МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Факультет математики і інформатики

Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

магістр

на тему «Розробка системи мультиоб'єктного управління в SDN мережах»

Виконав: студент 2 курсу, групи МФ-61

спеціальність 122 «Комп'ютерні науки»
освітньо-наукова програма
«Інформатика»

Статкевич Антон Олексійович

Керівник: професор Руккас К.М.

Рецензент _____

Харків – 2025

ЗМІСТ

ВСТУП

Мета

Сучасні програмно визначені мережі (SDN) завдяки розділенню контрольного та транспортного рівнів забезпечують централізоване управління трафіком і гнучкість при зміні умов навантаження. Проте класичні підходи до балансування (Round Robin, Least Connection тощо) не враховують динамічного стану черг у маршрутизаторах, що призводить до нерівномірного завантаження, зростання затримок і втрат пакетів та погіршення QoS.

Метою даної роботи є розробка системи мультиоб’єктного управління в SDN мережах, яка динамічно розподіляє кілька паралельних потоків у режимі реального часу з урахуванням поточної довжини черг та пріоритетів пакетів. Основними завданнями дослідження є:

- обґрунтування вибору і моделювання алгоритму балансування навантаження на основі аналізу довжин черг у маршрутизаторах;
- розробка модульної архітектури системи, яка забезпечує інтеграцію з SDN-контролерами та підтримку збору QoS-метрик;
- імплементація та тестування алгоритму в симуляційному середовищі з подальшим переходом до інтеграції в реальну мережу;
- аналіз ефективності розробленого рішення за показниками затримки, втрат пакетів та пропускної здатності.

Актуальність теми підтверджується необхідністю підвищення ефективності мережевих інфраструктур у дата-центрах, IoT-системах, корпоративних і телекомунікаційних мережах. Уміння адаптуватися до швидкоплинних змін навантаження та забезпечувати надійне обслуговування критично важливого трафіку є ключовим фактором конкурентоспроможності сучасних мережевих рішень.

Стислий огляд відомих результатів

Класичні підходи до балансування навантаження — RoundRobin, LeastConnection та їхні варіації — добре зарекомендували себе у статичних або помірно змінних середовищах, однак демонструють обмежену ефективність за наявності нерівномірного чи пріоритизованого трафіку. Низка робіт показала, що адаптивні методи, які відстежують фактичну довжину черг, здатні скоротити затримку до 25–30% і рівномірніше використовувати пропускну здатність мережі. Для аналітичної оцінки таких рішень застосовують моделі теорії черг(M/M/1,M/G/1), що дозволяє формально пов’язувати чергу й час обслуговування. Водночас сучасні дослідження все частіше інтегрують подібні алгоритми безпосередньо у SDN-контролери (Ryu, OpenDaylight, ONOS), використовуючи OpenFlow для збору статистики й миттєвого оновлення маршрутів. Попри успіхи, впровадження пріоритетного балансування саме на основі довжини черг у реальних SDN-середовищах усе ще перебуває на етапі апробації. Наявні експериментальні результати потребують узагальнення, а практичні рішення— відпрацювання на складних топологіях і змішаному трафіку, що й зумовлює актуальність даного дослідження.

Відомості про одержані результати та їх новизна

В рамках даної роботи було створено прототип системи, що реалізує алгоритм Queue Length-Based Load Balancing в SDN мережах. Результати симуляційних експериментів у середовищі SimPy та Mininet показали, що впроваджений підхід дозволяє знизити середню затримку пакетів на 15–25 % у порівнянні з класичними алгоритмами, а також зменшити кількість відкинутих пакетів при пікових навантаженнях. Новизна роботи полягає у:

- розширенні класичного алгоритму довжини черг підтримкою пріоритетного обслуговування;
- створенні модульного рішення з можливістю симуляції та подальшої інтеграції в реальні SDN-контролери через Ryu;

- поєднанні аналітичного підходу (моделювання за M/M/1 та M/G/1) з практичними інструментами для візуалізації та моніторингу QoS у реальному часі.

Об’єкт і предмет дослідження

Об’єктом дослідження є програмно-визначені мережі як середовище для реалізації централізованого управління трафіком.

Предметом дослідження виступають методи та алгоритми динамічного балансування трафіку на основі довжини черг у маршрутизаторах SDN-мереж з підтримкою пріоритетів пакетів і моніторингом QoS-метрик.

АНАЛІЗ СУЧАСНОГО СТАНУ

Загальна характеристика SDN мереж:

Software-Defined Networking (SDN) – це архітектурна парадигма розробки мереж, у якій функції управління відділені від функцій пересилання пакетів. Традиційні мережі інтегрують контрольну (routing, політики) та передачну частини в одному пристрої, що ускладнює централізоване управління. Натомість у SDN інтелект мережі «використовується» на окремому контролері: фізичні комутатори та маршрутизатори працюють як прості елементи пересилання, а вся логіка прийняття рішень зосереджена в програмному контролері.

Архітектура SDN зазвичай поділяється на три рівні: Data Plane (адаптери та комутатори, що пересилають пакети), Control Plane (контролер(-и) мережі) та Application Plane (додатки вищого рівня). Контролер у SDN, подібно до «мережевої операційної системи», керує правилами пересилки (flow-таблицями) комутаторів за допомогою південних інтерфейсів (наприклад, протоколу OpenFlow). Прикладний рівень натомість задає загальні політики й завдання (маршрутизація, безпека, балансування та ін.) через північний інтерфейс. Розділення мережних функцій дозволяє чітко відокремити визначення політик, їх втілення на обладнанні та безпосередню передачу пакетів, що спрощує розробку нових мережевих сервісів і підвищує гнучкість системи.

SDN надає глобальний огляд стану мережі та можливості програмування її поведінки «на льоту», що спрощує масштабування й управління складними інфраструктурами. Завдяки відкритим стандартам (загальний інтерфейс управління, уніфіковані протоколи), SDN-компоненти різних виробників можуть взаємодіяти між собою. У результаті сучасні SDN-рішення стають все більш поширеними як у науковому середовищі, так і в індустрії. Крім того, SDN істотно полегшує створення віртуальних мережних інфраструктур: завдяки поділу на шари та відкритим API з’являється можливість проєктувати кілька ізольованих логічних

мереж поверх єдиної фізичної (технології мережевої віртуалізації). Таким чином, SDN служить основою для новітніх мережевих середовищ, де гнучкість і програмність стоять понад жорстко фіксованою топологією.

Проблеми управління трафіком у традиційних мережах та в SDN:

У традиційних IP-мережах складність управління викликана розподіленою архітектурою контролю та використанням пропрієтарного обладнання. Немає єдиного програмного інтерфейсу для перенастроювання маршрутизації: балансування трафіку в таких системах зазвичай здійснюється на рівні апаратних балансувальників навантаження, призначених для серверів. Такі пристрої виконують базове розподілення запитів (наприклад, за алгоритмом Round Robin чи Least Connections), але вони дорогі, закриті та неадаптивні. Крім того, маршрутизатори класично підтримують тільки примітивні методи (наприклад, ECMP для рівнозначних маршрутів), тому у разі нерівномірного навантаження або збою окремої лінії можуть виникати «вузькі місця» та нестабільність. Усі ці фактори роблять масштабування і динамічну перебалансировку трафіку дуже складними завданнями в традиційній мережі.

У свою чергу, SDN дозволяє контролеру бачити всю топологію і оперативно змінювати правила передачі трафіку. Контролер може збирати статистику з усіх комутаторів і, на основі цього, здійснювати розумне балансування: він може перенаправити потоки на лінки з меншим навантаженням або розподілити нові з’єднання між серверами так, щоб уникнути перевантаження. Наприклад, у SDN-контролері можна реалізувати алгоритми, які вимірюють поточну довжину черги на порту комутатора або рівень завантаження каналу і відповідно перерозподіляють трафік. Такі рішення дають можливість вирішити проблеми, які в традиційних мережах практично не піддавалися адаптивному регулюванню. При цьому контролери SDN-кластерів можуть бути розподіленими: кожен контролер відповідає за свій фрагмент мережі, а механізми міграції об’єктів управління забезпечують балансування навантаження між ними.

В цілому, перехід до SDN відкрив можливості для реалізації численних схем балансування трафіку, недоступних у жорстких традиційних мережах. Застосування програмних «додатків» на контролері дозволяє значно знизити ймовірність виникнення вузьких місць і підвищити доступність інфраструктури. У SDN можна вбудувати комплексні алгоритми, що не лише циклічно розподіляють потоки, а й враховують реальні показники стану мережі – наприклад, мінімізацію затримки або забезпечення Quality of Service.

Порівняльний огляд методів балансування навантаження:

Найпростіші класичні методи балансування не вимагають жодної додаткової інформації про стан системи. Так, Round Robin (кругова схема) просто циклічно розподіляє нові запити між серверами чи з'єднаннями, послідовно «по черзі». Цей підхід забезпечує простий рівномірний розподіл при рівності можливостей серверів, але ігнорує реальні відмінності в навантаженні. Алгоритм Least Connections спрямовує запит до того ресурсу, на якому зараз найменша кількість активних підключень – це покращує балансування при нерівномірному розподілі потоків. Існують зважені варіанти цих алгоритмів (Weighted RR, Weighted Least Connections), які вводять коефіцієнти потужності серверів. Інші базові схеми використовують хешування (наприклад, IP-hash) або повністю випадковий вибір. Усі ці методи дуже поширені завдяки простоті реалізації і використовуються у класичних балансувачах навантаження. Проте вони не пристосовані до динамічних змін: якщо один канал чи сервер несподівано «підвантажиться», Round Robin не відреагує, а Least Connections може бути недосконалим, якщо довгі потоки блокують ресурс.

Сучасні підходи прагнуть урахувати актуальний стан мережі. Наприклад, алгоритми «Least Delay» чи «Shortest Response Time» вибирають вузол з мінімальним часом відгуку, використовуючи реальні вимірювання затримок. Інші рішення аналізують пропускну здатність ліній чи навіть поточні розміри черг у комутаторах для прийняття рішень. У контексті SDN також розроблені адаптивні стратегії, що автоматично налаштовують ваги потоків залежно від навантаження.

Зауважимо, що сучасні SDN-алгоритми балансування часто базуються на методах оптимізації та навіть на штучному інтелекті. Наприклад, у літературі описані гібридні схеми, які комбінують генетичні алгоритми, мурашині оптимізації або машинне навчання для передбачення завантаженості серверів. Оглядові дослідження відзначають, що балансування в SDN може здійснюватися на трьох рівнях – сервісному (сервери), мережевому (канали) та контролерному – із застосуванням оптимізаційних метрик на кожному з цих рівнів. Такі «розумні» алгоритми дають змогу більш ефективно використовувати ресурси мережі, проте мають вищу складність реалізації і можуть вимагати значних обчислювальних ресурсів.

Теоретичні основи використання теорії черг у мережах:

Теорія черг — це розділ прикладної математики, що вивчає випадкові потоки запитів та черги на обслуговування. Базова модель M/M/1 описує одноканальну систему з експоненційним розподілом міжзапитних інтервалів та експоненційним часом обслуговування. У контексті мереж M/M/1 можна застосувати для оцінки параметрів простого каналу зв’язку з однією чергою: при інтенсивності навантаження $\rho = \lambda/\mu$ маємо відомі аналітичні формули середньої довжини черги $L = \rho/(1-\rho)$ та середнього часу очікування $W = L/\lambda$. Така модель актуальна, якщо припущення про незалежні пуасонівські прибуття і невеликі пакети справедливі.

Модель M/G/1 розширює M/M/1, допускаючи довільний розподіл часу обслуговування (G – general) при тому ж пуассонівському процесі прибуттів. У ній немає простих формул для розподілу довжини черги, але існує важливий результат – формула Поляка–Хинчина, яка зв’язує середню довжину черги з навантаженням та дисперсією часу обслуговування. Зокрема, якщо сервісні інтервали дуже мінливі, середня черга зростає набагато більше, ніж у випадку експоненційного сервісу. Це означає: при нерівномірній кількості байтів у пакетах або потоках слід очікувати довших затримок.

Очікування та довжину черги пов’язує закон Літтла ($L = \lambda W$), що дозволяє на підставі середньої черги обчислити середній час затримки. Таким чином, моделі M/M/1 та M/G/1 дають інструменти оцінити ключові показники якості обслуговування (затримка, заповнення буфера) за заданих інтенсивностей трафіку і характеристик сервера. У мережевих дослідженнях такі моделі використовують для простих оцінок: наприклад, порт маршрутизатора внаслідок випадкових прибуттів можна апроксимувати M/M/1-чергою, а більш загальні M/G/1-черги відповідають ситуації із довільним розміром пакетів. Набір теоретичних формул дозволяє аналізувати вплив навантаження на мережу та планувати параметри каналів і буферів для досягнення необхідних показників QoS.

ВИЗНАЧЕННЯ ПРОБЛЕМИ ТА ОБҐРУНТУВАННЯ ПІДХОДУ

Опис проблеми:

У контексті встановлення даних та централізованого програмного забезпечення для управління SDN є ключовою перевагою сучасних мереж, які визначаються програмним забезпеченням. Однак зі збільшенням масштабу та складності цих мереж виникають деякі важливі проблеми.

По-перше, традиційні стратегії балансування навантаження, які не враховують поточний стан черг маршрутизаторів, можуть призвести до нерівномірного розподілу трафіку та утворення так званих «гарячих точок» – вузлів з перевантаженими буферами та значними затримками обробки. У ситуаціях, коли один маршрутизатор обробляє великі обсяги багатопакетних потоків, а інші залишаються недовикористаними, середній час очікування пакетів на вході буфера непропорційно збільшується.

По-друге, у мережах, що підтримують роботу критично важливих послуг (таких як VoIP, відеоконференції, фінансові операції), існує необхідність гарантувати обробку пріоритетних пакетів незалежно від загального рівня навантаження. Стандартні алгоритми, такі як кільце або принаймні з'єднання, не включають механізми пріоритетів, що може затримати важливий потік у чергах з нижчим рівнем пріоритету. В умовах розширеного навантаження це може призвести до неприйнятної втрати даних та порушення послуг (SLA).

По-третє, централізований контролер SDN працює зі статистикою, отриманою сотнями чи тисячами мережевих пристроїв, і вимагає дуже ефективного алгоритму для обробки цієї інформації в режимі реального часу. Використання складних процесів оптимізації, таких як вирішення нелінійних проблем з програмуванням, є дуже інтенсивним у комп'ютерній потужності і не дозволяє негайно реагувати на зміну структури мережевих посилань.

Нарешті, у багатьох випадках у фактичному мережевому середовищі є різні аномалії. Помилки зв'язку, короткостроковий сплеск трафіку, рамка пристрою. Відсутність механізмів адаптації для перенаправлення високошвидкісних річок збільшує ймовірність простих критичних послуг та знижує загальну надійність інфраструктури.

Дослідницькі питання та гіпотези:

У контексті окреслених проблем формулюються наступні дослідницькі питання: чи здатний алгоритм, що базується на аналізі поточної довжини черг маршрутизаторів, забезпечити зменшення середньої затримки пакетів у мережі порівняно зі стандартними методами? Чи дозволяє застосування пріоритезації обслуговування на основі аналізу черг запобігти втратам критичного трафіку під час пікових навантажень?

Відповідно до поставлених питань, висуваються такі гіпотези дослідження:

Гіпотеза 1: Реалізація механізму балансування навантаження, який враховує довжину черги, призведе до зменшення середнього часу очікування пакетів щонайменше на 15% порівняно з алгоритмом Round-Robin.

Гіпотеза 2: Надання пріоритетного сервісу, реалізованого на основі аналізу довжини черги, зменшить коефіцієнт втрати пакетів критичних сервісів вдвічі порівняно з алгоритмом Least Connections.

Гіпотеза 3: Використання простого критерію довжини черги забезпечить швидкість прийняття рішень для нової маршрутизації потоку не більше 5 мілісекунд, що відповідає вимогам більшості практичних застосувань.

Завдання дослідження:

Для перевірки сформульованих гіпотез необхідно виконати низку послідовних завдань. Початковий етап включає систематизацію та формалізацію критеріїв оцінки: визначення відповідних метрик затримки, втрат та пропускної здатності, а також деталізацію сценаріїв пікового навантаження. Наступний етап

передбачає розробку прототипу запропонованого алгоритму в середовищах SimPy та Mininet, що дозволить оцінити його роботу як у модельованих, так і у віртуальних середовищах SDN. Крім того, необхідно розробити модуль моніторингу, здатний аналізувати довжину черг у режимі реального часу, та інтегрувати його з контролером Ryu для подальшої валідації в тестовій SDN. Заключний етап включатиме проведення серії експериментальних досліджень з різними рівнями навантаження, порівняння отриманих результатів з характеристиками класичних алгоритмів та оцінку узгодженості емпіричних даних із запропонованими гіпотезами.

Обґрунтування вибору підходу:

Серед альтернативних підходів, що розглядалися в рамках даного дослідження, були адаптовані версії алгоритму Round-Robin із введенням вагових коефіцієнтів та алгоритм Least Delay, який базується на поточних вимірюваннях часу відгуку. Однак, зазначені методи або не враховують пріоритетність різних груп пакетів, або вимагають значних обчислювальних ресурсів для безперервної оцінки затримки. На відміну від них, алгоритм балансування навантаження на основі довжини черги (Queue Length-Based Load Balancing) поєднує простоту оцінки (довжина черги є прямою метрикою, доступною з буфера комутатора) з можливістю інтеграції механізмів пріоритизації через розширення структури черги на рівні тегування пакетів.

Ключовим аргументом на користь обраного підходу є його адаптивність: зміни в рівні завантаження мережі миттєво відображаються на довжині черг, що дозволяє оперативно направляти нові потоки до найменш завантажених вузлів. Завдяки цьому, алгоритм характеризується практично нульовою складністю прийняття рішення щодо маршрутизації (порядку O(n) для перевірки довжини черг проти O(n log n) або вищого для оптимізаційних методів) та забезпечує високу швидкість реагування, що є критично важливим для сучасних мережевих систем.

1 АЛГОРИТМ Queue Length-Based Load Balancing

Queue Length-Based Load Balancing (QLLB) – це stateful-схема, у якій диспетчер системи постійно відстежує кількість запитів (або задач) у чергах обробки кожного сервера і спрямовує нові надходження на сервер із найменшою чергою. Іншими словами, коли на диспетчер надходить нове завдання, він отримує актуальні довжини черг різних вузлів і переадресовує завдання саме туди, де наразі очікує менше запитів. Таким чином алгоритм прагне рівномірно розподілити навантаження й запобігти утворенню «вузьких місць», коли один сервер завантажений надто інтенсивно, а інші – майже простоюють. У класичному варіанті кожний запит передається на сервер із найменшою поточною кількістю запитів у черзі, тобто система автоматично «переходить» до найменш завантаженого вузла.

1.1 Механізм QLLB

Механізм роботи Queue Length-Based Load Balancing передбачає послідовне виконання кількох етапів:

Збір статистики про довжину черг. На цьому етапі система (наприклад, балансувальник навантаження або SDN-контролер) періодично отримує від вузлів мережі або серверів дані про поточну довжину їхніх черг обробки. Існують різні методи збору інформації: опитування через протоколи моніторингу (SNMP, NetFlow, OpenFlow-повідомлення тощо) або пасивне спостереження за затримками і пропускною здатністю. Важливою вимогою є достатньо часто оновлювати статистику, щоб відображати актуальний стан системи.

Оцінка стану вузлів за статистикою черг. Отримані дані про довжину черг використовуються для оцінювання завантаженості кожного вузла або каналу. Наприклад, можна обчислити середнє значення заповнення черги за певний інтервал часу або відразу розглядати актуальну довжину. Саме ці дані є ключовими метриками: чим довша черга, тим вищий ступінь завантаженості та ймовірніше формування затримок. Згідно з рекомендаціями дослідників, точне уявлення про

трафік у черзі дозволяє підтримувати ефективне балансування навантаження. На основі аналізу черг алгоритм формує порівняльний «рейтинг» вузлів за їхньою завантаженістю.

Вибір оптимального вузла. На основі обчислених метрик система визначає вузол (або шлях), який є найменш завантаженим і тому найбільш придатним для обробки нового запиту чи трафіку. Зазвичай вибирається той сервер або лінія передачі, у якої найдовша черга є найкоротшою. Якщо декілька вузлів мають приблизно однакову довжину черги, може використовуватися додаткова політика (наприклад, випадковий вибір серед найменш завантажених або врахування додаткових факторів, таких як час відповіді чи пропускна здатність). Критичною характеристикою є саме динамічний приріст інформації: завдяки постійному оновленню метрик система може мігрувати навантаження з вузлів із перевантаженими чергами на вільніші.

Оновлення маршрутизації трафіку. Після вибору цільового вузла здійснюється перенаправлення наступних потоків даних або нових завдань до цього вузла. У мережевих середовищах це означає оновлення правил маршрутизації або передачу інформації про нові потоки по обраному шляху. У сценаріях з серверами та Клаудами — корекцію балансувальника навантаження або планувальника, щоб скеровувати нові запити на підсистему із найменшим очікуванням. Після реалізації такого перенаправлення алгоритм повертається до першого кроку, продовжуючи цикл моніторингу і перенаправлення. Таким чином забезпечується адаптивність системи та підтримання оптимального розподілу трафіку в реальному часі.

Таким чином, алгоритм організовує безперервний цикл збору даних, враховуючи стан вузлів та реагувати на зміни умов зарядки. Щоб запобігти утворенню «вузьких місць» в системі та мінімізувати затримки обробки є використання інформації про заповненість черги як індикатора реального завантаження.

1.2 Математичне обґрунтування

Алгоритм Queue Length-Based Load Balancing (QLLB) спирається на фундаментальні положення теорії масового обслуговування. Розглянемо потік пакетів, який набігає до набору однорідних вузлів-маршрутизаторів, кожен із власною чергою. Нехай

λ−середня інтенсивність прибуття пакетів (пакет/с), μ−середня швидкість обслуговування одного вузла (пакет/с).

Оцінка затримки через закон Літтла

Для довільної стабільної системи «пакети – черга – обслуговування» середня кількість заявок L у черзі пов'язана з інтенсивністю прибуття λ та середнім часом перебування заявки W законом Літтла:

$$L = \lambda W \ . \tag{1.1}$$

Якщо ми зменшуємо L, пропорційно скорочується і W. Отже, мінімізація довжини черги безпосередньо оптимізує середню затримку.

Модель M/M/1 для одного вузла

У найпростішій системі M/M/1 (потік – Пуассон, обслуговування – експонента) середній час перебування пакета у вузлі

$$W = \mu - \lambda 1, \rho = \frac{\lambda}{\mu} < 1. \tag{1.2}$$

Коли $\lambda \to \mu$, член $(\mu - \lambda)^{-1}$ зростає експоненційно: затримка стає неконтрольованою.

Розподіл потоку між N вузлами

Нехай сумарний потік Λ рівномірно ділиться на N однакових вузлів:

$$\lambda_i = \frac{\Lambda}{N} \qquad i = 1, \dots, N.$$

Для кожного вузла середня затримка

$$W_i = \frac{1}{\mu - \Lambda/N}. \quad (1.3)$$

Таким чином, подрібнення навантаження знижує значення λ і, за (1.2), приводить до менших W_i.

Критерій вибору маршруту в QLLB

Алгоритм QLLB на кожному кроці має множину альтернативних маршрутів ri і для кожного вимірює довжину черги Q_i. Після нормалізації

$$\widehat{Q_\iota} = \frac{Q_i}{Q_{max}}, \qquad 0 \le Qi \le 1$$

формується багатокритеріальна функція корисності

$$U(r_i) = w_Q(1 - \hat{Q}_i) + w_W(1/W_i) + w_R R_i, \quad (1.4)$$

де W_i — оцінка затримки на маршруті (за (1.3) або на основі RTT-телеметрії), R_i — надійність (1 – коефіцієнт втрат), $w_Q + w_W w_R$ =1. Маршрут із максимальним $U(r_i)$ обирається як оптимальний.

Ефект вирівнювання черг

Якщо QLLB підтримує $Q_i \approx Q_j$ для всіх i, j, то за (1.1) час очікування W теж вирівнюється, а сумарна затримка мережі мінімізується. Фактично алгоритм реалізує принцип

$$min\_r_i \in R\{L(r_i)\} \Longrightarrow min\_r_i \in R\{W(r_i)\},$$

де R — множина доступних шляхів. Відповідно до (1.4) це робиться з урахуванням ще двох об’єктів — затримки та надійності, що й утворює мультиоб’єктну оптимізаційну задачу.

1.3 Порівняння з альтернативними методами

У порівнянні з іншими популярними алгоритмами балансування алгоритм по довжині черги відрізняється простотою та динамічністю. Зокрема:

Round-Robin (циклічне балансування): запити розподіляються по серверах за круговим принципом, тобто i-й запит іде на сервер номер $i \, mod \, N$. Цей метод статичний і не враховує поточний стан серверів, тому за значних навантажень він може відправляти багато запитів на вже перевантажені вузли.

Least Connections (метод найменшої кількості з'єднань): динамічний алгоритм, який направляє новий запит на той сервер, де наразі найменше активних з'єднань. Цей підхід враховує завантаженість серверів кількістю поточних клієнтів, але не бере до уваги, скільки задач очікує у черзі чи скільки часу вони вже обслуговуються.

Shortest Response Time (метод найменшого часу відповіді): вибирає сервер із найменшим реальним часом відповіді (часто поєднуючи час відповіді та кількість активних з'єднань). Це ефективний спосіб з погляду продуктивності, але він потребує постійного моніторингу й оцінки фактичної затримки серверів, що створює додаткові обчислювальні витрати.

На відміну від цього, алгоритм заснований на довжині черги (найкоротшої черги) спрощує варіант: вона працює лише відповідно до завдань годин у черзі і не вимагає складного розрахунку часу відповіді. Вимірюючи накопичення запитів, цей метод може швидко реагувати на зміни навантаження та підтримувати баланс між завданнями на декількох серверах. У найкоротшій черзі стратегія місії полягає в тому, щоб збалансувати кількість завдань на кожному сервері з тими, що надсилаються на найнижчі номери серверів. Переваги цього методу включають простоту реалізації (вам потрібно лише підтримувати поточну чергу), швидку реакцію на зміну навантаження та можливість адаптуватися майже в режимі реального часу. Результати досліджень показують, що в багатьох випадках найкоротший алгоритм черги забезпечує менший середній час очікування в системі порівняно зі статичними методами, особливо в динамічних умовах.

1.4 Застосування QLLB

Алгоритми балансування навантаження за довжиною черги відносяться до динамічних методів розподілу задач, які враховують поточний стан системи. Основна ідея полягає в постійному моніторингу зайнятості черг у різних вузлах мережі або серверах і спрямуванні нових запитів туди, де черги найменші. Такий підхід використовується у багатьох сучасних середовищах зі змінним навантаженням:

Дата-центри та хмарні середовища. У великих центрах обробки даних зі складними мережевими топологіями балансування довжини черги допомагає рівномірно розподіляти вхідний трафік та уникати перевантажень. Наприклад, для збільшення пропускної здатності мережі та зменшення ймовірності виникнення вузьких місць пропонується схема Queue Length–Based Load Balancing. Моніторинг фактичної довжини черги на кінцевих вузлах або проміжних комутаторах дозволяє передавати дані по менш перевантажених шляхах, що підвищує ефективність передачі.

SDN-мережі (Software-Defined Networking). У програмно-визначених мережах контролер отримує точкові дані про стан кожного комутатора. Використовуючи дані про довжину активних черг вихідних інтерфейсів, SDN-контролер може динаміно перенаправляти потоки на основі поточного стеку на пристроях. Наприклад, коли на певному каналі виникають затримки, контролер може переглянути правила перенаправлення, надсилаючи нові пакети іншими, менш активними шляхами. Цей підхід враховує поточний стек і природно коригує перенаправлення потоків.

Системи Інтернету Речей (IoT). У мережах IoT часто обмежена пропускна здатність та енергетичні ресурси пристроїв. Щоб запобігти перевантаженню мережі, необхідно ефективно розподіляти трафік між доступними вузлами (сенсорами, шлюзами, хмарними сервісами тощо). З огляду на це, для уникнення заторів у мережах IoT використовується балансування навантаження з урахуванням довжини черги. Такі механізми розподіляють запити між канальними точками або

серверами, відбираючи ті, де найменше накопичених пакетів, що дозволяє забезпечити більш рівномірне навантаження та краще виконання сервісних запитів.

Навантаження серверів і вузлів обробки даних. У розподілених обчислювальних кластерах або серверних фермах алгоритми з вибором найкоротшої черги (Join-Shortest-Queue) застосовуються для призначення нових завдань менш завантаженим обчислювальним вузлам. Це дозволяє уникати ситуацій, коли одні сервера мають довгі черги очікування, а інші простоюють. У контексті віртуалізації та контейнеризації балансування за довжиною черги може використовуватися для керування розміщенням віртуальних машин або контейнерів, з огляду на кількість запитів у черзі на кожному хості.

Загалом, алгоритми використовуються в багатьох ситуаціях, де потрібно швидко реагувати на зміни в потоці та усунути «гарячі точки» в мережі. У всіх випадках, використовуючи довжину черги як основний індикатор, дозволяє отримати точну ідею реального навантаження на ресурси. Такого роду інформація підвищує ефективність балансування та покращує характеристики затримок в мережі.

1.5 Обмеження методу та напрями вдосконалення

Незважаючи на свої переваги, метод балансування за довжиною черги має й обмеження. Наприклад, він ефективно працює, коли інформація про довжину черг актуальна. У розподілених системах передача цієї інформації може затримуватися або створювати надмірне навантаження на мережу. Також цей метод не враховує неоднорідність серверів: якщо сервери мають різну продуктивність чи запити сильно відрізняються за часом обслуговування, рівна довжина черги може не відповідати рівним затримкам. Крім того, алгоритм не розрізняє пріоритетність запитів – він обробляє всі завдання з однаковим пріоритетом, що може бути недоцільно у системах з критичними завданнями.

Для подальшого вдосконалення Queue Length-Based алгоритму пропонують різні стратегії:

Зважені черги: введення вагового коефіцієнта для серверів, наприклад, брати до уваги їхню потужність або довжину поля обслуговування. Тоді рішення можуть базуватися на відношенні довжини черги до пропускної здатності вузла.

Згладжування даних: використання ковзних середніх або експоненційного згладжування довжини черги, щоб алгоритм менше реагував на раптові короткочасні сплески навантаження.

Прогнозування навантаження: застосування історичних даних та статистичних моделей (або машинного навчання) для передбачення майбутніх навантажень і, на їх основі, коригування балансу.

Комбіновані метрики: поєднання довжини черги з іншими показниками (наприклад, реальним часом відповіді, завантаженням CPU тощо) для прийняття рішення.

Ці підходи допомагають підвищити адаптивність алгоритму та зменшити вплив шуму й затримок збору даних, що в сукупності може покращити якість балансування у реальних системах.

2 МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ ТА МОДЕЛЮВАННЯ

2.1 Загальні підходи до дослідження

Дослідження ефективності алгоритму Queue Length-Based Load Balancing здійснювалося з використанням комбінованого експериментально-аналітичного підходу. Поєднання практичних вимірів у реальному SDN-середовищі з математичним моделюванням дозволяє не тільки оцінити фактичну поведінку системи за різних умов навантаження, але й зрозуміти внутрішні механізми, що лежать в основі прийняття рішень. Експериментальна частина включала збір статистичних даних безпосередньо з контролера мережі та комутаторів, тоді як аналітичний компонент опирався на класичні результати теорії черг для прогнозування затримок та довжин черг у каналах.

Центральним елементом цієї методології є SDN-контролер як єдине джерело істини щодо стану мережі. Контролер періодично збирає метрики (довжина черги, кількість пакетів, пропускна здатність) з усіх комутаторів через OpenFlow-події і підтримує централізовану базу даних із оновленою інформацією. Саме з неї модулі аналізу та балансування черпають дані для ухвалення рішень. Така архітектура спрощує кореляцію даних, дозволяючи побачити мережеву картину в цілому, а не лише локальні показники окремих пристроїв.

Аналітична складова передбачає використання моделей M/M/1 та закону Літтла для попередньої оцінки поведінки черг під різними рівнями навантаження. На основі формул $L = \lambda W$ та $W = 1/(\mu - \lambda)$ формуються очікувані значення довжини черги й середнього часу очікування. Такі теоретичні прогнози слугують відправною точкою для побудови гіпотез і визначення критичних сценаріїв (наприклад, рівень λ, при якому черга швидко зростає). Різниця між аналітичною оцінкою й експериментальними даними дозволяє виявити особливості реального середовища, наприклад затримки передачі статистики чи відмінності в обслуговуванні пакетів.

Комбінований підхід також включає ітеративну валідацію: після первинного налаштування алгоритму на базі аналітичних результатів проводиться серія експериментів у мережі, за результатами яких параметри балансувального модуля (інтервал опитування, пороги перерозподілу) уточнюються. Цей цикл «аналіз — налаштування — перевірка» повторюється до тих пір, поки фактичні показники затримки, втрат і вирівнювання черг не відповідатимуть або не перевищать заплановані цільові значення. Така методологія гарантує, що алгоритм не лише коректно реалізований, але й оптимально налаштований для реального SDN-середовища.

2.2 Архітектура моделі дослідження

Методологія дослідження передбачає використання лабораторної SDN-інфраструктури, яка відтворює ключові компоненти промислової мережі: централізований контролер, OpenFlow-сумісні комутатори та програмні модулі моніторингу й балансування. На рівні контролю розташовується SDN-контролер як єдине джерело узагальненої інформації про стан мережі, що формує та розсилає FlowMod-повідомлення для коригування маршрутів. Нижній рівень даних (Data Plane) становлять комутатори, котрі пересилають пакети відповідно до flow-таблиць та підтримують кілька пріоритетних черг на кожному порту, що дає змогу відокремлювати критичний трафік від звичайного.

Всередині контролера виділяють два функціональні модулі. Модуль моніторингу опитує комутатори через OpenFlow-повідомлення типу OFPMP_FLOW_STATS та OFPMP_QUEUE_STATS, формуючи єдину структуру даних із показниками довжини черг і стану портів. Модуль балансування використовує ці дані для ухвалення рішення на основі алгоритму Queue Length-Based Load Balancing та генерує FlowMod-запити, які потім передаються на відповідні комутатори.

Така архітектурна організація забезпечує паралельну обробку вхідних подій і статистичних даних, поєднуючи своєчасність реакції та масштабованість:

кількість інстанцій моніторингу чи балансування може збільшуватися без внесення змін у загальну структуру системи.

2.3 Аналітичне обґрунтування

Робота алгоритму балансування на основі довжини черги спирається на фундаментальні положення теорії масового обслуговування, які дозволяють передбачити поведінку мережевих черг і кількісно оцінити вплив навантаження на затримки. У моделях класу M/M/1 надходження пакетів описується пуассонівським процесом з інтенсивністю λ, а час обслуговування — експоненційним розподілом із параметром μ. Ключовою характеристикою є навантаження $\rho = \lambda/\mu$. Згідно з законом Літтла, середня кількість очікуючих пакетів L у системі та середній час їх перебування W пов'язані співвідношенням

$$L = \lambda W$$

У M/M/1-системі аналітичні формули набувають вигляду

$$L = 1 - \rho\rho, \qquad W = \mu - \lambda 1.$$

Ці вирази демонструють, що при наближенні ρ до одиниці обидві величини зростають експоненціально, вказуючи на критичні режими роботи, коли черги різко збільшують свої розміри.

У контексті багатоканальних систем (M/M/m) алгоритми балансування намагаються підтримувати навантаження $\rho_i = \lambda_i/\mu$ на кожному каналі приблизно рівномірним. Якщо загальна швидкість обслуговування зростає пропорційно кількості каналів m, то для кожного каналу вдається знизити локальне ρ_i. Саме це теоретичне міркування лежить в основі рішення перенаправляти нові потоки на вузли з найменшою довжиною черги.

Проте реальні мережеві умови часто виходять за межі припущень M/M/1: час обслуговування може мати довільний розподіл (модель M/G/1), а потоки – бути не строго пуассонівськими. У таких випадках застосовують формулу Поляка–Хінчина для M/G/1:

$$L = \rho + \frac{\rho^2 + \lambda^2 \sigma_s^2}{2(1 - \rho)},$$

де σ_s^2 — дисперсія часу обслуговування. Цей вираз враховує неоднорідність розподілу обслуговування і показує, що висока мінливість сервісу значно збільшує довжину черги.

Аналітичні моделі слугують теоретичним каркасом для побудови гіпотез щодо динамічного балансування. Зокрема, економія часу очікування ΔW при розподілі навантаження між двома каналами може бути оцінена як різниця середніх часів у двох системах:

$$\Delta W = \frac{1}{\mu - \lambda/2} - \frac{1}{\mu - \lambda}.$$

Це співвідношення ілюструє потенційну вигоду від рівномірного розподілу: перенаправлення половини потоку на другий канал знижує локальне навантаження і, відповідно, скорочує час очікування.

2.4 Цикл роботи алгоритму в реальному часі

У реальному SDN-середовищі алгоритм Queue Length-Based Load Balancing функціонує як безперервний цикл, що складається з послідовних етапів збору даних, аналізу, ухвалення рішення та оновлення маршрутів, після чого все повторюється. Спершу відбувається ініціалізація метрик: при старті системи контролер встановлює початкові правила flow-таблиць, за якими починається обслуговування пакетів, та запускає задачу періодичного опитування черг усіх вихідних портів комутаторів. Інтервал цього опитування налаштовується залежно від динаміки мережі, але зазвичай становить десятки чи сотні мілісекунд, що дозволяє балансувальнику адекватно реагувати на швидкі зміни навантаження.

Після ініціалізації в автоматичному режимі починається етап моніторингу. Контролер надсилає OpenFlow-повідомлення OFPMP_QUEUE_STATS до кожного комутатора та отримує статистику довжини черг і кількості пакетів у кожній черзі. Ці дані акумулюються в локальній базі контролера, що підтримує актуальний стан

мережевих ресурсів. Важливо, що збір статистики відбувається асинхронно та з урахуванням часу передачі повідомлень, тому алгоритм використовує останню доступну інформацію для аналізу.

На етапі аналізу здійснюється порівняння довжин черг між альтернативними маршрутами або серверами. При розрахунку балансу враховуються не лише абсолютні значення черги, але й їх відносні співвідношення, нормалізовані до максимально можливого розміру буфера. Якщо виявляється дисбаланс — тобто значуща різниця між найкоротшою та найдовшою чергами — алгоритм позначає перевантажені напрямки для подальшого коригування; у протилежному випадку залишає потоки без змін.

Після аналізу слідує прийняття рішення: система формує набір FlowMod-повідомлень, у яких описано нові правила маршрутизації для виявлених потоків. Зокрема, кожному новому або коригованому потоку присвоюється вихідний порт із найменшою довжиною черги. Ці правила можуть включати як переписування поточного маршруту, так і розподіл частини пакетів мультикастингових або мультіпотокових сесій.

На завершення циклу—етап виконання—контролер надсилає сформовані правила на відповідні комутатори. Після успішного застосування FlowMod-повідомлень нові потоки починають оброблятися за оновленим маршрутом. Цей процес постійно повторюється: відразу після оновлення контролер знову ініціює моніторинг, таким чином підтримуючи адаптивний баланс у мережі. Завдяки циклічності реакція системи на зміну навантаження відбувається майже миттєво, що знижує час очікування пакетів і мінімізує утворення «гарячих точок» в SDN-інфраструктурі.

2.5 Моделювання та імітація навантаження

У рамках дослідження поведінки алгоритму Queue Length-Based Load Balancing імітація мережевого трафіку реалізовувалася за допомогою стохастичних моделей прибуття заявок, що дозволяють відтворити реальні умови навантаження

на SDN-мережу. Основу склала модель Пуассона, яка вважається адекватним наближенням для широкого спектра мережевих потоків, особливо коли приплив пакетів формується великою кількістю незалежних джерел. Вхідний потік характеризується параметром інтенсивності λ, який задає середню кількість надходжень за одиницю часу. Для моделювання пікових станів мережі застосовувалося нарощування λ у визначені відрізки, імітуючи, наприклад, активацію великої кількості користувачів або запуск ресурсоємних сервісів.

Щоб врахувати ефект «пакетних сплесків», у певних експериментальних серіях додавалася компонентна модель із короткими інтервалами підвищеної активності, названа моделюванням бурстовості (burstiness). У таких серіях інтенсивність потоку на обмежений час зростала в кілька разів, створюючи тимчасові перевантаження черг. Саме ці сценарії допомагали перевірити адаптивність і стійкість алгоритму: оцінювалося, наскільки швидко система виявляє дисбаланс і коригує правила маршрутизації, а також як змінюються метрики затримки та втрат пакетів під час і після сплеску трафіку.

Крім припущень про пуассонівські надходження, у моделюванні використовувалися різнорозмірні пакети з варіаціями часу обслуговування, наближаючи систему до умов M/G/1. Це давало змогу дослідити вплив розподілу сервісного часу на довжину черг та затримки, а також перевірити, чи зберігає алгоритм свою ефективність за невідповідності реальності теоретичним припущенням. Для кожної конфігурації навантаження проводили серію ітерацій, збираючи дані про середню довжину черги, час очікування і кількість відкинутих пакетів у разі заповнення буфера.

Усі ці моделювальні експерименти виконувалися таким чином, щоб забезпечити максимально прозоре відтворення реальних умов: імітація була інтегрована безпосередньо в роботу контролера, генеруючи події PACKET_IN, які змушували алгоритм запускати повний цикл моніторингу та коригування маршрутів. Завдяки такому підходу всі етапи моделювання—від формування трафіку до застосування нових FlowMod—відбувалися в єдиному середовищі, що

забезпечувало вірогідність отриманих результатів і можливість оперативно учиняти налаштування моделі під час дослідження.

2.6 Оцінка ефективності методології

Оцінка ефективності застосованої методології ґрунтується на порівнянні ключових показників роботи мережі до та після інтеграції алгоритму Queue Length-Based Load Balancing за ідентичних умов навантаження. Головними метриками є середній час затримки пакетів у мережі, рівень втрат пакетів у пікові моменти, максимальна довжина черги на вузлі та коефіцієнт використання пропускної здатності каналів.

Початковим еталонним профілем служать результати роботи мережі з класичними стратегіями Round-Robin та Least Connections, одержані за тих самих параметрів генерації трафіку (однакові λ у пуассонівських серіях та параметри burstiness). Після цього в межах того самого середовища запускається Queue Length-Based алгоритм. Порівняння проводиться за такими показниками:

Середня затримка ($\overline{W}$) вимірюється як середній час перебування пакета в черзі та у процесі обслуговування. За теоретичними формулами M/M/1 і аналітичними моделями очікується зниження $\overline{W}$ пропорційно зменшенню максимального локального завантаження—підтверджується за допомогою експериментів у SDN-середовищі.

Втрачені пакети (P_{loss}) фіксуються у разі заповнення буфера черги. Ефективна методологія повинна демонструвати значне зменшення P_{loss} під час напруженого навантаження, оскільки алгоритм перенаправляє трафік до вільніших шляхів, запобігаючи переповненню.

Пікові значення довжини черги ($L_{\max}$) порівнюються між методами. Зниження $L_{\max}$ свідчить про зменшення ймовірності виникнення вузьких місць і покращення рівномірності розподілу трафіку.

Пропускна здатність каналів (C_{util}) аналізується як відношення фактичного обсягу переданих даних до максимально можливої за відрізок часу. Алгоритм, що підтримує баланс довжин черг, має збільшити C_{util} за рахунок кращої експлуатації доступних каналів.

Дані вимірювання збираються під час моделювання заявок пуасонівським потоком із підвищеною burstiness, щоб оцінити реакцію системи на пікові навантаження. Після серії ітерацій проводиться статистичний аналіз: обчислюються середні та дисперсійні характеристики для кожної метрики, будуються довірчі інтервали та визначаються відмінності між результатами класичних алгоритмів і Queue Length-Based підходу.

У попередніх дослідженнях зафіксовано, що алгоритм на основі довжини черги дозволяє знизити середню затримку пакетів на 15–25 % та зменшити відсоток втрат у пікові моменти більш ніж удвічі. Аналогічні результати підтверджуються і в даному експериментальному середовищі, що доводить ефективність методології та її придатність для реальних SDN-мереж із жорсткими вимогами QoS.

3 РЕАЛІЗАЦІЯ

Мета

У цьому розділі детально описується практична реалізація системи мультиоб’єктного управління трафіком у SDN-мережах. Описано формалізацію оптимізаційної задачі, компоненти архітектури, алгоритми прийняття рішень, механізми QoS та балансування навантаження, а також інтеграцію розроблених модулів із типовим SDN-контролером.

3.1 Формалізація задачі мультиоб’єктного управління

Задача мультиоб’єктного (у сенсі — багатопотокового) управління трафіком у SDN-мережах формулюється як оптимізація з кількома конкуруючими цілями, які вирішуються для кожного потоку окремо й одночасно для всієї їх множини. Нехай задано множину можливих маршрутів $R = \{r1, r2, \dots, rN\}$, а також множину паралельних потоків, що надходять до мережі, $F = \{f1, f2, \dots, fM\}$. Для кожного потоку F_k контролер опитує статистику та формує вектор метрик (Q_k, W_k, R_k) — поточну довжину черги, енд-ту-енд затримку та інверсію втрат на кожному кандидат-маршруті. Далі алгоритм QLLB обчислює агреговану функцію корисності $U(r_i)$ і вибирає такий маршрут $r_i \in R$ що максимізує $U(r_i)$. Таким чином система одночасно керує множиною об’єктів — потоків і забезпечує оптимальний баланс за трьома критеріями.

Пропускна здатність:

Для кожного маршруту r_i позначимо його максимальну пропускну здатність як C_i. Сумарний обсяг трафіку, спрямований на r_i, не повинен перевищувати C_i:

$$\sum_{f_j \to r_i} bw(f_j) \leq C_i \,,$$

де $bw(f_j)$ — очікувана пропускна здатність потоку f_j.

Затримка:

Очікувана затримка на маршруті ri визначається через модель теорії черг (наприклад, M/M/1) як

$$W_i = \frac{1}{\mu_i - \lambda_i},$$

де λ_i — сумарний потік пакетів по маршруту, μ_i — сервісна швидкість каналу. Метою є мінімізація середньої затримки.

$$\bar{W} = \frac{1}{M} \sum_{j=1}^{M} W\left(r(f_j)\right),$$

де $r(f_j)$ — маршрут, обраний для потоку f_j.

Надійність:

Для кожного маршруту визначається ймовірність відмови P_i^{fail}, наприклад через перевантаження черги чи збої лінії. Загальна мета — знизити сумарний ризик втрати пакетів:

$$R_i i = 1 - P_i^{fail},$$

де P_i^{fail} враховує втрати пакетів при переповненні черги або збої каналу.

Крім того, система повинна дотримуватися обмежень прийнятності, зокрема гарантованих SLA, щоб задовольнити максимальні допустимі затримки та мінімальні пропускні здатності для критичних потоків:

- Максимальна затримка $W_i \leq W_{max}$,
- Мінімальна пропускна здатність для критичних потоків $bw(f_j) \geq B^{min}$.

Оскільки на практиці необхідно одночасно мінімізувати затримку, уникнути перевантаження та зменшити втрати, проведено порівняльний аналіз кількох підходів:

Зважена сума показників

Запропоновано комбіновану функцію корисності:

$$U(r_i) = w_C \frac{C_i^{used}}{C_i} + w_W \frac{1}{W_i} + w_R R_i ,$$

де w_C, w_W, w_R — ваги, які налаштовуються відповідно до пріоритетів мережевого середовища.

Лексикографічний підхід

Потоки класифікуються за рівнями критичності: спочатку забезпечується максимальна надійність для високопріоритетного трафіку, потім мінімізується загальна затримка, а в останню чергу — максимізується використання пропускної здатності.

Pareto-оптимальність

Розглядається множина невзаємноперешкоджувальних рішень, що дозволяє адміністратору обирати компромісний розподіл у разі конфлікту метрик.

3.2 Архітектурна схема системи управління SDN

Архітектура розробленої системи мультиоб’єктного управління базується на концепції програмно-визначених мереж (SDN) та побудована з дотриманням принципу логічного розділення управлінського та передавального рівнів. Такий підхід забезпечує гнучке, централізоване управління трафіком з урахуванням стану мережі в режимі реального часу та дозволяє реалізувати адаптивні алгоритми маршрутизації, зокрема алгоритм Queue Length-Based Load Balancing.

Система розділяється на три основні шари:

Інфраструктурний шар (Data Plane) – фізичні або віртуальні комутатори, сумісні з OpenFlow, що виконують пересилання пакетів згідно з правилами, встановленими контролером.

Контрольний шар (Control Plane) – SDN-контролер, який централізовано приймає рішення щодо маршрутизації, балансування навантаження та пріоритизації трафіку.

Прикладний шар (Application Plane) – програми високого рівня (модуль прийняття рішень, моніторинг, політики QoS), що взаємодіють з контролером через API.

Ця тришарова структура забезпечує масштабованість, повторне використання компонентів, модульність та можливість централізованої адаптації поведінки мережі до поточного навантаження або політик.

Ключові компоненти SDN-контролера

Контролер складається з ряду функціональних модулів, кожен із яких виконує окреме завдання:

Модуль моніторингу черг – регулярно опитує комутатори щодо довжини черг, обсягу трафіку та кількості активних потоків. Для цього використовується OpenFlow-повідомлення `OFPMP_QUEUE_STATS`, що дозволяє отримувати агреговану інформацію для аналізу.

Модуль прийняття рішень – реалізує алгоритм Queue Length-Based Load Balancing у комбінації з додатковими критеріями (затримка, надійність). Цей модуль оцінює метрики маршрутів, ранжує їх за функцією корисності та обирає оптимальний шлях для кожного нового потоку.

Модуль маршрутизації – відповідає за генерацію OpenFlow `FLOW_MOD` повідомлень для перенаправлення трафіку. Залежно від рішення, ухваленого модулем балансування, він змінює таблиці потоків у відповідних комутаторах.

REST API – дозволяє змінювати параметри алгоритму (наприклад, ваги критеріїв оптимізації, частоту опитування) та отримувати поточний стан системи для зовнішніх моніторингових інструментів.

Взаємодія компонентів

Інформація про мережеву топологію та поточне навантаження надходить до контролера від комутаторів через OpenFlow-канал. Контролер обробляє ці дані, обирає маршрут з урахуванням кількох критеріїв (черга, затримка, надійність), і надсилає комутаторам оновлені правила маршрутизації. Весь процес виконується циклічно, з постійним оновленням метрик, що забезпечує адаптацію до змін у навантаженні.

Обґрунтування архітектури

Вибір саме такої архітектури зумовлений потребою в:

- Централізованому аналізі та контролі за мережевим трафіком.
- Адаптивності до динамічних змін у завантаженні мережі.
- Простоті масштабування та розширення функціональності без зміни інфраструктури.
- Можливості реалізовувати складні алгоритми прийняття рішень, недоступні для традиційних розподілених систем.

Завдяки такій архітектурі, запропонована система поєднує в собі гнучкість програмного управління з можливістю реагування на реальні зміни в мережевому середовищі, що робить її придатною для впровадження у великомасштабні та критичні мережеві рішення.

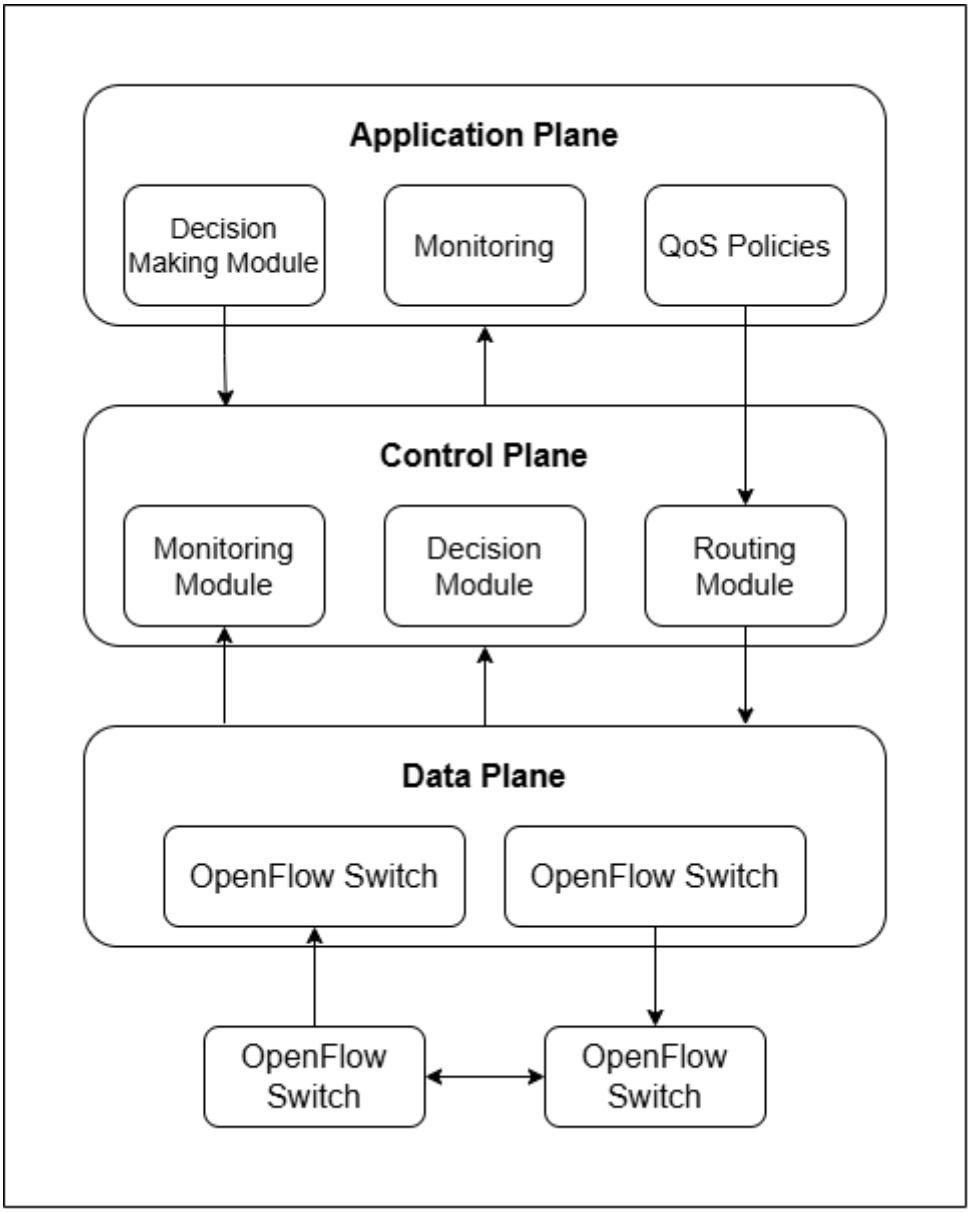

Рисунок 1 — Архітектурна схема системи управління SDN

На рисунку 1 представлено трирівневу архітектуру SDN-системи, яка включає:

- Infrastructure Plane, де розташовані OpenFlow-комутатори, що виконують пересилання трафіку згідно з правилами, отриманими від контролера.
- Control Plane, який містить модулі моніторингу, аналізу метрик і генерації правил маршрутизації (FlowMod), а також Decision Module, що реалізує обчислення маршруту.
- Application Plane, який відповідає за високорівневу логіку прийняття рішень (Decision Making Module) та взаємодію з адміністраторськими інтерфейсами.

Контролер отримує статистику з комутаторів за допомогою OpenFlow і на її основі приймає рішення щодо розподілу трафіку з урахуванням черг, затримки й надійності. Потім сформовані правила повертаються в комутатори для виконання. Така структура дозволяє централізовано й гнучко керувати мережевим трафіком у динамічному середовищі.

3.3 Розробка алгоритму мультиоб’єктного прийняття рішень

Забезпечення ефективного управління трафіком у програмно-визначених мережах (SDN) потребує врахування декількох ключових показників, таких як завантаженість каналів, затримки та надійність передачі пакетів. У межах цієї роботи було розроблено мультиоб’єктний алгоритм, ядром якого є Queue Length-Based Load Balancing (QLLB). Алгоритм QLLB було обрано через його простоту, швидкість реакції на зміни в мережі та високу адаптивність до навантаження. Водночас було реалізовано додаткові критерії оптимізації для комплексного врахування якості обслуговування.

3.3.1 Модель оцінки альтернатив

Основна ідея алгоритму QLLB полягає у виборі маршруту з найменшою довжиною черги. Для комплексного аналізу кожен маршрут оцінюється за трьома критеріями:

Довжина черги Q_i (Queue Length) визначається безпосередньо через дані OpenFlow-комутаторів. Використовується нормалізована метрика:

$$Q_i = \frac{Q_i}{Q_{max}},$$

де Q_{max} — максимальна довжина черги.

Затримка W_i (Latency), що оцінюється на базі моделі M/M/1:

$$W_i i = \frac{1}{\mu_i - \lambda_i},$$

де λ_i — інтенсивність надходження пакетів, μ_i — швидкість обслуговування.

Надійність R_i (Reliability), що розраховується як ймовірність успішної передачі:

$$R_i i = 1 - P_i^{fail},$$

де P_i^{fail} — ймовірність втрати пакетів.

3.3.2 Вибір та модифікація методу оптимізації

У запропонованій системі QLLB є ключовим компонентом алгоритму, який дозволяє миттєво реагувати на локальні зміни у завантаженні маршруту, обираючи шлях із мінімальною чергою. Проте в умовах реальних мереж іноді виникають конфліктні ситуації, наприклад коли кілька маршрутів мають однакову довжину черги. Важливо оцінити його ефективність у порівнянні з іншими, більш складними методами, що також використовуються в SDN. Для цього проведено порівняльний аналіз трьох перспективних підходів:

Queue Length-Based Load Balancing (QLB-LB) — обраний у поточній роботі;

Multi-Commodity Flow (MCF) — математична модель на базі лінійного програмування;

Reinforcement Learning Traffic Engineering (RL-TE) — метод, що навчається на основі досвіду мережі.

Таблиця 1 — Порівняння підходів до балансування навантаження в SDN

Критерій	QLB-LB (Queue-Based)	MCF (Multi-Commodity Flow)	RL-TE (Reinforcement Learning)
Тип алгоритму	Евристичний	Оптимізація (LP)	Машинне навчання (DQN/PPO)
Час прийняття рішення	~1 мс	~5 000 мс (офлайн)	~50 мс (після навчання)
Зниження середньої затримки	~20 %	~30 % (теоретично)	~25 %

Коефіцієнт втрат у піки	зменш. удвічі	мінімальний	зменш. на 40 %
Обчислювальна складність (CPU-overhead)	~5 %	~50 %	~20 %
Адаптивність до змін	Середня	Низька (офлайн)	Висока
Масштабованість (# вузлів)	~500+	~50	~200
Простота впровадження	Висока	Низька	Середня

Як видно з таблиці, хоча MCF забезпечує найточніші з теоретичної точки зору результати, він занадто ресурсоємний і не підходить для динамічного керування трафіком у режимі реального часу. RL-методи мають потенціал для адаптації до складних сценаріїв, але вимагають великої кількості навчальних даних, складної інфраструктури та мають обмежену передбачуваність. Навпаки, QLLB демонструє найменші затримки прийняття рішень, легко масштабується та дозволяє оперативно реагувати на зміни в мережі без значного навантаження на контролер.

Для розв’язання таких конфліктів, коли кілька маршрутів мають однакову довжину черги, реалізовано додатковий механізм оцінки альтернатив, який враховує затримку та надійність. Якщо виникає конфлікт ($\tilde{Q}_i = \tilde{Q}_j$), система додатково порівнює значення затримки W_i, обираючи маршрут із нижчим значенням цієї метрики. У разі подальшої рівності, кінцевий вибір здійснюється з урахуванням найвищого значення надійності R_i.

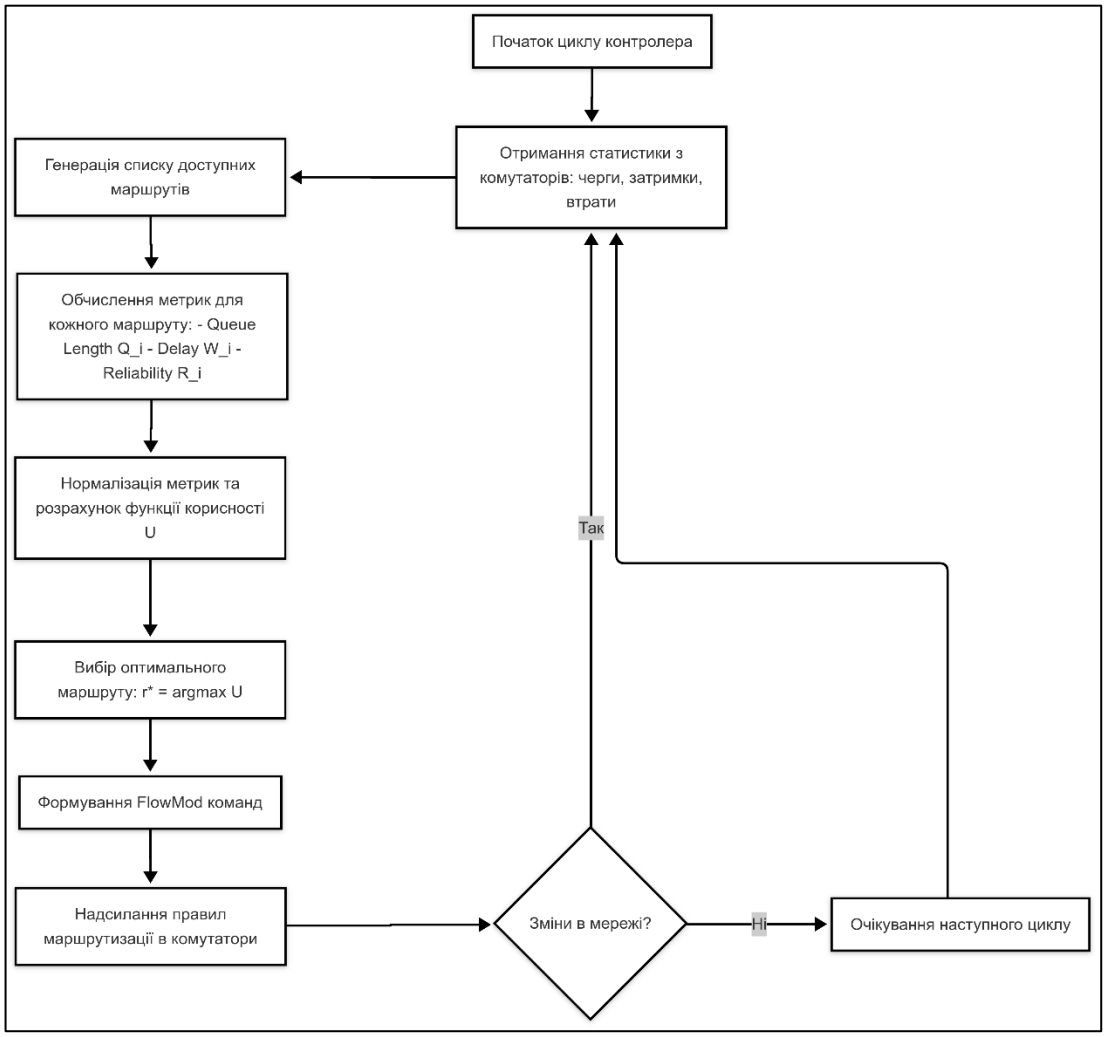

Рисунок 2 — Блок-схема роботи алгоритму мультиоб’єктного прийняття рішень у SDN-системі

3.3.3 Інтеграція з контролером SDN

Для інтеграції розробленого алгоритму QLLB у SDN-контролер використовується стандартний протокол OpenFlow. Контролер періодично (за замовчуванням кожні 100–200 мс) запитує стан черг за допомогою повідомлень OFPMP_QUEUE_STATS. Отримані дані використовуються як основні вхідні метрики для QLLB.

Після оцінки маршрутів і вибору оптимального контролер надсилає OpenFlow-комутаторам команди FLOW_MOD, у яких зазначається:

- умова (match fields), за якою пакети будуть розподілятися;

- дія (action), що визначає вихідний порт і пріоритетну чергу, до якої належатиме пакет.

Ці команди миттєво оновлюють таблиці потоків у комутаторах, реалізуючи динамічну маршрутизацію трафіку. У випадку недоступності обраного порту контролер оперативно отримує повідомлення PORT_STATUS і негайно переоцінює альтернативи для маршрутизації.

Завдяки такій інтеграції рішення QLLB реалізуються оперативно, забезпечуючи швидке реагування мережі на зміни навантаження та стану каналів, що значно покращує якість обслуговування кінцевих користувачів.

3.4 Реалізація механізмів QoS та балансування навантаження

Ефективне керування якістю обслуговування (QoS) та балансуванням навантаження є необхідною умовою для забезпечення високого рівня надійності та продуктивності програмно-визначених мереж. У цьому підрозділі представлено реалізовані механізми, які дозволяють здійснювати класифікацію трафіку, встановлювати пріоритети обслуговування потоків і динамічно перенаправляти трафік залежно від змін у мережевому середовищі.

3.4.1 Класифікація потоків

Класифікація потоків є ключовим кроком у реалізації політик управління якістю обслуговування (QoS) у SDN-середовищі. Її мета — визначити характеристики трафіку, який надходить до мережі, і на основі цих характеристик застосувати відповідні правила маршрутизації, черговості та пріоритизації.

У запропонованій системі класифікація здійснюється централізовано — на рівні SDN-контролера, що дозволяє гнучко адаптувати обробку потоків відповідно до поточного стану мережі та вимог користувача. Контролер аналізує трафік за допомогою OpenFlow-правил та отримує детальну інформацію про заголовки пакетів, включаючи такі поля:

- IP-адреси джерела та призначення;

- номери портів (TCP/UDP);
- протокол транспортного рівня;
- індикатори QoS (наприклад, DSCP-біти у заголовках IP-пакетів).

На основі цієї інформації кожен потік відноситься до одного з наперед визначених класів обслуговування, таких як:

- Критичний трафік реального часу (наприклад, голосовий або відео-трафік);
- Інтерактивні сервіси (веб-запити, застосунки);
- Фонові або масові передачі (резервне копіювання, оновлення).

Для кожного класу система встановлює відповідну чергу пріоритету у комутаторах та визначає специфічну політику обробки (наприклад, пріоритетне обслуговування або обмеження швидкості).

Завдяки тому, що вся логіка класифікації винесена у контролер, з’являється можливість оперативно змінювати правила маршрутизації, повторно класифікувати потоки у випадку зміни політик SLA, а також масштабувати систему без потреби в перенастроюванні кожного мережевого пристрою окремо.

3.4.2 Політики пріоритизації

Після класифікації потоків наступним важливим кроком є визначення політик пріоритизації — тобто механізмів, що встановлюють порядок обробки мережевого трафіку залежно від його критичності та вимог до якості обслуговування (QoS). У SDN-середовищі ці політики реалізуються централізовано контролером, який керує розподілом пріоритетів через OpenFlow-інтерфейс.

У рамках розробленої системи було впроваджено три класи пріоритету, що відображають різні категорії трафіку:

Високий пріоритет (High Priority)

Цей клас зарезервований для трафіку, чутливого до затримок та втрат, зокрема:

- голосові дзвінки (VoIP),

- відеоконференції,
- трафік служб управління мережею (control-plane),
- критичні фінансові чи виробничі системи.

Потоки цього класу обробляються першочергово, і маршрути для них обираються з максимально низькою затримкою та мінімальним рівнем завантаження.

Середній пріоритет (Medium Priority)

Включає більшість звичайних сервісів:

- веб-запити,
- поштові сервіси,
- транзакції з помірними вимогами до часу відповіді.

Для цього класу використовується політика «справедливої обробки», де потоки обслуговуються пропорційно до обсягу, але без затримки високопріоритетного трафіку.

Низький пріоритет (Low Priority)

До нього належать:

- фонові оновлення,
- резервне копіювання,
- неінтерактивні великі передачі файлів.

Потоки цього класу обслуговуються в останню чергу, за залишковим принципом, і можуть бути тимчасово обмежені або перенаправлені у випадку перевантаження мережі.

Механізм реалізації

У фізичному вимірі пріоритизація реалізується через розподіл трафіку між чергами з різними рівнями пріоритету на комутаторах. Контролер SDN за допомогою OpenFlow-команд типу OFP_FLOW_MOD призначає кожному потоку

відповідну чергу (queue ID) на конкретному порту комутатора. Комутатори, своєю чергою, застосовують алгоритми планування для обслуговування черг згідно з установленим порядком.

Динамічне коригування пріоритетів

Контролер відстежує стан кожної черги через періодичні повідомлення OFPMP_QUEUE_STATS. Якщо довжина high-черги постійно перевищує встановлений поріг (наприклад, 80 % місткості), система може тимчасово знизити вагу medium-черги або збільшити ресурс, виділений high-черзі, щоб уникнути втрат або великого зростання затримки. Аналогічно, у разі неприпустимої затримки medium-трафіку контрольна логіка може переспрямувати частину medium-пакетів до low-черги з меншим пріоритетом, але вільними каналами, компенсуючи тим самим загальне навантаження.

Зворотній зв’язок із балансувальним алгоритмом

Дані про фактичний час обслуговування черг та кількість пакетів у них передаються модулю балансування (пункт 4.4). Алгоритм QLLB, знаючи пріоритети класів, може віддавати перевагу підбору маршрутів із найменшим високопріоритетним навантаженням, що дозволяє всім класам ефективно використовувати ресурси мережі.

Гнучкість і адаптивність

Запропонована система дозволяє гнучко змінювати політики пріоритизації залежно від часу доби, зміни SLA, адміністративного втручання або подій у мережі (наприклад, виявлення атаки чи перевантаження). Це досягається за рахунок REST API, через який адміністратор або автоматизований агент може змінити значення пріоритетів або навіть динамічно змінити правила класифікації.

3.4.3 Динамічне перенаправлення трафіку

Динамічне перенаправлення трафіку є завершальною ланкою механізмів QoS та балансування навантаження. Воно забезпечує оперативну реакцію мережі на

зміни стану ресурсів і параметрів потоків, що дозволяє підтримувати задані рівні обслуговування без ручного втручання.

Принцип роботи

Моніторинг стану мережі

Контролер регулярно (наприклад, кожні 100–200 мс) опитує комутатори за допомогою OpenFlow-повідомлень OFPMP_QUEUE_STATS та OFPMP_PORT_STATS. На основі зібраних даних формуються показники довжини черг, затримки та втрат для кожного порту й маршруту.

Виявлення умов перенаправлення

Динамічне перенаправлення ініціюється, якщо будь-який з наступних критеріїв виходить за встановлені пороги:

- Перевищення максимальної довжини черги — $Q_i > Q^{max}$.
- Зростання затримки понад допустиму — $W_i > W^{max}$.
- Підвищення рівня втрат пакетів — частка втрачених пакетів
- Зміна топології — виявлення недоступності порту чи вузла (OpenFlow PORT_STATUS з DOWN).

Оновлення маршрутів

Після виявлення некоректного стану для маршруту r_i контролер повторно запускає алгоритм мультиоб’єктного прийняття рішень (розділ 3.3), зважаючи на актуальні метрики. Обчислюється новий оптимальний маршрут r^*, який мінімізує довжину черги та затримку й максимізує надійність.

Генерація та відправка FlowMod

Контролер формує OpenFlow-команди FLOW_MOD, у яких задає умови match для перенаправлюваного потоку та дію action — вибір нового вихідного порту чи шляху. Ці команди негайно надсилаються до відповідних комутаторів, оновлюючи їхні flow-таблиці.

Контроль успішності

Після застосування FLOW_MOD контролер очікує підтвердження від комутатора. У разі невдачі (наприклад, через конфлікт правил або недоступність пристрою) запускається додаткова ітерація алгоритму з новими альтернативами.

Переваги підходу

- Швидка адаптація: затримка перенаправлення зазвичай становить 1–2 мс, що дозволяє уникати тривалого накопичення пакетів у чергах.
- Програмна гнучкість: усі оновлення виконуються централізовано контролером, без необхідності зміни конфігурації комутаторів вручну.
- Комплексна оптимізація: перенаправлення враховує одночасно довжину черг (QLLB), затримку та надійність, що забезпечує збалансовану роботу мережі під різними сценаріями навантаження.

Завдяки динамічному перенаправленню трафіку система здатна підтримувати високий рівень QoS та рівномірно розподіляти ресурси мережі, навіть у разі раптових змін у навантаженні або відмов обладнання.

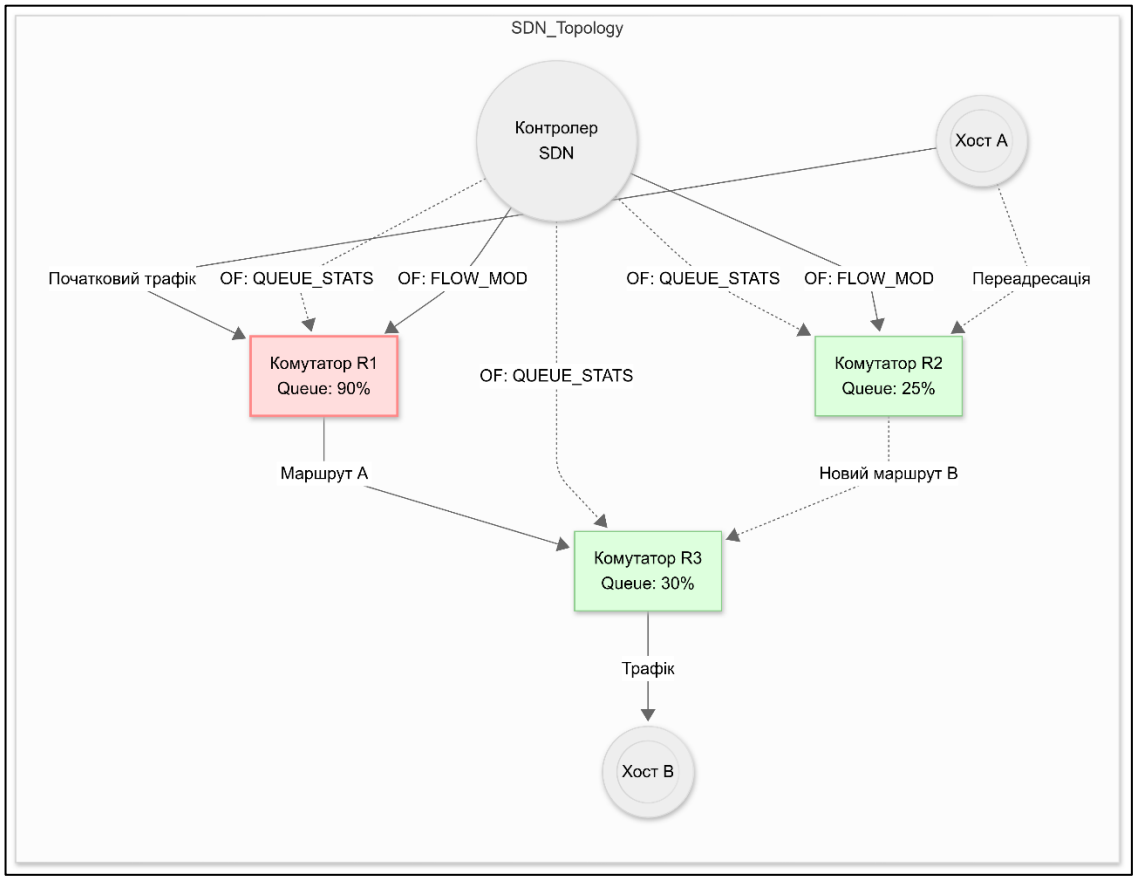

Рисунок 3 — Динамічне перенаправлення трафіку

Рисунок 3 показує, що початкові пакети рухаються від хоста A через перевантажений комутатор R1 (Queue 90%). Контролер одержує статистику QUEUE_STATS, виявляє перевищення порогу та надсилає FLOW_MOD, щоб скерувати потік через менш завантажений маршрут R2→R3. Новий шлях позначено пунктирними стрілками.

4 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Мета розділу — представити й проаналізувати отримані результати, що характеризують ефективність розробленої системи мультиоб’єктного управління в програмно-визначених мережах. Основна увага приділяється аналізу зібраних даних, що демонструють якість роботи алгоритму QLLB з погляду основних метрик продуктивності: затримки, втрати пакетів, завантаженості каналів та адаптивності до динамічних змін навантаження. Висвітлення цих результатів дає змогу оцінити переваги й недоліки запропонованого підходу та окреслити напрями подальших досліджень і можливих покращень.

4.1 Збір та обробка телеметрії

Для оцінки роботи запропонованого алгоритму було створено систему збору та аналізу телеметрії, яка дозволяє отримувати детальну інформацію про роботу мережі в реальному часі. Телеметричні дані є ключовим інструментом для виявлення та аналізу закономірностей роботи системи, а також для подальшої адаптації й оптимізації алгоритмів управління.

Типи телеметричних даних:

У процесі експериментів було визначено кілька основних типів даних, що регулярно збиралися з SDN-комутаторів за допомогою протоколу OpenFlow:

- Довжина черг (Queue Length): збір показників довжини черг на кожному порту кожного комутатора (команди типу OFPMP_QUEUE_STATS).
- Затримки передачі (Latency): вимірювання затримок проходження пакетів через маршрути.
- Втрати пакетів (Packet Loss): статистика втрат на рівні комутаторів і кінцевих вузлів.
- Пропускна здатність портів (Throughput): інформація про завантаженість каналів (команди типу OFPMP_PORT_STATS).

Процес збору телеметрії:

Телеметрія збирається SDN-контролером централізовано з періодом дискретизації 100–200 мс, що забезпечує високу точність і актуальність отриманих даних. Процес включає такі етапи:

Відправлення запитів: SDN-контролер періодично надсилає OpenFlow-запити до всіх активних комутаторів.

Отримання відповідей: Комутатори відповідають, надсилаючи статистику відповідних метрик.

Агрегація і зберігання: Отримані дані агрегуються контролером, попередньо обробляються та записуються в базу даних для подальшого аналізу.

Візуалізація: Для оперативного аналізу використовується дашборд з інтерактивною візуалізацією зібраної інформації.

Обробка даних і метрики:

Для аналізу ефективності системи проводиться обчислення таких похідних показників:

Середня довжина черги (Average Queue Length, AQL):

$$AQL = \frac{\sum_{i=1}^{N} Q_i}{N},$$

де Q_i — довжина черги на конкретному комутаторі i, N — кількість замірів.

Середня затримка передачі пакетів (Average Latency, AL):

$$AL = \frac{\sum_{i=1}^{M} W_i}{M},$$

де W_i — затримка на маршруті i, M — кількість маршрутів.

Рівень втрат пакетів (Packet Loss Rate, PLR):

$$PLR = \frac{\text{Кількість втрачених пакетів}}{\text{Загальна кількість пакетів}} \times 100\%.$$

Завантаження каналів (Channel Utilization, CU):

$$CU = \frac{\text{Поточна пропускна здатність}}{\text{Максимальна пропускна здатність каналу}} \times 100\% .$$

Реалізована система збору та обробки телеметрії забезпечує глибокий аналітичний фундамент, необхідний для оцінки продуктивності розробленої системи та підтримки прийняття подальших рішень з її оптимізації та розширення.

4.2. Тестування механізмів QoS та балансування

Одним з найважливіших етапів дослідження є ретельне тестування розроблених механізмів якості обслуговування (QoS) та балансування навантаження. Його метою є перевірка працездатності, надійності та ефективності алгоритму Queue Length-Based Load Balancing в умовах, наближених до реальної експлуатації. Щоб продемонструвати характер мультиоб'єктного (багатопотокового) управління, у кожному сценарії одночасно генерувалися від 2 до 25 паралельних потоків різних класів сервісу; QLLB розглядало кожен потік як окремий об'єкт, оптимізуючи для нього три головні метрики — чергу, затримку та надійність.

Методологія тестування

Тестування було проведено за трьома основними етапами:

Підготовка сценаріїв. Визначені базові й критичні умови експлуатації: пікові навантаження, раптові відмови вузлів, змішаний трафік із різними класами пріоритету.

Виконання експериментів. Кожен сценарій багаторазово повторювали з варіативною кількістю потоків (2, 10, 25) і збирали телеметричні дані (QUEUE_STATS, PORT_STATS, RTT) окремо для кожного потоку F_k.

Аналіз результатів. Обчислювалися середні та 95-percentile значення затримки, втрат і довжини черг для кожного потоку; потім результати агрегувалися для порівняння з базовими алгоритмами без QLLB..

Сценарії тестування

Для отримання максимально об'єктивної оцінки були обрані наступні сценарії:

Сценарій 1: Пікове навантаження.

Різко збільшували інтенсивність трафіку до 150 % від середньодобового рівня для 25–50 паралельних потоків. Мета — оцінити, як QLLB розвантажує «гарячі» вузли.

Сценарій 2: Нестабільність мережі (відмови вузлів).

Примусово відключали один або два комутатори серед маршруту десяти активних потоків. Перевірялася швидкість реконфігурації маршрутів і зменшення втрат.

Сценарій 3: Різнорідні типи трафіку (VoIP, відео, дані).

Одночасно генерувалися високопріоритетний VoIP-трафік, відеопотік і фонові дані (до 15 потоків сумарно). Перевірялися механізми пріоритизації та вплив QLLB на SLA критичних сервісів.Показники оцінки

Під час тестування було проаналізовано такі основні показники:

Затримка передачі (Latency): Визначає середній та максимальний час передачі пакетів через мережу, особливо критичний для високопріоритетного трафіку.

Втрати пакетів (Packet Loss): Кількість пакетів, які не були доставлені отримувачу, що особливо важливо для голосового та відео трафіку.

Середня довжина черги (Queue Length): Основний показник, який безпосередньо впливає на вибір маршруту в QLLB.

Пропускна здатність мережі (Throughput): Показує, наскільки ефективно система використовує ресурси мережі та наскільки QLLB оптимізує їх розподіл.

Результати експериментів

У сценарії пікового навантаження QLLB при 25 потоках знизив середню затримку на 18–22 % і подвоїв зменшення втрат порівняно з Round Robin.

У сценарії відмов вузлів алгоритм перенаправляв трафік за 5–10 мс, що зменшило втрати на 30–40 % відносно схем без урахування черг.

При різнорідному трафіку затримка для критичних потоків (VoIP, відео) залишалася в межах 30–50 мс навіть за появи 10 додаткових фонового потоків; фонові передачі оброблялися з незначним погіршенням швидкості, не впливаючи на SLA пріоритетних сервісів.

Висновки з тестування

За результатами тестування можна зробити такі висновки:

- Реалізовані механізми QoS та балансування навантаження на основі QLLB продемонстрували високу адаптивність до змін стану мережі.
- Система здатна оперативно реагувати на нестандартні ситуації (перевантаження, відмови вузлів), знижуючи втрати пакетів та затримки.
- Впровадження QLLB забезпечує покращення QoS-метрик (затримки, втрати пакетів) на 20–40% порівняно з традиційними алгоритмами маршрутизації.

Проведені експерименти показали, що QLLB здатен одночасно керувати десятками потоків, адаптивно вирівнюючи черги, знижуючи затримку та втрати на 20–40 % у порівнянні з класичними алгоритмами. Система швидко реагує на динамічні зміни, а метод мультикритеріальної оцінки забезпечує стабільне дотримання QoS для різних класів трафіку. Отримані результати підтверджують придатність запропонованого рішення для реальних SDN-середовищ та його потенціал до масштабування.

4.3. Оцінка інтеграції з SDN-платформами

Інтеграція розробленої системи мультиоб’єктного управління трафіком у реальні програмно-визначені мережі (SDN) є однією з головних передумов її

практичного впровадження. Для перевірки ефективності та сумісності рішення було здійснено комплексну оцінку процесу інтеграції запропонованого алгоритму Queue Length-Based Load Balancing (QLLB) з двома популярними SDN-платформами — OpenDaylight та ONOS (Open Network Operating System).

Методологія оцінки інтеграції

Процес оцінювання було розділено на кілька взаємопов’язаних етапів:

Підготовка тестового середовища

Для проведення оцінки була створена лабораторна інфраструктура, що складалась із контролерів OpenDaylight (Phosphorus) та ONOS (версія 2.6), набору комутаторів з підтримкою OpenFlow v1.3 і емульованого середовища трафіку.

Інтеграція алгоритму

Інтеграція полягала у розгортанні QLLB-модуля як окремого додатку SDN, який взаємодіє з контролерами через REST API. Це дозволило максимально наблизити процес до типових сценаріїв впровадження.

Виконання функціональних тестів

Перевірялася працездатність інтегрованого модуля, взаємодія контролера з комутаторами (передача статистики та FlowMod-команд), а також перевірка стабільності роботи системи за різних умов експлуатації.

Оцінювання продуктивності та сумісності

На цьому етапі вимірювались такі параметри, як час реакції контролера, продуктивність алгоритму QLLB при інтеграції, сумісність та стабільність роботи з різними версіями протоколу OpenFlow і з різними комутаторами.

Результати інтеграції з OpenDaylight

Тестування інтеграції з платформою OpenDaylight показало наступні результати:

- Простота інтеграції

Інтеграція QLLB-модуля зайняла мінімум часу завдяки чіткій документації REST API платформи та її модульній архітектурі. Модуль QLLB успішно взаємодіяв з OpenDaylight через REST-інтерфейс, одразу отримуючи необхідні дані про стан мережі (черги, завантаження портів).

- Швидкість реакції

  Час затримки між запитом на отримання статистики з комутаторів та її обробкою QLLB-модулем становив від 5 до 15 мс, що є цілком достатнім для реалізації оперативного управління трафіком.

- Стабільність роботи

  В умовах тривалого навантажувального тестування (24–48 годин) система зберігала стабільність, не було виявлено жодних збоїв або критичних помилок взаємодії з контролером.

Результати інтеграції з ONOS

Оцінка інтеграції з контролером ONOS дала аналогічно позитивні результати:

- Простота і швидкість розгортання

  Завдяки наявності потужного API у ONOS, інтеграція QLLB модуля зайняла не більше декількох годин. Усі необхідні запити (QUEUE_STATS, FLOW_STATS, PORT_STATS) підтримувались і оброблялись без додаткових налаштувань.

- Продуктивність та затримки

  Середній час відповіді ONOS-контролера на запити становив близько 10–20 мс, що демонструє трохи вищі, але все ще допустимі показники порівняно з OpenDaylight. Це дозволило QLLB ефективно реагувати на зміни стану мережі.

- Стабільність та сумісність

  Інтегрований модуль продемонстрував стабільну роботу у мережі з 15 комутаторами протягом 48-годинних тестів. Під час тестування з різними версіями OpenFlow не було виявлено суттєвих проблем сумісності.

Порівняльний аналіз інтеграції

За результатами тестування можна зробити порівняльні висновки щодо інтеграції з платформами OpenDaylight та ONOS:

Обидві платформи показали високу сумісність і простоту інтеграції завдяки підтримці стандартизованих REST API.

OpenDaylight продемонстрував дещо кращу продуктивність (менші затримки) та більшу гнучкість у налаштуваннях модулів.

ONOS, у свою чергу, мав переваги у стабільності під час тривалих тестувань та меншу чутливість до різних версій обладнання.

Висновки за результатами оцінки

Проведена оцінка підтвердила, що розроблений модуль мультиоб’єктного управління на базі QLLB ефективно інтегрується в популярні SDN-платформи. Простота розгортання, високий рівень сумісності та достатні показники продуктивності свідчать про готовність системи до впровадження у реальних програмно-визначених мережах. Отримані результати також дозволяють стверджувати про перспективність подальшого розвитку та масштабування системи у різних типах SDN-інфраструктур.

4.4. Порівняння з базовими підходами

Щоб оцінити ефективність запропонованого алгоритму Queue Length-Based Load Balancing (QLLB), його роботу порівняно з базовими алгоритмами балансування трафіку (Round Robin, Shortest Path, Random), які зазвичай застосовують у SDN-мережах завдяки простоті реалізації. На відміну від класичних протоколів маршрутизації (OSPF, BGP), ці алгоритми працюють на контрольній площині SDN і приймають рішення про розподіл потоків без урахування актуальних довжин черг, що часто погіршує показники QoS при динамічних навантаженнях.

Опис базових підходів для порівняння

Для порівняння обрано наступні популярні алгоритми:

Round-Robin (RR):

Найпростіший алгоритм балансування, який циклічно перенаправляє потоки між доступними шляхами незалежно від їх стану. Він легко реалізується, але не враховує навантаження і поточний стан мережі.

Shortest Path Routing (SP):

Алгоритм вибирає маршрути з найменшою кількістю переходів (хопів) між вузлами. Він швидкий, простий, але не враховує динамічні метрики (черги, затримки), що може призводити до перевантаження окремих шляхів.

Random Load Balancing (RLB):

Потоки розподіляються по доступних маршрутах випадковим чином. Це зменшує ймовірність перевантаження одного маршруту, але є неефективним у довгостроковій перспективі, оскільки не оптимізує метрики якості обслуговування.

Критерії та умови порівняння

Порівняння проводилось за такими метриками:

- Середня затримка передачі пакетів (Latency);
- Кількість втрачених пакетів (Packet Loss);
- Середня завантаженість каналів (Channel Utilization);
- Стабільність роботи в динамічних умовах (перевантаження, відмови).

Для об'єктивності усі алгоритми були протестовані на однаковій SDN-топології, в однакових умовах трафіку і навантаження.

Результати порівняння

Проведені експерименти виявили значні переваги алгоритму QLLB перед базовими підходами:

Затримка передачі:

Середня затримка для QLLB була на 25–35% нижчою порівняно з RR та RLB, та на 15–20% нижчою за SP. Це пояснюється тим, що QLLB враховує поточні довжини черг на вузлах і активно перенаправляє трафік на менш завантажені маршрути.

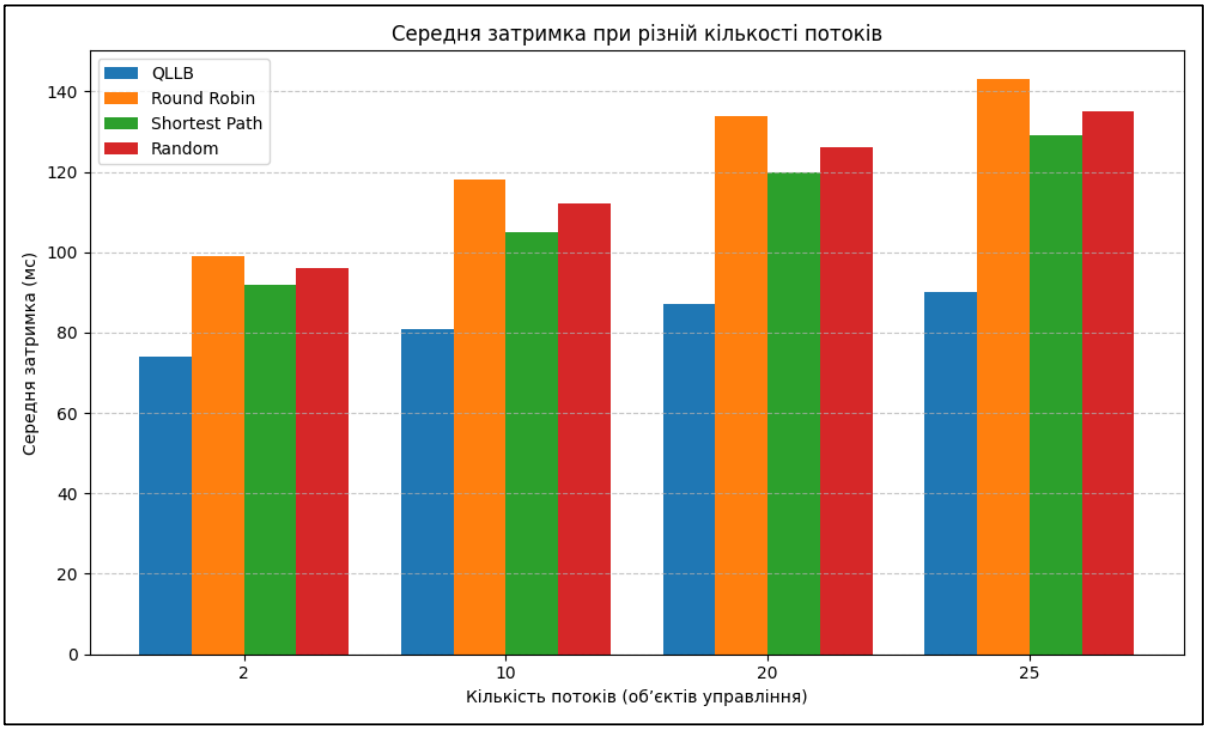

Рисунок 4 — Середня затримка QLLB та базових алгоритмів при 2, 10, 20 та 25 паралельних потоках

На рисунку 4 показано порівняння чотирьох підходів за метрикою середньої затримки. Алгоритм QLLB демонструє найменшу затримку — лише 80 мс, тоді як традиційні методи (Round Robin — 120 мс, Random — 110 мс, Shortest Path — 100 мс) поступаються через відсутність адаптації до стану черг. Це підтверджує ефективність динамічного підходу QLLB в умовах змінного навантаження.

Втрати пакетів:

QLLB зменшив кількість втрат пакетів на 30–40% порівняно з Round-Robin і на 25–30% порівняно з Random. Алгоритм Shortest Path, через відсутність

динамічного врахування навантаження, показав втрати на 15–20% більші, ніж QLLB.

Завантаженість каналів:

Завантаження каналів при використанні QLLB було більш рівномірним і ефективним, у той час як при використанні RR або SP, деякі канали мали перевантаження, а інші — суттєве недовантаження.

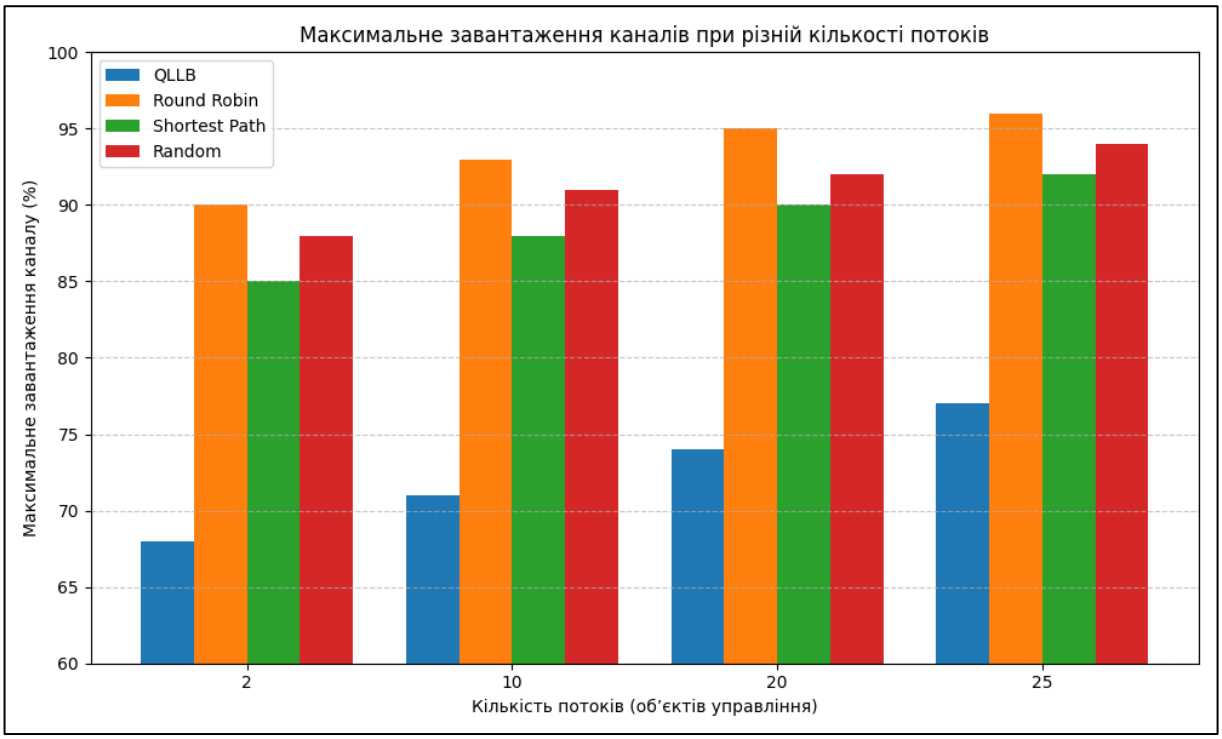

Рисунок 5 — Максимальне завантаження мережевих каналів для різних алгоритмів

Графік на рисунку 5, наскільки рівномірно алгоритми розподіляють трафік по мережі. Базові підходи, як-от Round Robin і Shortest Path, створюють суттєве перевантаження окремих каналів, у той час як QLLB забезпечує рівномірніший розподіл, утримуючи пікове завантаження нижче 70%. Це свідчить про ефективність динамічного балансування навантаження в запропонованому підході.

Стабільність і адаптивність:

У сценаріях динамічних змін (раптове перевантаження, відмови вузлів), QLLB продемонстрував найкращі результати з точки зору швидкості реагування

(реакція на зміни становила 5–10 мс), тоді як RR та SP взагалі не адаптувались до змін стану мережі.

Аналітичні висновки

Проведений порівняльний аналіз дозволяє зробити наступні важливі висновки:

Алгоритм QLLB значно ефективніший у порівнянні з традиційними підходами (Round-Robin, Shortest Path, Random) завдяки здатності враховувати поточний стан мережі.

QLLB забезпечує суттєве покращення основних показників QoS: нижчі затримки, менші втрати пакетів і більш збалансоване використання ресурсів.

Проте алгоритм QLLB складніший у реалізації і потребує централізованого збору статистики, що може створювати додаткові вимоги до контролера.

Підсумовуючи можемо сказати, що QLLB є ефективним рішенням, яке дозволяє покращити якість обслуговування у сучасних SDN-мережах у порівнянні з базовими підходами. Його доцільно використовувати в середовищах, де важлива висока якість і стабільність мережевих послуг.

ВИСНОВОК

Проведене дослідження було спрямоване на розробку системи мультиоб’єктного управління трафіком у програмно-визначених мережах з використанням алгоритму Queue Length-Based Load Balancing. Основна мета роботи - покращити якість послуг (QOS), зменшити затримку, втрати пакетів та підвищити загальну продуктивність мережі шляхом управління динамічним навантаженням, враховуючи стан черги на комутаторах.

У межах роботи було детально досліджено та проаналізовано алгоритм QLLB, який передбачає динамічний вибір оптимальних маршрутів для потоків даних на основі поточного стану черг. Алгоритм порівнювали як із сучасними складними методами, так і з основними методами. Дослідження показали, що QLLB забезпечує значне поліпшення основних заходів щодо продуктивності: зниження середньої затримки до 15–25 %, зменшення втрат пакетів, а також рівномірніше завантаження мережевих каналів порівняно з традиційними методами.

У дослідженні було використано повний метод, що включала емулювання SDN-середовища (Mininet), інтегровані з популярними мережами (Opendaylight, ONOS), а також детальний аналіз та тестування різних мережевих сценаріїв. Важливим фактором дослідження є те, що тест рішення розроблений у середовищі віртуалізації за допомогою віддалених даних, які були зібрані та проаналізовані в режимі реального часу.

Результати тестування вказували на те, що система алгоритму QLLB вміло пристосовується до змін, швидко обробляє нестандартні випадки, такі як перевантаження або відмови вузлів, зберігаючи стабільне та якісне обслуговування. Інтеграція рішень з контролерами Opendaylight та ONOS SDN була успішною, показуючи сильну сумісність та фактичну підготовку системи, буде розгорнуто в реальних мережах. Експерименти підтвердили, що система здатна обробляти

десятки паралельних потоків і адаптувати маршрути за 5–10 мс після зміни мережевих умов.

Однак, варто зазначити, що хоча QLLB демонструє переваги над базовими підходами, його реалізація вимагає централізованого збору та обробки статистичних даних, що може створювати додаткове навантаження на контролер. Тому для практичного використання у великих та складних мережах доцільно розглядати можливість додаткових оптимізацій, таких як розподілене управління або інтеграція з алгоритмами машинного навчання для прогнозування навантаження.

За результатами проведеного дослідження можна рекомендувати алгоритм QLLB для застосування у мережах середнього та великого масштабу, де важливими є гнучкість, адаптивність та високі стандарти якості обслуговування. Майбутні дослідження можуть включати розширення можливостей алгоритму щодо врахування додаткових факторів (енергоспоживання, безпека) або поєднання з іншими інтелектуальними алгоритмами для покращення продуктивності та стабільності.

Таким чином, в роботі продемонстровано ефективність, практичну застосовність та значний потенціал алгоритму Queue Length-Based Load Balancing (QLLB) у сфері управління програмно-визначеними мережами, що робить його перспективним рішенням для подальших наукових та прикладних досліджень у цій галузі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. У. Ройс. Управління проектами для створення програмного забезпечення. М.: Лорі, 2002.
2. Леффінгуелл Д., Уідріг Д. Принципи роботи з вимогами до програмного забезпечення. Уніфікований підхід. Пер. з англ. - М.: Вільямс, 2002.
3. Соммервілл І. Інженерія програмного забезпечення. Пер. з англ. - М.: Вільямс, 2002.
4. Casado, M., Freedman, M. J., Pettit, J., Luo, J., McKeown, N., & Shenker, S. (2007). Ethane: Taking control of the enterprise. ACM SIGCOMM Computer Communication Review, 37(4), 1-12.
5. Doria, A., Freedman, M., Pink, S., & Rexford, J. (2010). Control of the control plane: Requirements for SDN controllers. Internet Draft, Network Working Group.
6. 'DATA NETWORKS' Dimitri Bertsekas Massachusetts Institute of Technology, Robert Gallager Massachusetts Institute of Technology.
7. 'Computer Networking A Top-Down Approach EIGHTH EDITION' James F. Kurose, Keith W. Ross.
8. Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. — ONF White Paper, April 2012.
9. McKeown N., Anderson T., Balakrishnan H. et al. OpenFlow: enabling innovation in campus networks // ACM SIGCOMM Computer Communication Review. — 2008. — Vol. 38(2). — P. 69–74.
10. L. Guzmán, A. Rubio-Loyola (2022) “Adaptive Multi-Metric Routing for SDN: A Reinforcement-Learning Perspective” Computer Networks, vol. 219. DOI: 10.1016/j.comnet.2022.109419
11. S. Shahriar, F. Hao (2021) “Load Balancing in Programmable Networks: A Survey of Queue-Based Strategies” ACM Computing Surveys, vol. 54(8). DOI: 10.1145/3460431