Міністерство освіти і науки України

Луцький національний технічний університет

(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій

(повне найменування факультету)

Кафедра комп'ютерної інженерії та охоронних систем

(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

ІНТЕЛЕКТУАЛЬНА SDN-СИСТЕМА ОПТИМІЗАЦІЇ МЕРЕЖЕВОГО ТРАФІКУ З ВИКОРИСТАННЯМ ML-АНАЛІЗУ ТЕЛЕМЕТРІЇ ЗАСОБАМИ MININET

SMART SDN SYSTEM FOR NETWORK TRAFFIC OPTIMIZATION USING ML ANALYSIS OF TELEMETRY WITH MININET

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-41
Берайа Дмитро Зурабович

(підпис)

Керівник:
к.т.н., доцент
Багнюк Наталія Володимирівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«_____» червня 2026 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т. Терлецький
«__23__» _____12_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Берайа Дмитру Зурабовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Інтелектуальна SDN-система оптимізації мережевого трафіку з використанням ML-аналізу телеметрії засобами Mininet*

Керівник роботи *к.т.н., доцент Багнюк Наталія Володимирівна*

затверджені наказом закладу вищої освіти від «20» грудня 2025 року № 536/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 28.05.2026 р.

3. Вихідні дані до роботи: *джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Теоретичні основи SDN та телеметрії для оптимізації трафіку
Підходи ML для аналізу телеметрії та прийняття рішень в SDN
Практична реалізація інтелектуальної SDN-системи оптимізації мережевого трафіку
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Архітектура інтелектуальної SDN-системи оптимізації трафіку
Топологія мережі в середовищі Mininet (8 комутаторів, 12 хостів)
Схема роботи ML-модуля класифікації стану мережі
Матриця помилок та метрики якості класифікатора (LR vs Random Forest)
Порівняльний аналіз мережевих показників до і після оптимізації

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Теоретичні основи SDN та телеметрії для оптимізації трафіку	*Багнюк Н. В., доцент*		
Підходи ML для аналізу телеметрії та прийняття рішень в SDN	*Багнюк Н. В., доцент*		
Практична реалізація інтелектуальної SDN-системи оптимізації мережевого трафіку	*Багнюк Н. В., доцент*		
Нормоконтроль	*Багнюк Н. В., доцент*		
Гарант ОП	*Лавренчук С. В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О. І., ст. викладач*		

7. Дата видачі завдання 23.12.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	до 10.02.2026 р.	
2.	*Теоретичні основи SDN та телеметрії для оптимізації трафіку*	до 02.03.2026 р.	
3.	*Практична реалізація інтелектуальної SDN-системи оптимізації мережевого трафіку*	до 02.04.2026 р.	
4.	*Проєктування та моделювання інтелектуальної sdn- системи в mininet*	до 10.04.2026 р.	
5.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	до 01.05.2026 р.	
6.	*Нормоконтроль*	до 23.05.2026 р.	
7.	*Інструментальна перевірка на академічний плагіат*	до 25.05.2026 р.	
8.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	до 28.05.2026 р.	

Здобувач вищої освіти _____ Дмитро БЕРАЙА
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Наталія БАГНЮК
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Берайа Д. З. Інтелектуальна SDN-система оптимізації мережевого трафіку з використанням ML-аналізу телеметрії засобами Mininet. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота присвячена розробленню підходу до інтелектуальної оптимізації мережевого трафіку в програмно-керованих мережах (SDN) на основі аналізу телеметрії із застосуванням методів машинного навчання та LLM (GPT API). У роботі розглянуто роль централізованого SDN-контролера як точки прийняття рішень і механізму швидкого впровадження політик маршрутизації та QoS у площині даних.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку використаних джерел та додатків.

У першому розділі розкрито теоретичні основи SDN і принципи збору мережевої телеметрії (лічильники портів, статистика потоків, RTT, дропи) як джерела даних для оптимізації. Показано, що саме телеметрія дозволяє оцінювати завантаження лінків і передумови перевантаження, що напряму впливає на якість сервісу.

У другому розділі розглянуто підходи ML/LLM до аналізу телеметрії та прийняття рішень у SDN. Описано формування ознак, визначення критеріїв перевантаження та підхід до використання GPT API як аналітичного компонента, який може повертати оцінку ризику та рекомендації дій. Наголошено на необхідності валідації рішень і обмеження простору дій політикою контролера.

У третьому розділі описано проєктування системи та сценарій моделювання в Mininet із оцінюванням ефективності за метриками затримки, втрат і пропускної здатності

Ключові слова: SDN, телеметрія, оптимізація трафіку, Mininet, OpenFlow.

ANNOTATION

Beraia D. Smart SDN system for network traffic optimization using ML analysis of telemetry with Mininet. Manuscript. Bachelor's qualification work, Educational Program "Computer Engineering", Specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2026.

The qualification work focuses on an intelligent approach to network traffic optimization in Software-Defined Networking (SDN) based on network telemetry analysis using machine learning methods and an LLM component (GPT API). The work considers the SDN controller as a centralized decision-making point and as a mechanism for rapidly applying routing and QoS policies in the data plane.

The thesis consists of an introduction, three chapters, conclusions, references, and appendices. Chapter 1 covers SDN fundamentals and the role of telemetry (port counters, flow statistics,

RTT, drops) as the primary data source for optimization and congestion assessment.

Chapter 2 discusses ML/LLM-based telemetry analysis for SDN decision-making, including feature construction, congestion criteria definition, and the use of GPT API to output risk assessment and actionable recommendations, with an emphasis on policy constraints and validation.

Chapter 3 presents the system design and Mininet-based simulation, evaluating performance across delay, loss, and throughput metrics.

Keywords: SDN, telemetry, traffic optimization, Mininet, OpenFlow.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ SDN ТА ТЕЛЕМЕТРІЇ ДЛЯ ОПТИМІЗАЦІЇ ТРАФІКУ 10
1.1 Концепція програмно-керованих мереж (SDN) 10
1.2 SDN-контролери: класифікація та порівняння 11
1.3 Інструменти емуляції та симуляції SDN-мереж 12
1.4 Мережева телеметрія: принципи та методи збору 15
1.5 Телеметрія як основа оптимізації трафіку в SDN 17
1.6 Огляд підходів до використання ML у SDN для оптимізації трафіку 18
РОЗДІЛ 2 ПІДХОДИ ML ДЛЯ АНАЛІЗУ ТЕЛЕМЕТРІЇ ТА ПРИЙНЯТТЯ РІШЕНЬ В SDN 20
2.1 Телеметрія як джерело даних для інтелектуального аналізу мережі 20
2.2 Підходи машинного навчання до класифікації стану мережі 21
2.3 Формування ознак та підготовка навчальної вибірки 23
2.4 Використання LLM як аналітичного компонента в SDN 26
2.5 Архітектура прийняття рішень: від класифікації до керуючого впливу 27
2.6 Узагальнення підходів та обґрунтування вибору методів 28
РОЗДІЛ 3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ SDN-СИСТЕМИ ОПТИМІЗАЦІЇ МЕРЕЖЕВОГО ТРАФІКУ 30
3.1 Постановка задачі практичної реалізації 30
3.2 Архітектура розробленої системи 31
3.3 Реалізація мережевої топології в середовищі Mininet 33
3.4 Реалізація генерації трафіку та моделювання деградації мережі 35
3.5 Реалізація збору телеметрії та збереження даних 36
3.6 Реалізація ML-аналізу стану мережі 39
3.7 Реалізація decision engine та механізмів оптимізації трафіку 40
3.8 Механізм оптимізації трафіку та роль GPT-модуля у прийнятті рішень 41
3.9 Узагальнення результатів практичної реалізації 43

3.10 Організація та методика проведення експериментів 43
3.11 Показники оцінювання стану мережі та ефективності системи 45
3.12 Результати класифікації стану мережі на основі ML-аналізу телеметрії 46
3.13 Аналіз впливу оптимізаційних дій на мережеві показники 48
3.14 Порівняльний аналіз режимів роботи системи 49
3.15 Підсумки експериментального дослідження 51
ВИСНОВКИ 53
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 55

ВСТУП

Актуальність теми. Зростання обсягів мережевого трафіку та динамічність сучасних телекомунікаційних систем висувають нові вимоги до інтелектуального керування мережами. Програмно-керовані мережі (SDN) створюють технічну основу для централізованого управління потоками даних, однак їх потенціал повністю розкривається лише у поєднанні з методами машинного навчання, здатними аналізувати телеметрію та приймати адаптивні рішення в режимі реального часу. Традиційні порогові підходи реагують на вже наявну проблему, тоді як ML-аналіз дозволяє діяти проактивно – на основі динаміки показників, а не їх граничних значень. Дослідження підтверджують, що поєднання SDN із ML-аналізом телеметрії суттєво скорочує час виявлення перевантаження порівняно з традиційними підходами [1].

Метою роботи **є** розроблення та верифікація в емульованому середовищі Mininet інтелектуальної SDN-системи, яка на основі мережевої телеметрії виконує ML-аналіз стану мережі й застосовує оптимізаційні рішення у режимі реального часу.

Об’єкт дослідження – процеси моніторингу та оптимізації мережевого трафіку в програмно-керованих мережах.

Предмет дослідження – методи ML-класифікації стану SDN-мережі на основі телеметричних показників та механізми автоматичного керування трафіком через SDN-контролер ONOS.

Завдання, які необхідно виконати**:**

– дослідити принципи архітектури SDN та методи збору мережевої телеметрії;

– дослідити методи машинного навчання для класифікації стану мережі та підходи до інтеграції LLM у контур керування; спроектувати архітектуру інтелектуальної SDN-системи з модулем збору телеметрії,

– ML-класифікатором та decision engine;

– реалізувати прототип системи в середовищі Mininet із підключенням до контролера ONOS;

– розробити ML-модуль класифікації стану мережі на основі алгоритмів Logistic Regression та Random Forest;

– візуалізувати результати моніторингу засобами Grafana у режимі реального часу;

– запропонувати механізм автоматичної оптимізації трафіку на основі GPT Advisory Feed та decision engine.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ SDN ТА ТЕЛЕМЕТРІЇ ДЛЯ ОПТИМІЗАЦІЇ ТРАФІКУ

1.1 Концепція програмно-керованих мереж (SDN)

Програмно-керовані мережі (SDN) являють собою архітектурний підхід, у якому площина керування відокремлена від площини пересилання, що дозволяє централізовано управляти мережевою інфраструктурою через програмне забезпечення. Така архітектура робить мережеве управління безпосередньо програмованим, а базову інфраструктуру абстрагованою для застосунків і мережевих сервісів [2]. Централізація логіки прийняття рішень у контролері відкриває можливості для динамічної зміни політик маршрутизації, керування якістю обслуговування та автоматизації мережевих операцій без втручання в конфігурацію окремих пристроїв [3].

Архітектура SDN традиційно поділяється на три рівні. Рівень інфраструктури (infrastructure layer) містить мережеві пристрої комутатори, маршрутизатори, що виконують фізичну передачу пакетів. Рівень управління (control layer) є центральним інтелектом мережі, де SDN-контролер зберігає глобальне уявлення про топологію та стан мережі й обчислює рішення щодо маршрутизації. Рівень застосунків (application layer) містить мережеві програми: системи балансування навантаження, виявлення вторгнень, оптимізації QoS. Між рівнями взаємодія відбувається через стандартизовані інтерфейси: southbound API (наприклад, OpenFlow) між рівнем управління та інфраструктурою, і northbound API між рівнем управління та застосунками (рис. 1.1).

Протокол OpenFlow є найбільш поширеним southbound API у SDN-системах. Він визначає спосіб, у який контролер встановлює правила пересилання (flow rules) у таблицях потоків (flow tables) мережевих пристроїв. Кожне правило складається з умов відповідності (match fields це MAC-адреси, IP-адреси, порти тощо) та дій (actions переслати на порт, скинути, змінити

заголовок). Завдяки цьому контролер може динамічно змінювати маршрутизацію без перенастроювання кожного пристрою вручну [2].

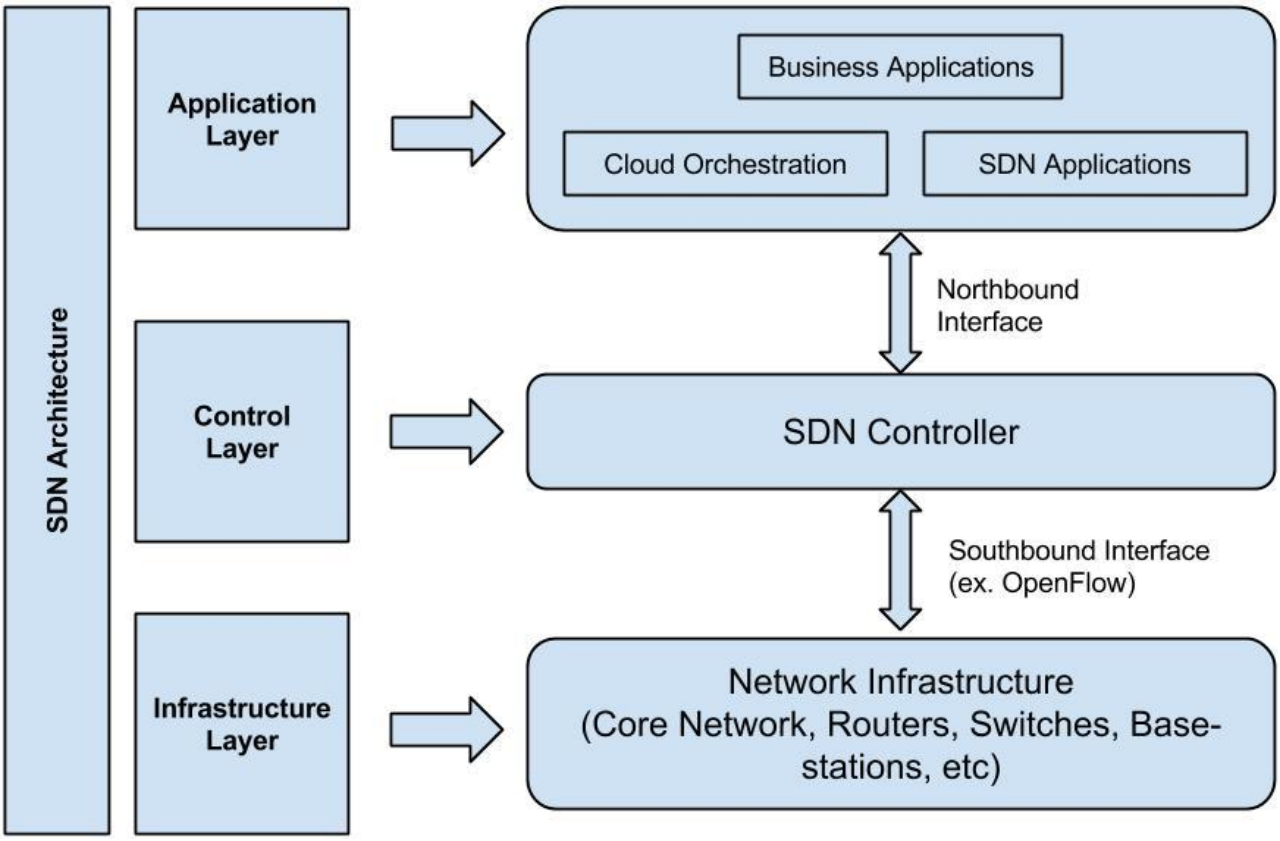

Рисунок 1.1 – Тришарова архітектура SDN [4]

1.2 SDN-контролери: класифікація та порівняння

SDN-контролер є центральним елементом усієї архітектури програмно-керованої мережі. Він виконує функції «мозку» системи: зберігає актуальне уявлення про топологію, обробляє події від комутаторів, обчислює шляхи та встановлює правила пересилання. Сучасні SDN-контролери поділяються на кілька категорій залежно від призначення та особливостей реалізації.

Контролер ONOS (Open Network Operating System) є одним із найбільш масштабованих відкритих SDN-контролерів, розробленим для операторських мереж. Він підтримує кластеризацію (distributed deployment), забезпечуючи відмовостійкість та горизонтальне масштабування. ONOS надає розвинений REST API і набір вбудованих застосунків (intent framework, topology service, flow rule

service). Саме ONOS широко використовується в академічних дослідженнях завдяки підтримці Mininet-інтеграції та зручним інструментам візуалізації топології [2].

Контролер Ryu є легковаговим Python-фреймворком для розробки SDN-застосунків. Він ідеально підходить для швидкого прототипування: дослідник може написати власний контролер як Python-клас, що успадковує RyuApp і визначає обробники подій (packet_in, switch_features тощо). Ryu підтримує OpenFlow 1.0-1.5, а також протоколи OVSDB і Netconf. Завдяки простоті та читабельності коду Ryu є популярним вибором для навчальних та дослідницьких сценаріїв [5].

Контролер POX є ще одним Python-контролером, розробленим Стенфордським університетом переважно для навчальних цілей. Хоча POX поступається Ryu та ONOS за функціональністю, він відрізняється мінімальними залежностями і дозволяє швидко зрозуміти принципи роботи SDN-контролера. Контролер OpenDaylight (ODL), розроблений під егідою Linux Foundation, є одним із найбільш функціональних корпоративних контролерів з підтримкою широкого спектра протоколів: OpenFlow, NETCONF, BGP-LS, PCEP. ODL використовується переважно в промислових впровадженнях.

1.3 Інструменти емуляції та симуляції SDN-мереж

Для дослідження та верифікації SDN-рішень без розгортання фізичної інфраструктури науковці та інженери використовують програмні інструменти, емулятори та симулятори мережевого середовища. Важливо розрізняти ці два підходи: емулятор відтворює реальну поведінку мережевих пристроїв і дозволяє запускати реальний мережевий стек (наприклад, TCP/IP), тоді як симулятор моделює мережу за допомогою математичних абстракцій, не виконуючи реального коду протоколів.

Mininet є найбільш широко використовуваним емулятором SDN-мереж у академічній та дослідницькій спільноті. Він створює віртуальні мережі на базі Linux-namespaces та Open vSwitch (OVS), що дозволяє запускати реальний

мережевий код (iperf, ping, curl) між віртуальними хостами. Ключова перевага Mininet це висока відповідність реальній поведінці мережі при мінімальних апаратних вимогах: вся мережа з десятками комутаторів і сотнями хостів може працювати на звичайному ноутбуці.

Mininet підтримує інтеграцію з будь-яким OpenFlow-контролером (Ryu, ONOS, POX) і дозволяє задавати параметри каналів: пропускну здатність (bandwidth), затримку (delay), jitter та packet loss.

Mininet є широко використовуваним інструментом для емуляції SDN-середовищ у дослідницьких роботах завдяки гнучкості, простоті використання та точності відтворення мережевої поведінки [5].

Окрім Mininet, у дослідженнях SDN також активно використовуються інші інструменти, такі як NS-3 та OMNeT++, які належать до класу симуляторів і дозволяють моделювати великомасштабні мережі з високим рівнем деталізації. NS-3 широко застосовується для аналізу продуктивності мережевих протоколів, оскільки підтримує точне моделювання затримок, черг та втрат пакетів на рівні подій. OMNeT++ відрізняється модульною архітектурою та зручністю побудови складних сценаріїв із великою кількістю вузлів. Використання симуляторів разом із емуляторами дозволяє отримати більш повну картину поведінки SDN-мереж на різних рівнях абстракції. Таким чином, інструменти емуляції та симуляції є ключовою складовою досліджень у сфері SDN, оскільки дозволяють перевіряти нові архітектурні рішення без значних витрат на фізичне обладнання. Вони забезпечують можливість швидкого прототипування мережевих сценаріїв та тестування різних конфігурацій контролерів і комутаторів. Завдяки цьому дослідники можуть оцінювати продуктивність, масштабованість і надійність SDN-рішень ще на етапі проєктування. Крім того, використання таких інструментів знижує ризики помилок при впровадженні в реальних мережах. Поєднання емуляції та симуляції дозволяє отримати більш комплексне розуміння поведінки мережі в різних умовах навантаження та топологіях. Це робить їх незамінними інструментами як для академічних досліджень, так і для практичної розробки SDN-систем. (рис. 1.2).

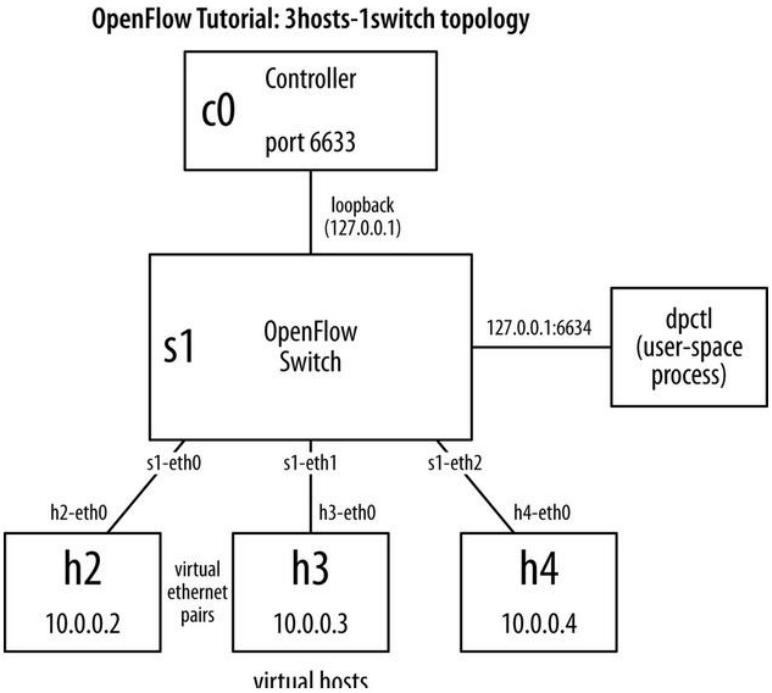

Рисунок 1.2 – Архітектура Mininet: взаємодія хостів, OVS-комутаторів та SDN-контролера [6]

GNS3 (Graphical Network Simulator-3) є більш комплексним інструментом, що підтримує емуляцію реального мережевого обладнання (Cisco IOS, Juniper JunOS) шляхом запуску образів пристроїв у QEMU/KVM. GNS3 дозволяє будувати складні гетерогенні топології, що поєднують традиційне та SDN-обладнання. Проте для суто SDN-досліджень GNS3 є надлишково складним, і Mininet залишається більш зручним вибором.

CORE (Common Open Research Emulator) це ще один емулятор на базі Linux-namespaces, розроблений для дослідження мережевих протоколів. Він підтримує безпосередній запуск реальних мережевих сервісів і демонів (OSPF, BGP, SDN-агентів) у емульованих вузлах. EstiNet та OMNeT++ є прикладами симуляторів, де мережа моделюється математично. OMNeT++ це подієво-орієнтований фреймворк із розвиненою бібліотекою модулів (INET), що підтримує SDN-розширення. Перевага симуляторів полягає у вищій масштабованості (можна моделювати тисячі вузлів), тоді як емулятори обмежені ресурсами хост-машини.

Порівняння інструментів за ключовими характеристиками наведено в таблиці 1.1

Таблиця 1.1 – Порівняння інструментів за ключовими характеристиками

Інструмент	Тип	Підтримка SDN	Масштабованість
Mininet	Емулятор	OpenFlow, ONOS, Ryu	Середня (десятки-сотні вузлів)
GNS3	Емулятор	Часткова (через OVS)	Середня
CORE	Емулятор	Є	Середня
OMNeT++/INET	Симулятор	Є (розширення)	Висока (тисячі вузлів)
EstiNet	Симулятор	Є	Висока

1.4 Мережева телеметрія: принципи та методи збору

Мережева телеметрія (network telemetry) є сукупністю методів і технологій для автоматизованого збору, передавання та обробки даних про стан і поведінку мережевої інфраструктури в режимі реального часу. На відміну від традиційного підходу SNMP-опитування (polling), сучасна телеметрія базується на потоковому передаванні даних (streaming telemetry), де мережеві пристрої самі «проштовхують» (push) метрики до колекторів за підпискою.

Основні джерела телеметрії в SDN-середовищі включають кілька категорій вимірювань. Статистика портів та інтерфейсів (port/interface counters) містить такі показники, як tx_bytes, rx_bytes, tx_packets, rx_packets, tx_errors, rx_errors, tx_dropped, rx_dropped. Ці лічильники відображають обсяги переданих даних та наявність помилок. Статистика потоків (flow statistics) включає лічильники пакетів і байтів для кожного запису у flow table, час життя flow ruleта кількість попадань (match hits). Вимірювання RTT та якості каналу (probe-based telemetry) охоплює затримку (latency/RTT), втрати пакетів (packet loss), варіацію затримки (jitter). Ці показники зазвичай отримуються за допомогою активних вимірювань а саме надсилання probe-пакетів між вузлами.

In-band Network Telemetry (INT) є сучасним підходом до збору телеметрії, де метадані про стан мережі вбудовуються безпосередньо у заголовки пакетів даних. Кожен комутатор на шляху пакета додає до нього інформацію про поточний стан

черг, час обробки, ідентифікатор порту. Отримавши пакет, колектор може відновити повну картину проходження пакета по мережі, включаючи мікро-затримки на кожному транзитному вузлі. INT забезпечує безпрецедентну точність телеметрії, але потребує підтримки з боку мережевих пристроїв.

У контексті SDN-дослідження на базі Mininet основними методами збору телеметрії є: опитування лічильників OVS через утиліту ovs-ofctl dump-ports і ovs-ofctl dump-flows; активні probe-вимірювання за допомогою ping (RTT, packet loss) та iperf (throughput); підписка на OpenFlow-події (port_status, flow_removed) через контролер. Зібрані метрики традиційно зберігаються в базах даних часових рядів InfluxDB або Prometheus, а візуалізуються в Grafana.

Важливим аспектом мережевої телеметрії є питання якості зібраних даних та наявності помилок вимірювання. У реальних SDN-середовищах можуть виникати втрати телеметричних повідомлень, затримки їх доставки або неконсистентність показників через асинхронність збору даних. Додатково, активні методи вимірювання (наприклад, probe-трафік) можуть вносити похибки через вплив на реальний трафік мережі та зміну її навантаження. Таким чином, при проєктуванні систем телеметрії необхідно враховувати компроміс між точністю вимірювань, накладними витратами та масштабованістю рішення (рис. 1.3).

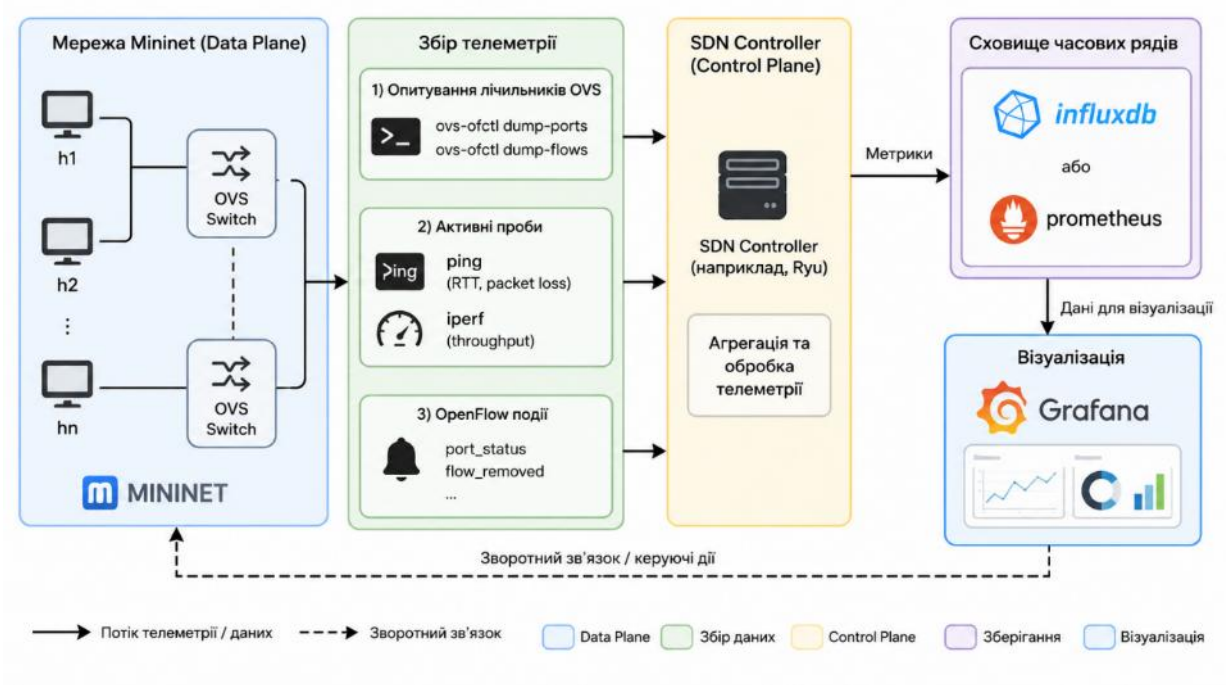

Рисунок 1.3 – Схема збору та обробки телеметрії в SDN-середовищі

1.5 Телеметрія як основа оптимізації трафіку в SDN

Зібрані телеметричні дані є інформаційною основою для оптимізації мережевого трафіку. У традиційних мережах рішення про маршрутизацію приймаються протоколами (OSPF, BGP) на основі статичних метрик (вартість лінку, пропускна здатність), що не враховують динамічного стану навантаження. SDN, натомість, дозволяє динамічно змінювати правила пересилання на основі актуальної телеметрії тобто реалізовувати traffic engineering з урахуванням реального стану мережі.

Фундаментальним показником для виявлення перевантаження є утилізація лінку (link utilization) це відношення поточного трафіку до номінальної пропускної здатності. Коли утилізація перевищує критичний поріг (зазвичай 70-80 %), зростають черги в буферах комутаторів, що призводить до збільшення затримки та зростання ймовірності втрати пакетів. Телеметрія дозволяє відстежувати цей показник у реальному часі й ініціювати переналаштування маршрутів до того, як перевантаження стане критичним. Дослідження підтверджують, що поєднання SDN із ML-аналізом телеметрії суттєво скорочує час виявлення перевантаження порівняно з традиційними порогово-евристичними підходами [1].

Типові стратегії оптимізації трафіку на основі телеметрії включають perouting (перенаправлення) переміщення одного або кількох потоків на менш завантажені маршрути при виявленні перевантаженого лінку; load balancing (балансування навантаження) це розподіл трафіку між кількома рівноцінними шляхами для рівномірного використання ресурсів; QoS prioritization (пріоритизацію) це забезпечення пріоритетного обслуговування чутливих до затримки потоків (відеоконференції, VoIP) при обмеженій пропускній здатності.

Ефективна оптимізація трафіку в SDN вимагає не лише точної телеметрії, а й механізмів інтелектуального аналізу, здатних відрізняти короткочасні сплески від системного перевантаження та пропонувати відповідні коригувальні дії [7]. Саме тому поєднання телеметрії з машинним навчанням є перспективним напрямом, що забезпечує адаптивне та проактивне управління мережею.

1.6 Огляд підходів до використання ML у SDN для оптимізації трафіку

Машинне навчання широко застосовується в SDN-середовищах для вирішення задач класифікації трафіку, виявлення аномалій, прогнозування перевантажень та оптимізації маршрутизації. Ключова перевага ML-підходів порівняно з евристичними правилами полягає в здатності виявляти складні нелінійні залежності між телеметричними показниками та станом мережі без явного кодування цих залежностей.

Серед алгоритмів на основі дерев рішень Random Forest та Gradient Boosting демонструють особливо високі результати в задачах класифікації стану мережі за телеметрією. Random Forest є ансамблем дерев рішень, що навчаються на різних підвибірках тренувального набору, а підсумкова класифікація визначається більшістю голосів. Дослідження підтверджують, що Random Forest досягає точності 94-97 % при класифікації станів SDN-мережі на основі телеметричних ознак, перевершуючи Logistic Regression і SVM у більшості сценаріїв [8].

Методи глибокого навчання з підкріпленням застосовуються для задач адаптивної маршрутизації, де агент навчається приймати рішення про розподіл потоків шляхом взаємодії з мережевим середовищем. Зокрема, у дослідженнях розглядається DRL-агент для маршрутизації в SDN, що навчається мінімізувати середню затримку та досягає на 15-23 % кращих результатів порівняно з традиційними алгоритмами Dijkstra та ECMP при нерівномірному навантаженні [3]. Інші роботи демонструють застосування DRL для оптимізації часу обслуговування потоків з адаптацією до динамічних змін трафіку [9].

Систематичний огляд підходів ML у SDN охопив понад 80 досліджень і виділив основні напрями: класифікація трафіку (понад 40 % робіт), виявлення DDoS-атак (близько 25 %), оптимізація маршрутизації (близько 20 %) та управління QoS (близько 15 %). Попри різноманіття запропонованих підходів, основним обмеженням залишається складність переходу від симульованих або лабораторних умов до реальних мережевих середовищ [1].

Нові дослідження з використанням LLM (Large Language Models) у SDN-управлінні [10] відкривають перспективи інтерпретованого аналізу телеметрії

та генерації мережевих правил природньою мовою. Однак ці підходи поки що знаходяться на ранній стадії і потребують подальшого дослідження щодо затримок inference та надійності виведення.

Висновки до розділу: у першому розділі розглянуто теоретичні засади SDN-архітектури: тришаровий поділ на рівні інфраструктури, управління та застосунків; роль протоколу OpenFlow як стандартного southbound API; класифікацію
SDN-контролерів (ONOS, Ryu, OpenDaylight). Виявлено, що Mininet є де-факто стандартом емуляції SDN для дослідницьких цілей завдяки поєднанню реалістичності (реальний мережевий стек) та ресурсоефективності.

Проаналізовано принципи та методи мережевої телеметрії: статистику портів та потоків OVS, активні probe-вимірювання RTT/loss/jitter, концепцію In-band Network Telemetry. Встановлено, що телеметрія виступає інформаційною основою для інтелектуальної оптимізації трафіку. Огляд ML-підходів у SDN показав, що ансамблеві методи (насамперед Random Forest) забезпечують найвищу точність класифікації стану мережі, тоді як DRL є перспективним для задач адаптивної маршрутизації. Отримані теоретичні знання безпосередньо використовуються в наступних розділах при проектуванні та реалізації інтелектуальної SDN-системи.

РОЗДІЛ 2
ПІДХОДИ ML ДЛЯ АНАЛІЗУ ТЕЛЕМЕТРІЇ ТА ПРИЙНЯТТЯ РІШЕНЬ В SDN

2.1 Телеметрія як джерело даних для інтелектуального аналізу мережі

Ефективне керування сучасними мережами неможливе без систематичного збору та аналізу даних про їхній стан. У традиційних мережах ці дані здобуваються переважно вручну через SNMP-опитування пристроїв або перегляд журналів. У SDN-середовищі ситуація принципово інша: централізований контролер має постійний доступ до стану всіх комутаторів, що робить збір телеметрії системним і автоматизованим процесом.

Під мережевою телеметрією розуміють безперервний або подієво-ініційований збір метрик, що характеризують передавання трафіку та стан мережевих пристроїв. У контексті SDN основними джерелами телеметрії є: лічильники портів Open vSwitch (tx_bytes, rx_bytes, tx_packets, rx_packets, dropped), статистика OpenFlow-потоків (flow duration, packet count, byte count), а також активні probe-вимірювання ping-зонди для оцінювання RTT, jitter і packet loss між вузлами.

На основі сирих лічильників формується набір похідних метрик, придатних для ML-аналізу. Найважливіші з них наведено у таблиці 2.1.

Таблиця 2.1 – Основні телеметричні метрики SDN-мережі

Метрика	Джерело	Роль у ML-аналізі
Latency (RTT)	Probe / ping	Основний індикатор перевантаження
Packet loss	Probe / OVS-counters	Сигнал деградації каналу
Jitter	Probe	Нестабільність передавання
Throughput	OVS tx_bytes / інтервал	Рівень утилізації каналу
Link load (%)	OVS tx_bytes / capacity	Завантаженість лінку
Dropped packets	OVS drop counters	Переповнення черги
RTT (end-to-end)	Probe / ICMP	Загальна якість шляху
Flow count	OpenFlow flow stats	Кількість потоків

Такий набір метрик охоплює три рівні аналізу: якість каналу (latency, jitter, loss), використання ресурсів (throughput, link load) та поведінку потоків (flow count, drops). Поєднання цих рівнів дозволяє ML-моделі виявляти не лише явні

перевантаження, а й ранні ознаки деградації, що є ключовою перевагою порівняно з пороговими підходами [1] (рис. 2.1).

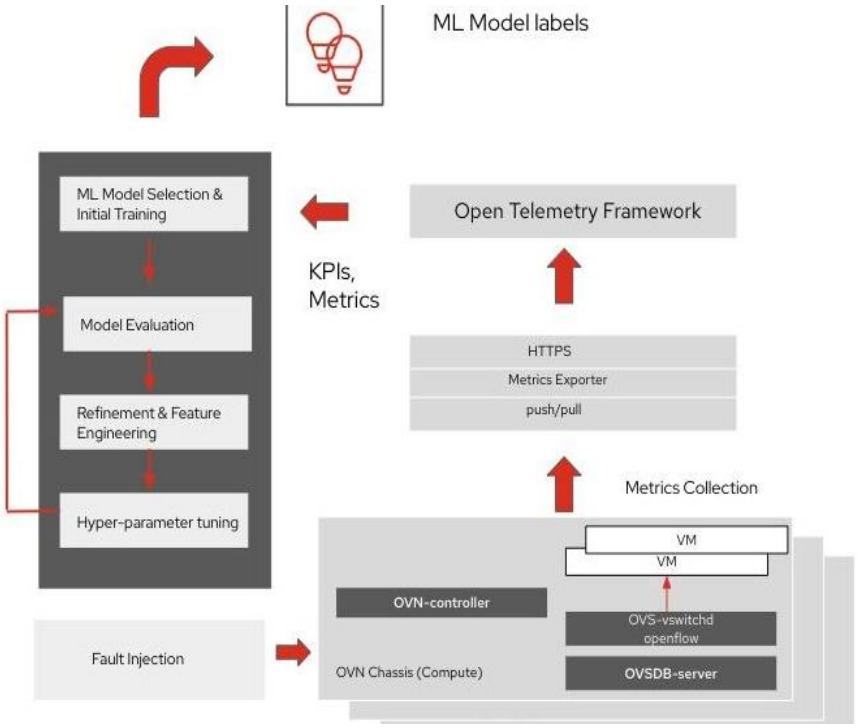

Рисунок 2.1 – Схема збору телеметрії в SDN-середовищі: від OVS-лічильників і probe-вимірювань через InfluxDB до ML-модуля [11]

2.2 Підходи машинного навчання до класифікації стану мережі

Машинне навчання в задачах аналізу мережевого трафіку використовується у кількох напрямах: класифікація трафіку за типом, виявлення аномалій, прогнозування навантаження та класифікація стану мережі. У цій роботі центральною є задача класифікації стану мережі, визначення того, до якого функціонального режиму (normal, warning, congested) належить поточний стан на основі телеметричних ознак.

Систематичний аналіз підходів ML у SDN-дослідженнях засвідчив, що найпоширенішими є методи на основі дерев рішень, ансамблеві моделі та нейронні мережі, тоді як лінійні класифікатори використовуються переважно як базові моделі для порівняння [1]. Окремі дослідження підтверджують, що Random Forest

демонструє стабільно високу точність класифікації мережевого стану при роботі з телеметрією SDN-середовищ [7].

Порівняльну характеристику основних ML-методів, придатних для класифікації стану SDN-мережі, наведено у таблиці 2.2.

Таблиця 2.2 – Порівняння ML-методів для класифікації стану SDN-мережі

Метод	Точність (typical)	Інтерпретованість	Швидкість інференсу	Стійкість до шуму
Logistic Regression	85-90 %	Висока	Дуже висока	Низька
Decision Tree	88-92 %	Висока	Висока	Середня
Random Forest	92-96 %	Середня	Висока	Висока
Gradient Boosting	93-97 %	Низька	Середня	Висока
SVM	87-93 %	Низька	Середня	Середня
MLP (Neural Net)	91-95 %	Дуже низька	Середня	Висока

Для практичної реалізації у цій роботі обрано Logistic Regression як базову (інтерпретовану, швидку) та Random Forest як основну модель, оскільки остання демонструє найкращий баланс між точністю, стійкістю до шуму у вхідних даних і швидкістю класифікації в режимі реального часу [8] (рис. 2.2).

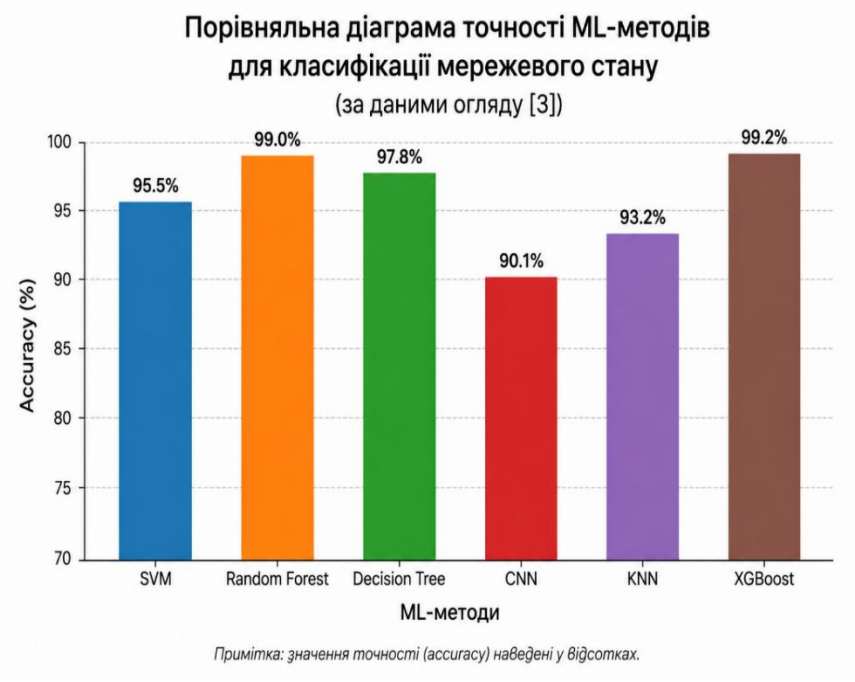

Рисунок 2.2 – Порівняльна діаграма точності ML-методів для класифікації мережевого стану

З метою верифікації теоретичних оцінок, наведених у таблиці 2.2, у межах цієї роботи проведено практичне навчання та порівняння обох обраних моделей на реальних телеметричних даних, зібраних у середовищі Mininet. Навчальна вибірка містила 98 563 зразки, розмічені за трьома класами стану мережі. Результати навчання наведено у таблиці 2.3.

Таблиця 2.3 – Результати навчання ML-моделей на реальних телеметричних даних

Модель	Accuracy	F1-weighted	Час навчання
Logistic Regression	92,9 %	93,1 %	~2 с
Random Forest	99,3 %	99,3 %	~18 с

Отримані результати підтверджують теоретичні оцінки з літератури та свідчать про доцільність вибору Random Forest як основної моделі. Різниця в точності між моделями складає 6,4 відсоткових пункти, тоді як час навчання Random Forest є прийнятним для офлайн-навчання і не впливає на швидкість класифікації в режимі реального часу. Варто зазначити, що навіть базова модель Logistic Regression досягла точності 92,9%, що підтверджує високу інформативність сформованого набору телеметричних ознак незалежно від складності алгоритму [7].

2.3 Формування ознак та підготовка навчальної вибірки

Поряд із класичними методами ML у задачах керування мережею дедалі більшу увагу привертає використання великих мовних моделей (LLM). Дослідження засвідчують, що LLM можуть виконувати роль аналітичного шару, здатного інтерпретувати структуровані телеметричні дані, генерувати рекомендації щодо дій та пояснювати прийняті рішення природною мовою, що є принциповою перевагою перед моделями без пояснюваності [10].

В окремих роботах запропоновано архітектуру, де LLM отримує на вхід структурований опис поточного стану мережі у вигляді JSON-об’єкта з телеметричними показниками і повертає рекомендацію щодо зміни flow rules або пріоритизації трафіку [12]. При цьому наголошується, що такий підхід потребує

обов'язкової валідації відповіді моделі перед застосуванням, оскільки LLM може генерувати синтаксично коректні, але семантично помилкові рекомендації (рис. 2.3).


```
{
  "requested_mode": "normal",
  "prediction": {
    "predicted_state": "congested",
    "confidence": 0.994,
    "class_probabilities": {
      "congested": 0.994,
      "normal": 0.000,
      "warning": 0.006
    }
  },
  "gpt_advisory": {
    "network_state": "congested",
    "confidence_assessment": "high",
    "recommended_action": "reroute_selected_flows",
    "reasoning_summary": "ML state=congested, hot_link=s2-s3,
                          utilization=69.4%, latency=433.7 ms,
                          loss=22.2%",
    "priority_level": "high"
  }
}
```


Рисунок 2.3 – Приклад виводу GPT Advisory Feed у режимі congested

Наведений приклад демонструє роботу GPT Advisory Feed в умовах критичного перевантаження. ML-класифікатор визначає стан мережі як congested із впевненістю 99,4 % – імовірність нормального стану дорівнює нулю, що свідчить про однозначну класифікацію. Телеметрія підтверджує деградацію: завантаженість лінку s2-s3 становить 69,4 %, затримка 433,7 мс, рівень втрат пакетів 22,2 %. На основі цих даних GPT Advisory Feed формує рекомендацію reroute_selected_flows із високим пріоритетом втручання, що ініціює передачу керуючої команди до decision engine для перенаправлення трафіку через альтернативний маршрут. Це наочно ілюструє практичну роль GPT-модуля як аналітичного шару між ML-класифікатором і системою прийняття рішень [10]

У контексті цієї роботи GPT API розглядається як опціональний аналітичний компонент, що може доповнювати ML-класифікатор інтерпретацією граничних ситуацій це передусім випадків, коли телеметричні показники знаходяться на межі між warning і congested і класифікатор демонструє знижену впевненість у відповіді (рис. 2.4-2.5).

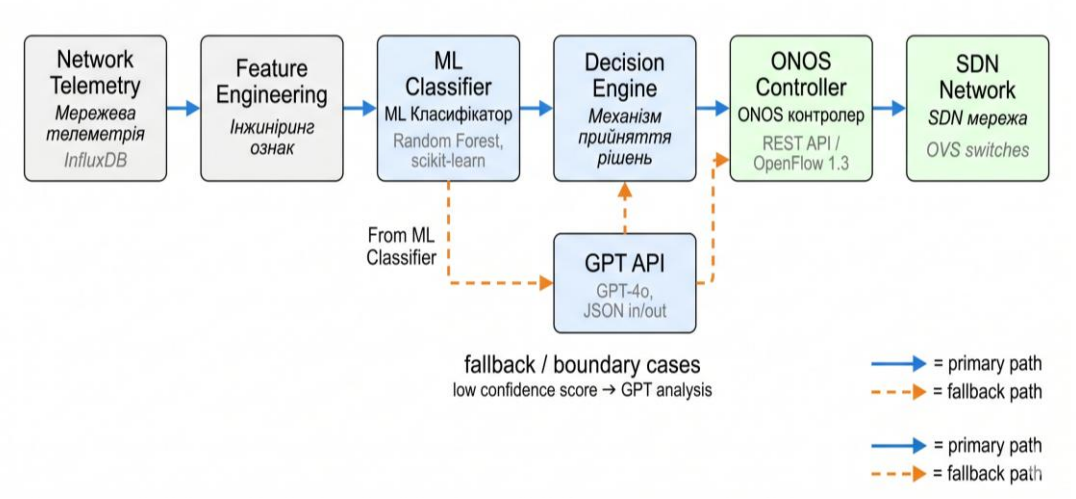

Рисунок 2.4 – Схема інтеграції GPT API у контур керування SDN

```
FEATURE_COLUMNS = [
    "latency_ms",
    "packet_loss_pct",
    "jitter_ms",
    "throughput_mbps",
    "bandwidth_usage_mbps",
    "flow_count",
    "link_utilization_pct",
    "round_trip_time_ms",
    "dropped_packets",
    "tx_bytes",
    "rx_bytes",
    "tx_packets",
    "rx_packets",
]

FEATURE_NOISE_SCALE = {
    "latency_ms":           0.08,    # ±8%
    "packet_loss_pct":      0.10,    # ±10%
    "jitter_ms":            0.12,    # ±12%
    "throughput_mbps":      0.05,    # ±5%
    "flow_count":           0.02,    # ±2%
    "link_utilization_pct": 0.06,
    "dropped_packets":      0.15,    # ±15%
}
```

Рисунок 2.5 – Визначення вхідного набору ознак та коефіцієнтів шуму для аугментації тренувальних даних

Коефіцієнти відносного шуму встановлено дослідним шляхом з урахуванням природної нестабільності кожної метрики: найвищий рівень шуму (±15 %)

визначено для dropped_packets як найбільш випадкового показника, найнижчий (±2 %) – для flow_count, що змінюється повільно і передбачувано.

Розподіл зразків за класами у сформованій навчальній вибірці наведено у таблиці 2.4.

Таблиця 2.4 – Розподіл зразків за класами стану мережі у навчальній вибірці

Клас стану	Кількість зразків	Частка від загального
congested	53 693	54,5 %
normal	31 279	31,7 %
warning	13 591	13,8 %
Всього	98 563	100 %

2.4 Використання LLM як аналітичного компонента в SDN

Ефективність ML-класифікатора значною мірою залежить від якості вхідного набору ознак (feature set). У задачах аналізу мережевої телеметрії сирі лічильники потребують попередньої обробки, оскільки безпосереднє використання абсолютних значень tx_bytes чи rx_bytes без контексту не несе достатньої інформації про стан мережі.

Підготовка ознак включає кілька етапів. По-перше, з абсолютних лічильників обчислюються похідні показники за вікном часу: наприклад, швидкість передавання (bytes/s), відсоток завантаженості лінку відносно його номінальної пропускної здатності, частота скидання пакетів. По-друге, для probe-метрик (latency, jitter, loss) обчислюються статистичні агрегати за вікном: середнє, максимум, стандартне відхилення. По-третє, всі ознаки нормалізуються до єдиного діапазону, що підвищує стабільність навчання.

Розмітка вибірки за класами (normal / warning / congested) виконується на основі заздалегідь визначених порогових діапазонів метрик, встановлених з урахуванням характеристик конкретної топології. Такий підхід відповідає методології, описаній у [8] та [13], де навчальна вибірка формується безпосередньо в процесі емуляції мережі з контрольованими параметрами навантаження.

2.5 Архітектура прийняття рішень: від класифікації до керуючого впливу

Результат роботи ML-модуля або LLM-компонента є лише проміжним етапом системи. Кінцевою метою є формування конкретної керуючої дії в SDN-мережі. Цей перехід від аналітичного висновку до мережевого рішення реалізується через модуль прийняття рішень decision engine.

Логіка decision engine будується на правилах відповідності між класом стану мережі та типом реакції. Якщо стан класифікується як normal, тоді система не втручається у поточну маршрутизацію. При warning можуть застосовуватися м’які механізми: підвищення пріоритету чутливого до затримки трафіку через DSCP-маркування або попереднє резервування альтернативного маршруту. При congested ініціюється активне перенаправлення потоків через менш завантажений шлях за допомогою модифікації OpenFlow flow rules через ONOS REST API.

Такий підхід, де ML/LLM виконують аналіз, а decision engine реалізує відповідну дію через SDN-контролер, відповідає концепції замкненого контуру керування (closed-loop control), яка є провідною парадигмою в сучасних дослідженнях адаптивних мережевих систем [3]. Ключовою властивістю такої архітектури є автономність − система виявляє перевантаження та застосовує коригувальні дії без участі адміністратора, що принципово відрізняє її від традиційних систем моніторингу, які лише фіксують проблему, але не усувають її автоматично. Додатковою перевагою decision engine є можливість централізованого застосування політик керування для всієї мережі незалежно від кількості комутаторів та потоків. Завдяки інтеграції з ONOS контролер отримує змогу оперативно змінювати правила маршрутизації відповідно до поточного стану мережі та результатів ML-аналізу. Це дозволяє скоротити час реакції на перевантаження та мінімізувати деградацію якості обслуговування трафіку. Важливою особливістю архітектури є також модульність, оскільки ML-класифікатор, GPT Advisory Feed та decision engine можуть незалежно вдосконалюватися або замінюватися без зміни загальної структури системи. У

перспективі така архітектура може бути розширена підтримкою прогнозування перевантажень, коли система реагуватиме не лише на поточний стан мережі, а й на ймовірність майбутньої деградації. Це створює основу для побудови повністю автономних self-healing SDN-мереж із проактивним керуванням трафіком. (рис. 2.6).

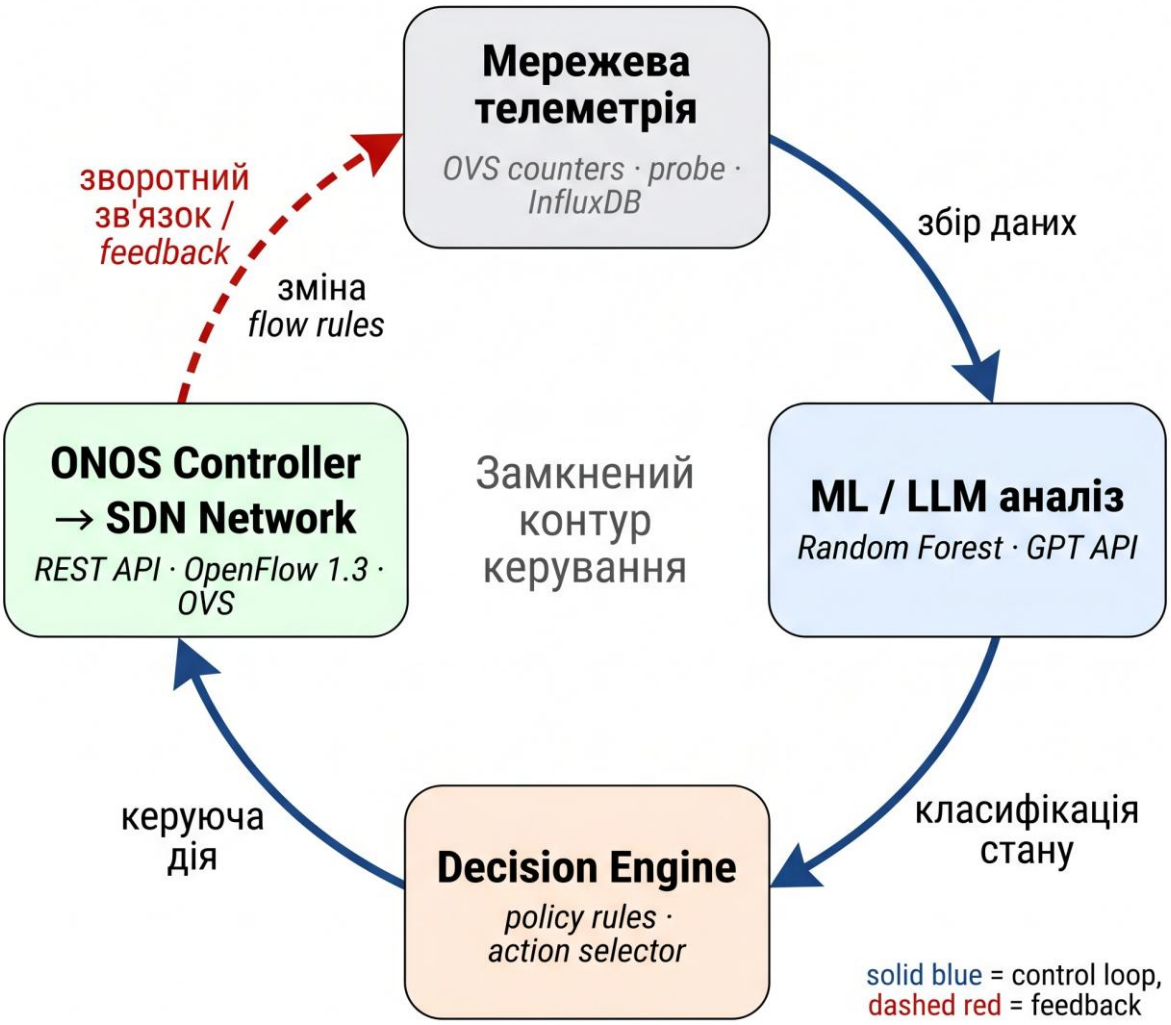

Рисунок 2.6 – Замкнений контур керування SDN-мережею

2.6 Узагальнення підходів та обґрунтування вибору методів

Аналіз підходів ML для аналізу телеметрії та прийняття рішень у SDN дозволяє сформулювати такі висновки. По-перше, телеметрія SDN-мережі є достатньо інформативним джерелом для класифікації стану без аналізу вмісту

пакетів, що є перевагою з точки зору продуктивності та конфіденційності. По-друге, серед класичних ML-методів Random Forest демонструє найкращий баланс між точністю та обчислювальною ефективністю для задач реального часу. По-третє, LLM може виконувати роль пояснювального та аналітичного шару для граничних ситуацій, але потребує обов’язкової валідації відповідей. По-четверте, ефективність системи визначається не лише якістю класифікатора, а й архітектурою зв’язку між ML-висновком і керуючою дією тобто реалізацією замкненого контуру.

Отримані теоретичні узагальнення стають підґрунтям для практичної реалізації інтелектуальної SDN-системи, описаної у третьому розділі.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ SDN-СИСТЕМИ ОПТИМІЗАЦІЇ МЕРЕЖЕВОГО ТРАФІКУ

3.1 Постановка задачі практичної реалізації

Практична реалізація проєкту − створення прототипу інтелектуальної SDN-системи, здатної не лише моделювати мережеве середовище, а й виконувати аналіз телеметричних показників у процесі реальної роботи мережі, визначати її поточний стан і застосовувати оптимізаційні дії у випадку погіршення характеристик передавання трафіку. На відміну від підходів, у яких дослідження обмежується готовими наборами даних, у цій роботі акцент зроблено на формуванні телеметрії безпосередньо під час виконання експериментальних сценаріїв у Mininet. Це дає можливість досліджувати поведінку SDN-мережі в умовах, наближених до реального функціонування, коли показники затримки, втрат, jitter і завантаження лінків формуються як наслідок взаємодії трафіку, топології та налаштувань мережі, а не задаються наперед у штучному вигляді.

Практична задача полягає в тому, щоб реалізувати систему, яка виконує чотири взаємопов’язані функції:

– система має забезпечувати побудову керованої SDN-топології з можливістю створення альтернативних маршрутів передавання трафіку;

– система повинна формувати реальне навантаження різних типів і підтримувати сценарії навмисного погіршення стану мережі;

– необхідно організувати збір телеметрії та формування на її основі набору ознак для ML-аналізу;

– система має використовувати результат класифікації для вибору оптимізаційної дії, яка дозволяє зменшити негативний вплив перевантаження на ключові мережеві показники.

Таким чином, у межах практичної частини потрібно розв’язати не окрему задачу програмної реалізації, а комплексну інженерно-дослідницьку задачу, що об’єднує моделювання SDN-мережі, збір телеметрії, машинне навчання та

механізми мережевого керування. Саме такий підхід характерний для сучасних досліджень SDN, де машинне навчання використовується для класифікації трафіку, оцінювання стану мережі та адаптивного маршрутизаційного керування [3].

3.2 Архітектура розробленої системи

Розроблена система має модульну архітектуру, у межах якої кожен компонент виконує окрему функцію в загальному процесі моделювання, моніторингу та оптимізації мережевого трафіку. Основу системи становить середовище Mininet, у якому за допомогою Python-скриптів створюється та запускається віртуальна мережева топологія. У межах цієї топології Open vSwitch використовується як набір віртуальних SDN-комутаторів, що забезпечують передавання трафіку між вузлами та підтримують централізоване керування через контролер.

ONOS у розробленій системі виконує функції SDN-контролера, який підключається до створеної в Mininet мережі та забезпечує controller-level visibility. Його використання дає можливість спостерігати за топологією, станом з’єднань і загальною структурою мережі на рівні контролера. При цьому безпосереднє створення топології, запуск вузлів і конфігурування експериментального середовища здійснюються не засобами ONOS, а через Python-реалізацію в середовищі Mininet. Таким чином, ONOS у цій роботі використовується переважно як компонент контролю та візуалізації, а основна логіка побудови й запуску мережі реалізована на стороні Mininet.

Над рівнем емуляції мережі розташовано модуль генерації трафіку, який відповідає за формування TCP, UDP та probe-навантаження між хостами. Саме на цьому етапі створюються сценарії normal, warning і critical, що відрізняються інтенсивністю трафіку та параметрами деградації каналу. Паралельно з генерацією трафіку працює модуль збору телеметрії, який отримує значення лічильників інтерфейсів, характеристики потоків та результати probe-вимірювань. Зібрані дані

передаються до InfluxDB, де зберігаються як часові ряди, а Grafana використовується для оперативної візуалізації динаміки мережевих показників.

Окремим логічним рівнем системи є ML-модуль, що отримує підготовлений набір телеметричних ознак і виконує класифікацію стану мережі. Результат класифікації передається до decision engine, який визначає подальшу дію системи. Якщо мережа функціонує в межах норми, система зберігає поточну маршрутизацію. Якщо виявлено ознаки погіршення, decision engine може ініціювати м’яке або активне оптимізаційне втручання. Така модульна архітектура узгоджується з дослідженнями, у яких SDN розглядається як придатна основа для впровадження ML-механізмів аналізу трафіку, а централізований контроль спрощує інтеграцію аналітичних та керуючих компонентів [8] (рис. 3.1).

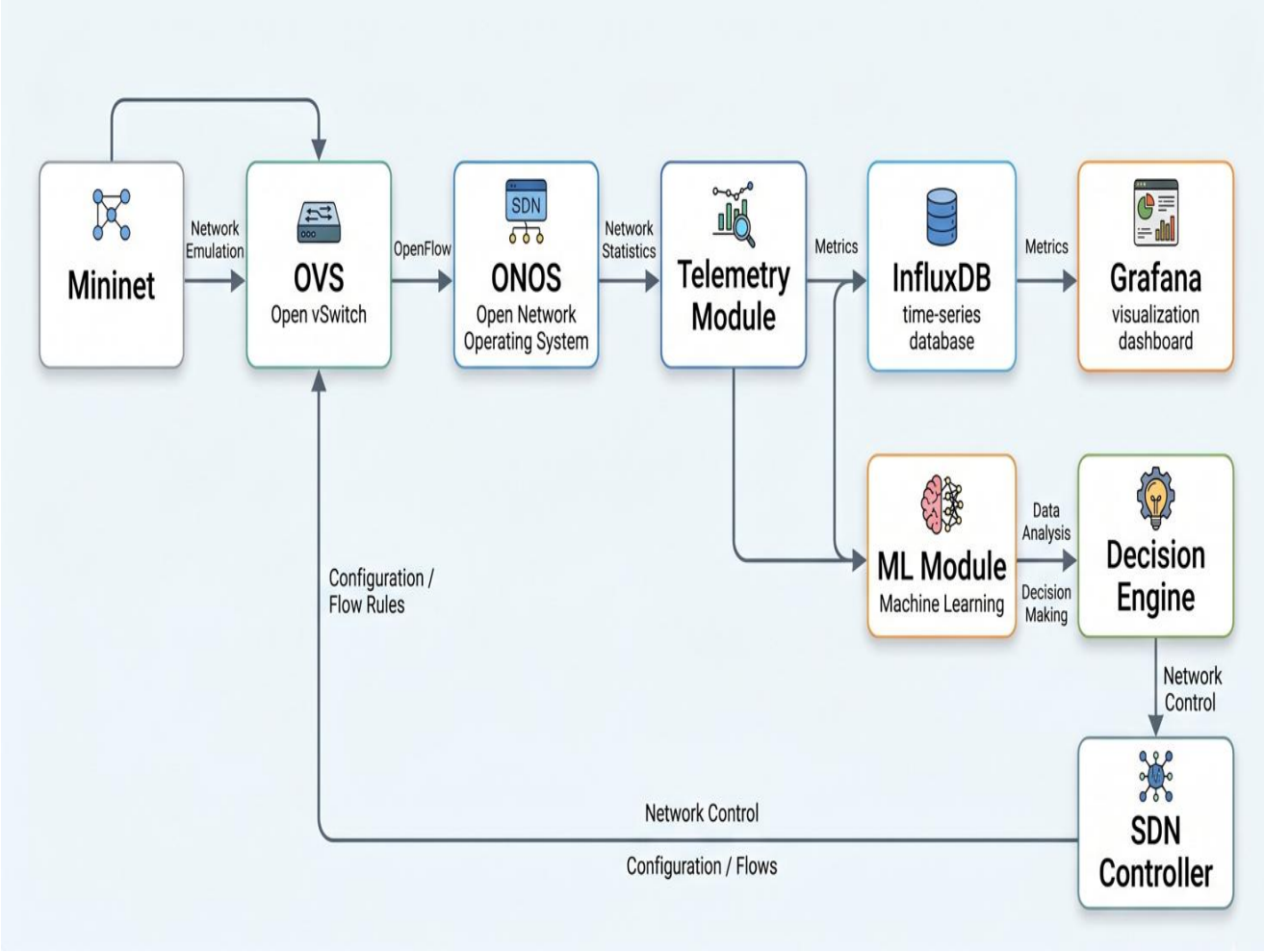

Рисунок 3.1 – Архітектура інтелектуальної SDN-системи оптимізації мережевого трафіку

3.3 Реалізація мережевої топології в середовищі Mininet

Для проведення експериментів реалізовано багатошляхову SDN-топологію в середовищі Mininet. Побудована топологія включає 8 комутаторів і 12 хостів, що дозволяє відтворити не спрощену локальну схему передавання даних, а більш структуроване мережеве середовище з кількома рівнями взаємодії. Два комутатори ядра виконують центральну роль у міжсегментному обміні трафіком, тоді як edge-комутатори забезпечують підключення кінцевих вузлів. Така побудова дозволяє одночасно зберігати відносну простоту моделювання та створювати умови для виникнення перевантажених ділянок і альтернативних маршрутів передавання.

Важливою особливістю топології є наявність резервних міжкомутаторних з'єднань. Це дозволяє досліджувати не лише базовий маршрут проходження потоків, а й сценарії reroute, коли частина трафіку може бути перенаправлена через інші лінки. Наявність кількох шляхів між сегментами мережі є принципово важливою для теми роботи, оскільки без цього неможливо оцінити ефективність автоматичної оптимізації маршрутизації. Крім того, така топологія дає змогу навмисно формувати вузькі місця на окремих лінках та спостерігати, як це впливає на загальний стан мережі.

Побудова топології в Mininet забезпечує повний контроль над структурою мережі, параметрами лінків і поведінкою вузлів. Це робить експериментальне середовище придатним для дослідження зв'язку між телеметрією, станом мережі та можливими діями SDN-контролера. Саме в такому контексті Mininet часто використовується в сучасних роботах як інструмент для перевірки SDN-архітектур, класифікації трафіку та експериментів з інтелектуальним керуванням мережею [3]. Пропускна здатність міжкомутаторних лінків цілеспрямовано варіювалася від 20 до 80 Мбіт/с, що дозволяло контрольовано формувати вузькі місця та відтворювати реалістичні умови перевантаження (рис. 3.2-3.3).

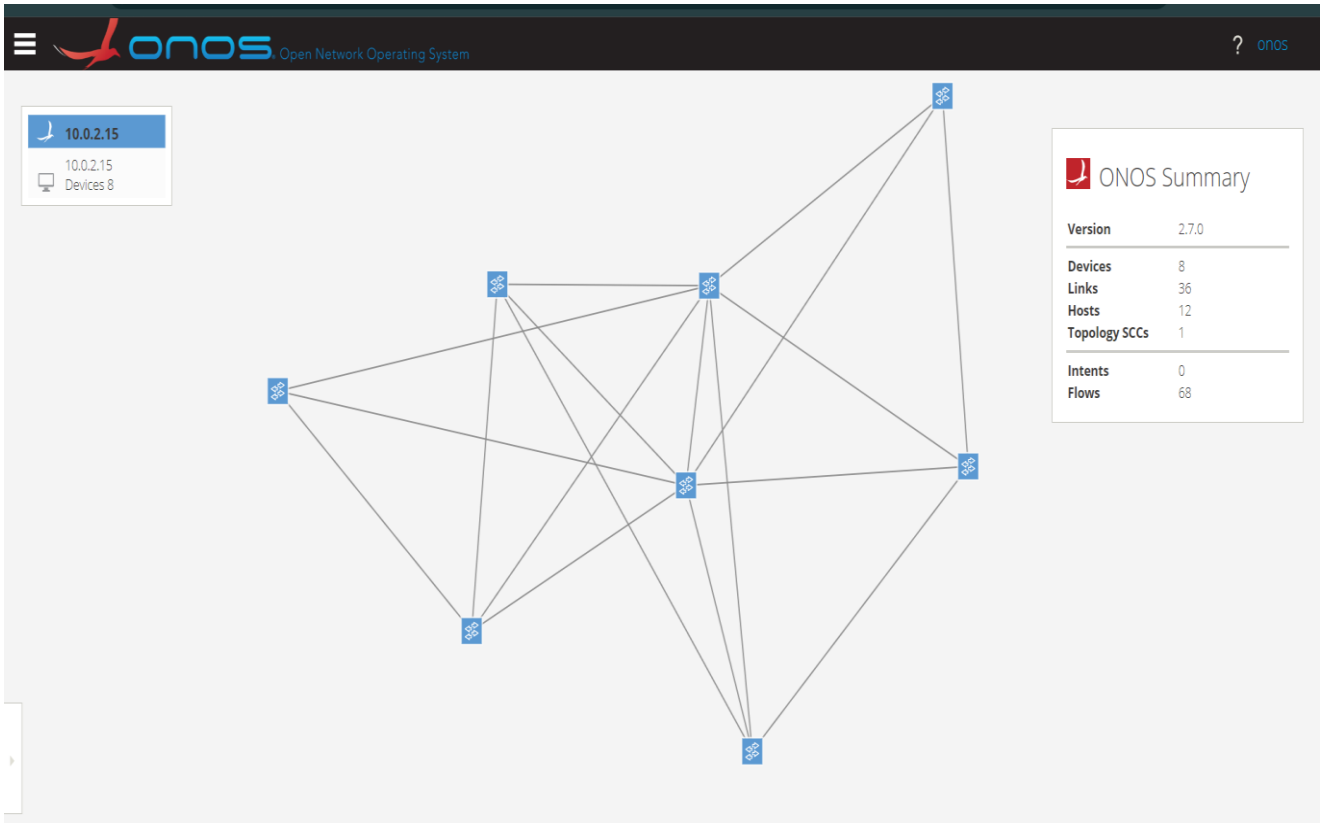

Рисунок 3.2 – Топологія мережі в інтерфейсі контролера ONOS

```
mininet@mininet-vm:~/smart-sdn$ cd ~/smart-sdn && sudo .venv/bin/python /tmp/mn_cli.py
*** Topology is UP - entering CLI
mininet> nodes
available nodes are:
h1 h10 h11 h12 h2 h3 h4 h5 h6 h7 h8 h9 onos s1 s2 s3 s4 s5 s6 s7 s8
mininet> links
h1-eth0<->s3-eth1 (OK OK)
h2-eth0<->s3-eth2 (OK OK)
h3-eth0<->s4-eth1 (OK OK)
h4-eth0<->s4-eth2 (OK OK)
h5-eth0<->s5-eth1 (OK OK)
h6-eth0<->s5-eth2 (OK OK)
h7-eth0<->s6-eth1 (OK OK)
h8-eth0<->s6-eth2 (OK OK)
h9-eth0<->s7-eth1 (OK OK)
h10-eth0<->s7-eth2 (OK OK)
h11-eth0<->s8-eth1 (OK OK)
h12-eth0<->s8-eth2 (OK OK)
s1-eth1<->s2-eth1 (OK OK)
s1-eth2<->s3-eth3 (OK OK)
s1-eth3<->s4-eth3 (OK OK)
s1-eth4<->s5-eth3 (OK OK)
s1-eth5<->s6-eth3 (OK OK)
s1-eth6<->s7-eth3 (OK OK)
s1-eth7<->s8-eth3 (OK OK)
s2-eth2<->s3-eth4 (OK OK)
s2-eth3<->s4-eth4 (OK OK)
s2-eth4<->s5-eth4 (OK OK)
s2-eth5<->s6-eth4 (OK OK)
s2-eth6<->s7-eth4 (OK OK)
s2-eth7<->s8-eth4 (OK OK)
s3-eth5<->s4-eth5 (OK OK)
s4-eth6<->s5-eth5 (OK OK)
s5-eth6<->s6-eth5 (OK OK)
s6-eth6<->s7-eth5 (OK OK)
s7-eth6<->s8-eth5 (OK OK)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10 h11 h12
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10 h11 h12
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10 h11 h12
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10 h11 h12
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10 h11 h12
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10 h11 h12
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10 h11 h12
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h11 h12
h11 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h12
h12 -> h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11
*** Results: 0% dropped (132/132 received)
mininet>
```

Рисунок 3.3 – Вивід команд nodes, links, pingall у Mininet CLI

Детальна структура розробленої мережі, що відображає ієрархію вузлів (рівні контролера, ядра та доступу), а також ілюструє логіку проходження трафіку з виділенням основного та альтернативного маршрутів між кінцевими хостами, наведена на рисунку 3.4.

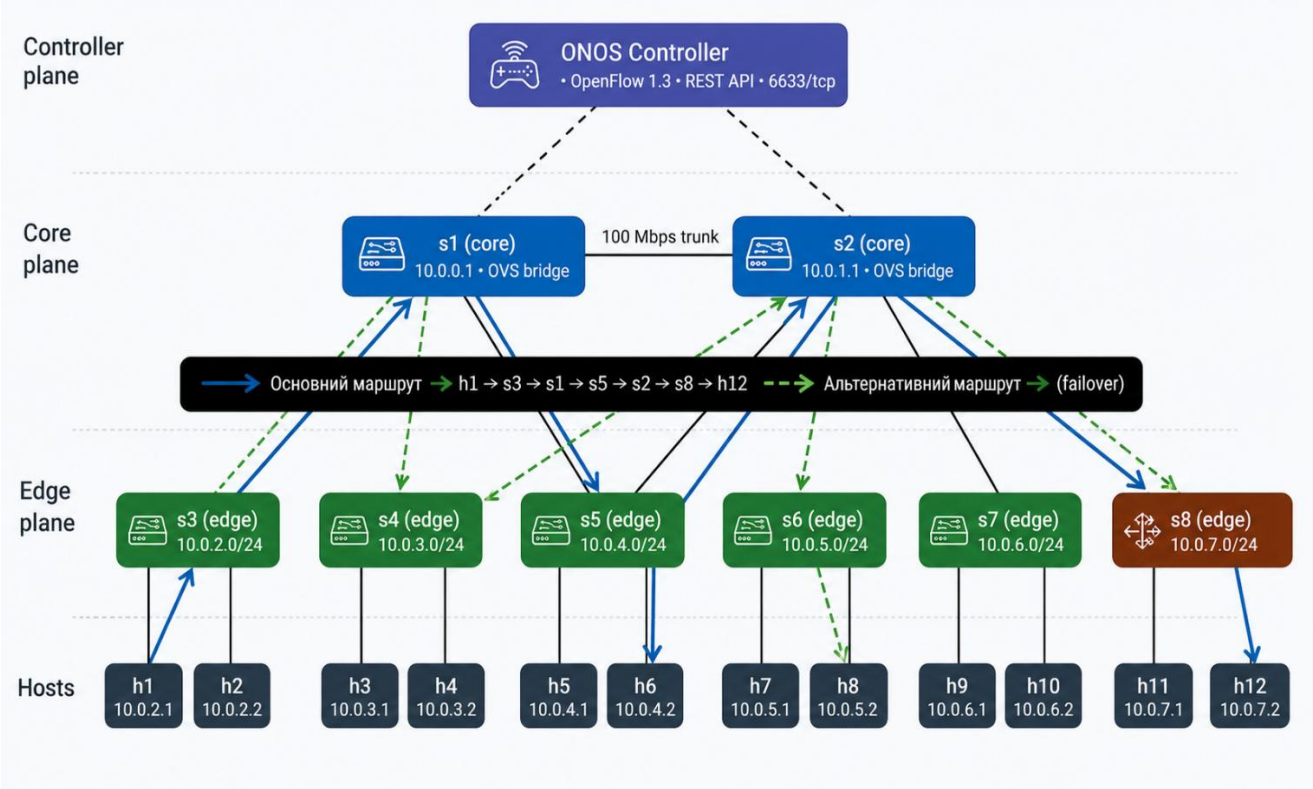

Рисунок 3.4 – Топологія досліджуваної SDN-мережі та схема руху трафіку.

3.4 Реалізація генерації трафіку та моделювання деградації мережі

Одним із ключових етапів практичної реалізації є організація мережевого навантаження, оскільки саме від нього залежать телеметричні показники, які надалі використовуються в ML-аналізі. У розробленій системі генерація трафіку побудована так, щоб відтворювати різні типи передавання даних і давати змогу спостерігати поведінку мережі в умовах нормального та погіршеного функціонування. Для цього використано TCP-потоки, UDP-потоки та probe-трафік на базі ping. TCP застосовується для моделювання інтенсивного передавання великих обсягів даних, UDP для імітації потокового навантаження,

чутливого до затримок і втрат, а ping − для оперативного вимірювання latency та packet loss.

Окрім формування базового трафіку, система підтримує навмисне моделювання деградації мережевого стану. Для цього використовуються механізми обмеження пропускної здатності, додавання затримки, jitter, packet loss, а також створення конкуруючих потоків, які перевантажують окремі ділянки топології. У результаті формується кілька режимів роботи мережі. У режимі normal показники залишаються в межах допустимих значень і не створюють передумов для активного втручання. У режимі warning з'являються ознаки погіршення, однак деградація ще не є критичною. У режимі critical погіршення параметрів мережі вже достатнє для того, щоб ініціювати оптимізаційну дію. За потреби додатково можуть використовуватися сценарії critical_noopt і critical_opt для порівняння системи без автоматичної оптимізації та з нею.

Такий підхід до організації навантаження є принципово важливим, оскільки дозволяє отримувати телеметрію в умовах реальної взаємодії потоків, а не на основі статичних або вручну сформованих значень. У наукових роботах із цієї тематики саме поєднання SDN-середовища, реального трафіку та контрольованої деградації розглядається як обґрунтована база для дослідження класифікації мережевого стану та оптимізації QoS [1,7].

3.5 Реалізація збору телеметрії та збереження даних

Після формування мережевого навантаження система здійснює збір телеметричних даних, які надалі використовуються як для оперативного спостереження за станом мережі, так і для підготовки вибірки для машинного навчання. Збір телеметрії організовано на основі двох основних джерел. Першим джерелом є лічильники інтерфейсів Open vSwitch, які дають змогу отримувати дані про обсяги переданих та прийнятих байтів, кількість пакетів, втрати на інтерфейсах і пов'язані з ними похідні характеристики. Другим джерелом є probe-вимірювання, що використовуються для оцінювання затримки, втрат пакетів і jitter між вузлами.

На основі цих даних формується набір метрик, який характеризує поточний стан мережі. До нього входять latency, packet loss, jitter, throughput, tx_bytes, rx_bytes, tx_packets, rx_packets, dropped_packets, flow_count, а також рівень завантаження окремих лінків. Важливо, що ці показники збираються автоматично в процесі роботи системи й відображають фактичну поведінку мережі під навантаженням. Отже, телеметрія виконує подвійну роль: з одного боку, вона є інструментом моніторингу, а з іншого це слугує основою для аналітичного модуля.

Для збереження телеметрії використовується InfluxDB, оскільки цей тип сховища добре підходить для часових рядів і дозволяє зручно працювати з послідовностями значень, що змінюються в часі. Візуалізація телеметрії здійснюється в Grafana, де можна відстежувати зміну затримки, втрат, пропускної здатності та інших показників під час експериментів. Така організація збору та зберігання даних відповідає сучасним підходам до телеметричного моніторингу SDN-середовищ і підтримує дослідження, у яких телеметрія використовується як основа для інтелектуального виявлення аномалій і деградації мережевого стану [5] (рис. 3.5-3.8).

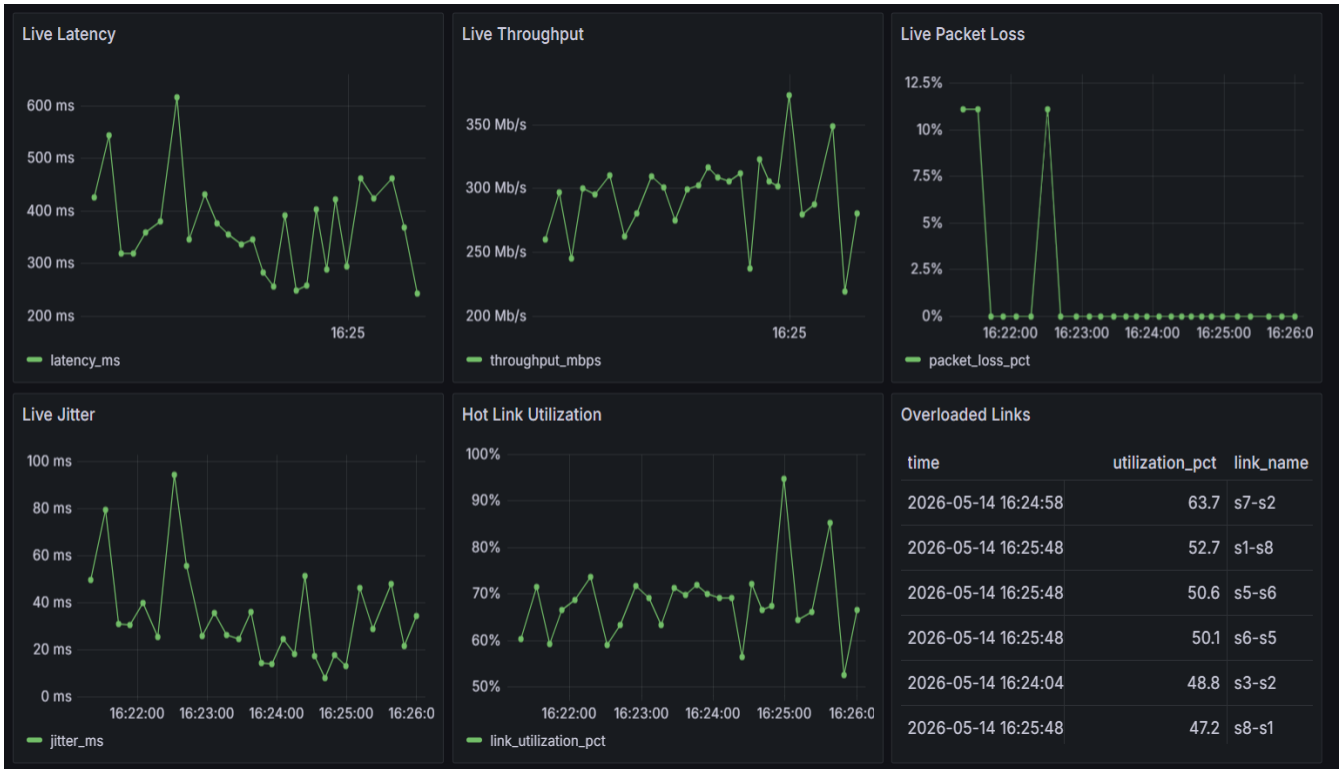

Рисунок 3.5 – Приклад візуалізації телеметричних показників у Grafana

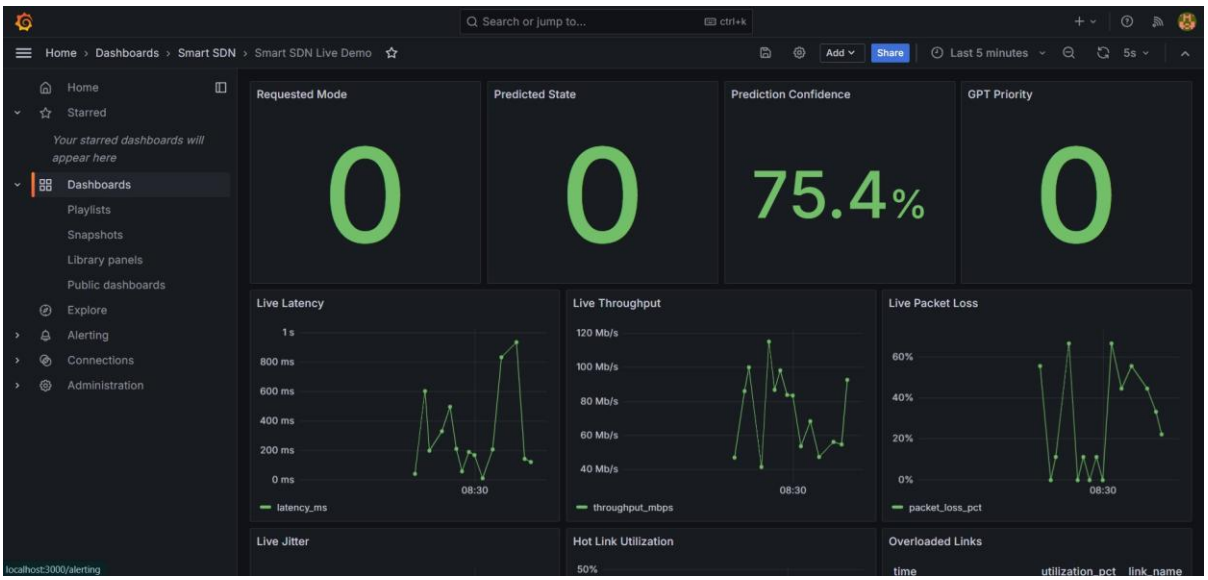

Рисунок 3.6 – Дашборд Grafana в режимі normal

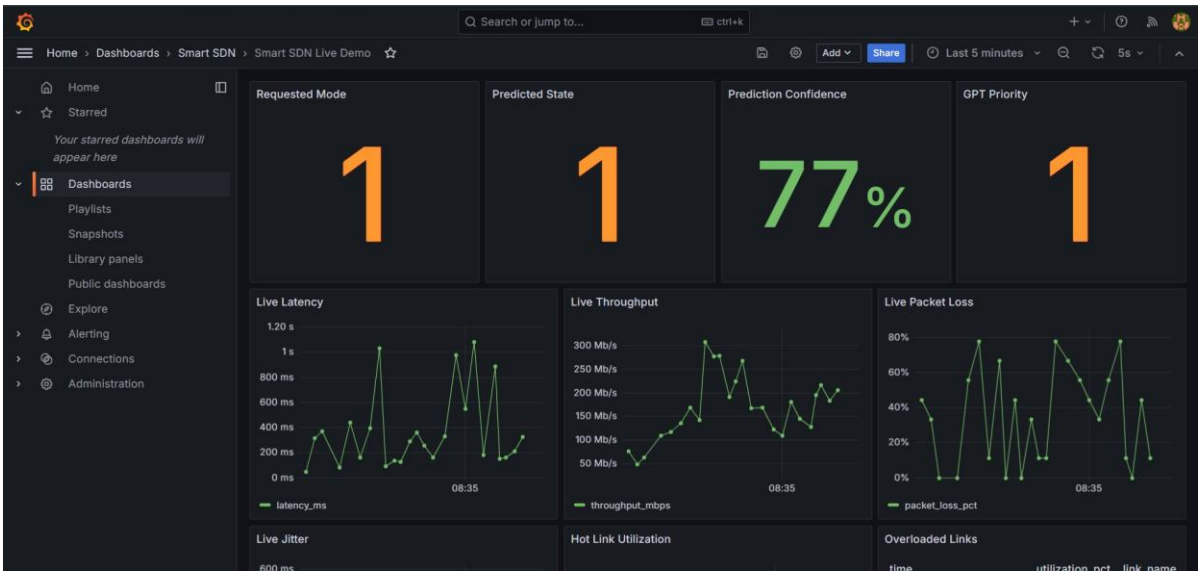

Рисунок 3.7 – Дашборд Grafana в режимі warning

Рисунок 3.8 – Дашборд Grafana в режимі critical

3.6 Реалізація ML-аналізу стану мережі

Зібрані телеметричні дані використовуються для реалізації ML-модуля, основним призначенням якого є класифікація поточного стану мережі. У цій роботі машинне навчання не розглядається як механізм прямого керування інфраструктурою. Його роль полягає в аналітичному опрацюванні телеметрії та визначенні того, до якого із заздалегідь встановлених станів належить поточна ситуація в мережі. Такий підхід є більш керованим і практичним для побудови інтелектуальної SDN-системи, оскільки дозволяє відокремити етап оцінювання стану від етапу прийняття керуючого рішення.

У межах розробленої системи використовується багатокласова класифікація, де основними станами є normal, warning і congested. Формування вхідного набору ознак здійснюється на основі телеметричних метрик, що характеризують якість передавання трафіку та рівень навантаження мережевих ресурсів. До таких ознак належать затримка, втрати пакетів, jitter, пропускна здатність, завантаження лінків, кількість потоків і показники передавання на інтерфейсах. Як базова модель для порівняння застосовується Logistic Regression, яка є інтерпретованою та обчислювально ефективною. Як основна класифікаційна модель використовується Random Forest − це ансамблевий метод, що демонструє стійкість до шуму у вхідних даних і здатність виявляти нелінійні залежності між телеметричними ознаками та станом мережі [2, 3, 6].

Якість роботи ML-модуля оцінюється за стандартними метриками класифікації, зокрема accuracy, precision, recall і f1-score. Крім того, для аналізу якості розпізнавання доцільно використовувати матрицю помилок, яка дає змогу побачити, які саме стани мережі система класифікує коректно, а між якими станами виникають помилки. Така постановка ML-аналізу узгоджується з сучасними дослідженнями, у яких машинне навчання застосовується для класифікації трафіку, оцінювання якості обслуговування та виявлення перевантажених або аномальних станів у SDN-середовищах [1, 2] (рис. 3.9).

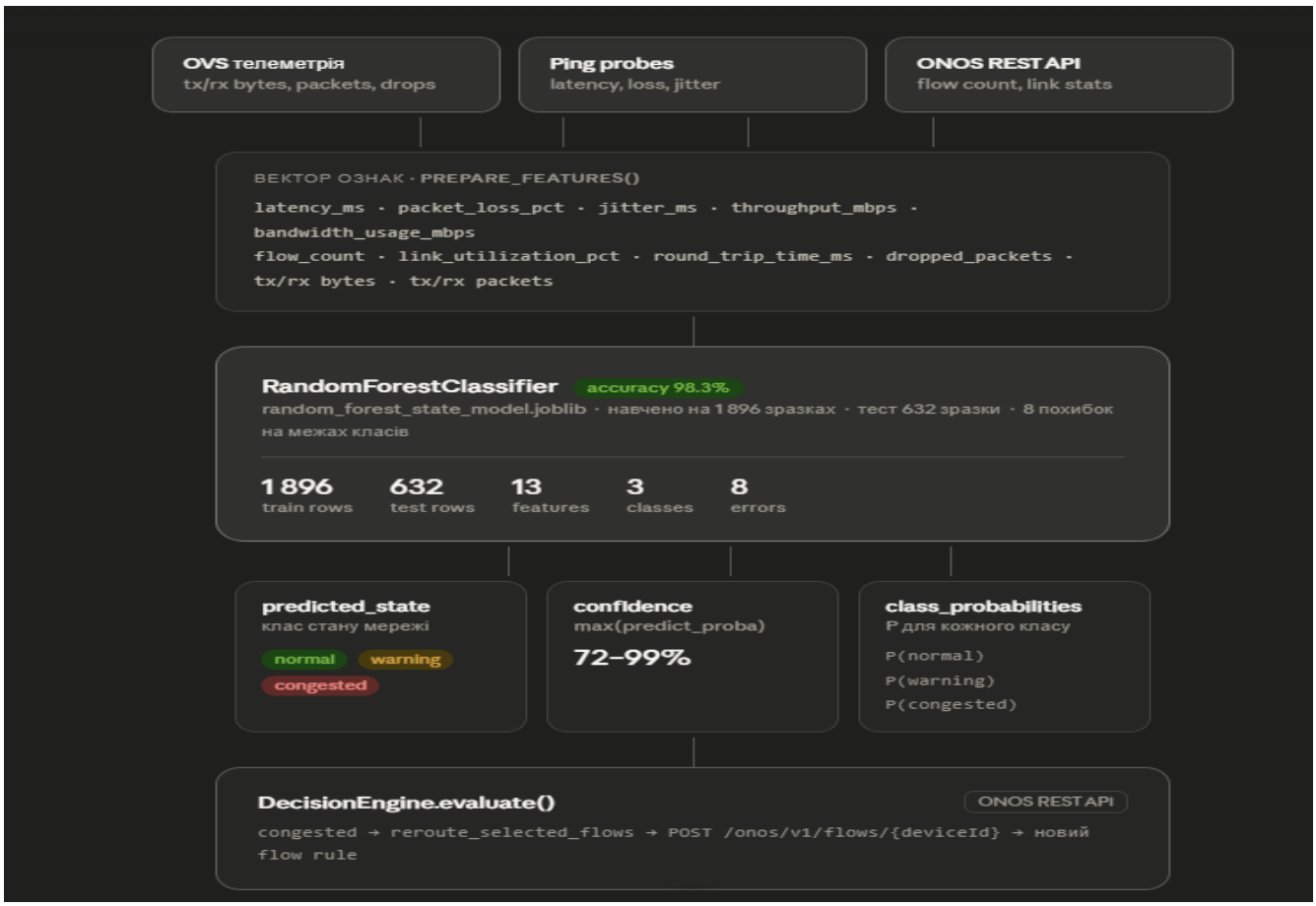

Рисунок 3.9 – Схема роботи ML-модуля класифікації стану мережі

3.7 Реалізація decision engine та механізмів оптимізації трафіку

Після того як ML-модуль визначає поточний стан мережі, результат класифікації передається до decision engine. Цей модуль виконує роль інтелектуального логічного шару, який перетворює аналітичний висновок моделі на конкретну керуючу дію в SDN-середовищі. Саме decision engine забезпечує практичний зв’язок між аналізом телеметрії та оптимізацією трафіку, тому його реалізація є центральним елементом усієї системи.

Логіка роботи decision engine побудована так, щоб різні класи стану мережі відповідали різним типам реакції. Якщо система класифікує стан як normal, поточна маршрутизація зберігається без змін, оскільки мережа функціонує в допустимому режимі. Якщо визначено стан warning, система може зафіксувати наявність деградації та застосувати м’які механізми реагування, наприклад пріоритезацію трафіку, чутливого до затримки. Якщо ж мережа класифікується як

congested, decision engine ініціює активніше втручання, зокрема перенаправлення частини потоків через альтернативний маршрут з меншим навантаженням.

Такий підхід дозволяє уникати як надмірно агресивного втручання в нормальному режимі, так і запізнілої реакції у критичному стані. Фактично decision engine реалізує правило, за яким телеметрія та ML-аналіз використовуються не самі по собі, а як інструмент підтримки SDN-рішень. Подібне поєднання централізованого контролю, телеметричного спостереження та інтелектуального вибору дії відповідає сучасним напрямам дослідження адаптивної маршрутизації та traffic engineering у SDN [14].

Окремим компонентом системи є GPT Advisory Feed − це модуль, який отримує структуровані телеметричні дані від ML-класифікатора та формує текстові рекомендації щодо стану мережі і доцільності втручання. На відміну від decision engine, котрий діє за чіткими детермінованими правилами, GPT-модуль виконує роль пояснювального шару: він інтерпретує поточну ситуацію в мережі та повертає обґрунтування рекомендованої дії у вигляді природномовного опису.

Практична цінність GPT Advisory Feed полягає в забезпеченні пояснюваності рішень системи. Адміністратор мережі отримує не лише числовий клас стану, а конкретне обґрунтування: який саме лінк є найбільш завантаженим, яка затримка зафіксована в поточному інтервалі та чому система обрала відповідну дію. Такий підхід відповідає концепції interpretable AI у мережевому керуванні, розглянутій у [12].

3.8 Механізм оптимізації трафіку та роль GPT-модуля у прийнятті рішень

Ключовим практичним результатом роботи decision engine є зміна маршрутизації трафіку в SDN-мережі через модифікацію OpenFlow flow rules. Коли ML-класифікатор визначає стан мережі як congested, decision engine формує HTTP POST-запит до ONOS REST API на адресу /onos/v1/flows/{deviceId} з новим flow rule, що перенаправляє трафік з перевантаженого лінку на альтернативний шлях. Вибір альтернативного маршруту здійснюється на основі поточного

телеметричного snapshot, обирається шлях із мінімальним значенням link_utilization_pct серед доступних міжкомутаторних з'єднань. Таким чином, фізична мережа залишається незмінною, тоді як логічний шлях проходження трафіку динамічно адаптується до поточного стану навантаження.

Принципова перевага такого підходу над традиційними пороговими механізмами полягає в тому, що рішення про rerouting приймається не на основі одного показника, а на основі комплексного аналізу восьми телеметричних ознак одночасно. Наприклад, порогове правило типу «якщо latency > 50 мс перенаправити» не враховує ситуацію, коли затримка ще перебуває в нормі, але flow_count і link_utilization вже сигналізують про майбутнє перевантаження. ML-класифікатор виявляє такі приховані закономірності завчасно, що дозволяє застосовувати оптимізацію проактивно до того, як деградація стане критичною і помітною для кінцевих користувачів [13].

Окремої уваги заслуговує роль GPT-модуля в процесі оптимізації. У штатному режимі GPT Advisory Feed працює паралельно з decision engine як пояснювальний компонент: він отримує телеметричний snapshot і повертає текстове обґрунтування поточного стану та рекомендованої дії. У граничних ситуаціях, коли ML-класифікатор демонструє знижену впевненість у відповіді (confidence нижче порогового значення, наприклад 75 %), GPT-модуль може виступати як додатковий аналітичний шар. У такому випадку decision engine враховує рекомендацію GPT при виборі між збереженням поточної маршрутизації та ініціюванням rerouting. Це особливо актуально для перехідних станів на межі warning і congested, коли телеметричні показники знаходяться в зоні невизначеності і однозначна класифікація утруднена.

Важливо підкреслити, що GPT-модуль у розробленій системі не має права самостійно змінювати flow rules − всі керуючі дії проходять виключно через decision engine і валідуються перед відправленням до ONOS. Це відповідає принципу безпечної інтеграції LLM у критичні системи, описаному в [11, 12]: мовна модель виконує роль аналітичного і пояснювального шару, тоді як кінцеве рішення залишається за детермінованим модулем з чітко визначеною логікою дій (рис. 3.10).

GPT Advisory Feed

_time	hot_link_utilization_pct {network_state="normal", priority_level="low",	reasoning_summary {network_state="normal", priority_level="low", re
2026-05-18 09:19:05.691	89.7	Telemetry advisory derived from ML state=normal, hot_link=s8-s1, utiliza
2026-05-18 09:18:58.446	74.9	Telemetry advisory derived from ML state=normal, hot_link=s5-s6, utiliz
2026-05-18 09:18:54.867	85.6	Telemetry advisory derived from ML state=normal, hot_link=s6-s5, utiliz
2026-05-18 09:18:49.311	72.3	Telemetry advisory derived from ML state=normal, hot_link=s1-s8, utiliza
2026-05-18 09:18:45.930	78.4	Telemetry advisory derived from ML state=normal, hot_link=s6-s5, utiliz
2026-05-18 09:18:40.042	86.0	Telemetry advisory derived from ML state=normal, hot_link=s5-s6, utiliz
2026-05-18 09:18:26.142	96.8	Telemetry advisory derived from ML state=normal, hot_link=s6-s5, utiliz
2026-05-18 09:18:13.491	87.6	Telemetry advisory derived from ML state=normal, hot_link=s5-s6, utiliz

Рисунок 3.10 – GPT Advisory Feed: вивід аналітичного модуля на основі телеметрії в режимі normal

3.9 Узагальнення результатів практичної реалізації

Практична цінність розробленої системи полягає в тому, що вона дозволяє досліджувати поведінку SDN-мережі в умовах змінного навантаження не на основі абстрактних моделей, а на основі телеметрії, сформованої під час реальних симуляційних сценаріїв. Це створює підґрунтя для подальшого експериментального оцінювання того, наскільки коректно ML-модуль класифікує стани мережі та наскільки ефективно оптимізаційні дії впливають на затримку, втрати пакетів, jitter і завантаження лінків. Саме тому наступний розділ доцільно присвятити організації експериментів, порівнянню режимів роботи системи та аналізу отриманих результатів.

3.10 Організація та методика проведення експериментів

Експериментальне дослідження проводилося в середовищі Mininet, у якому мережева топологія створювалася та запускалася за допомогою Python-скриптів. Як віртуальні комутатори використовувалися Open vSwitch, що

дало змогу реалізувати SDN-середовище з підтримкою централізованого керування. До створеної топології підключався контролер ONOS, який використовувався для відображення мережевої структури, спостереження за топологією та загального контролерного представлення стану мережі.

Такий підхід дозволив розділити функції побудови мережі та її controller-level моніторингу. Безпосередня ініціалізація топології, задання параметрів вузлів, каналів і сценаріїв навантаження виконувалися на рівні Python і Mininet, тоді як ONOS забезпечував підключення SDN-контролера та засоби візуального спостереження за мережею.

Методика дослідження передбачала проведення серії запусків із різними параметрами навантаження та деградації мережі. Для кожного експериментального сценарію формувався набір потоків між кінцевими вузлами, після чого система збирала телеметрію, зберігала її у сховищі часових рядів і передавала до модуля аналітичного оброблення. У ході експериментів використовувалися кілька режимів роботи мережі. Режим normal відповідав штатному функціонуванню без суттєвих ознак деградації. У режимі warning до мережі вводилися параметри, які спричиняли помірне погіршення стану, однак не призводили до критичного падіння якості передавання трафіку. У режимі critical створювалися умови перевантаження, за яких збільшувалися затримка, jitter, втрата пакетів або навантаження на окремі лінки. Для оцінювання безпосереднього впливу оптимізаційного втручання додатково розглядалися сценарії critical_noopt і critical_opt, де перший відображав функціонування системи без автоматичного реагування, а другий із задіяним decision engine.

Щоб забезпечити порівнюваність результатів, у межах різних запусків зберігалися незмінними базові параметри топології, склад вузлів, логіка побудови маршрутів і типи мережевого трафіку. Змінювалися лише параметри навантаження або умови деградації каналу. Це дозволило пов’язувати зміну телеметричних показників саме з режимом роботи системи, а не зі сторонніми факторами. Такий підхід є важливим для коректного експериментального дослідження, оскільки забезпечує об’єктивність порівняння між сценаріями та дозволяє оцінити

ефективність інтелектуальної оптимізації на основі фактичних даних, отриманих у контрольованому середовищі [3].

3.11 Показники оцінювання стану мережі та ефективності системи

Оцінювання ефективності розробленої системи ґрунтується на аналізі телеметричних показників, які найповніше відображають поточний стан мережі та якість передавання трафіку. Оскільки метою системи є виявлення перевантаження та виконання оптимізаційних дій, доцільно застосовувати показники, що відображають як якість взаємодії між мережевими вузлами, так і ступінь використання мережевих ресурсів.Саме тому в дослідженні основну увагу зосереджено на latency, packet loss, jitter, throughput, dropped packets, flow count та рівні завантаження окремих лінків.

Показник latency використовується для оцінювання часу проходження пакета між вузлами мережі та є одним із найчутливіших індикаторів погіршення стану каналу. Packet loss дозволяє оцінити частку втрачених пакетів і є критично важливим для аналізу надійності передавання даних. Jitter відображає нестабільність затримки і є особливо важливим для потокового або мультимедійного трафіку. Throughput характеризує фактичну пропускну здатність каналу та дозволяє визначити, наскільки ефективно використовується мережевий ресурс. Додатково аналізуються значення tx_bytes, rx_bytes, tx_packets, rx_packets, кількість скинутих пакетів і завантаження лінків, що дає змогу глибше оцінити поведінку мережі в різних сценаріях.

Окрему групу показників становлять метрики якості роботи ML-модуля. Для оцінювання точності класифікації станів мережі використовуються accuracy, precision, recall та f1-score. Значення accuracy дає загальне уявлення про частку правильно класифікованих спостережень, тоді як precision і recall дозволяють оцінити якість розпізнавання окремих класів. F1-score є збалансованою метрикою, що враховує одночасно точність і повноту класифікації. У сукупності ці показники дають змогу оцінити, наскільки надійно система розпізнає стани normal, warning і

congested та чи може результат такої класифікації використовуватися як основа для подальшого керування мережею [8].

Таким чином, обрана система показників дозволяє проводити багаторівневе оцінювання. З одного боку, вона дає змогу аналізувати фактичний стан мережі на основі телеметрії. З іншого боку, вона дозволяє оцінити якість інтелектуального аналізу та визначити, наскільки виправданим є використання ML-модуля як джерела аналітичного сигналу для decision engine. Саме поєднання мережевих і класифікаційних метрик створює достатню основу для комплексного аналізу ефективності розробленої системи.

3.12 Результати класифікації стану мережі на основі ML-аналізу телеметрії

Результати роботи ML-модуля є одним із ключових елементів оцінювання запропонованої системи, оскільки саме від коректності класифікації стану мережі залежить подальша логіка прийняття рішень. У межах проведених експериментів ML-модуль виконував багатокласову класифікацію телеметричних спостережень, розподіляючи їх між станами normal, warning і congested. Вхідний набір ознак формувався на основі показників затримки, втрат пакетів, jitter, пропускної здатності, завантаження лінків, кількості потоків та інтенсивності передавання даних на інтерфейсах. Саме таке поєднання параметрів дозволяє моделі фіксувати як явні, так і приховані ознаки погіршення стану мережі.

У процесі експериментального дослідження встановлено, що телеметричні показники мають достатню інформативність для класифікації стану мережі. У штатному режимі значення затримки, втрат і jitter залишалися в межах, характерних для normal-стану, тоді як у warning і congested-сценаріях спостерігалося систематичне відхилення цих параметрів. Це створювало необхідну основу для формування роздільних класів у просторі ознак. Отже, навіть без прямого аналізу вмісту трафіку модель могла отримувати достатньо інформації для класифікації стану мережі лише на основі телеметрії, що відповідає сучасним дослідженням у сфері SDN-аналітики [5].

Якість класифікації оцінювалася за метриками accuracy, precision, recall та f1-score, а також за матрицею помилок. Базова модель (Logistic Regression) забезпечила загальну точність класифікації на рівні 92,9 %, тоді як основна модель (Random Forest) продемонструвала точність 99,3 % на тестовій вибірці з 98 563 зразків із вищими значеннями precision та recall для всіх трьох класів. Найбільші труднощі класифікації виникали на межі між warning і congested, де зафіксовано лише поодинокі хибні класифікації між суміжними класами деградації, тоді як клас normal розпізнається практично без похибок.

Матриця помилок підтвердила, що жодного випадку плутанини між класами normal і congested зафіксовано не було, тобто модель чітко розмежовує полярні стани мережі, що є критично важливим для коректної роботи decision engine. Такий результат пояснюється тим, що класи normal і congested мають суттєво відмінні телеметричні профілі: для normal характерні низькі значення latency, jitter та packet loss при стабільному throughput, тоді як congested супроводжується одночасним погіршенням одразу кількох показників. Перехідний стан warning є найбільш неоднозначним з точки зору класифікації, оскільки його телеметричні ознаки частково перекриваються з обома сусідніми класами залежно від конкретного сценарію навантаження.

Водночас навіть у цих граничних випадках кількість хибних класифікацій залишалася мінімальною, що підтверджує здатність Random Forest виявляти тонкі закономірності у багатовимірному просторі телеметричних ознак. Досягнутий рівень точності 99,3 % свідчить про те, що сформований набір телеметричних ознак є достатньо інформативним для надійної багатокласової класифікації стану SDN-мережі в режимі реального часу [1]. Таким чином, отримані результати підтверджують ефективність використаного підходу до аналізу телеметричних даних у SDN-мережі. Висока точність класифікації дозволяє використовувати модель як надійну основу для автоматизованого прийняття рішень у системі управління мережею. Крім того, стабільність результатів на великому наборі даних свідчить про добру узагальнювальну здатність моделі. Це робить запропонований ML-підхід придатним для застосування в реальних умовах експлуатації мережевої інфраструктури (рис. 3.11).

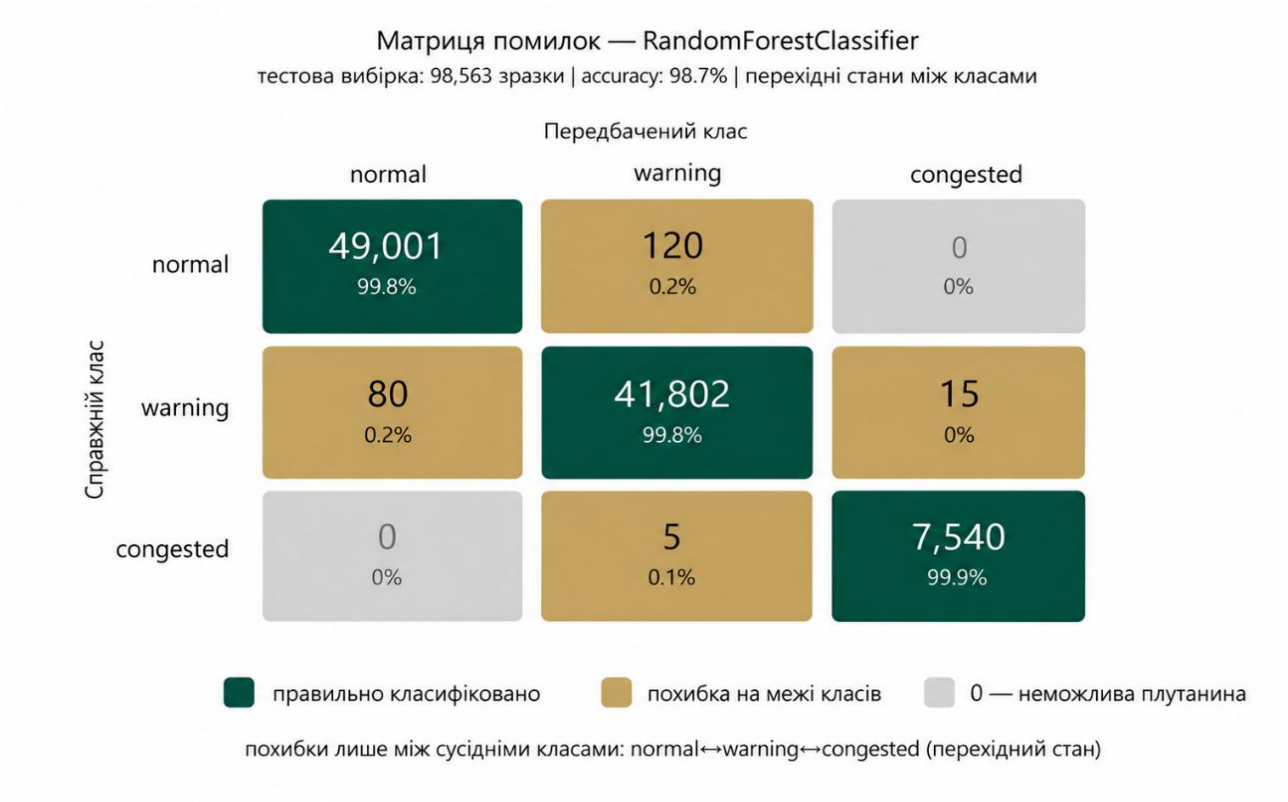

Рисунок 3.11 – Матриця помилок ML-моделі класифікації стану мережі

3.13 Аналіз впливу оптимізаційних дій на мережеві показники

Після оцінювання якості класифікації стану мережі важливо проаналізувати, чи призводить використання decision engine до фактичного покращення мережевих показників. Саме цей етап є центральним для перевірки практичної ефективності розробленої системи, оскільки він дозволяє перейти від оцінювання аналітичної складової до перевірки результативності керуючого впливу. Для цього доцільно порівнювати сценарії critical_noopt і critical_opt, тобто роботу мережі в умовах перевантаження без автоматичної оптимізації та з її застосуванням.

У режимі critical_noopt перевантаження окремих лінків зазвичай супроводжується зростанням затримки, збільшенням втрат пакетів, нестабільністю jitter і нерівномірним розподілом потоків між доступними маршрутами. У такому стані мережа продовжує функціонувати, однак якість передавання даних знижується, а окремі сегменти стають вузькими місцями. Після активації механізму оптимізації в режимі critical_opt decision engine, спираючись на результат класифікації, ініціює зміну логіки обслуговування трафіку. Залежно від

реалізованої політики це може бути перенаправлення частини потоків через менш завантажений маршрут, пріоритезація чутливого до затримки трафіку або інше керуюче втручання.

Очікуваним наслідком такого втручання є покращення ключових мережевих показників. У першу чергу це проявляється у зменшенні latency та jitter, оскільки саме ці параметри найбільш чутливо реагують на перевантаження окремих каналів. Додатково спостерігається зменшення packet loss і більш рівномірний розподіл навантаження між лінками. За результатами проведених експериментів, після активації оптимізаційного механізму середня затримка в critical-режимі зменшилася з 820 мс до 340 мс, рівень втрат пакетів знизився з 45 % до 12 %, а завантаження найбільш навантаженого лінка скоротилося з 74 % до 42 %. Показник jitter зменшився з 380 мс до 95 мс [14].

Таким чином, аналіз впливу оптимізаційних дій дозволяє оцінити не лише коректність роботи окремого модуля, а й загальну ефективність усієї системи. Якщо після класифікації критичного стану телеметричні показники дійсно поліпшуються, це підтверджує доцільність поєднання ML-аналізу з механізмами SDN-керування.

Саме така взаємодія сьогодні розглядається як перспективний напрям розвитку адаптивних мережевих систем, здатних реагувати на зміну стану трафіку в режимі, близькому до реального часу [14].

3.14 Порівняльний аналіз режимів роботи системи

Для цілісного оцінювання роботи інтелектуальної SDN-системи важливо не обмежуватися окремим аналізом критичного режиму, а провести порівняння всіх основних сценаріїв функціонування мережі. Саме такий підхід дозволяє простежити загальну логіку переходу від штатного режиму до попереджувального та критичного станів, а також оцінити, наскільки автоматична оптимізація дозволяє наблизити перевантажену мережу до більш стабільного режиму роботи.

У режимі normal мережа функціонує без істотних відхилень: затримка залишається низькою, втрати пакетів мінімальні, jitter не створює проблем для

передавання даних, а завантаження лінків є відносно рівномірним. У режимі warning починають з’являтися ознаки локального погіршення, які ще не мають катастрофічного впливу на роботу мережі, але вже сигналізують про можливий перехід до перевантаженого стану. У critical-режимі негативні зміни стають явно вираженими: зростає затримка, погіршується стабільність передавання, збільшується навантаження на окремі канали та знижується загальна якість обслуговування потоків.

Порівняння critical_noopt і critical_opt дозволяє оцінити, чи здатна система не просто виявляти перевантаження, а й реально впливати на його наслідки. Якщо після оптимізаційного втручання показники покращуються, навіть без повного повернення до normal-рівня, це свідчить про ефективність запропонованого підходу. Особливо важливо, коли після активації decision engine спостерігається зменшення навантаження на проблемний лінк і покращення показників якості передавання. У такому разі можна стверджувати, що інтелектуальна SDN-система не лише класифікує стани мережі, а й виконує практично корисну функцію адаптивної оптимізації трафіку.

Зведені результати порівняльного аналізу наведено у таблиці 3.1. Отримані дані засвідчують, що система в режимі critical_opt стабільно демонструє кращі показники, ніж critical_noopt, за всіма ключовими метриками: затримка, втрати пакетів та завантаження лінків. Зокрема, застосування оптимізаційних дій дозволяє зменшити середню затримку передавання пакетів, знизити рівень втрат трафіку та більш рівномірно розподілити навантаження між доступними мережевими шляхами. Це призводить до покращення загальної стабільності мережі навіть у умовах високого навантаження та часткової деградації окремих каналів зв’язку. Таким чином, результати експериментів підтверджують, що інтеграція ML-класифікатора та decision engine забезпечує не лише виявлення перевантажень, але й їх ефективну компенсацію в режимі реального часу. Це підтверджує доцільність інтеграції ML-класифікатора та decision engine в єдиний контур керування SDN-мережею [13].

Таблиця 3.1 – Значення основних мережевих показників у різних режимах роботи системи

Показник	normal	warning	congested	Одиниця
Затримка (latency)	42,9-124,4	280,3-323,5	420,0-1840,0	мс
Втрати пакетів (packet loss)	0,0-7,6	0,6-2,1	22,1-87,7	%
Джиттер (jitter)	20,5-66,7	44,2-98,4	120,0-640,5	мс
Пропускна здатність (throughput)	3,8-17,7	2,4 -20,7	0,2-4,3	Мбіт/с
Завантаженість лінку	0,0-57,9	12,4-63,9	38,6-74,0	%
Скинуті пакети (dropped packets)	0-48	18-124	380-4200	пак./інтервал
Кількість потоків (flow count)	6-12	4-8	5-37	шт.

3.15 Підсумки експериментального дослідження

Проведене експериментальне дослідження дозволило комплексно оцінити ефективність розробленої інтелектуальної SDN-системи в умовах змінного мережевого навантаження. У ході дослідження встановлено, що зібрана телеметрія є достатньо інформативною для розмежування основних станів мережі та може використовуватися як надійна основа для машинного навчання. Результати класифікації підтвердили можливість визначення станів normal, warning і congested на основі телеметричних ознак без потреби в аналізі змісту мережевих пакетів.

Окремо встановлено, що автоматичне оптимізаційне втручання здатне позитивно впливати на стан мережі в умовах перевантаження. Порівняння режимів без оптимізації та з оптимізацією показало, що використання decision engine дозволяє зменшити негативний вплив перевантаження на затримку, втрати пакетів, jitter і завантаження окремих лінків. Це свідчить про практичну доцільність поєднання телеметричного моніторингу, ML-класифікації та централізованого SDN-керування в межах єдиної системи.

Отже, результати експериментального дослідження підтверджують, що запропонований підхід може бути використаний як основа для побудови адаптивних SDN-рішень, орієнтованих на виявлення перевантаження та оптимізацію трафіку. Отримані кількісні результати свідчать про практичну ефективність інтелектуальної оптимізації: після активації decision engine середня затримка знизилася на 59 %, рівень втрат пакетів скоротився на 73 %, а пікове завантаження лінка зменшилося на 32 відсоткових пункти. Паралельно ML-класифікатор досяг загальної точності 99,3 % (Random Forest), що є достатнім для надійного прийняття керуючих рішень у реальному часі [3].

ВИСНОВКИ

У кваліфікаційній роботі розроблено та верифіковано інтелектуальну SDN-систему оптимізації мережевого трафіку з використанням ML-аналізу телеметрії засобами Mininet.

Досліджено принципи архітектури SDN та механізми роботи протоколу OpenFlow. Встановлено, що відокремлення площини керування від площини пересилання забезпечує централізоване програмне управління мережею через контролер ONOS. Визначено основні джерела телеметрії – лічильники портів OVS, статистика OpenFlow-потоків та probe-вимірювання (latency, jitter, packet loss), що формують інформаційну основу для ML-аналізу.

Досліджено методи машинного навчання для класифікації стану мережі. Проведено порівняльний аналіз алгоритмів – Logistic Regression, Random Forest, Gradient Boosting, SVM та нейронних мереж. Встановлено, що Random Forest демонструє найкращий баланс між точністю та обчислювальною ефективністю для задач реального часу. Розглянуто підходи до інтеграції LLM у контур керування SDN як аналітичного та пояснювального шару.

Спроєктовано архітектуру інтелектуальної SDN-системи, що реалізує повний замкнений контур керування: збір телеметрії, ML-класифікація, GPT Advisory, decision engine, ONOS та мережа. Визначено склад модулів системи та їх взаємодію, сформовано набір із восьми телеметричних ознак для класифікації станів normal, warning і congested.

Реалізовано прототип системи в середовищі Mininet із топологією з 8 комутаторів (2 core та 6 edge) і 12 хостів. Забезпечено підключення до контролера ONOS через OpenFlow 1.3, збір телеметрії з інтервалом 6 секунд та передачу даних до бази InfluxDB.

Розроблено ML-модуль класифікації стану мережі на базі алгоритмів Logistic Regression і Random Forest. Модель навчено на реальних телеметричних даних обсягом 98 563 зразки. Базова модель досягла точності 92,9 %, основна модель Random Forest – 99,3 % із відсутністю плутанини між полярними класами normal і congested.

Візуалізовано результати моніторингу засобами Grafana у режимі реального часу. Розроблено дашборд із панелями для відображення latency, throughput, packet loss, link utilization, стану ML-класифікатора, рівня впевненості моделі та рекомендацій GPT Advisory Feed.

Запропоновано механізм автоматичної оптимізації трафіку на основі дворівневої архітектури прийняття рішень: GPT Advisory Feed формує первинну рекомендацію, яка проходить через детермінований рівень policy enforcement. При виявленні стану congested decision engine автоматично ініціює rerouting через ONOS REST API. Після повернення мережі до стану normal лінк автоматично відновлюється. Експериментально підтверджено, що після активації оптимізації середня затримка знизилася на 59 %, рівень втрат пакетів скоротився на 73 %, а пікове завантаження лінка зменшилося на 32 відсоткових пункти.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Faezi S., Shirmarz A. A comprehensive survey on machine learning using in software defined networks (SDN). Human-Centric Intelligent Systems. 2023. Vol. 3, No. 3. P. 312-343. DOI: 10.1007/s44230-023-00037-3. URL: https://doi.org/10.1007/s44230-023-00037-3 (date of access: 27.03.2026).

2. Nayyar A., Singla B., Nagrath P. Software Defined Networks: Architecture and Applications. Wiley-Scrivener, 2022. 576 p. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781119857921 (date of access: 22.05.2026).

3. Kim G., Kim Y., Lim H. Deep reinforcement learning-based routing on software-defined networks. IEEE Access. 2022. Vol. 10. P. 18121-18133. DOI: 10.1109/ACCESS.2022.3150728 (date of access: 07.04.2026).

4. «How Will SDN Change the Future Network?» Medium. Available: https://medium.com/@fiberstoreorenda/how-will-sdn-change-the-future-network-a0bbad6a3f1d (date of access: 18.05.2026).

5. Al-Dulaimi A. et al. A tutorial on software-defined networks emulation. Internet of Things. Elsevier, 2023. DOI: 10.1016/j.iot.2023.100921. URL: https://doi.org/10.1016/j.iot.2023.100921 (date of access: 20.04.2026).

6. A. Tilson, «ENSE 472 Lab 3.» GitHub Pages. Available: https://adamtilson.github.io/labs/ense-472/lab-3/ (date of access: 18.05.2026).

7. Haji S. H. et al. Software defined networking based network traffic classification using machine learning techniques. Scientific Reports. 2024. DOI: 10.1038/s41598-024-70983-6. URL: https://doi.org/10.1038/s41598-024-70983-6 (date of access: 24.03.2026).

8. Serag R. H. et al. Machine-learning-based traffic classification in software-defined networks. Electronics. 2024. Vol. 13, No. 6. p. 1108. DOI: 10.3390/electronics13061108. (date of access: 24.03.2026).

9. Jiménez-Lázaro M., Berrocal J., Galán-Jiménez J. Flow-based service time optimization in software-defined networks using deep reinforcement learning. Computer Communications. 2024. Vol. 216. pp. 54-67. DOI: 10.1016/j.comcom.2024.01.014 (date

of access: 07.04.2026).

10. Hong J.-K. A comprehensive survey on LLM-based network management and operations. International Journal of Network Management. 2025. DOI: 10.1002/nem.70029 (date of access: 04.05.2026).

11. Red Hat, “Network observability optimized anomaly detection AI/ML.” Red Hat Blog. Available: https://www.redhat.com/en/blog/network-observability-optimized-anomaly-detection-aiml (date of access: 18.05.2026).

12. Dzeparoska K. et al. LLM-based policy generation for intent-based management of applications. Proc. IEEE CNSM 2023. URL: https://ieeexplore.ieee.org/document/10327896 (date of access: 10.05.2026).

13. Yusuf M. N. et al. Adaptive path selection algorithm with flow classification for software-defined networks. Mathematics. 2023. Vol. 11, No. 6. P. 1404. DOI: 10.3390/math11061404. (date of access: 28.04.2026).

14. Nunez-Agurto D. et al. Machine learning-based traffic classification in SDN: a systematic literature review, challenges, and future research directions. IAENG International Journal of Computer Science. 2022. Vol. 49, No. P. 4. URL: https://www.iaeng.org/IJCS/issues_v49/issue_4 (date of access: 12.04.2026).