жорстких обмежень на QoS, таких як пропускна здатність, неперервність потоків та інтегральна затримка з урахуванням часу апаратного пробудження портів. З огляду на приналежність задачі оптимізації до класу NP-складних, обґрунтовано доцільність застосування евристичного методу, який забезпечує отримання раціональних рішень за поліноміальний час, що є критично важливим для динамічних SDN-середовищ.

Для верифікації розробленого методу визначено оптимальний інструментарій імітаційного моделювання, що включає мову Python, SDN-контролер Ryu та середовище Mininet. Такий вибір дозволяє проводити безпечні та повторювані експерименти на складних топологіях ЦОД, забезпечуючи високу точність оцінки потенційного енергозбереження та продуктивності системи без ризиків деградації реальної мережевої інфраструктури.

3 МЕТОД ЗАБЕЗПЕЧЕННЯ ЗЕЛЕНИХ ОБЧИСЛЕНЬ НА БАЗІ ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖ

3.1 Розробка евристичного алгоритму консолідації трафіку

Сформульована задача мінімізації цільової функції загального енергоспоживання із суворим дотриманням нелінійних обмежень якості обслуговування належить до класу NP-складних задач комбінаторної оптимізації. Пряме математичне розв'язання такої задачі методами повного перебору або гілок і меж вимагає часу, що експоненційно зростає зі збільшенням кількості вузлів $|V|$ та ліній зв'язку $|E|$. В умовах динамічних програмно-конфігурованих мереж, де зміна патернів трафіку відбувається в мілісекундних інтервалах, використання точних аналітичних методів є неприпустимим.

З огляду на це, ядром розробленого підходу є створення швидкого субоптимального евристичного методу енергоефективної консолідації трафіку (ЕМЕКТ), здатного функціонувати в реальному часі на рівні SDN-контролера.

На відміну від традиційних мереж, де кожен маршрутизатор приймає рішення ізольовано на основі локальних протоколів, запропонований метод ЕМЕКТ використовує парадигму глобальної видимості SDN. Робота алгоритму базується на безперервному циклі взаємодії між площиною управління та площиною даних, що складається з таких кроків:

1. Першим кроком є моніторинг топології та збір статистики. Алгоритм ЕМЕКТ регулярно ініціює запити показників стану всіх підключених OpenFlow-комутаторів. У відповідь комутатори надсилають дані про кількість переданих байтів та пакетів, що дозволяє контролеру з високою точністю розраховувати поточний потік f_e для кожної лінії зв'язку $e \in E$, а також визначати коефіцієнт завантаженості. Завдяки протоколу LLDP контролер підтримує актуальну матрицю суміжності графа $G = (V, E)$.

2. Далі відбувається динамічний перерахунок ваг маршрутів. Замість класичної метрики найкоротшого шляху, алгоритм ЕМЕКТ розраховує «енергетичну ціну» кожного транзитного переходу з урахуванням поточного стану

змінних x_v та y_e.

3. Потім виявлені нові інформаційні потоки класифікуються, після чого алгоритм шукає для них такий маршрут, який дозволить задіяти вже активне обладнання, уникаючи увімкнення нових портів.

4. Після оновлення таблиць потоків, алгоритм визначає множину портів, через які не проходить жоден потік і надсилає команди на переведення їх у режим низького енергоспоживання LPI. Якщо всі порти комутатора переходять у стан $y_e = 0$, контролер ініціює переведення всього шасі вузла у режим глибокого сну ($x_v = 0$).

Щоб змусити класичні алгоритми пошуку шляхів (наприклад, алгоритм Дейкстри) працювати в інтересах енергозбереження, необхідно трансформувати вагу кожного ребра графа мережі. У запропонованому методі вага лінії зв'язку W_e не є сталою величиною. Вона динамічно обчислюється SDN-контролером на основі поточного енергетичного стану та рівня завантаження:

$$W_e = \alpha * \Delta P_{dyn}^{v}(f_{new}) + \beta * \tau_{wake} * \left(1 - y_e^{prev}\right) + \gamma * \frac{1}{C_e - f_e - f_{new}}, \qquad (3.1)$$

де $\Delta P_{dyn}^{v}(f_{new})$ – прогнозований приріст динамічної потужності цільового вузла у разі додавання до нього нового потоку обсягом f_{new}. Згідно з поліномом, завантажені вузли споживають енергію нелінійно, тому алгоритм шукає баланс, уникаючи як недовантажених, так і критично перевантажених вузлів;

$\tau_{wake} * \left(1 - y_e^{prev}\right)$ – штраф за «пробудження». Якщо лінія зв'язку перебуває у сплячому режимі ($y_e^{prev} = 0$), її використання призведе до затримки τ_{wake} та витрати енергії на зміну стану порту. Тому алгоритм штучно завищує вагу таких ліній, надаючи абсолютний пріоритет уже активним каналам ($y_e^{prev} = 1$, штраф дорівнює нулю);

$\frac{1}{C_e - f_e - f_{new}}$ – бар'єрна функція перевантаження. У міру того, як консолідований трафік f_e наближається до фізичної межі каналу C_e, знаменник прямує до нуля, а

загальна вага W_e стрімко зростає до нескінченності. Це математичне рішення гарантує виконання обмеження та запобігає переповненню буферів;
α, β, γ – нормалізаційні коефіцієнти, які налаштовуються адміністратором мережі для балансування пріоритетів між економією енергії та показниками QoS.

Процес консолідації трафіку в алгоритмі ЕМЕКТ виконується ітеративно. На початку кожного циклу оптимізації контролер формує матрицю вимог на передачу даних D, де кожна вимога визначається як $D_k = (S_k, d_k, R_k)$ – джерело, отримувач та необхідна пропускна здатність відповідно.

Спершу відбувається сортування потоків. Оскільки порядок маршрутизації в евристиці суттєво впливає на кінцевий результат, алгоритм спочатку сортує всі потоки з матриці D у порядку спадання їхнього обсягу R_k. Спочатку маршрутизуються масивні потоки, наприклад, передача 3D-моделей або бекапів, які займають левову частку смуги пропускання. Дрібні потоки маршрутизуються пізніше, оскільки їх легко додати у залишкові ємності вже активних каналів зв'язку.

Далі відбувається побудова маршруту та валідація QoS. Для кожного потоку k у відсортованому списку алгоритм виконує пошук найкоротшого шляху P_k на графі $G = (V, E)$ з використанням динамічних ваг W_e. Отримавши потенційний шлях, алгоритм не застосовує його миттєво, а спочатку проводить строгу перевірку на відповідність системі обмежень:

1. Перевіряється умова наскрізної затримки: $D_{total} \leq D_{max}$. Для цього контролер оцінює час у чергах систем масового обслуговування D^v_{queue} на обраних комутаторах з урахуванням додавання нового трафіку.

2. Перевіряється ризик втрати пакетів: оновлений коефіцієнт завантаження підставляється у формулу Ерланга, щоб переконатися, що $p^v_{loss} \leq p_{target}$.

Якщо хоча б одне обмеження порушується, алгоритм тимчасово видаляє найбільш перевантажене (або найповільніше) ребро зі своєї внутрішньої копії графа і повторює пошук альтернативного шляху. Якщо маршрут знайдено і він

проходить перевірку QoS, контролер резервує ємність R_k на всіх лініях шляху P_k.

Після успішного розподілу всього масиву потоків мережа переходить до стану високої консолідації. На цьому етапі SDN-контролер аналізує кінцевий стан інфраструктури:

– для кожного ребра $e \in E$: якщо сумарний зарезервований потік $f_e = 0$, контролер надсилає команду модифікації порту для його програмного вимкнення або ініціює перехід у режим LPI;

– для кожного комутатора $v \in V$: якщо всі його транзитні порти переведено у стан сну, контролер ініціює команду глибокого сну для шасі, задовольняючи ієрархічне обмеження.

Для чіткого програмування логіки контролера у третьому розділі, евристичний метод формалізовано у вигляді псевдокоду (Алгоритм 3.1).

Алгоритм 3.1. Евристичний метод консолідації трафіку EMEKT

Вхідні дані:

Граф мережі $G(V, E)$;

Матриця потоків $D = \{D_1, D_2, \dots, D_m\}$;

Поточні стани $X = \{x_v\}, Y = \{y_e\}$;

Вимоги QoS: D_{max}, p_{target}.

Вихідні дані:

Оптимальні маршрути P;

Оновлені стани X', Y' (для мінімізації P_{total}).

Відсортувати D за спаданням вимог пропускної здатності R_k

Для кожного потоку $D_k(S, d, R)$ з матриці D:

Поки шлях не знайдено:

Для кожного $e \in E$ розрахувати вагу W_e за формулою (3.1)

Знайти шлях P_k від S до d за допомогою алгоритму Дейкстри з вагами W_e

Якщо шлях P_k не знайдено:

Повернути помилку маршрутизації (брак ресурсів)

Обчислити очікувану затримку D_{total} та втрати p_{loss} вздовж P_k

Якщо ($D_{total} \leq D_{max}$) ТА ($p_{loss} \leq p_{target}$):

Зарезервувати ресурси: $f_e = f_e + R_k$ для всіх $e \in P_k$

Зафіксувати шлях P_k для потоку D_k

Вийти з циклу пошуку (Break)

Інакше:

Тимчасово видалити «вузьке місце» з графа G

Для кожного $e \in E$:

Якщо f_e == 0:

y_e = 0

Інакше:

y_e = 1

Для кожного $v \in V$:

Якщо SUM(y_e) для інцидентних портів == 0:

x_v = 0

Інакше:

x_v = 1

Надіслати команди налаштування потоків та інструкції з модифікації портів на всі активні комутатори

Ефективність впровадження алгоритмів у SDN-мережах прямо залежить від їхньої асимптотичної складності. Це зумовлено необхідністю централізованого опрацювання даних без зниження продуктивності керуючого елемента.

На відміну від точного розв'язання задачі MILP, яка має експоненційну складність $O(2^{|V|+|E|})$ і є обчислювально непідйомною для топологій рівня дата-центру (наприклад, багаторівнева комутаційна архітектура з сотнями комутаторів), запропонований алгоритм EMEKT працює за поліноміальний час.

Сортування масиву з $|D|$ потоків виконується за час $O(|D|log|D|$. Основний цикл для кожного потоку використовує алгоритм Дейкстри з пріоритетною чергою, складність якого становить $O(|E| + |V|log|V|$. У найгіршому випадку алгоритму доведеться відкидати несумісні за QoS канали і повторювати пошук, що обмежено константою C_{retry}. Фаза деактивації потребує лише лінійного проходу по множинах ребер та вершин $O(|E| + |V|)$. Таким чином, загальна асимптотична складність алгоритму EMEKT становить:

$$O(|D|log|D| + |D| * C_{retry} * (|E| + |V|log|V|)) \quad (3.2)$$

де $|D|$ – кількість інформаційних потоків, які необхідно маршрутизувати;
$|V|$ – кількість вузлів (комутаторів) у графі мережі;
$|E|$ – кількість ребер (ліній зв'язку);
C_{retry} – константа, що обмежує кількість повторних спроб пошуку шляху при порушенні умов QoS.

Оскільки точне розв'язання багатовимірної цільової функції з нелінійними обмеженнями QoS належить до класу NP-складних задач і є критично повільним для динамічних середовищ, розроблено евристичний метод енергоефективної консолідації трафіку. Алгоритм, використовуючи спеціальну енергетичну метрику ваги ребер, інтелектуально консолідує потоки даних на найменш енергоємних ділянках мережі та проактивно переводить вивільнене обладнання у режим сну без порушення угоди про рівень обслуговування. Доведена поліноміальна складність алгоритму гарантує його життєздатність та ефективність при інтеграції в SDN-контролер реального часу. Таким чином, отримана теоретична база та алгоритмічна логіка методу консолідації EMEKT виступають повноцінною основою для наступного етапу дослідження – програмної реалізації керуючого модуля мовою Python та проведення імітаційного моделювання у середовищі Mininet.

3.2 Архітектура програмно-технічного засобу управління інфраструктурою

При розробці системи основна увага приділялася використанню програмно-визначених мереж. Вони мають вбудовану шарову структуру, де кожен шар виконує свою функцію. Керування мережею відокремлене від передачі даних, а не об'єднане з ними в один фізичний вузол, як це було раніше. Це означає, що існує поділ пристроїв, призначених для виконання різних завдань, які за замовчуванням перебувають у повністю окремих режимах роботи. Структура SDN загалом складається з трьох частин: внизу ієрархії – апаратна частина, посередині – керування, а нагорі – обробка всього цього. Хоча в системі є різні рівні функцій, що дозволяють працювати окремим компонентам, вони взаємодіють один з одним, надсилаючи їм дані, тобто передаються сигнали, події, інші мережеві повідомлення.

Найнижчий рівень забезпечує зв'язок між пристроями, які, по суті, є байтами. Крім комутаторів, існують також хости, одна частина яких передає дані, інша – чекає на їх надходження.

Основну роль у функціонуванні мереж SDN відіграють комутатори OpenFlow. Вони мають дуже схожий вигляд зі звичайним апаратним забезпеченням, що саме робить їх придатними для мереж SDN: функція маршрутизації, яка більше не розкидана по всій мережі, як це було раніше, і є у всіх пристроїв, централізована. Таким чином, під контроль ставиться і робота таких протоколів, як OSPF або BGP. Комутатори просто дотримуються інструкцій, записаних у таблицях потоків. Обробка таблиць потоків здійснюється апаратним забезпеченням комутатора, надзвичайно швидко, оскільки це робиться на рівні кремнію, а не на програмному рівні. Кожен порт може незалежно обмежувати потужність. Це називається принципом локальної оптимізації споживання енергії. Все це координується з одного місця. Крім того, використання стандарту Energy Efficient Ethernet 802.3az також сприяє економії енергії апаратним забезпеченням.

Замість громіздкого фізичного обладнання, яким були б десятки пристроїв

SDN, розташованих у ієрархічній деревоподібній топології, використовується система емуляції Mininet, в якій, по суті, можна працювати зі справжньою мережею. Кожен з вузлів працює з програмними комутаторами Open vSwitch на основі мережевого рівня ядра операційної системи Linux. Система Mininet дозволяє нам створювати віртуальну мережу між мережевими вузлами, і навіть якщо все тут віртуальне, воно чудово працює. Mininet сумісний зі справжнім обладнанням, оскільки ми можемо застосувати нашу реалізацію EMEKT до справжнього обладнання без модифікації вихідного коду.

OpenFlow 1.3 використовувався як протокол для зв'язку між рівнем керування та мережею, щоб зробити обмін повідомленнями між ними більш надійним. Він реалізований у захищеному TCP-каналі. Таким чином, існує багато корисних функцій: підраховується кількість даних, що передаються кожним пакетом, що проходить через порт, щоб ми могли точно бачити, скільки даних проходить через кожен порт. Оскільки лічильники автоматично оновлюються, система може постійно відстежувати та аналізувати дані для обмеження кількості та обсягу пакетів, забезпечуючи цілодобове спостереження за станом кожного порту. Правила таблиці потоків також можуть бути змінені під час роботи системи, дозволяючи програмно перемикати стан живлення порту, просто вмикаючи або вимикаючи його. Це зручно для керування живленням та контролю доступу без необхідності окремого налаштування.

Апаратно-програмне рішення на базі програмного контролера Ryu SDN відповідає за реалізацію центру керування мережею. Сам контролер є основною програмною платформою для взаємодії з мережею. Це програма на Python, оскільки її архітектура була ключовим фактором при її розробці: можливість обробки подій без блокування була обов'язковою, і вимога полягала в суворому дотриманні версії OpenFlow 1.3 (без прогалин у сумісності). Зручний API програмування на Python дозволив легко та швидко розробляти та тестувати нестандартні протоколи, наприклад, для аналізу топології мережі за допомогою NetworkX. Контролер також забезпечує базовий зв'язок TCP з кожним комутатором у локальній мережі та одночасно встановлює контекст для програмної реалізації на

рівні комутатора.

Рівень додатків визначається програмною реалізацією. Пакет розробки програмного забезпечення також відповідає за визначення інтелектуального маршрутизуючого пристрою ЕМЕКТ та його функціональності. Щоб це сталося, ЕМЕКТ інтегрується в платформу через адаптерний модуль. Архітектура ЕМЕКТ складається з чотирьох логічних компонентів, і цей модульний поділ був необхідний для забезпечення гнучкого розширення логіки ЕМЕКТ та заміни окремих компонентів. Усі компоненти працюють паралельно, але взаємодіють один з одним для підтримки цілісності системи.

Моніторинг стану мережевого середовища є основною функцією компонента контролю. Він автоматично визначає топологію мережі за допомогою LLDP, без ручного налаштування. Після аналізу отриманих повідомлень автоматично оновлюється граф мережі. Для розрахунку навантаження на кожен з мережевих каналів цей компонент періодично надсилає повідомлення комутаторам у мережі. Ці повідомлення складаються з кількох підкомпонентів. Частка використання каналу обчислюється за цими даними.

Компонент динамічного обчислення реалізує математичну модель визначення ваг для кожного фізичного каналу зв'язку. У цьому компоненті кожному фізичному каналу даних призначається метрика вартості для кожного фізичного каналу даних. Вартість встановлюється на максимальне значення, якщо порт був неактивним (у режимі енергозбереження). Тим часом, метрики вартості встановлюються на мінімальне значення, якщо порт активний і повністю функціонує. Це гарантує, що наступні компоненти обробки враховують лише ті фізичні канали даних, які фактично використовуються.

Компонент маршрутизації та перевірки якості обслуговування спрацьовує при появі нового потоку даних у мережі. Модель побудована на графовому підході, де довжина комунікаційних каналів періодично перераховується. Таким чином, оптимальний шлях вибирається не з точки зору найкоротшої відстані у традиційному розумінні, а з точки зору майже оптимального. Алгоритм Дейкстри, що використовується тут, модифікований відповідно до поточної ситуації в

каналах, і перед остаточним вибором він проходить ретельну перевірку системою нерівностей з параметрами, що контролюються. Для забезпечення стабільності параметрів сервісу QoS спочатку оцінюється пропускна здатність, а потім можлива затримка всіх комунікаційних каналів.

Після завершення перерозподілу даних активується блок керування живленням та статусом. Він виявляє канали, які більше не передають пакети даних, і генерує набір керуючих сигналів для переведення цих каналів у режим енергозбереження. Якщо всі порти комутаторів переходять у режим низького енергоспоживання, автоматично запускається логіка вимкнення шасі, що дозволяє ефективно економити енергетичні ресурси без втручання людини.

Запропоноване рішення на базі EMEKT вирізняється масштабованістю завдяки відокремленню площини керування від площини даних. Централізація функцій маршрутизації та контролю QoS у контролері звільняє мережеве обладнання від складних обчислень, що дозволяє використовувати простіші та енергоефективніші пристрої.

Це забезпечує повну видимість топології та енергоспоживання мережі в реальному часі. На відміну від традиційних методів, контролер здійснює глобальне управління ресурсами: він миттєво реагує на сплески трафіку, консолідуючи навантаження на активних лініях та переводячи надлишкові елементи в режим глибокого енергозбереження (наприклад, LPI).

Перенесення логіки керування у таблиці потоків дозволяє апаратним вузлам фокусуватися виключно на швидкій передачі пакетів із мінімальними витратами потужності. Тісна інтеграція програмного та апаратного рівнів мінімізує навантаження на процесори у станах очікування та забезпечує швидку реактивацію ресурсів (рис. 3.1).

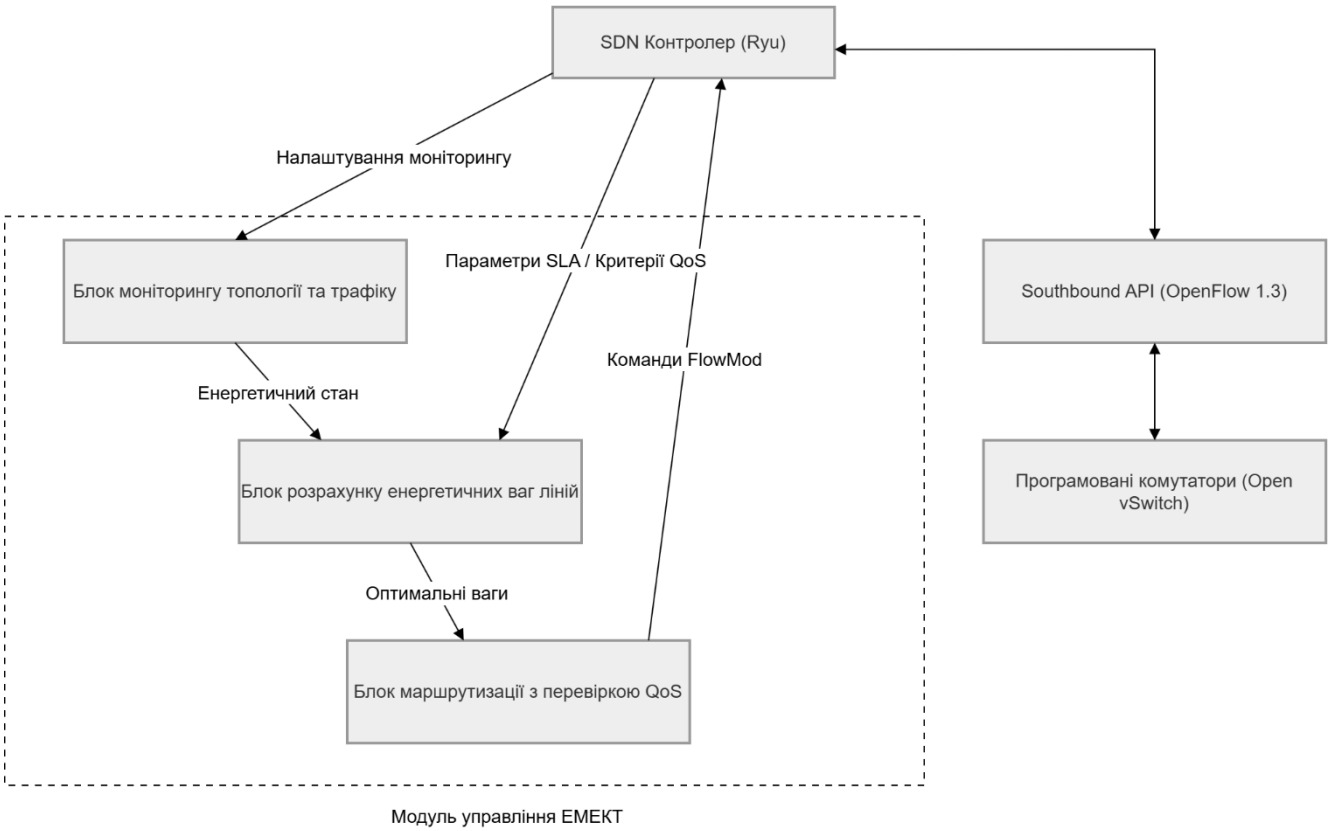


Рисунок 3.1 – Структурна схема розробленого програмно-технічного засобу

Для практичної реалізації запропонованого підходу до оптимізації енергоспоживання було розроблено структурно-функціональну архітектуру системи управління, яка базується на концепції програмно-конфігурованих мереж. Вона передбачає чітке розділення площини передачі даних, що складається з програмованих комутаторів, та логічно централізованої площини управління. Таке рішення дозволяє гнучко збирати низькорівневу метрику про стан інфраструктури та оперативно впроваджувати керуючі впливи для динамічної зміни станів живлення портів і комутаційних вузлів.

Організація внутрішніх зв’язків архітектури забезпечує узгоджену взаємодію між сервісами моніторингу топології, обчислення динамічних енергетичних ваг каналів та блоками маршрутизації з жорстким контролем параметрів угод про рівень послуг. Усі інформаційні потоки та керуючі команди проходять через стандартизовані інтерфейси взаємодії з обладнанням, що гарантує високу масштабованість та живучість мережевого середовища в умовах інтенсивного навантаження.

Як видно з рисунка 3.1, центральним елементом прийняття рішень у розробленій архітектурі є програмний модуль EMEKT, що функціонує всередині SDN-контролера. Для забезпечення його коректної роботи необхідна сувора послідовність виконання дій, детальний опис якої наведено у наступному підрозділі.

3.3 Алгоритмічне забезпечення модуля енергоефективної маршрутизації

Щоб втілити ідею інтелектуального управління мережевою інфраструктурою, потрібен структурований і зрозумілий алгоритм для контролера SDN. У класичній мережі кожен маршрутизатор приймає рішення незалежно, на основі протоколів маршрутизації, таких як OSPF або BGP. В SDN вся логіка реалізована на рівні контролера, тому програмний модуль EMEKT працюватиме як системна програма контролера SDN Ryu.

Для архітектури алгоритм EMEKT реалізує подієву модель у поєднанні з фоновими процесами. Він постійно реагує на події в мережі, що надходять асинхронно (поява нового пакета, зміна стану порту) і періодично запускає цикл оновлення стану системи, збираючи статистику.

Логіку алгоритмічної підтримки запропонованого методу можна представити як серію з п'яти основних кроків.

На початку відбувається налаштування системи, конфігурація мережевої топології та створення базового графа. Коли контролер SDN запускається і встановлюються безпечні TCP-з'єднання з усіма доступними на даний момент комутаторами OpenFlow на рівні інфраструктури, цей момент є початковою точкою алгоритму. Під час певної процедури зв'язку, яка називається "рукостисканням", кожен комутатор надсилає контролеру повідомлення про свої параметри, вказуючи свій особистий номер, ідентифікатор DPID та інформацію про технічні характеристики пристрою.

Для побудови шляху необхідно отримати нову карту топології. Процес виявлення топології активується контролером, який надсилає широкомовні пакети

LLDP з кожного порту всіх підключених комутаторів. Пакет LLDP містить DPID пристрою-відправника та номер вихідного порту. Коли сусідній комутатор отримує цей пакет, він, згідно із заздалегідь визначеним правилом, миттєво перенаправляє його контролеру як запит на обробку невідомого трафіку.

Після аналізу інкапсульованих пакетів LLDP, EMEKT ідентифікує двонаправлені фізичні зв'язки між комутаторами інфраструктури та починає будувати абстракцію мережі контролера. Топологія мережі описується як зважений математичний граф $G = (V, E)$, де вершини V представляють комутатори, а ребра E – канали передачі. У коді це представлено як структури суміжності, що є списковим або матричним представленням. Бібліотека NetworkX використовується для представлення мережі в Python, і вона дозволяє системі одночасно розраховувати максимальну пропускну здатність кожного ідентифікованого каналу зв'язку.

Початкова ініціалізація енергетичних станів виконується на другому кроці. При запуску мережі передбачається, що всі комутатори та канали активні ($x_v = 1, y_e = 1$). Усі ребра графа отримують метрику за замовчуванням 1, або обернено пропорційну його пропускній здатності.

Збір статистичних даних стає циклічним процесом. Неможливо консолідувати потоки даних без чіткого розуміння навантаження каналу. Система EMEKT має таймер, який регулярно запускає процедуру моніторингу стану мережі.

Коли виконується перевірка, контролер надсилає запити статистики до кожного доступного вузла $v \in V$. У відповідь пристрої передають лічильники байтів, що пройшли через кожен порт з моменту їх активації.

Отримані відповіді зчитуються та обробляються алгоритмом. Поточна швидкість передачі даних (пропускна здатність) BW_{used}^{e} для кожної лінії e обчислюється шляхом знаходження різниці переданих байтів між двома послідовними циклами опитування, поділеної на інтервал часу:

$$BW_{used}^{e} = \frac{(B_{current} - B_{previous}) * 8}{\Delta t}, \tag{3.3}$$

де BW_{used}^{e} – поточна завантажена смуга пропускання лінії e (вимірюється в біт/с);
$B_{current}$ – значення лічильника переданих байтів, отримане в поточному циклі опитування;
$B_{previous}$ – значення того ж лічильника, зафіксоване в попередньому циклі опитування;
Δt – часовий інтервал між двома послідовними запитами контролера (інтервал дискретизації).

Часовий інтервал – це ще один параметр алгоритму, який слід відповідно налаштувати, щоб він не створював додаткового навантаження (наприклад, якщо сервіс викликається кожні менше однієї секунди Δt) або якщо алгоритм не встигне вчасно вловити піки навантаження в системі, якщо він буде занадто великим. Найкраще значення можна вибрати експериментально, залежно від масштабу мережі.

На основі розрахованої швидкості алгоритм обчислює поточну залишкову ємність кожного каналу:

$$C_{residual}^{e} = C_e - BW_{used}^{e}, \tag{3.4}$$

де $C_{residual}^{e}$ – поточна залишкова ємність каналу e.

Цей індикатор є важливим для дотримання обмеження пропускної здатності лінії, що допомагає запобігти перевантаженню комунікаційної лінії та втраті пакетів через переповнення буфера на наступних етапах алгоритму.

Наступним кроком є обробка події надходження нового потоку та маршрутизація. Це «інтелектуальна» частина модуля ЕМЕКТ. Комутатори в SDN не знають, як пересилати дані, якщо у них немає відповідного правила. Коли новий потік ініціюється кінцевим вузлом, перший пакет цього потоку надходить у мережу через вхідний порт комутатора рівня Edge. Через відсутність відповідного запису в

таблиці потоків, спрацьовує дія за замовчуванням, і пакет перенаправляється контролеру як запит на отримання інструкцій.

Отримавши відповідну подію, алгоритм EMEKT розпочинає процедуру обробки нового потоку. Спочатку він витягує дані з пакета та створює унікальний ідентифікатор потоку з заголовків, використовуючи MAC- та IP-адреси, порти TCP/UDP тощо на рівнях L2/L3/L4.

Наступним кроком є перерахунок ваги. На відміну від традиційних метрик, які є статичними і залежать лише від апаратного забезпечення, EMEKT використовує динамічну метрику енергії на основі стану матриці. Він зменшує вагу, якщо лінія активна і недовантажена, щоб стимулювати консолідацію трафіку на вже активних лініях. Навпаки, вага сплячих ліній штучно збільшується з коефіцієнтом штрафу τ_{wake}, щоб запобігти непотрібному пробудженню обладнання. Крім того, якщо залишкова пропускна здатність лінії наближається до нуля, то вага відповідної лінії наближається до нескінченності. Таким чином, така лінія не братиме участі в найкоротшому шляху через перевантаження.

На основі енергетичних ваг W_e блок маршрутизації обчислює субоптимальний маршрут з мінімальною сумарною вагою. Після цього обчислений маршрут проходить обов'язкову перевірку для забезпечення вимог якості обслуговування. Щоб перевірити це, обчислюється загальна затримка доставки пакета по вибраному маршруту D_{total} з урахуванням часу очікування в чергах. У разі, якщо отриманий результат порушує угоду про QoS, маршрут відхиляється EMEKT. Після цього алгоритм видаляє найбільш перевантажений сегмент з внутрішнього графа і повертається до одного з попередніх кроків, щоб спробувати інший варіант маршруту, доки не буде знайдено придатний варіант, що відповідає угоді про якість обслуговування.

Як тільки алгоритм EMEKT знайшов енергоефективний та безпечний маршрут, він повинен виконати встановлення цього маршруту в інфраструктурі. Таким чином, маршрут P, знайдений алгоритмом, розкладається на набір дій для кожного комутатора, що бере участь у маршруті.

Для налаштування маршруту контролер надсилає комутаторам команди

модифікації потоків. Таке повідомлення містить:

– критерії співпадіння (IP/MAC адреси потоку);

– дію, яка вказує, на який фізичний вихідний порт комутатор повинен пересилати пакет для його передачі по обчисленому маршруту;

– пріоритет правила.

Важливою деталлю цього етапу є керування життєвим циклом правил. Оскільки пам'ять TCAM у комутаторі є обмеженою та дорогою, алгоритм EMEKT у наборі правил встановлює тайм-аути. Коли кінцеві хости завершують передачу даних, і якщо комутатор не має інших пакетів цього потоку протягом цього тайм-ауту, він безумовно видаляє правило зі своєї пам'яті TCAM на апаратному рівні. Потім комутатор асинхронно надсилає повідомлення контролеру, дозволяючи модулю EMEKT дізнатися, що передачу завершено, і ресурси, зайняті правилом, тепер доступні для іншого потоку.

Енергоефективність алгоритму EMEKT забезпечується останнім кроком циклу, де алгоритм аналізує топологію мережі та вимикає неактивні вузли одночасно зі збором та аналізом статистики під час другого кроку алгоритму.

Алгоритм досліджує матрицю навантаження: якщо аналіз статистичних даних показує, що протягом кількох послідовних циклів опитування Δt не було корисного транзитного трафіку з порту p_i жодного з комутаторів, алгоритм робить висновок, що це з'єднання є зайвим.

Для забезпечення енергоефективної роботи мережевої інфраструктури контролер ініціює процедуру, під час якої значення математичної змінної стану лінії зв'язку y_e встановлюється в нуль. У межах цього процесу керуючий вузол генерує та спрямовує комутатору команду модифікації параметрів порту, яка активує режим програмного вимкнення у поточній конфігурації. Після отримання та обробки даної команди комутаційний пристрій припиняє роботу апаратного забезпечення трансивера на відповідному порту, що забезпечує його перехід у режим низького енергоспоживання згідно зі специфікаціями стандарту Energy Efficient Ethernet (IEEE 802.3az).

Крім того, алгоритм EMEKT перевіряє умову ієрархії: якщо на комутаторі v

не залишилося транзитних портів, і всі порти цього комутатора переведено в режим низького енергоспоживання, цей комутатор може бути виключений з обчислення корисної інформації. У цьому випадку контролер ініціює процес переведення його процесора в режим глибокого сну, усуваючи таким чином споживання енергії на рівні базової статичної потужності P^{v}_{static}.

Виконавши ці дії, цикл замикається, і алгоритм повертається до моніторингу та оптимізації інфраструктури.

Щоб більш повно роз'яснити структуру програмного модуля, реалізованого у вигляді алгоритму 3.2, весь логічний потік розглянутого циклу описано за допомогою псевдокоду.

Алгоритм 3.2. Логіка модуля енергоефективної маршрутизації EMEKT

Ініціалізація $G(V, E)$ через LLDP Discovery
Встановити всі $x_v = 1, y_e = 1$
Поки контролер Ryu активний:
  Якщо таймер Δt спрацював:
    Надіслати запити статистики на всі V
    Обчислити BW_{used} та $C_{residual}$ для всіх E
    Для кожного порта p_i у E:
      Якщо $BW_{used}(p_i)$ == 0 протягом k циклів:
        y_e = 0
        Надіслати команду вимкнення порту (режим енергозбереження)
    Для кожного комутатора v у V:
      Якщо Sum(y_e) для інцидентних портів == 0:
        x_v = 0
  Якщо отримано запит на маршрутизацію нового потоку від хоста H_s до H_d:
    Витягти ідентифікатор потоку (Flow_ID)
    Для кожного ребра e у E:

Розрахувати енергетичну вагу W_e

Знайти шлях P за алгоритмом Дейкстри: P = Dijkstra(G, H_s, H_d, W_e)

Якщо P задовольняє QoS Delay та Capacity:

Для e у P:

Якщо y_e == 0 то y_e = 1, надіслати команду активації порту

Надіслати інструкції з модифікації таблиць потоків (встановити правила з часовим лімітом бездіяльності)

Інакше:

Видалити критичне ребро з G та повторити пошук

Алгоритм EMEKT забезпечує безперервну адаптацію SDN-мережі через централізований аналіз трафіку в реальному часі. У періоди низького навантаження метод консолідує потоки на мінімальній кількості ліній, переводячи неактивні порти та пристрої в режими LPI або глибокого сну для досягнення значної економії енергії.

Водночас метод проактивно реагує на сплески активності, динамічно перераховуючи ваги маршрутів та моніторячи залишкову ємність каналів. Це дозволяє миттєво активувати резервні ресурси через Southbound API, запобігаючи втратам пакетів та деградації QoS. Така інтелектуальна система із замкнутим контуром інтегрує безперервний моніторинг та енергоефективну оптимізацію (рис. 3.2).

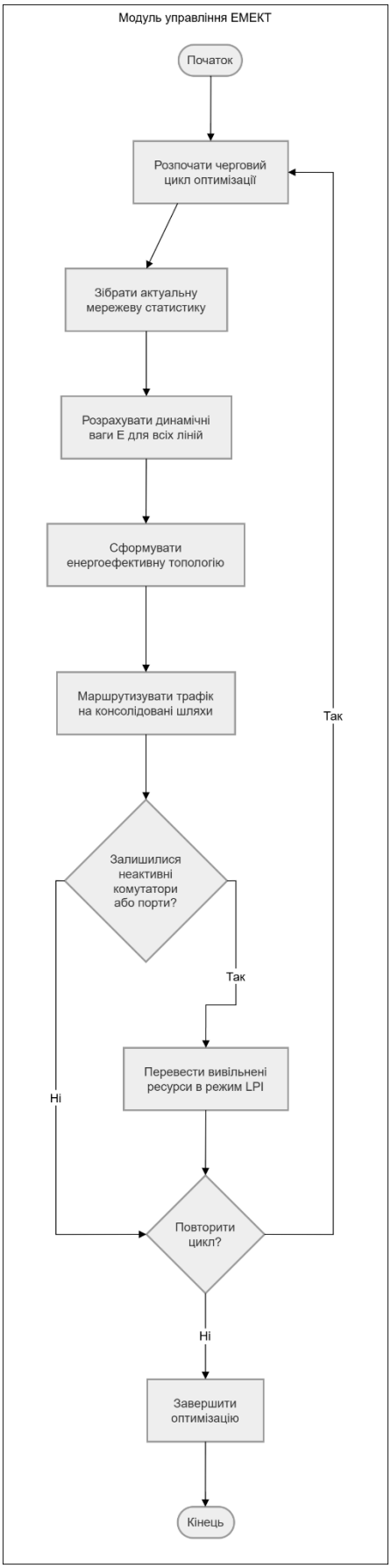


Рисунок 3.2 – Блок-схема алгоритму енергоефективної консолідації трафіку ЕМЕКТ

Як видно з наведеної блок-схеми, практична реалізація запропонованих методів керування живленням та маршрутизації повністю базується на безперервному обміні службовими повідомленнями. Забезпечення надійної інформаційної комунікації між обчислювальними компонентами програмного модуля ЕМЕКТ та фізичними пристроями інфраструктурного рівня вимагає використання стандартизованого інтерфейсу. Специфіка цієї взаємодії та структура необхідних керуючих команд розглядаються у наступному підрозділі.

3.4 Інформаційна взаємодія компонентів за протоколом OpenFlow

Реалізація функціональності розроблених алгоритмів маршрутизації та керування живленням передбачає надійну взаємодію між елементом керування та мережевою інфраструктурою, тобто площиною даних. Оскільки сучасне мережеве середовище є гетерогенним і включає обладнання різних виробників, для забезпечення необхідного рівня інтеграції інтерфейс взаємодії повинен бути суворо стандартизований.

У розробленій програмно-апаратної системі роль Southbound API виконує протокол OpenFlow. Розробляються також інші протоколи, такі як ForCES, де логічний функціональний блок може використовуватися як елемент керування, але сьогодні галузевим стандартом для побудови SDN є OpenFlow. Він використовує протокол TCP для встановлення захищеного з'єднання між контролером і комутатором і надає Ryu прямий контроль над таблицями потоків і станами портів на комутаторі.

Для роботи модуля ЕМЕКТ використовується версія специфікації OpenFlow 1.3 (і вище), оскільки попередні версії стандарту не підтримують таких функцій, необхідних для сучасних центрів обробки даних, як розширена статистика портів, багатотаблична конвеєрна обробка та підтримка протоколу IPv6.

Специфікація OpenFlow визначає структуру обміну мережевою інформацією як три типи повідомлень: від контролера до комутатора, асинхронні повідомлення та симетричні повідомлення. Модуль ЕМЕКТ використовує всі три типи для

встановлення процедури замкнутого контролю, що включає моніторинг, оптимізацію та деактивацію ресурсів.

Асинхронні повідомлення надсилаються лише комутатором площини даних і призначені для сповіщення контролера про певні події в мережі. Щоразу, коли кінцевий хост випускає пакет, він надходить на вхідний порт комутатора, який бере заголовок пакета та перевіряє таблицю потоків відповідно до MAC-адреси, IP-адреси, портів TCP/UDP та метаданих у ній. Якщо відповідне правило відсутнє, виникає ситуація неуспішного пошуку, за якої комутатор інкапсулює пакет у спеціальний запит на отримання інструкцій і передає його контролеру Ryu для модуля EMEKT. Таке сповіщення розглядається як програмне переривання, яке ініціює виконання кроку евристичного алгоритму, коли динамічні енергетичні ваги графа перераховуються та визначається новий енергетично-оптимальний шлях.

Потім контролер надсилає інструкції керування потоком до всіх транзитних комутаторів з фільтрами трафіку для відповідності трафіку та вибору портів. Одним з основних принципів дизайну EMEKT є те, що він намагається мінімізувати використання пам'яті з асоціативним доступом, оскільки це один з найбільш енергоємних компонентів комутатора. Через високе енергоспоживання TCAM споживає значно більше енергії, ніж модулі RAM, тому EMEKT не призначає правило до постійного запису. Натомість алгоритм призначає правилам тайм-аут неактивності, і після закінчення часу він видаляє запис правила. Таким чином, енергетична вартість P_{flow} зменшується і сповіщає про генерацію видалення правила. Ця зворотна інформація є життєво важливою для алгоритму, щоб він міг негайно отримувати інформацію про звільнення ресурсів каналу. Алгоритм зменшення навантаження на TCAM, який є одним з найбільш енергоємних пристроїв (з енергоспоживанням у 100 разів вищим, ніж у звичайної RAM) на пріоритеті для модуля EMEKT, зображено нижче.

Для реалізації одного з етапів алгоритму EMEKT, а саме моніторингу стану мережі, використовується багатоелементний механізм запитів: таймер контролера ініціює запит статистики портів кожні Δt, після чого комутатори надсилають набір апаратних лічильників байтів, пакетів та помилок для кожного каналу. З динаміки

зміни цих характеристик (перш за все з обсягів отриманих та переданих даних) розраховується значення залишкової пропускної здатності каналу $C^e_{residual}$, яке використовується для завчасного запобігання можливого перевантаження та забезпечення QoS мережі відповідно до обмежень математичної моделі щодо пропускної здатності та затримок мережі.

Проте ключова особливість ЕМЕКТ – це пряме керування обладнанням у системі, коли замість простих так/ні вирішень запускаються справжні мережеві операції. Такий підхід стає можливим завдяки спеціальній функції, що змінює стан портів через програмний інтерфейс, даючи контролеру змогу налаштовувати інтерфейси без фізичного доступу.

Коли блок моніторингу ЕМЕКТ фіксує, що один із каналів комутатора не пересилає корисних даних протягом декількох циклів опитування – система визнає його зайвим. Замість нього контролер готує налаштування для порту, де вказує точний номер фізичного інтерфейсу та активує режим вимкнення. Цю інструкцію комутатор потім спрямовує шару апаратної взаємодії, щоб точно регулювати роботу трансивера. На фізичному рівні це призводить до:

− вимкнення живлення лазерного або мідного трансивера на відповідному порту;

− переведення інтерфейсу низького енергоспоживання в режимі очікування, що регламентується стандартом Energy Efficient Ethernet;

− зниження тактової частоти внутрішніх обчислювальних шин комутаційної матриці, які відповідали за обслуговування цього порту.

Таким чином, у цільовій функції енергоспоживання P_{total} компонент $P^e_{link} * y_e$ фізично зводиться до нуля. У випадку, коли алгоритм ЕМЕКТ прогнозує зростання трафіку та необхідність активації резервних шляхів для запобігання втраті пакетів, генерується аналогічна за структурою команда модифікації порту. Відмінність полягає в тому, що замість вимкнення встановлюється статус активного стану. У цьому випадку алгоритм враховує час, що виникає після пробудження системи, як штраф за пробудження τ_{wake}.

Незважаючи на позитивні риси централізованого управління, архітектура

SDN страждає від суттєвого недоліку, генеруючи високі рівні службового трафіку між комутаторами та контролером. У великих мережах це може призвести до надмірної кількості запитів до контролера під час масової появи нових з'єднань, що перевантажує процесор SDN-контролера, суттєво збільшуючи затримку та призводячи до втрати ефекту загальної економії енергії.

Для зменшення згаданих накладних витрат у модулі EMEKT реалізовано низку інженерних оптимізацій протокольного обміну, наприклад, реалізацію агрегації потоків, встановлюються не за конкретними IP-адресами хостів, а за конкретними підмережами, що суттєво зменшує кількість запитів комутаторів до контролера, а також адаптивне таймування моніторингу, шляхом збільшення часового інтервалу для надсилання запитів статистики у періоди низького навантаження для економії енергії, та автоматичного зменшення цього часового інтервалу у разі зростання активності, що дозволяє алгоритму залишатися дуже швидким. Крім того, щоб не блокувати керуючий канал алгоритму через деактивацію портів, реалізовано позасмугове управління.

Таким чином, запропонований підхід оптимізованого застосування повідомлень протоколу OpenFlow для алгоритму EMEKT дозволить зменшити накладні витрати службового трафіку, абстрагуватися від специфіки реалізації обладнання конкретних вендорів та створити ефективний інструмент для мікрорівневого контролю енергоспоживання інфраструктури.

3.5 Висновки до третього розділу

Третій розділ кваліфікаційної роботи описує перехід від теоретичного та математичного моделювання до практичного проєктування програмно-апаратного інструменту для управління інфраструктурою SDN. Зокрема, розроблено багаторівневу архітектуру системи, розділено площини керування та даних, в якій центральним елементом системи є SDN-контролер Ryu та програмний модуль EMEKT. Апаратна частина системи включає програмовані комутатори з підтримкою OpenFlow. Для покращення масштабованості системи та розподілу

обчислювального навантаження між апаратними комутаторами, модуль керування був логічно розкладений на чотири взаємопов'язані блоки (моніторинг топології, динамічний розрахунок енергетичних ваг, маршрутизація з перевіркою QoS та управління живленням).

Також детально описано алгоритмічне забезпечення системи: покрокова логіка роботи програмного забезпечення, від створення графа мережевої інфраструктури та збору статистики з комутаторів до розрахунку динамічних ваг портів комутаторів та зміни таблиць потоків на комутаторах в результаті. В результаті алгоритм ЕМЕКТ створює замкнутий цикл, де можливе швидке ущільнення трафіку зі зменшенням навантаження та проактивне включення резервних каналів при його різкому збільшенні. Також введено та описано чіткий формат обміну інформацією між компонентами системи з використанням протоколу OpenFlow. Він включає використання запитів на обробку невідомого трафіку, інструкцій з налаштування потоків, звітів мережевої статистики та команд модифікації портів. Ці повідомлення слугують тригерами для виклику алгоритму оптимізації, визначенням часових інтервалів для алгоритму оптимізації з урахуванням енергозбереження для TCAM, аналізом рівня поточного навантаження та безпосереднім перемиканням портів з режиму очікування на режим низького енергоспоживання відповідно.

Таким чином, розроблена багаторівнева архітектура, деталізована алгоритмічна схема ЕМЕКТ та уніфікований формат протокольного інтерфейсу формують комплексну науково-практичну основу для програмної реалізації системи управління. Сформований логічний базис дозволяє перейти до етапу написання коду мовою Python з використанням функціональних можливостей SDN-контролера Ryu, а також створює необхідні умови для проведення масштабного симуляційного моделювання в середовищі Mininet. Це забезпечує можливість всебічної перевірки життєздатності запропонованих рішень у динамічних сценаріях, створюючи передумови для переходу до наступного етапу – кількісної оцінки ефективності енергозбереження та аналізу стабільності показників QoS в умовах реалістичного мережевого навантаження.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МЕТОДУ ЗАБЕЗПЕЧЕННЯ ЗЕЛЕНИХ ОБЧИСЛЕНЬ НА БАЗІ ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖ

4.1 Програмна реалізація модуля енергоефективної маршрутизації

Для реалізації контрольної площини програмно-конфігурованої мережі було обрано мову програмування Python. Ця мова обґрунтована своїм високим рівнем абстракції, здатністю до швидкого та динамічного прототипування складних мережевих алгоритмів та широким спектром спеціалізованих бібліотек для роботи з графами та науковими обчисленнями.

Фреймворк контролера Ryu SDN використовується як базова операційна система мережі. Ryu повністю розроблений на Python і використовує бібліотеку Eventlet, щоб дозволити коду виконуватися одночасно, на відміну від деяких інших популярних контролерів таких як POX або Floodlight. Завдяки архітектурі Eventlet, контролер може обслуговувати тисячі вхідних запитів від комутаторів без блокування основного потоку виконання, що важливо для зменшення затримки маршрутизації. Крім того, Ryu має повну та стабільну підтримку протоколу OpenFlow версії 1.3, що необхідно для мікроуправління потужністю портів.

Для написання та реалізації математичної моделі мережі, а також для виконання алгоритмів пошуку найкоротшого шляху, у вихідному коді застосовується бібліотека NetworkX. Вона підтримує динамічне оновлення ваг ребер у пам'яті контролера та оптимізовану реалізацію алгоритму Дейкстри.

Об'єднання цих інструментів у єдину екосистему дозволяє повністю реалізувати логіку інтелектуальної енергозберігаючої маршрутизації. Загальну ієрархію взаємодії обраного інструментарію та стек технологій розробленого програмно-технічного засобу наочно представлено на рис. 4.1.

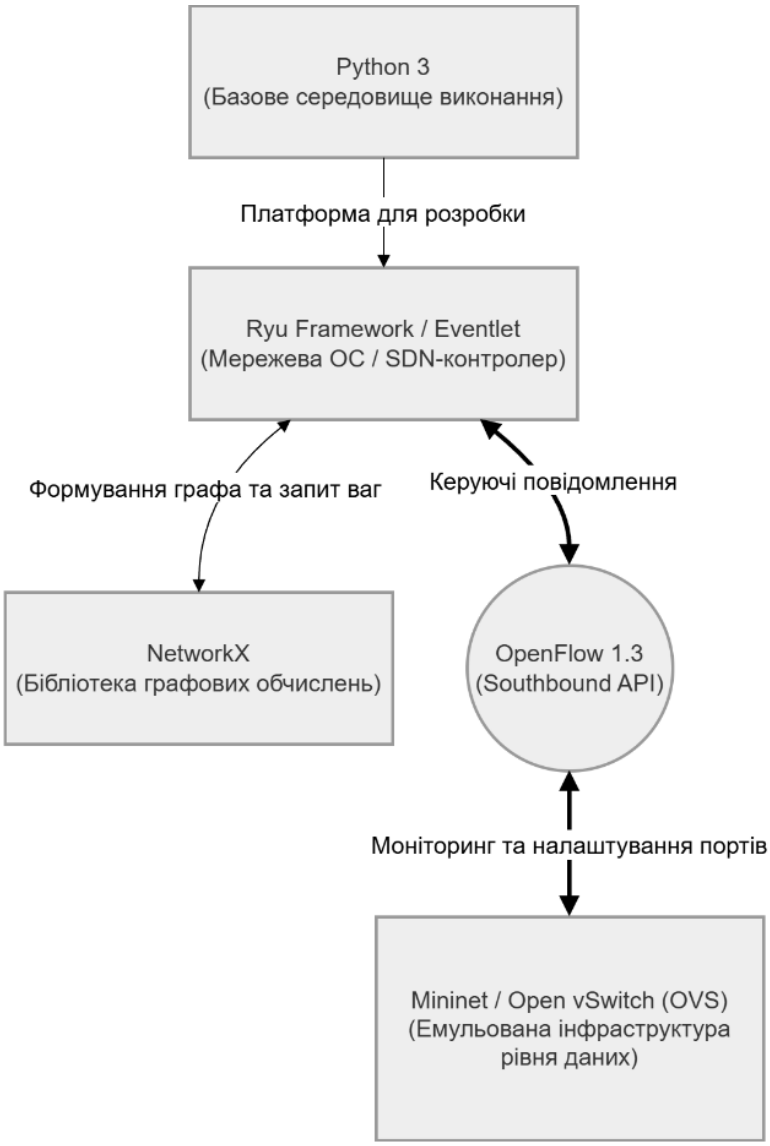


Рисунок 4.1 – Стек технологій та ієрархія взаємодії програмно-технічного засобу

Програмний модуль ЕМЕКТ реалізовано як окремий системний додаток на основі фреймворку Ryu. Логіка програми ЕМЕКТ має свої функціональні методи та структури даних, засновані на архітектурних блоках системи, щоб забезпечити її модульність.

Контролер ЕМЕКТ є основним компонентом, який керує всіма процесами розробки. Архітектура внутрішніх модулів може містити:

– динамічну модель графа, яка зберігає поточну топологію мережі у вигляді неорієнтованого графа;

– механізм відстеження вузлів, що служить таблицею навчання MAC-адрес для кінцевих хостів;

– блок збору статистики та аналізу продуктивності, який накопичує дані про обсяг переданої інформації та визначає швидкість на кожному мережевому порту;

– реєстр стану обладнання, що зберігає значення змінної енергетичного стану (активний режим або режим сну).

Автоматичне виявлення структури мережі модуля досягається за допомогою вбудованих сервісів топології, які аналізують службові пакети у фоновому режимі та генерують події щоразу, коли виявляються нові комутатори або зв'язки.

Для наочного представлення архітектури розробленого програмного забезпечення на рис. 4.2 наведено спрощену UML-діаграму класу EMEKTController. Вона відображає наслідування від базового компонента фреймворку Ryu, а також інкапсуляцію ключових атрибутів (математичного графа мережі та блоків статистики) і методів, що реалізують логіку збору даних та обробку нових потоків трафіку.

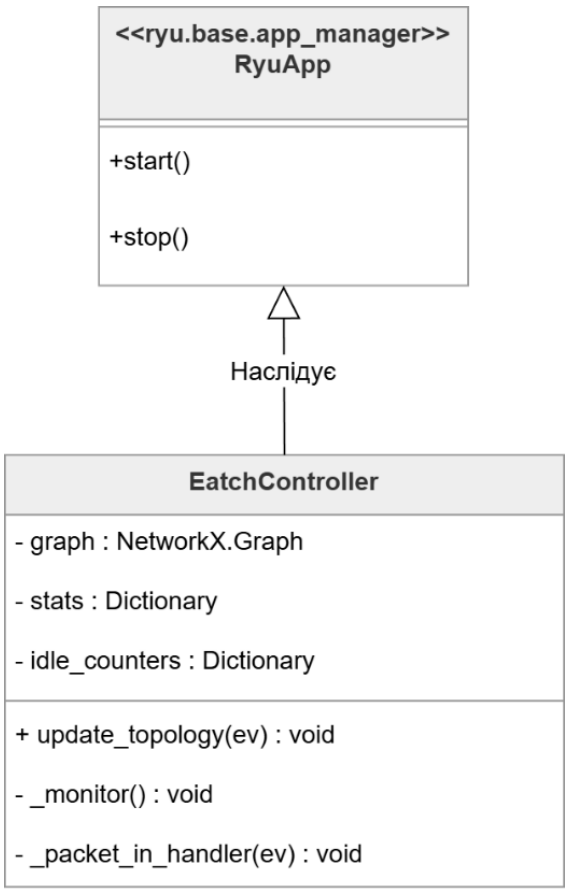


Рисунок 4.2 – UML-діаграма класу модуля EMEKT

Згідно з розробленим алгоритмом, керуючий модуль здійснює циклічний збір мережевої статистики для оцінки її завантаженості. Для цього в контролері реалізовано окремий фоновий процес моніторингу, який функціонує паралельно з основними операціями.

Цей процес працює як нескінченний цикл, який, з урахуванням періодичності Δt, опитує всі комутатори, зареєстровані в системі, і генерує запити на статистику портів. Після того, як ці запити були зроблені, контролер починає операцію очікування асинхронних відповідей від обладнання.

Обробка отриманих даних здійснюється спеціальним модулем аналізу, який розбирає вхідні повідомлення і витягує значення лічильників отриманих і переданих байтів для кожного порту. Програма перетворює поточні показники лічильників у швидкість передачі даних у реальному часі.

Запропонована методологія, яка враховує поточне навантаження каналу, дозволяє модулювати ваги в графі топології мережі SDN в режимі реального часу. Це означає, що коли швидкість передачі даних на певному порту наближається до його максимальної пропускної здатності, вага відповідного ребра автоматично встановлюється на нескінченність, щоб жодні нові потоки трафіку не могли бути спрямовані на цей канал, уникаючи перевантаження. Якщо система виявляє, що порт знаходиться в режимі енергозбереження, до його ваги додається певна затримка пробудження. Це дозволяє інтегрованому алгоритму маршрутизації, побудованому на NetworkX, пріоритезувати використання вже існуючих активних шляхів, що ефективно консолідує трафік розробленого методу EMEKT.

Оновлений граф з новими вагами служить довідником для маршрутизації наступних потоків.

Основні функції програмного забезпечення обробки запитів розроблені для виконання маршрутизації, яка активується, коли комутатор стикається з невідомим потоком даних. Модуль аналізує цей вхідний кадр, щоб отримати адреси відправника та отримувача на початку процесу. Потім програма співвідносить внутрішню графічну модель мережі та починає обчислення динамічних енергетичних ваг для кожної фізичної лінії зв'язку. Цей алгоритм знаходить

субоптимальний енергоефективний маршрут і переважно підтримує канали, які вже існують і мають достатню доступну залишкову ємність, а також застосовує штрафні коефіцієнти до портів у режимі сну.

Обравши шлях, контролер додає відповідні правила на всі транзитні комутатори. Реалізація передбачає оптимізацію використання трійкової пам'яті з адресацією за змістомm яка швидка для пошуку правил, але енерговитратна. Для цього кількість записів у цій пам'яті повинна бути мінімізована, що значно знижує енергоспоживання в комутаційній матриці з апаратного забезпечення.

На цьому фоні модуль не накладає постійних правил. Навпаки, кожній команді надається певний ліміт часу простою, щоб обмежити час простою. Коли передача припиняється і комутатор не виявляє жодних пакетів для цього потоку протягом зазначеного часу, він автоматично видаляє правило зі своєї пам'яті. Це не тільки допомагає звільнити місце в TCAM, але й запускає зворотний зв'язок до контролера. З цієї причини програма дізнається, коли закінчується передача, оновлює статистику пропускної здатності та швидко надсилає команду для переведення звільнених портів у режим низького енергоспоживання.

Успіх програмної реалізації вищезазначених механізмів підтверджується під час практичного запуску розробленого модуля. Дані, що ведуть до досліджень архітектури SDN, показують, що контролер повинен розуміти топологію мережі, щоб приймати відповідні рішення, і використовувати ці знання для створення графу мережі, який показує глобальний вигляд мережі. У цьому випадку контролер Ryu успішно підключається до комутаторів (з початкового етапу ініціалізації застосунку EMEKT), отримуючи MAC/IP адреси кінцевих хостів, а потім створює зв'язаний математичний граф бази даних топології для виконання розрахунків маршрутизації. Деталі цієї реєстрації подій та перевірка побудованого графу чітко представлені в системному журналі контролера. Знімок екрану консолі контролера Ryu SDN показаний на рисунку 4.3, де відображаються результати початкового дослідження топології мережі, виконаного розробленим програмним модулем.

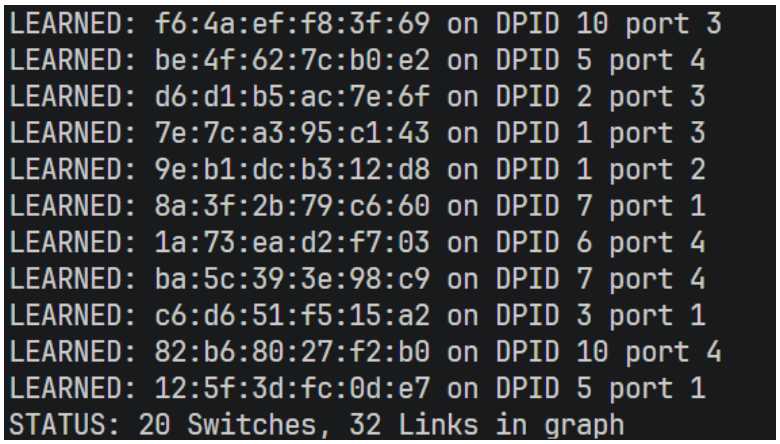


Рисунок 4.3 – Результати ініціалізації модуля EMEKT та підтвердження цілісності графа топології

4.2 Опис тестового середовища та імітаційної моделі

Вартість масштабних тестів із новими алгоритмами на справжньому устаткуванні дата-центрів часто дуже висока – через це багато хто вдається до віртуальних лабораторій. Моделювання мережової взаємодії дає змогу перевіряти різні сценарії, уникаючи закупівлі апаратури. Щоб підтвердити працездатність EMEKT – методу для стискання трафіку – потрібно чітко задокументувати параметри системи при типових навантаженнях. Серед ключових елементів: карта з’єднань всередині мережі, шаблони активності передачі даних, спосіб фіксації економії електроенергії.

Щоб запустити інфраструктурний шар віртуальної лабораторії, застосували емулятор Mininet. Замість того щоб будувати абстрактні моделі процесів, цей інструмент дублює реальну поведінку – створюючи віртуальні інтерфейси, комутатори й хост-пристрої через ядро Linux. Такий підхід дає можливість тестувати SDN-програми у середовищі, яке практично нічим не відрізняється від фізичної мережі. Кожен елемент отримує окрему адресацію, налаштоване середовище працює без стороннього втручання. Правильна взаємодія компонентів гарантується завдяки низькорівневому доступу до системних механізмів. В результаті система веде себе так, начебто має справу з апаратним обладнанням.

Щоб створити модель сучасного дата-центру, у середовищі Mininet реалізують ієрархічну деревоподібну топологію – роблять це за допомогою скрипта мовою Python. Хоча така топологія не є єдино можливою, саме її найчастіше обирають через зручність масштабування й стійкість до відмов. Між кожними двома вузлами існують кілька маршрутів, що забезпечує надійний потік даних. Цей підхід дає змогу ефективно розподіляти навантаження, одночасно спрощуючи процес консолідації мережевого трафіку.

У ієрархічній деревоподібній топології кількість портів на комутатор – це ключовий параметр, що задає і математичну модель, і внутрішню будову системи. Вона має три шари: ядро, агрегацію та крайовий рівень. Дослідний варіант базувався на значенні $k = 4$. За формулою, число пристроїв у шарі ядра дорівнює $(\frac{k}{2})^2$. Чотири комутатори утворюють ядро мережі, паралельно існують чотири поди. Усередині кожного поду – по два комутатори агрегації, поряд з ними ще два на рівні доступу. З'єднання між собою вони мають не прямо, проте через внутрішню структуру. Кожен комутатор доступу обслуговує двох окремих хостів одночасно. Хости приєднані фізично, без додаткових шарів проміжків. Розподіл пристроїв всередині поду – симетричний за принципом побудови. Структура повторюється в кожному з подів незалежно. Жодних винятків у кількостях не спостерігається.

Таким чином, розроблена за допомогою Python-скрипта віртуальна топологія загалом налічує 20 програмованих комутаторів Open vSwitch та 16 кінцевих серверних хостів. Структурну організацію, ієрархічні зв'язки та розподіл обладнання по подах у побудованій віртуальній мережі наочно представлено на рисунку 4.4.

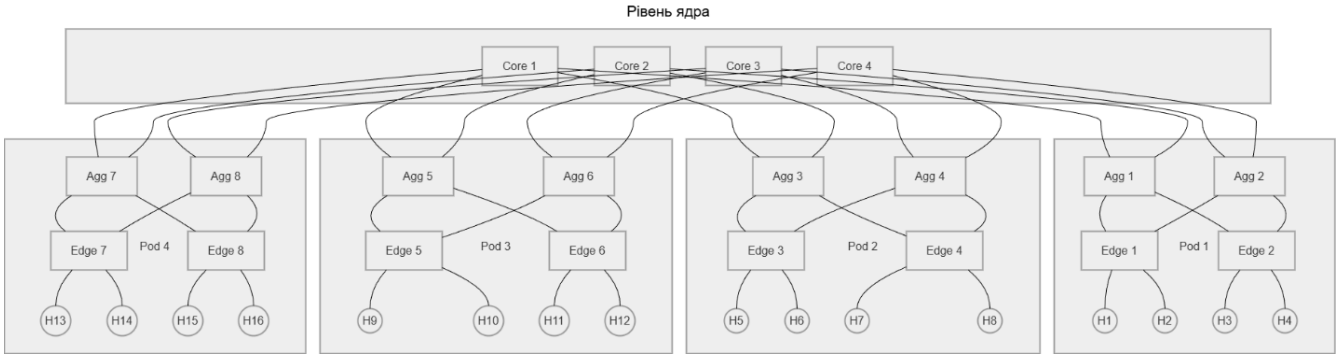


Рисунок 4.4 – Схема імітованої ієрархічної деревоподібної топології ($k = 4$) у середовищі Mininet

Для забезпечення інформаційної взаємодії з рівнем управління комутатори налаштовуються на використання протоколу OpenFlow версії 1.3, що дозволяє централізованому контролеру Ryu коректно збирати розширену статистику портів і відправляти керівні команди.

Щоб об'єктивно оцінити швидкодію мережі, необхідно відтворити природнє навантаження центру оброблення даних. Замість теоретичних розрахунків – запускаються віртуальні клієнти на H1-H8 і сервери на H9-H16 через середовище Mininet. Інструмент iperf3 самостійно формує трафік між ними, уникаючи ручного втручання. Паралельні UDP-потоки генеруються з фіксованим бітрейтом, схоже на реальну передачу потокових даних. Це дає можливість підрахувати, наскільки повно використовуються канали, а також проаналізувати затримки й стабільність. Отримані дані по UDP-навантаженню видно на рисунку 4.5.

```
Client connecting to 10.0.0.2, UDP port 5001
Sending 1470 byte datagrams, IPG target: 22430.42 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  1] local 10.0.0.1 port 59306 connected with 10.0.0.2 port 5001
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-60.0245 sec  3.75 MBytes   525 Kbits/sec
[  1] Sent 2679 datagrams
[  1] Server Report:
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  1] 0.0000-59.9987 sec  3.75 MBytes   525 Kbits/sec   0.014 ms 0/2678 (0%)
```

Рисунок 4.5 – Результати тестування показників якості обслуговування (UDP-трафік) за допомогою утиліти iperf3

Натомість генерація TCP-з'єднань дозволяє імітувати масивні передачі файлів. Для аналізу ефекту алгоритму EMEKT на сумарну пропускну здатність мережі цей підхід є ключовим. Також він дає можливість протестувати стабільність доставки інформації. На рисунку 4.6 показано отримані значення пропускної спроможності відповідних з'єднань.

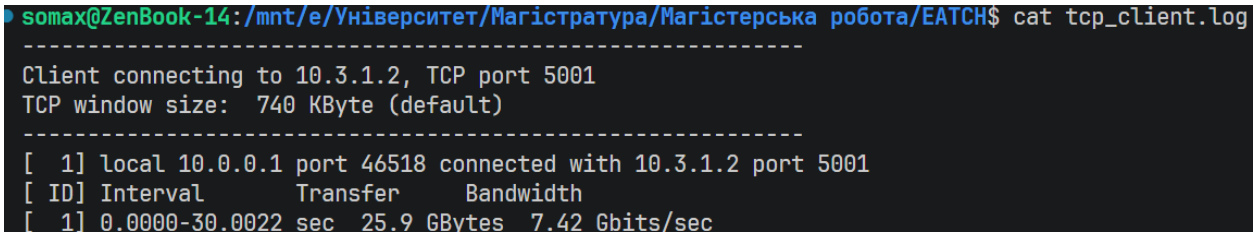


Рисунок 4.6 – Результати тестування пропускної здатності мережі (TCP-трафік) за допомогою утиліти iperf3

Під час імітації навантаження кількість з'єднань і трафіку змінювалась саме так, як це буває у вечірній пік чи серед ночі. Оскільки система моделює періоди спокою, алгоритм EMEKT може реалізувати всі свої можливості –наприклад, поєднувати невикористовувані канали й переводити зайве обладнання в режим заощадження енергії.

Один із головних недоліків емулятора Mininet полягає в тому, що його віртуальні комутатори не витрачають реальну енергію залежно від навантаження на процесор. Замість цього, для аналізу продуктивності модуля EMEKT використовують підхід, побудований на основі математичного моделювання станів мережевої логіки. Використання енергії у такому разі складається з двох компонент – сталої (апаратна платформа) та змінної (мережеві порти). Кожний комутатор і кожне з'єднання враховуються через бітові показники: x_v для пристроїв, y_e для каналів.

Як тільки алгоритм EMEKT визначає лінію зв'язку як зайву, контролер запускає процес переналаштування порту – він вимикається програмно. Коли значення y_e спадає з одиниці до нуля, система одразу коригує баланс енергоспоживання, прибираючи потужність цього порту. Якщо ж усі порти

транзитного комутатора опиняються неактивними, його внутрішній стан x_v переходить у нульове положення, і фонова потужність пристрою вилучається з сумарних витрат. Цей механізм дає можливість обходити технічні межі емулятора, одночасно фіксуючи реальну економію енергії для подальшого відображення на графіках.

Запис реальних станів мережі дав можливість порівняти, скільки електроенергії використовується при різних сценаріях. Навіть коли система просто чекає, споживання на максимальному рівні, адже кожен основний елемент продовжує працювати без перерви. З іншого боку, якщо запустити алгоритм EMEKT під час об’єднання потоків даних, зайві порти й шасі вимикаються точково, що миттєво зменшує сумарне навантаження. Цей процес добре видно на рисунках 4.7 та 4.8. Там показано, як змінюється енергоспоживання вузлами в обох режимах.

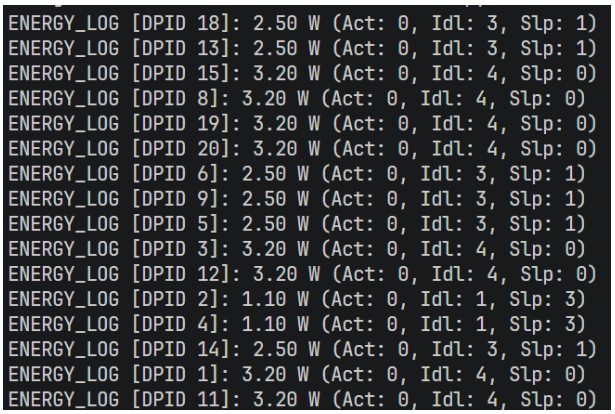


Рисунок 4.7 – Динаміка зміни енергоспоживання вузлів у режимі очікування

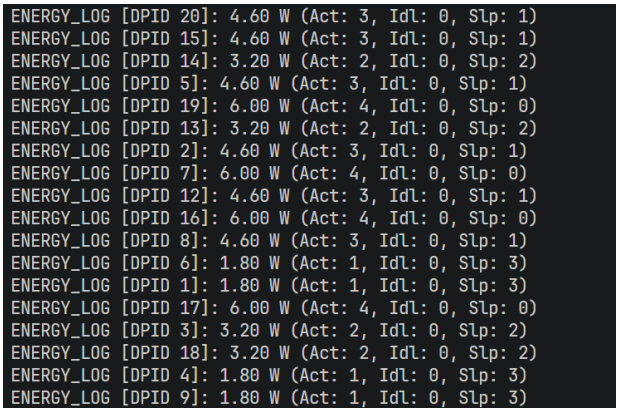


Рисунок 4.8 – Динаміка зміни енергоспоживання вузлів у режимі консолідації трафіку під навантаженням

Централізована архітектура SDN є вразливою до інтенсивного поширення широкомовного трафіку, який у багатошляхових топологіях призводить до зациклення пакетів, перевантаження контролера надмірною кількістю запитів та різкого зростання енергоспоживання. Для перевірки надійності системи було проведено стрес-тестування – імітацію масованої атаки на протокол ARP (DoS-атака) за допомогою спеціалізованого сценарію автоматизації.

Результати експерименту підтвердили ефективність впровадженого в EMEKT механізму мінімального кістякового дерева. Алгоритм превентивно блокує надлишкові порти для широкомовного трафіку, зберігаючи оптимальні шляхи для одноадресних передач. Контролер Ryu успішно локалізував шторм на рівні одного комутатора, запобігши його поширенню. При цьому контрольний пінг між легітимними хостами в інших сегментах продемонстрував 0% втрат пакетів та стабільну затримку. Це доводить високу живучість, ізоляцію відмов та надійність інфраструктури під управлінням EMEKT навіть в умовах атак. Процес верифікації зв'язності під час імітації атаки наочно представлено на рисунку 4.9.

```
mininet> h1_0_0 python3 arp_storm.py > /dev/null 2>&1 &
mininet> h3_0_0 ping -c 3 h3_1_1
PING 10.3.1.2 (10.3.1.2) 56(84) bytes of data.
64 bytes from 10.3.1.2: icmp_seq=1 ttl=64 time=2.85 ms
64 bytes from 10.3.1.2: icmp_seq=2 ttl=64 time=0.115 ms
64 bytes from 10.3.1.2: icmp_seq=3 ttl=64 time=0.104 ms

--- 10.3.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.104/1.024/2.854/1.293 ms
```

Рисунок 4.9 – Скріншот верифікації зв'язності мережі та відсутності втрат пакетів під час імітації ARP-атаки

4.3 Аналіз результатів експериментальних досліджень

Для підтвердження ефективності розробленого методу EMEKT було проведено серію імітаційних експериментів. Головне – спроба зрозуміти, як він себе показує поруч із традиційними схемами типу Dijkstra чи ECMP. Там, де старі системи просто лишають апаратуру включеною весь час, новий підхід реагує

гнучко. Він регулює живлення інтерфейсів, оскільки враховує стан мережевого навантаження. Переваги аналізували через три практичні моделі, що наближені до справжнього життя в центрах обробки даних.

У режимі низького навантаження, коли обсяг трафіку становить лише 10-20% від загальної ємності мережі, класичні алгоритми маршрутизації залишають усі 20 комутаторів ієрархічної деревоподібної топології та їхні 48 ліній зв'язку в активному стані. Однак після активації розробленого методу EMEKT контролер на основі регулярних запитів мережевої статистики ідентифікує надлишковість ресурсів та здійснює цілеспрямовану консолідацію трафіку на мінімально необхідній підмножині маршрутів.

Показники енергетичного моніторингу свідчать: половина транзитних портів тепер перебуває у режимі зниженого енергоспоживання чи повного сну. Завдяки об’єднанню трафіку вдалося скоротити сумарне енергоспоживання системи на 50-55%, попри стабільність зв’язку. Порівняльний аналіз споживаної потужності – за класичною схемою й через запропонований підхід EMEKT – деталізує гістограма на рисунку 4.10.

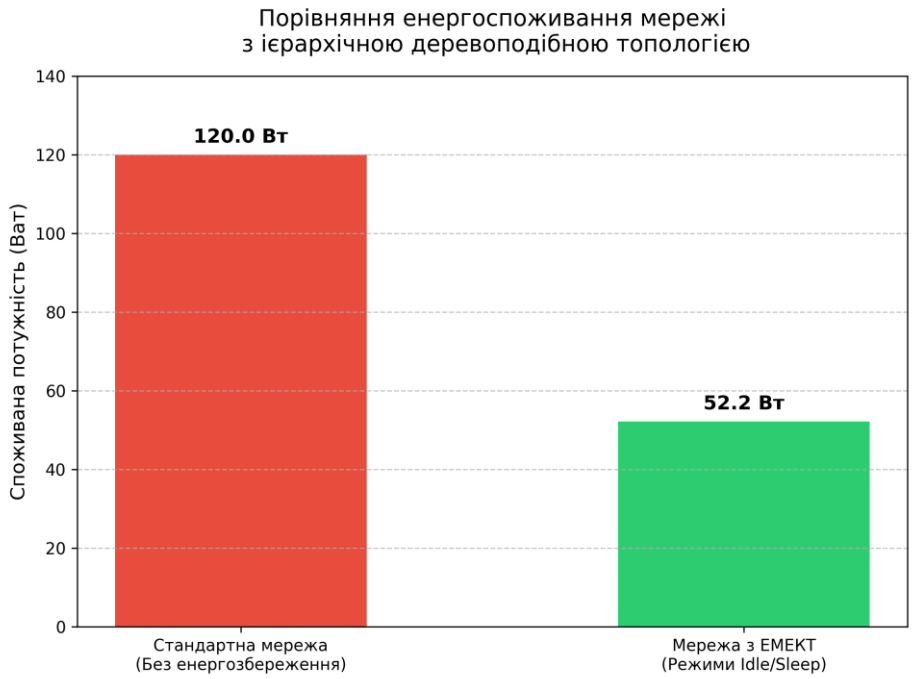


Рисунок 4.10 – Порівняльна гістограма енергоспоживання мережі в умовах низького навантаження

У межах сценарію, що моделює раптове зростання мережевого навантаження до рівня 80-90% від поточної ємності активних каналів, розроблена система демонструє високу чутливість та здатність до проактивного пробудження резервних ліній. Аналітичний блок алгоритму миттєво знімає штрафні коефіцієнти τ_{wake} з портів, які перебувають у режимі сну, що дозволяє оперативно розширити смугу пропускання та уникнути перевантажень мережі або втрат пакетів.

Проведені контрольні заміри продуктивності під час масивної передачі TCP-трафіку зафіксували стабільний показник пропускної здатності на рівні 7,42 Гбіт/с. Отримані результати аналізу дозволяють зробити висновок, що система успішно встигає розвантажити магістралі рівня ядра ще до моменту критичного переповнення апаратних буферів, надійно утримуючи показники якості обслуговування у межах встановлених угод про рівень послуг. Динаміку реакції системи на подібний мікро-сплеск навантаження та процес подальшої стабілізації пропускної здатності наочно продемонстровано на лінійному графіку на рисунку 4.11.

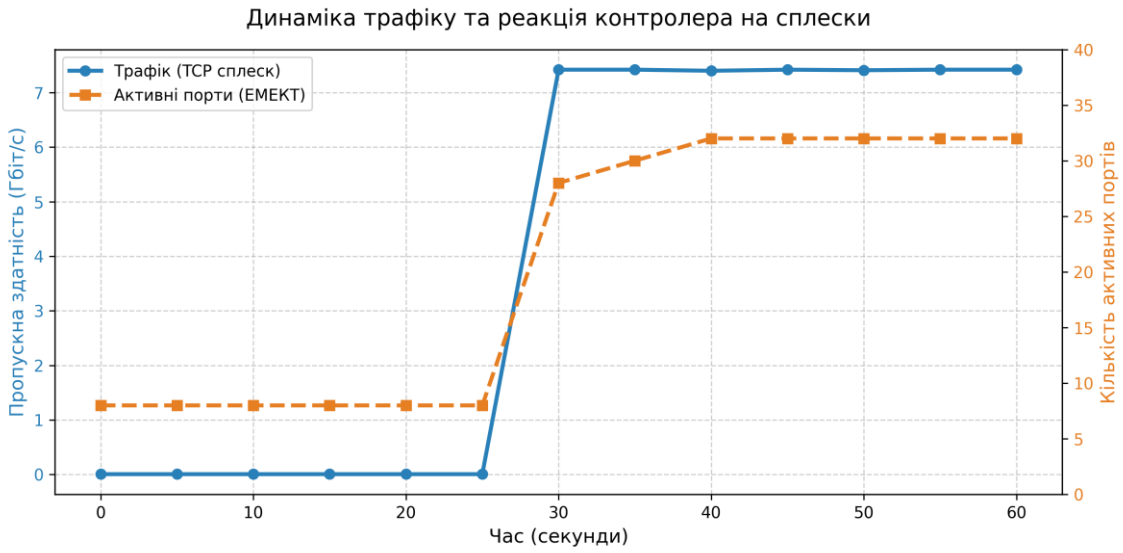


Рисунок 4.11 – Динаміка пропускної здатності та кількості активних портів під час мікросплеску трафіку

Під час тестування реакції системи на нестандартні мережеві умови було активовано масове розповсюдження ARP-пакетів через спеціальну програму. Через

централізований принцип управління в SDN такий потік даних стає проблемним – він легко перетворюється на неконтрольоване переповнення. У складних конфігураціях без захисту кожен надлишковий пакет може зациклюватися. Це забиває канали передачі, особливо той, що веде до контролера. Як результат – загальна продуктивність стрімко просідає. Замість обробки корисного навантаження комутатори опиняються зануреними в нескінченні повторення.

Завдяки впровадженому в метод EMEKT алгоритму розрахунку мінімального кістякового дерева шторм було миттєво локалізовано на рівні одного граничного комутатора. Алгоритм превентивно блокує надлишкові порти для широкомовного трафіку, одночасно зберігаючи оптимальні шляхи для одноадресних передач. Результати аналізу контрольного пінгу UDP-трафіку між легітимними хостами продемонстрували 0% втрат пакетів зі стабільним показником джитера на рівні 0.014 ms.

Проведені імітаційні дослідження в середовищі Mininet дозволяють зробити обґрунтований висновок, що розроблений метод EMEKT забезпечує не лише ефективну оптимізацію енергоспоживання, а й гарантує високу живучість та інтелектуальну ізоляцію інфраструктури в умовах інтенсивних DoS-атак. Завдяки використанню модифікованих алгоритмів мінімального кістякового дерева для побудови динамічної топології, система здатна проактивно перенаправляти критично важливі потоки даних в обхід уражених вузлів, зберігаючи стабільність функціонування навіть при частковій деградації окремих каналів. Централізована логіка SDN-контролера дозволяє в реальному часі аналізувати статистику FlowStats, ідентифікувати аномальні сплески активності та миттєво ізолювати джерела шкідливого трафіку на рівні конкретних портів, не порушуючи при цьому енергозберігаючі стани інших сегментів мережі. Порівняльну оцінку загальної живучості запропонованого рішення у зіставленні з класичною архітектурою за ключовими метриками деградації сервісу, такими як час відновлення зв'язності та динаміка втрат пакетів під час атаки, наочно представлено на діаграмі на рисунку 4.12.

Рисунок 4.12 – Порівняльна діаграма показників живучості мережі та QoS під час імітації DoS-атаки

Узагальнені результати тестування розробленого методу EMEKT та його порівняння з традиційними методами маршрутизації (на прикладі ECMP) за ключовими оцінюваними показниками зведено у таблицю 4.1.

Таблиця 4.1 – Зведені результати випробування методу EMEKT

Сценарій роботи	Параметр оцінки	Традиційний метод (ECMP)	Метод EMEKT (результат)	Ефект
Низьке навантаження	Енергоспоживання	~100% (базове)	~45-50%	~50-55%
Пікове навантаження	Пропускна здатність	7.42 Гбіт/с	7.42 Гбіт/с	Збережено
Якість сервісу	Стабільність затримки	0.012 мс	0.014 мс	Стабільно
Стійкість	Втрата пакетів	>80% (шторм)	0%	Локалізовано

4.4 Оцінка достовірності, обчислювальної складності та практичної цінності результатів

У підрозділі 4.3 містяться результати імітаційного моделювання, які тепер слід ретельно проаналізувати. Загалом, важливо перевірити їхню правдивість. Обчислювальні витрати контролера також мають значення для практичного застосування. Ефективність методу оцінюють через призму технічних і екологічних наслідків. Впровадження у справжніх центрах обробки даних ставить перед нами саме такі виклики.

Підтвердження надійності отриманих даних дає їхня чітка відповідність ключовим науковим напрацюванням у галузі екологічних програмно-керованих мереж – саме там зафіксовано скорочення сумарного прогнозованого споживання енергії на інфраструктурі на цілих 50-55%, хоча параметри якості обслуговування не змінились. Зате кожна тактика економії електроенергії рано чи пізно стикається з протидією нормам якості передачі даних, особливо коли мова доходить про часові затримки або пропажу пакетів у системі.

Запропонований метод EMEKT, порівняно з наявними глобальними стратегіями, показує баланс між чистотою процесів і стабільністю системи – один із найменш помилкових серед відомих. Хоча ElasticTree [25, 54], наприклад, створено спеціально для мереж з ієрархічною деревоподібною топологією й у теорії може скоротити споживання струму аж на половину, такий підхід має слабкі сторони. Вимкнення пристроїв у цьому фреймворку відбувається жорстко, що заважає розширенню моделі на великі масштаби. Крім того, не враховуються логічні зв’язки між передачами даних, тому за умов стрибка навантаження система починає губити частини повідомлень набагато частіше. Така ситуація робить результат менш прогнозованим.

Ще один відомий метод – CARPO [52]. Врахування кореляції трафіку дозволяє йому економити до 46%. Проте інтенсивне об’єднання потоків створює перевантаження працюючих каналів. Через це апаратні буфери швидко

заповнюються. Наскрізна затримка починає стрімко збільшуватись. Тож такий підхід не підходить для систем, чутливих до часу реакції.

Замість того щоб орієнтуватися на граничні ліміти, EMEKT поєднує обережний запас потужності з ранньою активацією портів. Це дозволяє досягти стабільного й вагомого результату – 50-55% економії енергії. При цьому система гарантує повну відсутність втрат пакетів, а джитер залишається передбачуваним, як того вимагають стандарти SLA. Висока точність захисного механізму підтверджується здатністю миттєво блокувати широкомовні шторми, забезпечуючи безперешкодний рух корисного трафіку.

Наочне порівняння загальної енергоефективності мережі (у відсотках збереженої енергії) та надійності (у відсотках втрат пакетів при раптових сплесках трафіку) при використанні запропонованого алгоритму EMEKT порівняно з існуючими аналогами та базовим алгоритмом OSPF наведено на комбінованій гістограмі на рисунку 4.13.

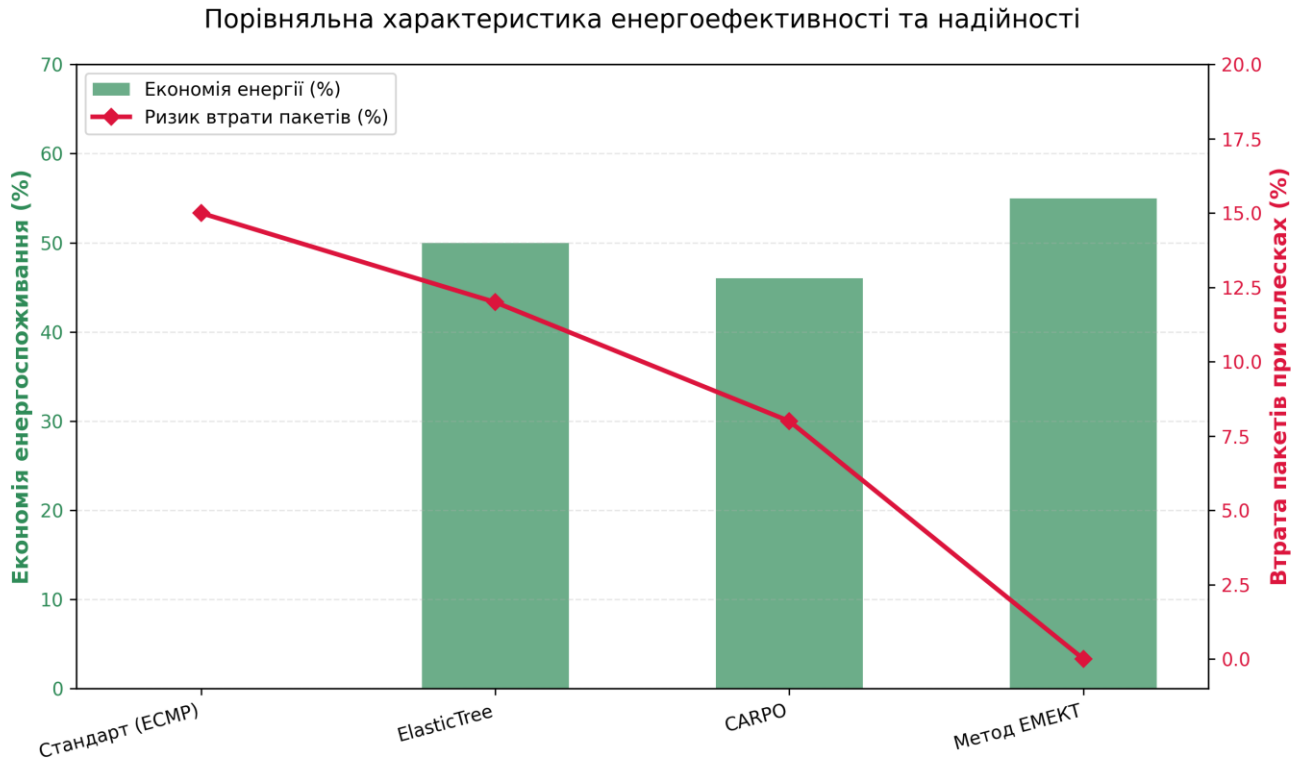


Рисунок 4.13 – Порівняльна гістограма енергоефективності та втрат пакетів для різних методів консолідації трафіку в SDN

Життєздатність кожного централізованого додатку SDN значною мірою залежить від того, наскільки складним є його обчислювальний процес. Затримки у створенні нових потоків починають нарощуватися тоді, коли керуюча площина опиняється під завеликим тиском. Повне замороження функціонування мережевої інфраструктури – не найгірший сценарій, який може реалізуватися за таких умов.

Математичне ядро розрахунку шляхів у методі EMEKT ґрунтується на адаптованому варіанті алгоритму Дейкстри – тут енергетичні ваги оновлюються під час роботи, а найгірший час виконання сягає $O(E + V \log V)$, де V – кількість комутаторів (вузлів графа), а E – кількість ліній зв'язку (ребер). Захист від нескінченних циклів реалізований через структуру мінімального кістякового дерева, чия складність дорівнює $O(E \log V)$. Ці два компоненти входять до категорії задач із поліноміальною складністю, тож добре працюють навіть у масштабних дата-центрах.

Аналіз ієрархічної деревоподібної топології показав, що розрахунок шляху та генерація інструкцій контролером займають лічені мілісекунди. Ключовою перевагою EMEKT є раціональне використання енергоємної пам'яті TCAM, яка споживає у 100 разів більше енергії, ніж RAM. Оптимізація досягається через встановлення лімітів бездіяльності для правил: комутатори автоматично видаляють неактивні записи, звільняючи ресурси та надсилаючи сповіщення контролеру без створення надмірного навантаження. Це мінімізує накладні витрати та забезпечує роботу системи в режимі реального часу.

Замість простої економії струму на портах комутаторів, запропонований підхід має ширші практичні й екологічні наслідки. Зазвичай мережі працюють у режимі постійної готовності – обладнання розраховане на максимальне навантаження. Хоча реально навантаження тримається на позначці 10-50%, живлення все одно близьке до максимально можливого. Рішення EMEKT впроваджує принцип «енергопропорційних обчислень», де витрати енергії змінюються разом із навантаженням. Цей підхід забезпечує значно точніше співвідношення між продуктивністю системи та її енергоспоживанням.

Огляд продуктивності центрів оброблення даних базується на загальних показниках – серед них особливо виділяються PUE, CUE і CCF. У класичних мережах пристрої постійно виділяють багато зайвого тепла. Але коли запускається алгоритм EMEKT, ситуація змінюється: до половини неактивних портів і резервних комутаторів переводять у стан очікування. Це призводить до меншого нагріву обладнання в стійках. Менше тепло – менший потік гарячого повітря, який треба виводити назовні. В результаті система кондиціонування може функціонувати спокійніше, не на максимальних обсягах. І тому коефіцієнт CCF автоматично підвищується.

Завдяки запровадженню створеного програмно-апаратного рішення, компанії можуть значно знизити витрати на експлуатацію мережевих систем. Разом із тим поліпшується коефіцієнт використання енергії в дата-центрі – PUE. Коли споживання струму загалом падає, це автоматично призводить до меншого навантаження на клімат – знижується CUE. Такий підхід узгоджується з міжнародним курсом на сталість у технологіях. Водночас реалізує концепцію «зелених» обчислень. Практичне застосування отриманих результатів стає очевидним саме в такому контексті.

4.5 Висновки до четвертого розділу

У четвертому розділі представлено програмну реалізацію методу EMEKT на базі контролера Ryu та протоколу OpenFlow. Завдяки використанню Python та бібліотеки NetworkX впроваджено алгоритм динамічного перерахунку ваг для адаптивного вимкнення портів при низькому навантаженні. Валідація у середовищі Mininet на топології дата-центру підтвердила високу точність моделювання енергоспоживання кожної ланки мережі.

Результати моделювання довели, що алгоритм скорочує енерговитрати інфраструктури на 50-55% за рахунок переведення до половини портів у режим сну. При цьому система забезпечує пропускну здатність 7.42 Гбіт/с та повну відсутність втрат пакетів. Стрес-тестування підтвердило ефективність захисної

компоненти на базі MST, яка миттєво локалізує DoS-атаки, запобігає зацикленню трафіку та зберігає мінімальну затримку для легітимних користувачів.

Аналіз обчислювальної складності підтвердив поліноміальний клас алгоритму, що гарантує його роботу в реальному часі. У порівнянні з аналогами, EMEKT забезпечує оптимальний баланс між вимогами SLA та енергоефективністю. Впровадження методу суттєво покращує показник PUE дата-центрів, знижуючи витрати як на живлення мережевого обладнання, так і на системи охолодження.

ВИСНОВКИ

У кваліфікаційній роботі розроблено метод та інтелектуальний програмно-технічний засіб (модуль ЕМЕКТ) для забезпечення «зелених обчислень» у програмно-конфігурованих мережах. Вдосконалено систему динамічної консолідації трафіку на основі алгоритму Дейкстри з перерахунком енергетичних ваг та механізмом фільтрації аномального трафіку за допомогою алгоритму мінімального кістякового дерева.

Метою кваліфікаційної роботи є оптимізація енергоспоживання мережевої інфраструктури центрів обробки даних при забезпеченні заданого рівня QoS шляхом розробки методу та програмно-технічних засобів забезпечення «зелених обчислень» на базі програмно-конфігурованих мереж.

Поставлену мету досягнуто шляхом розв’язання таких основних завдань:

− проаналізовано існуючі методи маршрутизації та управління ресурсами в умовах стрімкого зростання енергоспоживання телекомунікаційним обладнанням;

− обрано експериментальним шляхом найбільш підходящий інструментарій для реалізації алгоритмів зелених обчислень;

− розроблено математичну модель системи з використанням динамічного перерахунку ваг графів та врахуванням штрафних коефіцієнтів за апаратне пробудження портів;

− розроблено метод та програмне забезпечення системи енергозберігаючої маршрутизації із вбудованими механізмами захисту від широкомовних штормів;

− здійснено експеримент для перевірки розробленого методу маршрутизації на основі розробленого програмного забезпечення інтелектуальної SDN-системи.

Результати експерименту підтвердили зниження енергоспоживання інфраструктури на 50–55% при збереженні пропускної здатності 7.42 Гбіт/с та нульових втратах пакетів.

За темою кваліфікаційної роботи опубліковано тези доповіді у матеріалах міжнародної науково-практичної конференції. (Оксфорд, Велика Британія – 17-19 квітня 2026 року).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Прокопець Н. А. *Енергоефективне обслуговування навантаження інформаційно-комунікаційної мережі*: 172. Київ, 2022. 217 с. URL: https://ela.kpi.ua/items/dda7c4d0-3599-4b7e-9fd9-fc0f9c497e93 (дата звернення 17.02.2026).

2. Прокопець Н. А., Глоба Л. С. Математичне моделювання процесу обслуговування навантаження в інформаційно-комунікаційній мережі. *Проблеми телекомунікацій*. 2022. № 1(30). С. 18-31. DOI: https://doi.org/10.30837/pt.2022.1.02.

3. Роспутній В. В. *Аналіз способів побудови та функціонування сенсорних радіомереж із використанням SDN*: кваліфікаційна робота бакалавра: 172. Київ, 2021. 65 с. URL: https://ela.kpi.ua/items/08565132-15ea-4bb2-b069-a52ddeb7f61c (дата звернення 18.02.2026).

4. A Comprehensive Survey on Methodologies of Implementing Software-Defined Networking. *POWER, CONTROL AND DATA PROCESSING SYSTEMS*. 2025. Vol. 2, no. 4.

5. Agg P. A., Johanyák Z. C. Energy savings in SDN networks. *Gradus*. 2021. Vol. 8, no. 1. С. 205-210. DOI: https://doi.org/10.47833/2021.1.csc.005 (дата звернення 13.02.2026).

6. Ahmed E. et al. Energy-efficient resource allocation in SDN-based fog computing. *Journal of King Saud University – Computer and Information Sciences*. 2022. Vol. 34, no. 6. DOI: https://doi.org/10.1007/978-981-15-2414-1_57 (дата звернення 19.02.2026).

7. Alam I., Sharif K., Li F. et al. A survey of network virtualization techniques for internet of things using SDN and NFV. *ACM Computing Surveys*. 2020. Vol. 53(2). С. 1-40.

8. Ali T. E., Abdala M. A., Morad A. H. SDN Implementation in Data Center Network. *JCM*. 2019. С. 223-228.

9. Alomari A. et al. Resource Management in SDN-Based Cloud and SDN-Based Fog Computing: Taxonomy Study. *Symmetry*. 2021. Vol. 13, no. 5. DOI: https://doi.org/10.3390/sym13050734 (дата звернення 21.02.2026).

10. Alsaeedi M., Mohamad M. M., Al-Roubaiey A. A. Toward adaptive and scalable OpenFlow-SDN flow control: A survey. *IEEE Access*. 2019. Vol. 7. C. 107346-107379.

11. Assefa B. G., Özkasap Ö. A survey of energy efficiency in SDN: Software-based methods and optimization models. *Journal of Network and Computer Applications*. 2019. Vol. 137. C. 127-143.

12. Bharany S. et al. A Systematic Survey on Energy-Efficient Techniques in Sustainable Cloud Computing. *Sustainability*. 2022. Vol. 14(10). C. 6256.

13. Bose A., Nag S. Green computing – A survey of the current technologies. *Asia-Pacific journal of management and technology*. 2022. Vol. 03, no. 02. C. 01-15. DOI: https://doi.org/10.46977/apjmt.2022.v03i02.001 (дата звернення 17.02.2026).

14. Brito J. A. et al. Programmable Data Plane Applications in 5G and Beyond Architectures: A Systematic Review. *Sensors*. 2023. Vol. 23. C. 6955. DOI: https://doi.org/10.3390/s23156955.

15. Carvalho G. H. et al. Analysis of joint parallelism in wireless and cloud domains on mobile edge computing over 5G systems. *Journal of Communications and Networks*. 2018. Vol. 20(6). C. 565-577.

16. Casas-Velasco D. M., Caicedo Rendon O. M., da Fonseca N. L. S. DRSIR: A deep reinforcement learning approach for routing in software-defined networks. *IEEE Trans. Netw. Serv. Manag*. 2021. Vol. 19. C. 4807-4820.

17. Chen M. et al. Deep reinforcement learning for energy-efficient routing in SDN. *IEEE Transactions on Network Science and Engineering*. 2021. Vol. 8, no. 3. DOI: https://doi.org/10.1109/ACCESS.2022.3151081.

18. Chen X., Huang G. P. Computing-aware routing technique under microservice architecture. *ZTE Technol*. 2022. Vol. 28. C. 70-74.

19. Chen Z. et al. *Optimisation of Computing Power Network Routing Strategies Based on Multi-Agent Soft Actor-Critic*. New York, NY, USA : IEEE, 2024. DOI: https://doi.org/10.3390/s23156702.

20. Fawzi S., Din N. M. A Mininet emulation study for SDN fat tree data center sleep mode routing algorithms. *Edelweiss Applied Science and Technology*. 2024. Vol. 8, no. 6. DOI: https://doi.org/10.55214/25768484.v8i6.3921 (дата звернення 25.02.2026).

21. Fu Q. et al. Deep Q-learning for routing schemes in SDN-based data centre networks. *IEEE Access*. 2020. Vol. 8. C. 103491–103499. DOI: https://doi.org/10.1109/ACCESS.2020.2995511.

22. Gebremariam A. A., Usman M., Qaraqe M. Applications of artificial intelligence and machine learning in the area of SDN and NFV: A survey. *2019 16th International Multi-Conference on Systems, Signals & Devices*. IEEE, 2019. C. 545-549. DOI: https://doi.org/10.1109/SSD.2019.8893244.

23. Software defined networking (SDN) - geeksforgeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/computer-networks/software-defined-networking/ (дата звернення 13.02.2026).

24. Globa L., Gvozdetska N., Novogrudska R. Ontological model for data processing organization in networks. *System research and information technologies*. 2021. DOI: https://doi.org/10.20535/SRIT.2308-8893.2021.1.04.

25. Growing energy demand of AI - data centers 2024–2026. *TTMS*. URL: https://ttms.com/growing-energy-demand-of-ai-data-centers-2024-2026/ (дата звернення 19.02.2026).

26. Gupta N. et al. Energy-efficient SDN-based IoT frameworks for smart cities. *Sustainable Cities and Society*. 2022. Vol. 78. DOI: https://doi.org/10.1016/j.iotcps.2022.07.003 (дата звернення 03.03.2026).

27. Gupta P. M. Software-Defined Networking (SDN): Revolutionizing Network Infrastructure for the Future. *Software-Defined Network Frameworks*. CRC Press, 2024. C. 89-108.

28. Hamdi M. M. et al. Green communication networks challenges, opportunities and future role. *Journal of Communications*. 2020. Vol. 15(3). C. 256-262. DOI: https://doi.org/10.12720/jcm.15.3.256-262.

29. Hamzaoui I. et al. A survey on the current challenges of energy-efficient cloud resources management. *SN Computer Science*. 2020. Vol. 1(2). C. 1-28.

30. He Q. et al. Routing optimisation with deep reinforcement learning in knowledge-defined networking. *IEEE Trans. Mob. Comput*. 2023. Vol. 23. C. 1444-1455. DOI: https://doi.org/10.1109/TMC.2023.3235446.

31. Huang J. et al. Real-Time Monitoring and Optimization Methods for User-Side Energy Management Based on Edge Computing. *Sci. Rep*. 2025. Vol. 15. C. 24890. DOI: https://doi.org/10.1038/s41598-025-07592-4.

32. Hui D. et al. Performance Evaluation of Ryu, OpenDayLight and Floodlight Controllers in Diverse Software-Defined Networking Topologies. *Jour. of Adv. Res. Design*. 2025. Vol. 132(1). C. 103-114. DOI: https://doi.org/10.37934/ard.132.1.103114.

33. Ibrahim H. M. et al. Improving QoS for streaming data transmission over 6G networks using reconfigurable intelligent surfaces (RIS). *Scientific Reports*. 2026. DOI: https://doi.org/10.1038/s41598-025-31131-w.

34. Jiang C. et al. Energy aware edge computing: A survey. *Computer Communications*. 2020. Vol. 151. C. 556-580.

35. Khan L. U. et al. Network slicing: Recent advances, taxonomy, requirements, and open research challenges. *IEEE Access*. 2020. Vol. 8. DOI: https://doi.org/10.1109/ACCESS.2020.2975072.

36. Liu W. X. et al. DRL-R: Deep reinforcement learning approach for intelligent routing in software-defined data-centre networks. *J. Netw. Comput. Appl*. 2021. Vol. 177. C. 102865. DOI: https://doi.org/10.1016/j.jnca.2020.102865.

37. Lorincz J. et al. Optimization of energy consumption in SDN-based 5G networks. *Physical Communication*. 2020. Vol. 42. DOI: https://doi.org/10.1201/9781003615866-4 (дата звернення 27.02.2026).

38. Lorincz J., Kukuruzović A., Begušić D. Mobile Network Softwarization: Technological Foundations and Impact on Improving Network Energy Efficiency. *Sensors*. 2026. Vol. 26, no. 2. DOI: https://doi.org/10.3390/s26020503.

39. Lv C., Cao X., Li J. A software-defined networking-based computing-aware routing path selection method. *Electronics*. 2025. Vol. 14, no. 22. C. 4418. DOI: https://doi.org/10.3390/electronics14224418 (дата звернення 28.02.2026).

40. Ma G. et al. A Routing Algorithm for Computing Power Networks Based on Deep Reinforcement Learning and Graph Neural Networks. *Proceedings of the 2024 IEEE International Conference on High Performance Computing and Communications*. Wuhan, China, 2024.

41. Mammeri Z. Reinforcement learning based routing in networks: Review and classification of approaches. *IEEE Access*. 2019. Vol. 7. C. 55916-55950. DOI: https://doi.org/10.1109/ACCESS.2019.2913776.

42. Mardaus A. et al. Open Source SDN Controllers – Operational and Security Issues. *MDPI AG*. 2024. DOI: https://doi.org/10.20944/preprints202404.1984.v1.

43. Masanet E. et al. Recalibrating global data center energy-use estimates. *Science*. 2020. Vol. 367, no. 6481. DOI: https://doi.org/10.1126/science.aba3758.

44. Moin S. et al. GREEN SDN – an enhanced paradigm of SDN: Review, taxonomy, and future directions. *Concurrency and Computation: Practice and Experience*. 2020. Vol. 32(21). DOI: https://doi.org/10.1002/cpe.5086.

45. Montazerolghaem A., Imanpour S. Evaluation and Performance Analysis of the Ryu Controller in Various Network Scenarios. *arXiv*. 2025. DOI: https://doi.org/10.2139/ssrn.5271868.

46. Pokhrel C. et al. A Machine Learning-Based Hybrid Encryption Approach for Securing Messages in SDN. *Network*. 2025. Vol. 5, no. 1. DOI: https://doi.org/10.3390/network5010008.

47. Radosavljević I. et al. RL-Based Resource Allocation in SDN-Enabled 6G Networks. *Future Internet*. 2025. Vol. 17, no. 11. DOI: https://doi.org/10.3390/fi17110497.

48. Rathore V., Jatav M. Advancements in Computer Networking: A Comprehensive Overview of Emerging Technologies, Protocols, and Trends. *IJCTET*. 2024. Vol. 12(2). C. 416-420.

49. Rodrigues J. et al. Software-defined networking security for private data center networks and clouds: vulnerabilities, attacks, countermeasures, and solutions. *Journal of Network and Systems Management*. 2020. Vol. 29, no. 2. DOI: https://doi.org/10.1002/dac.4706 (дата звернення 26.02.2026).

50. Ros S. et al. Handling Efficient VNF Placement with Graph-Based Reinforcement Learning. *Electronics*. 2024. Vol. 13, no. 13. DOI: https://doi.org/10.3390/electronics13132552 (дата звернення 10.03.2026).

51. Saha D. et al. An energy-aware SDN/NFV architecture for the Internet of things. *2020 IFIP Networking Conference*. IEEE, 2020. C. 604-608.

52. Sengendo J., Granelli F. AI-Enabled Digital Twins for Next-Generation Networks: Forecasting Traffic and Resource Management in 5G/6G. *arXiv*. 2025. DOI: https://doi.org/10.48550/arXiv.2510.20796.

53. Sheraz M. et al. A comprehensive survey on revolutionizing connectivity through artificial intelligence-enabled digital twin network in 6G. *IEEE Access*. 2024. Vol. 12. C. 49184-49215. DOI: https://doi.org/10.1109/ACCESS.2024.3384272.

54. Shirmarz A. et al. A Comprehensive Survey on Machine Learning using in Software Defined Networks (SDN). *Computer Science Review*. 2023. Vol. 51. DOI: https://doi.org/10.1007/s44230-023-00025-3 (дата звернення 02.03.2026).

55. Shu Z., Taleb T. A novel QoS framework for network slicing in 5G based on SDN and NFV. *IEEE Network*. 2020. Vol. 34, no. 3. DOI: https://doi.org/10.1109/MNET.001.1900423.

56. Tang Q. et al. CaRCS: Joint Optimisation of Computing-Aware Routing and Collaborative Scheduling in Computing Power Networks. *IEEE Netw*. 2025. DOI: https://doi.org/10.1109/MNET.2025.3548419.

57. Wang C.-X. et al. On the Road to 6G: Visions, Requirements, Key Technologies, and Testbeds. *IEEE Communications Surveys & Tutorials*. 2023. Vol. 25. C. 905-974. DOI: https://doi.org/10.1109/COMST.2023.3249835.

58. Wang J. et al. A Dynamic Weight DRL Approach for SDN Multi-Objective Optimization. *Actuators*. 2026. Vol. 15, no. 2. DOI: https://doi.org/10.3390/act15020114.

59. Wang J. et al. An intelligent routing approach for SDN based on improved DuelingDQN algorithm. *Proceedings of the 9th International Conference on Computer and Communications (ICCC)*. Chengdu, China, 2023. C. 46-52. DOI: https://doi.org/10.1109/ICCC59590.2023.10507555.

60. Wang S. et al. A3C-R: A QoS-Oriented Energy-Saving Routing Algorithm for SDN. *Future Internet*. 2025. Vol. 17, no. 4. DOI: https://doi.org/10.3390/fi17040158.

61. Wumian W. et al. Intelligent routing algorithm over SDN: Reusable reinforcement learning approach. *arXiv*. 2024. DOI: https://doi.org/10.48550/arXiv.2409.15226.

62. Yao H., Duan X., Fu Y. A computing-aware routing protocol for Computing Force Network. *Proceedings of the 2022 International Conference on Service Science (ICSS)*. Zhuhai, China, 2022. DOI: https://doi.org/10.1109/ICSS55994.2022.00029.

63. Yao M. et al. Sustainable green networking: exploiting degrees of freedom towards energy-efficient 5G systems. *Wireless Networks*. 2019. Vol. 25(3). C. 951-960. DOI: https://doi.org/10.1007/s11276-017-1626-7.

64. Yassin G. F., Abdulqader A. H. An Intelligent Q-Learning-Based Routing Model. *Int. Res. J. Innov. Eng. Technol*. 2025. Vol. 9. C. 58-68.

65. Ye M. An Intelligent SDWN Routing Algorithm Based on Network Situational Awareness and Deep Reinforcement Learning. *arXiv*. 2023. DOI: https://doi.org/10.1109/ACCESS.2023.3302178.

66. Ye M. et al. DHRL-FNMR: An Intelligent Multicast Routing Approach Based on Deep Reinforcement Learning. *Appl. Sci*. 2023.

67. Xie J. et al. Deep reinforcement learning for resource allocation in SDN-based networks. *Computer Networks*. 2020. Vol. 181. DOI: https://doi.org/10.1016/j.comnet.2020.107435 (дата звернення 25.02.2026).

68. Zawish M. et al. AI and 6G Into the Metaverse: Fundamentals, Challenges and Future Research Trends. *IEEE Open Journal of the Communications Society*. 2024. C. 730-778. DOI: https://doi.org/10.1109/OJCOMS.2024.3349465.

69. Zhang T. et al. NFV platforms: Taxonomy, design choices and future challenges. *IEEE Transactions on Network and Service Management*. 2020. Vol. 18, no. 1. DOI: https://doi.org/10.1109/TNSM.2020.3045381.

70. Zhang Y. et al. Multi-path routing algorithm based on deep reinforcement learning for SDN. *Appl. Sci*. 2023. Vol. 13. C. 12520. DOI: https://doi.org/10.3390/app132212520.

71. Zhao Y. et al. Power optimization with less state transition for green software defined networking. *Future Generation Computer Systems*. 2020. Vol. 114. C. 69-81. DOI: https://doi.org/10.1016/j.future.2020.07.027.

ДОДАТОК А
(обов'язковий)

Лістинг коду програмного забезпечення системи енергоефективної маршрутизації EMEKT в програмно-конфігурованих мережах

topology.py

```
from mininet.topo import Topo
from mininet.node import OVSSwitch

class FatTree4(Topo):
    def build(self):
        k = 4
        dpid_count = 1

        cores = []
        for i in range((k//2)**2):
            cores.append(self.addSwitch(f'c{i+1}', cls=OVSSwitch, dpid=hex(dpid_count)[2:], protocols='OpenFlow13'))
            dpid_count += 1

        for pod in range(k):
            aggs = []
            edges = []
            for i in range(k // 2):
                aggs.append(self.addSwitch(f'a{pod}_{i}', cls=OVSSwitch, dpid=hex(dpid_count)[2:], protocols='OpenFlow13'))
                dpid_count += 1
            for i in range(k // 2):
                edges.append(self.addSwitch(f'e{pod}_{i}', cls=OVSSwitch, dpid=hex(dpid_count)[2:], protocols='OpenFlow13'))
                dpid_count += 1

            for i, edge in enumerate(edges):
                for j in range(k // 2):
                    ip_addr = f'10.{pod}.{i}.{j+1}'
                    h = self.addHost(f'h{pod}_{i}_{j}', ip=ip_addr)
                    self.addLink(edge, h)
```

```
            for agg in aggs:
                self.addLink(edge, agg)

        for i, agg in enumerate(aggs):
            core_offset = i * (k // 2)
            for j in range(k // 2):
                self.addLink(agg, cores[core_offset + j])

topos = {'fattree': (lambda: FatTree4())}
```

eatch_controller.py:

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import CONFIG_DISPATCHER, MAIN_DISPATCHER, set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet, ethernet, arp
from ryu.lib import hub
from ryu.topology import event as topo_event
from ryu.topology.api import get_switch, get_link
import networkx as nx
import time

class EatchController(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(EatchController, self).__init__(*args, **kwargs)
        self.topology_api_app = self
        self.graph = nx.Graph()
        self.mst = nx.Graph()
        self.mac_to_port = {}
        self.datapaths = {}
        self.stats = {}
        self.idle_counters = {}
        self.sleep_threshold = 5
        self.switch_ports = set()
        self.link_to_port = {}
```

```
    self.monitor_thread = hub.spawn(self._monitor)

def _monitor(self):
    while True:
        self.update_topology()

        for dp in list(self.datapaths.values()):
            self._request_stats(dp)
        hub.sleep(5)

def _request_stats(self, datapath):
    parser = datapath.ofproto_parser
    req = parser.OFPPortStatsRequest(datapath, 0, datapath.ofproto.OFPP_ANY)
    datapath.send_msg(req)

@set_ev_cls(ofp_event.EventOFPPortStatsReply, MAIN_DISPATCHER)
def _port_stats_reply_handler(self, ev):
    body = ev.msg.body
    dpid = ev.msg.datapath.id

    active_ports = 0
    idle_ports = 0
    sleep_ports = 0

    for stat in body:
        port_no = stat.port_no
        if port_no > 100: continue

        key = (dpid, port_no)
        curr_bytes = stat.tx_bytes + stat.rx_bytes
        prev_bytes, prev_time = self.stats.get(key, (0, time.time()))

        duration = time.time() - prev_time
        bw_used = ((curr_bytes - prev_bytes) * 8) / duration if duration > 0 else 0
        self.stats[key] = (curr_bytes, time.time())

        if bw_used < 1000:
            self.idle_counters[key] = self.idle_counters.get(key, 0) + 1
```

```
            if self.idle_counters[key] >= self.sleep_threshold:
                sleep_ports += 1
            else:
                idle_ports += 1
        else:
            self.idle_counters[key] = 0
            active_ports += 1

    total_watts = (active_ports * 1.5) + (idle_ports * 0.8) + (sleep_ports * 0.1)
    self.logger.info("ENERGY_LOG [DPID %s]: %.2f W (Act: %d, Idl: %d, Slp: %d)",
                     dpid, total_watts, active_ports, idle_ports, sleep_ports)

@set_ev_cls(ofp_event.EventOFPSwitchFeatures, CONFIG_DISPATCHER)
def switch_features_handler(self, ev):
    datapath = ev.msg.datapath
    self.datapaths[datapath.id] = datapath

    self.graph.add_node(datapath.id)

    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser

    self.add_flow(datapath, 0, parser.OFPMatch(),
                  [parser.OFPActionOutput(ofproto.OFPP_CONTROLLER)], idle_timeout=0)

    self.logger.info("!!! Table-miss FLOW SET for DPID %s", datapath.id)

@set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
def _packet_in_handler(self, ev):
    msg = ev.msg
    datapath = msg.datapath
    dpid = datapath.id
    parser = datapath.ofproto_parser
    in_port = msg.match['in_port']
    pkt = packet.Packet(msg.data)
    eth = pkt.get_protocols(ethernet.ethernet)[0]

    if eth.ethertype == 35020: return
```

```
    dst = eth.dst
    src = eth.src

    if src not in self.mac_to_port:
        self.mac_to_port[src] = (dpid, in_port)

    if dst == 'ff:ff:ff:ff:ff:ff':
        self._do_flood(msg, in_port)
        return

    if dst in self.mac_to_port:
        dst_dpid, dst_port = self.mac_to_port[dst]
        if dpid == dst_dpid:
            out_port = dst_port
        else:
            try:
                path = nx.shortest_path(self.graph, dpid, dst_dpid)
                next_hop = path[path.index(dpid) + 1]
                out_port = self.link_to_port[(dpid, next_hop)]
            except:
                self._do_flood(msg, in_port)
                return

        actions = [parser.OFPActionOutput(out_port)]
        if eth.ethertype == 0x0800:
            match = parser.OFPMatch(in_port=in_port, eth_dst=dst, eth_type=0x0800)
            self.add_flow(datapath, 10, match, actions)

        self._send_packet_out(datapath, msg.buffer_id, in_port, out_port, msg.data)
    else:
        self._do_flood(msg, in_port)

def _do_flood(self, msg, in_port):
    datapath = msg.datapath
    dpid = datapath.id
    parser = datapath.ofproto_parser
```

```
    flood_ports = []
    mst_neighbors = list(self.mst.neighbors(dpid)) if dpid in self.mst else []

    is_topo_ready = len(self.switch_ports) > 0

    for p_no in datapath.ports:
        if p_no == in_port or p_no > 100:
            continue

        if not is_topo_ready:
            if p_no in [1, 2]:
                flood_ports.append(p_no)
        else:
            if not self._is_switch_port(dpid, p_no):
                flood_ports.append(p_no)
            elif any(self.link_to_port.get((dpid, n)) == p_no for n in mst_neighbors):
                flood_ports.append(p_no)

    if flood_ports:
        actions = [parser.OFPActionOutput(p) for p in flood_ports]
        self._send_packet_out(datapath, msg.buffer_id, in_port, None, msg.data, actions)

def _is_switch_port(self, dpid, port_no):
    return (dpid, port_no) in self.switch_ports

def _send_packet_out(self, datapath, buffer_id, in_port, out_port, data, actions=None):
    parser = datapath.ofproto_parser
    if actions is None:
        actions = [parser.OFPActionOutput(out_port)]
    data_to_send = data if buffer_id == datapath.ofproto.OFP_NO_BUFFER else None
    out = parser.OFPPacketOut(datapath=datapath, buffer_id=buffer_id,
                              in_port=in_port, actions=actions, data=data_to_send)
    datapath.send_msg(out)

def add_flow(self, datapath, priority, match, actions, idle_timeout=0):
    ofproto = datapath.ofproto
    parser = datapath.ofproto_parser
    inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS, actions)]
```

```
        mod = parser.OFPFlowMod(datapath=datapath, priority=priority,
                                match=match, instructions=inst, idle_timeout=idle_timeout)
        datapath.send_msg(mod)

    def update_topology(self):
        try:
            links_list = get_link(self.topology_api_app, None)
            self.graph.add_nodes_from(list(self.datapaths.keys()))

            for link in links_list:
                self.graph.add_edge(link.src.dpid, link.dst.dpid)
                self.link_to_port[(link.src.dpid, link.dst.dpid)] = link.src.port_no
                self.link_to_port[(link.dst.dpid, link.src.dpid)] = link.dst.port_no
                self.switch_ports.add((link.src.dpid, link.src.port_no))
                self.switch_ports.add((link.dst.dpid, link.dst.port_no))

            if self.graph.edges:
                self.mst = nx.minimum_spanning_tree(self.graph.to_undirected())

            self.logger.info("STATUS: %d Switches, %d Links in graph",
                             self.graph.number_of_nodes(), self.graph.number_of_edges())
        except Exception as e:
            self.logger.error("Topology error: %s", e)
```

run_experiment.py:

```
from mininet.net import Mininet
from mininet.node import RemoteController, OVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
from topology import FatTree4
import time

def run_experiment():
    setLogLevel('info')
    topo = FatTree4()

    net = Mininet(topo=topo, controller=RemoteController, switch=OVSSwitch, build=False)
    net.addController('c0', controller=RemoteController, ip='127.0.0.1', port=6653)
```

```
    net.build()
    net.start()

    print("\n[*] Чекаємо 15 секунд для вивчення топології та побудови графа (Dijkstra/MST)...")
    time.sleep(15)

    h0_0_0 = net.get('h0_0_0')
    h0_0_1 = net.get('h0_0_1')
    h3_1_1 = net.get('h3_1_1')

    print("\n[*] Запуск фонового UDP трафіку (Intra-Pod) [10.0.0.1 -> 10.0.0.2]")
    h0_0_1.cmd('iperf -s -u -i 1 > udp_server.log &')
    h0_0_0.cmd('iperf -c 10.0.0.2 -u -b 0.5M -t 60 > udp_client.log &')

    print("[*] Запуск TCP сплесків (Inter-Pod) [10.0.0.1 -> 10.3.1.2]")
    h3_1_1.cmd('iperf -s -i 1 > tcp_server.log &')
    h0_0_0.cmd('iperf -c 10.3.1.2 -t 30 > tcp_client.log &')

    print("\n[*] Передаємо керування в Mininet CLI. Для завершення введіть 'exit'\n")
    CLI(net)
    net.stop()

if __name__ == '__main__':
    run_experiment()
```

arp_storm.py:

```
from scapy.all import Ether, ARP, sendp

def generate_arp_storm(interface="h1_0_0-eth0"):
    print(f"[*] Починаємо ARP шторм на інтерфейсі {interface}...")
    print("[*] Ціль: перевірити, чи утримає MST контролер захист від широкомовних петель.")

    pkt = Ether(dst="ff:ff:ff:ff:ff:ff")/ARP(op=1, pdst="10.9.9.9")

    try:
        sendp(pkt, iface=interface, loop=1, inter=0.005, verbose=0)
    except KeyboardInterrupt:
        print("\n[*] ARP шторм зупинено.")
```

```
if __name__ == "__main__":
    generate_arp_storm(interface="h1_0_0-eth0")
```

draw_charts.py:

```
import matplotlib.pyplot as plt
import numpy as np

def draw_energy_chart():
    labels = ['Стандартна мережа\n(Без енергозбереження)', 'Мережа з EMEKT\n(Режими Idle/Sleep)']
    energy_values = [120, 52.2]

    plt.figure(figsize=(8, 6))
    bars = plt.bar(labels, energy_values, color=['#e74c3c', '#2ecc71'], width=0.5)

    for bar in bars:
        yval = bar.get_height()
        plt.text(bar.get_x() + bar.get_width()/2, yval + 2, f'{yval} Вт', ha='center', va='bottom', fontsize=12, fontweight='bold')

    plt.title('Порівняння енергоспоживання мережі з ієрархічною деревоподібною топологією', fontsize=14, pad=15)
    plt.ylabel('Споживана потужність (Ват)', fontsize=12)
    plt.ylim(0, 140)
    plt.grid(axis='y', linestyle='--', alpha=0.7)

    plt.tight_layout()
    plt.savefig('energy_comparison.png', dpi=300)
    print("Графік 'energy_comparison.png' збережено!")
    plt.close()

def draw_traffic_dynamics():
    time = np.arange(0, 65, 5)
    bandwidth = [0.005, 0.005, 0.005, 0.005, 0.005, 0.005, 7.42, 7.42, 7.40, 7.42, 7.41, 7.42, 7.42]
    active_ports = [8, 8, 8, 8, 8, 8, 28, 30, 32, 32, 32, 32, 32]

    fig, ax1 = plt.subplots(figsize=(10, 5))
    color1 = '#2980b9'
```

```
    ax1.set_xlabel('Час (секунди)', fontsize=12)
    ax1.set_ylabel('Пропускна здатність (Гбіт/с)', color=color1, fontsize=12)
    line1 = ax1.plot(time, bandwidth, color=color1, marker='o', linewidth=2.5, label='Трафік (TCP сплеск)')
    ax1.tick_params(axis='y', labelcolor=color1)
    ax1.grid(True, linestyle='--', alpha=0.6)

    ax2 = ax1.twinx()
    color2 = '#e67e22'
    ax2.set_ylabel('Кількість активних портів', color=color2, fontsize=12)
    line2 = ax2.plot(time, active_ports, color=color2, marker='s', linestyle='--', linewidth=2.5, label='Активні порти (EMEKT)')
    ax2.tick_params(axis='y', labelcolor=color2)
    ax2.set_ylim(0, 40)

    plt.title('Динаміка трафіку та реакція контролера на сплески', fontsize=14, pad=15)
    lines = line1 + line2
    labels = [l.get_label() for l in lines]
    ax1.legend(lines, labels, loc='upper left')

    plt.tight_layout()
    plt.savefig('traffic_dynamics.png', dpi=300)
    print("Графік 'traffic_dynamics.png' збережено!")
    plt.close()

def draw_survivability():
    labels = ['Класична ієрархічна деревоподібна топологія \n(Під час атаки)', 'Ієрархічна деревоподібна топологія + EMEKT\n(Під час атаки)']
    packet_loss = [100, 0]
    jitter = [1200, 0.104]
    cpu_load = [100, 15]

    fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(14, 5))
    colors = ['#c0392b', '#27ae60']

    ax1.bar(labels, packet_loss, color=colors, width=0.5)
    ax1.set_title('Втрати пакетів (Packet Loss)', fontsize=12)
    ax1.set_ylabel('% втрат')
    ax1.set_ylim(0, 110)
```

```
    for i, v in enumerate(packet_loss):
        ax1.text(i, v + 2, f"{v}%", ha='center', fontweight='bold')

    ax2.bar(labels, jitter, color=colors, width=0.5)
    ax2.set_title('Тремтіння затримки (Jitter)', fontsize=12)
    ax2.set_ylabel('Мілісекунди (мс)')
    ax2.set_yscale('log')
    for i, v in enumerate(jitter):
        ax2.text(i, v + (v*0.1), f"{v} мс", ha='center', fontweight='bold')

    # CPU Load
    ax3.bar(labels, cpu_load, color=colors, width=0.5)
    ax3.set_title('Завантаженість контролера/CPU', fontsize=12)
    ax3.set_ylabel('% навантаження')
    ax3.set_ylim(0, 110)
    for i, v in enumerate(cpu_load):
        ax3.text(i, v + 2, f"{v}%", ha='center', fontweight='bold')

    plt.suptitle('Показники якості обслуговування під час атаки ARP Storm', fontsize=16, y=1.05)
    for ax in (ax1, ax2, ax3):
        ax.grid(axis='y', linestyle='--', alpha=0.5)

    plt.tight_layout()
    plt.savefig('network_survivability.png', dpi=300, bbox_inches='tight')
    print("Графік 'network_survivability.png' збережено!")
    plt.close()

def draw_comparative_analysis():
    methods = ['Стандарт (ECMP)', 'ElasticTree', 'CARPO', 'Метод EMEKT']
    energy_savings = [0, 50, 46, 55]
    packet_loss_risk = [15, 12, 8, 0]

    x = np.arange(len(methods))
    width = 0.5

    fig, ax1 = plt.subplots(figsize=(10, 6))

    bars = ax1.bar(x, energy_savings, width, color='seagreen', alpha=0.7, label='Економія енергії (%)')
```

```
    ax1.set_ylabel('Економія енергоспоживання (%)', color='seagreen', fontsize=12, fontweight='bold')
    ax1.set_ylim(0, 70)
    ax1.tick_params(axis='y', labelcolor='seagreen')

    ax2 = ax1.twinx()
    line = ax2.plot(x, packet_loss_risk, color='crimson', marker='D', linewidth=2.5, label='Ризик втрати пакетів (%)')
    ax2.set_ylabel('Втрата пакетів при сплесках (%)', color='crimson', fontsize=12, fontweight='bold')
    ax2.set_ylim(-1, 20)
    ax2.tick_params(axis='y', labelcolor='crimson')

    plt.title('Порівняльна характеристика енергоефективності та надійності', fontsize=14, pad=20)
    ax1.set_xticks(x)
    ax1.set_xticklabels(methods)
    ax1.grid(axis='y', linestyle='--', alpha=0.3)

    lines, labels = ax1.get_legend_handles_labels()
    lines2, labels2 = ax2.get_legend_handles_labels()
    ax1.legend(lines + lines2, labels + labels2, loc='upper left')

    plt.tight_layout()
    plt.savefig('comparative_analysis.png', dpi=300)
    print("Графік 'comparative_analysis.png' збережено!")
    plt.close()

if __name__ == "__main__":
    draw_energy_chart()
    draw_traffic_dynamics()
    draw_survivability()
    draw_comparative_analysis()
```

ДОДАТОК Б
(обов'язковий)

Публікація

DOI: https://doi.org/10.64828/conf-115-2026-1

УДК 004.7

Сойко Максим Олександрович

магістрант

Лисенко Сергій Миколайович

д.т.н., професор

Хмельницький національний університет

м. Хмельницький, Україна

МЕТОД ЗАБЕЗПЕЧЕННЯ ЗЕЛЕНИХ ОБЧИСЛЕНЬ НА БАЗІ ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖ

Анотація. У статті досліджується проблема зростання енергоспоживання ІКТ-сектору та пропонується комплексний метод «зелених обчислень» на базі програмно-конфігурованих мереж (SDN). На основі аналізу сучасних промислових мереж обґрунтовано доцільність консолідації трафіку. Сформовано математичну модель та запропоновано евристичний підхід до оптимізації для SDN-контролера, що дозволяє динамічно переводити вільні транзитні вузли у режим сну без деградації параметрів якості обслуговування (QoS).

Ключові слова: програмно-конфігуровані мережі, зелені обчислення, енергоефективність, SDN-контролер, маршрутизація, консолідація трафіку, якість обслуговування.

Розвиток цифрової економіки тісно пов'язаний із впровадженням 5G/6G, IoT і хмарних обчислень. Ці технології створюють величезні обсяги даних. Для прикладу, у 2018 році було 33 зетабайти, а до 2030 року очікується 175 зетабайтів [2, с. 129]. Поширення розумних пристроїв, потокового відео високої якості та автономних систем змушує постійно збільшувати потужності мереж

для обробки даних майже миттєво. На комунікаційні мережеві пристрої (комутатори, маршрутизатори та канали зв'язку) припадає від 15 до 20% від загального обсягу електроенергії, спожитої дата-центрами [3, с. 985]. При цьому значна частина цієї енергії витрачається марно, оскільки традиційні мережі проєктуються з розрахунком на пікові навантаження, і мережеве обладнання працює на повну потужність навіть у періоди мінімального трафіку.

Сучасний виробничий цикл критично залежить від швидкості обміну даними між конструкторськими бюро та цехами. Передача масивних 3D-моделей із середовищ SolidWorks та AutoCAD, а також трансляція складних керуючих програм для верстатів із числовим програмним керуванням, створюють специфічні нерівномірні «піки» навантаження. У періоди затишшя або нічних змін така інфраструктура працює марно [8, с. 45]. Застосування парадигми програмно-конфігурованих мереж (SDN) дозволяє відійти від статичної маршрутизації. Контролер виступає інтелектуальним арбітром: він ідентифікує критично важливі виробничі потоки та надає їм пріоритет. Програмно-керована топологія дає змогу гнучко консолідувати трафік на мінімальній кількості портів, автоматично переводячи незадіяні вузли у сплячий режим [5, с. 189].

Однак складнощі у створенні енергоефективних програмно-керованих мереж часто виникають через протиріччя – треба одночасно економити електроенергію й дотримуватись жорстких норм продуктивності. Агресивне вимкнення надлишкових шляхів може викликати зниження пропускної здатності та підвищення ризику втрати пакетів у разі раптових сплесків трафіку. Замість стабільної трансляції даних виникають затримки, що призводить до розриву мережевих сесій [6, с. 208]. Це пояснюється тим, що переведення мережевих пристроїв з режиму сну в активний стан вимагає певного часу на апаратне «пробудження», протягом якого вхідні пакети змушені очікувати в чергах комутаторів. Крім того, надто часте перемикання станів живлення генерує додатковий обсяг службових повідомлень, що перевантажує центральний SDN-контролер та збільшує час його реакції на динамічні зміни. В

умовах надмірної консолідації трафіку мережа стає вкрай вразливою до непередбачуваних перевантажень (мікро-сплесків) або відмов окремих ліній зв'язку. З огляду на це, критично важливо впроваджувати гнучкі алгоритми, які не просто сліпо вимикають обладнання, а підтримують інтелектуальний баланс (залишаючи необхідний запас пропускної здатності). Порівняльний аналіз ефективності існуючих підходів та запропонованого SDN-методу, який дозволяє уникнути цих недоліків, наведено в табл. 1.

Таблиця 1

Порівняльний аналіз методів забезпечення енергоефективності

Метод / Підхід	Основний механізм	Недоліки / Ризики
Традиційна маршрутизація	Робота обладнання на повну потужність 24/7	Максимальні витрати енергії (до 90% марно в режимі простою)
ElasticTree / CARPO	Жорстка консолідація мережевих потоків	Висока обчислювальна складність, ризик втрати пакетів при сплесках трафіку
Запропонований SDN-метод	Динамічне переведення портів у стан сну (LPI) з урахуванням якості обслуговування (QoS)	Оптимальний баланс між економією енергії (OPEX) та продуктивністю

Для досягнення мети роботи обрано комплексний підхід, що поєднує методи математичного моделювання, теорії графів, розробку евристичних алгоритмів та імітаційне моделювання. Для математичної формалізації задачі топологію мережі задано у вигляді графа $G = (V, E)$. Стан кожного комутатора $v \in V$ та каналу зв'язку $e \in E$ описується бінарними змінними x_v, $y_e \in \{0, 1\}$ де значення «1» означає активний стан пристрою, а «0» – стан сну. Цільова функція полягає у мінімізації сумарних енерговитрат інфраструктури та має наступний вигляд (1):

$$P_{total} = \sum_{v \in V}(P^v_{static} * x_v + P^v_{dyn} * u_v) + \sum_{e \in E} P^e_{link} * y_e \rightarrow min \qquad (1)$$

де: P^v_{static} – статична потужність вузла v;

P^v_{dyn} – динамічна потужність вузла, що залежить від його завантаження u_v;

P^{e}_{link} – енергоспоживання активної лінії зв'язку e;

x_v, y_e – бінарні змінні стану вузлів та ліній відповідно.

Оскільки пряме математичне розв'язання такої багатокритеріальної задачі (особливо в умовах динамічної зміни топології) належить до класу NP-складних і працює надто повільно для оперативного прийняття рішень SDN-контролером [1, с. 132; 7, с. 37], основним ядром запропонованого методу виступає легкий евристичний алгоритм. Цей алгоритм у реальному часі здійснює агрегацію потоків, свідомо віддаючи пріоритет під час прокладання нових маршрутів тим вузлам, які вже перебувають в активному стані. Це дозволяє уникнути деградації якості обслуговування та ліквідувати затримки, пов'язані з часом апаратного «пробудження» обладнання [4, с. 136]. Процес взаємодії мережевих пристроїв та механізми оптимізації живлення реалізуються через централізований вузол (Рис. 1. Архітектура управління енергоспоживанням на базі SDN-контролера).

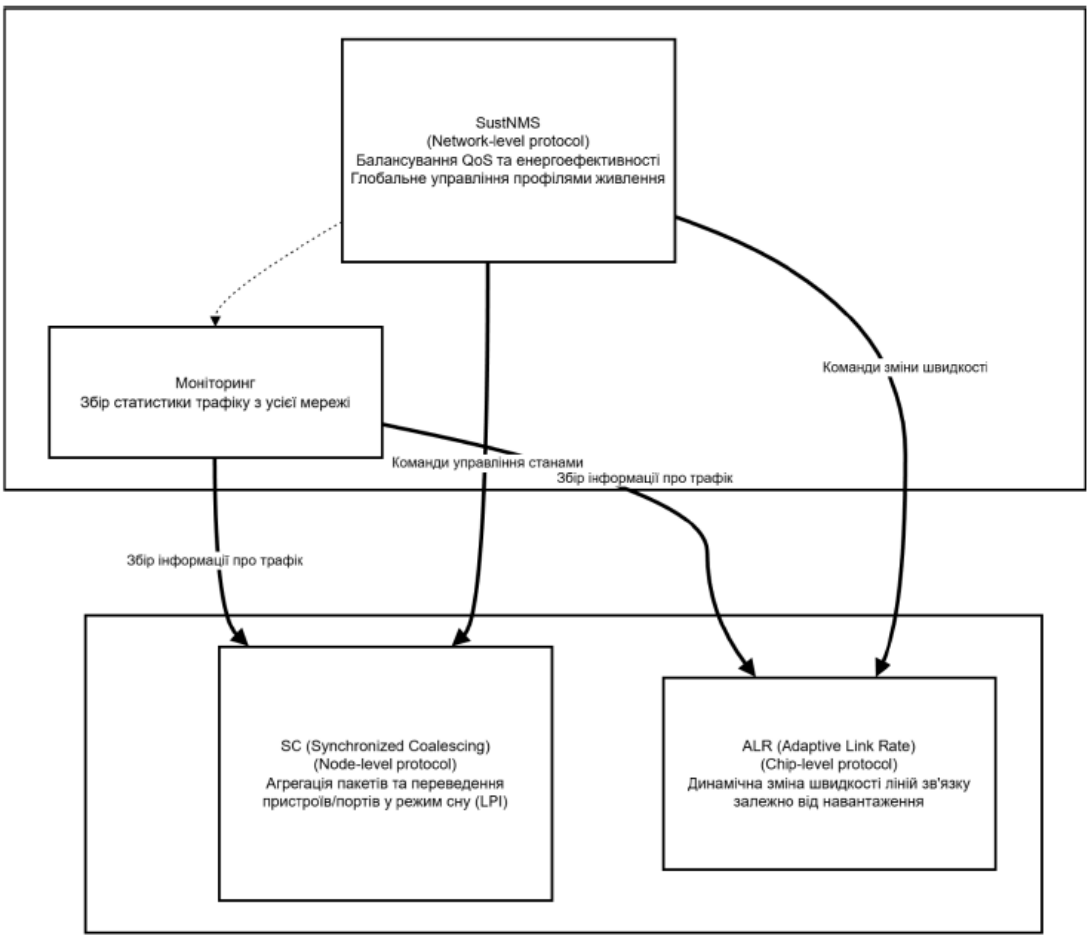


Рис. 1. Архітектура управління енергоспоживанням на базі SDN-контролера

Оскільки розгортання та тестування нових алгоритмів маршрутизації на реальному апаратному обладнанні ЦОД є дороговартісним і не завжди можливим, основним підходом для валідації розробленого методу стане імітаційне моделювання. Програмна реалізація логіки управління здійснюватиметься на мові Python з використанням SDN-контролера Ryu, який забезпечує повну підтримку протоколу OpenFlow 1.3+. В якості середовища емуляції обрано Mininet, що дозволяє створювати віртуальні топології типу Fat-Tree з високим ступенем достовірності мережевих процесів [7, с. 164].

Отже, практична цінність отриманих результатів виходить далеко за межі суто академічного інтересу. Інтелектуальне управління інфраструктурою на базі

SDN дозволяє перетворити енергоспоживання з некерованої статті витрат на прогнозований і оптимізований ресурс. Шляхом інтелектуальної консолідації трафіку на обмеженій кількості активних вузлів стає можливим не лише пряме заощадження електрики, а й суттєве покращення показника PUE (Power Usage Effectiveness) у центрах обробки даних, гарантуючи при цьому безперебійність передачі критично важливих потоків [4, с. 4].

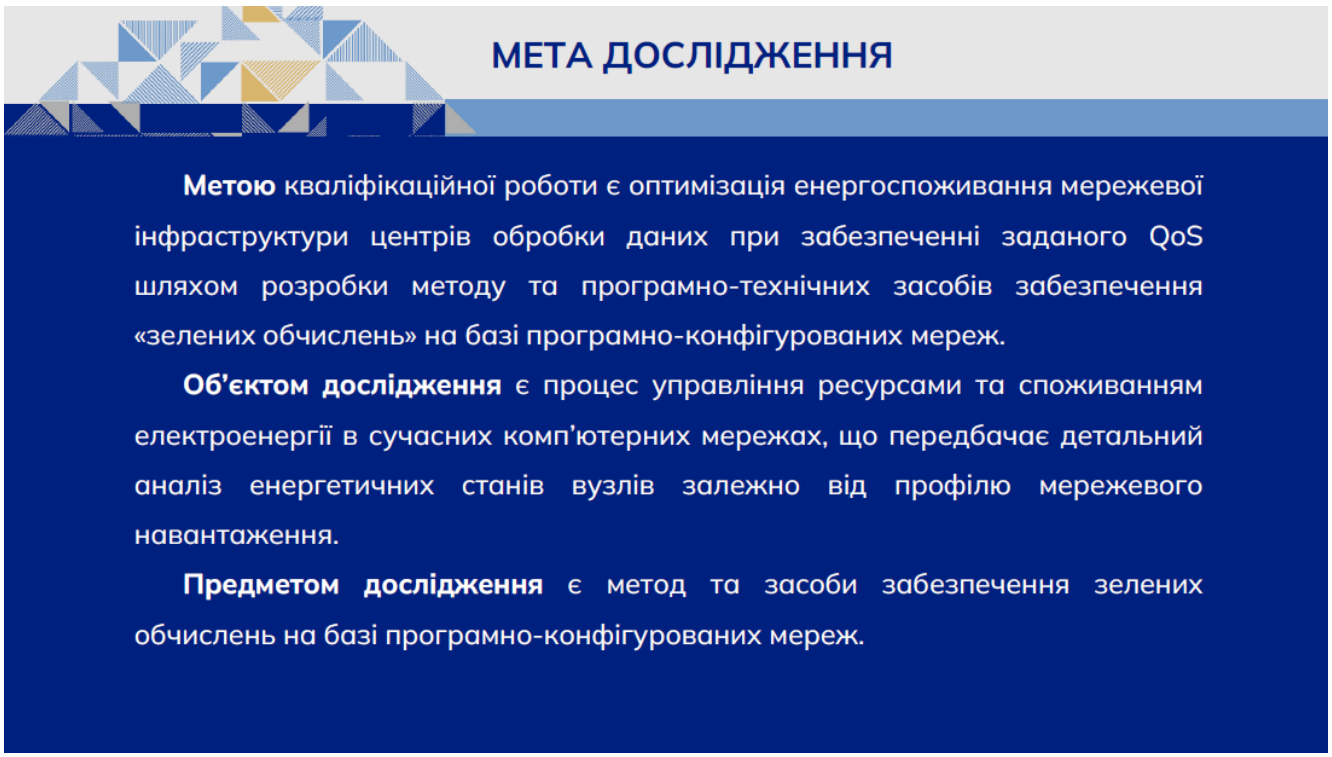
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Agg P. A., Johanyák Z. C. Energy savings in SDN networks // Gradus. 2021. Vol. 8, no. 1. P. 205–210.
2. Assefa B. G., Özkasap Ö. A survey of energy efficiency in SDN: Software-based methods and optimization models // Journal of Network and Computer Applications. 2019. Vol. 137. P. 127–143.
3. Masanet E. [et al.]. Recalibrating global data center energy-use estimates. Science. 2020. Vol. 367, no. 6481. DOI: 10.1126/science.aba3758.
4. Safitra M. F., Lubis M. Green Networking: Challenges, Opportunities for Sustainable Development. Proceedings of the 11th International Conference on IT: IoT and Smart City. 2023. DOI: 10.1145/3613245.3613251.
5. Tuysuz M. F., Ankarali Z. K., Gözüpek D. A survey on energy efficiency in software defined networks // Computer Networks. 2017. Vol. 113. P. 188–204.
6. Гніденко М. П., Гніденко М. М., Вишнівський О. В., Зінченко В. В. Забезпечення енергоефективності програмно визначених мереж (SDNs) при впровадженні різних схем безпеки // Наукові записки ДУІКТ. 2024. № 2(6). С. 204–228.
7. Прокопець Н. А. Енергоефективне обслуговування навантаження інформаційно-комунікаційної мережі: 172. Київ, 2022. 217 с. URL: https://ela.kpi.ua/items/dda7c4d0-3599-4b7e-9fd9-fc0f9c497e93 (дата звернення: 30.03.2026).
8. Роспутній В. В. Аналіз способів побудови та функціонування сенсорних

радіомереж із використанням SDN : кваліфікаційна робота бакалавра : 172. Київ, 2021. 65 с.

ДОДАТОК В
(обов'язковий)

Презентація



ЗАДАЧІ ДОСЛІДЖЕННЯ

Для досягнення поставленої мети необхідно вирішити такі завдання:

- проаналізувати та систематизувати сучасний стан проблеми енергоспоживання в мережах, а також існуючі підходи до реалізації «зелених» програмно-конфігурованих мереж;
- обґрунтувати вибір архітектурних компонентів технології програмно-конфігурованих мереж як інструменту для оптимізації енергозбереження мережевої інфраструктури;
- сформувати математичні та алгоритмічні моделі для забезпечення динамічної зміни топології та ефективної консолідації трафіку;
- розробити програмне забезпечення для імітаційного моделювання запропонованих рішень у середовищі Mininet і провести оцінку їхньої результативності.

НАУКОВА НОВИЗНА ТА ПРАКТИЧНА ЦІННІСТЬ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Наукова новизна отриманих результатів полягає у вдосконаленні евристичного методу енергоефективної консолідації трафіку через використання динамічного перерахунку ваг ребер графа з урахуванням штрафних коефіцієнтів за апаратне пробудження портів, а також інтеграцію механізму ізоляції широкомовних штормів на основі алгоритму мінімального кістякового дерева.

Практична значимість отриманих результатів полягає у програмній реалізації модуля управління на базі SDN-контролера Ryu, що дозволяє знизити енергоспоживання інфраструктури при збереженні стабільної пропускної здатності та нульових втратах пакетів.

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

- В умовах стрімкого впровадження 5G/6G, систем IoT та Edge Computing обсяги глобального трафіку зростають експоненціально, що за прогнозами призведе до збільшення частки ІКТ-сектору в загальносвітовому споживанні електроенергії до 20% до 2030 року.
- Класична мережева архітектура ЦОД є фундаментально негнучкою через принцип «надлишкового резервування», внаслідок чого типові комутатори марнують до 90% енергії у стані очікування, ігноруючи принципи енергопропорційності.
- Ключовою науково-технічною проблемою стає необхідність переходу від статичних конфігурацій до динамічного управління ресурсами на базі SDN, що дозволило б консолідувати трафік та вимикати надлишкові вузли без деградації критичних показників якості обслуговування.

АНАЛІЗ ВІДОМИХ МЕТОДІВ

- Метод ElasticTree забезпечує консолідацію трафіку в деревоподібних структурах, проте має низьку масштабованість і нездатність адаптуватися до різких змін навантаження через відсутність аналізу динаміки інформаційних потоків.
- Алгоритм CARPO використовує кореляцію потоків для щільної агрегації даних, але висока обчислювальна складність перерахунку матриць у реальному часі створює надмірне навантаження на SDN-контролер та призводить до зростання мережевих затримок.
- GreenSDN інтегрує апаратні режими низького енергоспоживання, проте складна багаторівнева архітектура та жорстка залежність від специфічних моделей обладнання суттєво обмежують гнучкість і універсальність такого рішення.

МАТЕМАТИЧНА МОДЕЛЬ

- Мережа представлена як зважений граф $G = (V, E)$. Стан пристроїв та портів описується бінарними змінними $x_v, y_e \in \{0, 1\}$.
- Мінімізація сумарної потужності:

$$P_{total} = \sum(P_{static} + P_{dyn}) * x_v + \sum P_{link} * y_e \rightarrow min$$

- Система обмежень QoS базується на дотриманні граничної ємності каналів згідно з умовою $f_e \leq C_e * y_e$ та мінімізації сумарної затримки на шляху $L_{path} \leq L_{max}$ з обов'язковим урахуванням апаратного штрафу за пробудження портів τ_{wake}, при цьому цілісність інформаційних потоків гарантується через безумовне виконання закону збереження потоку в кожному комутаційному вузлі інфраструктури.

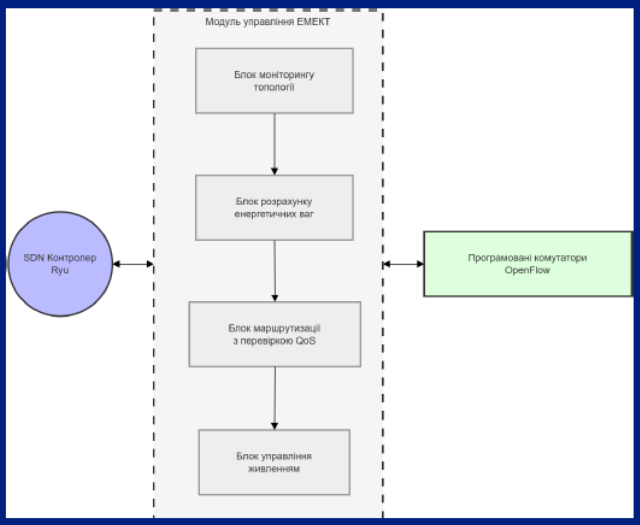
ЕВРЕСТИЧНИЙ МЕТОД ЕНЕРГОЕФЕКТИВНОЇ КОНСОЛІДАЦІЇ ТРАФІКУ

Схема алгоритму евристичного методу енергоефективної консолідації трафіку

- Алгоритм використовує енергетичну метрику ваги ребер. Вага зростає при низькій залишковій ємності та додає штраф для сплячих вузлів.
- Ітеративний пошук шляхів (алгоритм Дейкстри) з пріоритетом на вже активні лінії. Потоки сортуються за обсягом для щільного ущільнення трафіку.
- Автоматична деактивація портів і шасі при відсутності корисного трафіку протягом циклів моніторингу через інтерфейс OpenFlow.

АРХІТЕКТУРА СИСТЕМИ

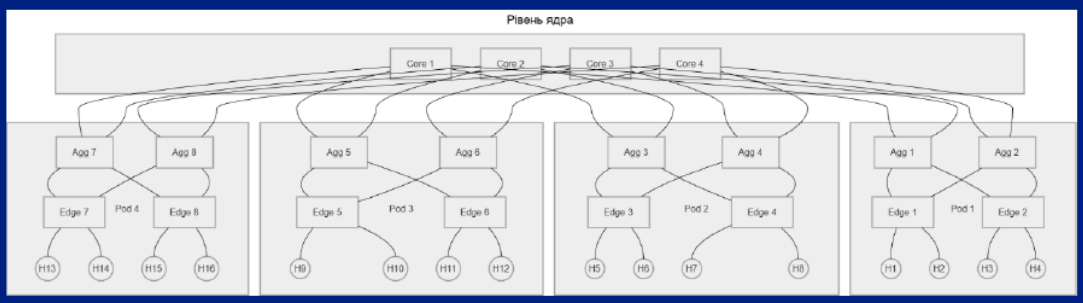
Схема архітектури системи



Логіка системи базується на Python 3 та Ryu SDN з асинхронною обробкою подій через Eventlet, тоді як бібліотека NetworkX забезпечує підтримку топології та розрахунок енергоефективних траєкторій. Взаємодія у площині Southbound реалізована через протокол OpenFlow 1.3, що дозволяє збирати статистику та та механізми модифікації портів для мікроуправління живленням.

ТЕСТОВЕ СЕРЕДОВИЩЕ

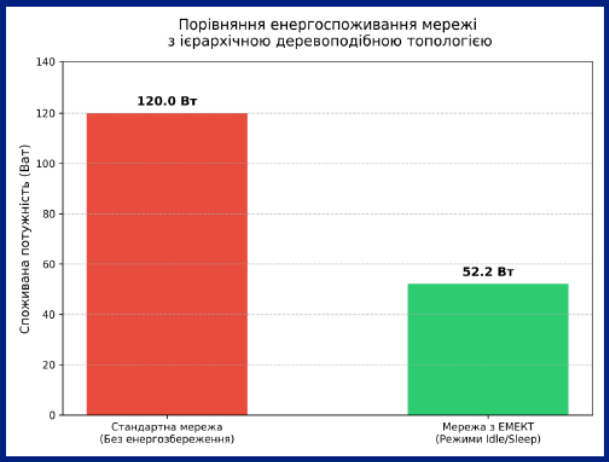
Схема імітованої ієрархічної деревоподібної топології у середовищі Mininet



Ієрархічна структура деревоподібної топології: 20 комутаторів Open vSwitch, 16 серверних хостів. Побудована за допомогою Python-скриптів. Використано утиліту iperf3 для імітації реального навантаження ЦОД (масивні TCP-передачі та UDP-потоки з фіксованим бітрейтом). Циклічне опитування лічильників байтів для розрахунку поточної завантаженості каналів у реальному часі.

ЕНЕРГОЕФЕКТИВНІСТЬ ІНФРАСТРУКТУРИ

Порівняльна гістограма енергоспоживання мережі в умовах низького навантаження



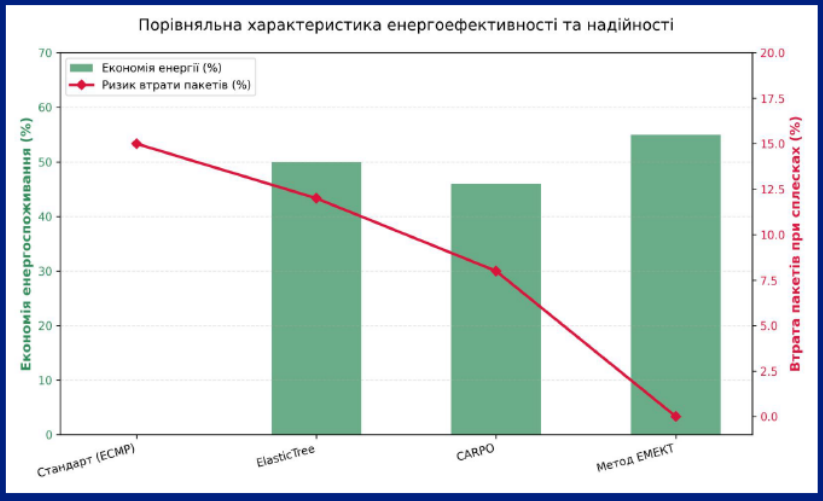
Аналіз результатів імітаційного моделювання підтвердив високу ефективність розробленого методу, що дозволяє знизити сумарне енергоспоживання інфраструктури на 50–55% у періоди низької активності мережі. Отримана залежність демонструє здатність алгоритму EMEKT адаптивно керувати станами живлення портів, забезпечуючи стабільно високий показник економії незалежно від інтенсивності вхідних потоків даних.

DoS-ЗАХИСТ

Порівняльна діаграма показників живучості мережі та QoS під час імітації DoS-атаки

ПОРІВНЯЛЬНА ОЦІНКА ТА ЦІННІСТЬ

Порівняльна характеристика енергоефективності та надійності розробленого методу відносно існуючих рішень



ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень:

- проаналізовано існуючі методи маршрутизації та управління ресурсами в умовах стрімкого зростання енергоспоживання телекомунікаційним обладнанням;
- обрано експериментальним шляхом найбільш підходящий інструментарій для реалізації алгоритмів зелених обчислень;
- розроблено математичну модель системи з використанням динамічного перерахунку ваг графів та врахуванням штрафних коефіцієнтів за апаратне пробудження портів;
- розроблено метод та програмне забезпечення системи енергозберігаючої маршрутизації із вбудованими механізмами захисту від широкомовних штормів;

Результати експерименту підтвердили зниження енергоспоживання інфраструктури на 50-55% при збереженні пропускної здатності 7.42 Гбіт/с та нульових втратах пакетів.

ПУБЛІКАЦІЇ

- Лисенко С., Сойко М. Метод забезпечення зелених обчислень на базі програмно-конфігурованих мереж. Oxford International Science Forum. Матеріали міжнародної науково-практичної конференції (17-19 квітня 2026 р.). Оксфорд, Велика Британія: Oxford University Press, 2026. С. 92-98.

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Максим СОЙКО

Співавтор:

Назва: Метод забезпечення зелених обчислень на базі програмно-конфігурованих мереж

Експерт: Сергій ЛИСЕНКО

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:3.91%

Коефіцієнт подібності 2:0.66%

Мікропробіли: 3

Заміна букв: 2

Інтервали: 0

Білі знаки: 9

Дата створення звіту: 2026-05-13 17:44:53.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-13 Доцент Андрій Нічепорук

Дата експерт

Wed May 13 18:20:50 EEST 2026, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 23.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 271472 Назва: МКР Метод забезпечення зелених обчислень на базі програмно-конфігурованих мереж Додано в БД: 2026-05-13 Автора: Максим СОЙКО Керівники: Сергій ЛИСЕНКО Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	137139	1108	32968 (24%)	286 (26%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
269947	Назва: Звіт з НДП Метод забезпечення зелених обчислень на базі програмно-конфігурованих мереж Додано в БД: 2026-03-23 Автора: Сойку М.О. Керівники: Гнатчук Є. Г Консультанти: Опоненти:	31552 (23.0%)	258 (23.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Здобувач: Максим СОЙКО

Тема: Метод забезпечення зелених обчислень на базі програмно-конфігурованих мереж

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи магістра:

Кількість листів креслень – ; кількість сторінок записки 82

1. Короткий зміст роботи та прийнятих рішень У роботі запропоновано метод забезпечення зелених обчислень на базі програмно-конфігурованих мереж

2. Висновок про відповідність роботи дипломному завданню Кваліфікаційна робота магістра відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведено глибокий аналіз проблеми зростання енергоспоживання в ІКТ-секторі та обґрунтовано переваги парадигми SDN для реалізації концепції Green Computing. У другому розділі розроблено математичну модель оптимізації енергоспоживання, яка, на відміну від аналогів, враховує штрафні коефіцієнти за апаратне пробудження портів. У третьому розділі запропоновано алгоритмічне забезпечення методу EMEKT, інтегроване з механізмами фільтрації аномального трафіку. У четвертому розділі проведено розширене імітаційне моделювання у середовищі Mininet з використанням контролера Ryu, що підтвердило ефективність розроблених рішень.

4. Позитивні сторони роботи: Вдалося досягти високого показника енергозбереження при збереженні високої пропускної здатності. Особливої уваги заслуговує впровадження захисту від DoS-атак на базі мінімального кістякового дерева, що забезпечує стабільність якості обслуговування.

5. Негативні сторони роботи: Основним недоліком методу є зростання обчислювальної складності при масштабуванні на топології понад 1000 вузлів, що потребує подальшої оптимізації для зниження навантаження на контролер. Також метод потребує вдосконалення математичного опису перехідних процесів у гетерогенних мережевих середовищах.

6. Оцінка графічного оформлення та пояснювальної записки роботи: –

7. Відгук про роботу в цілому: В загальному робота виконана на хорошому рівні.

8. Інші зауваження: –

9. Оцінка кваліфікаційної роботи магістра:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи магістра вважаю, що робота заслуговує оцінки «добре» 83.00 (B)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Бедрачова Леся Петрівна, зав. каф. ІПЗ, ХНУ

"15" грудня 2026р.

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Максим СОЙКО

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-24-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

01.05.2026

Дата

Підпис

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Метод забезпечення зелених обчислень на базі програмно-конфігурованих мереж
Автор Максим СОЙКО
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: д.т.н., професор Сергій ЛИСЕНКО

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 3,91% і адресується до 71 першоджерела; та системою Anti-Plagiarism складає 23%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

13.05.2025

Завідувач кафедри _____ Ольга ПАВЛОВА
Підпис Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми _____ Олег САВЕНКО
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Сергій ЛИСЕНКО
Підпис Ім'я, ПРІЗВИЩЕ