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 багатоагентної системи моніторингу SDN мереж»

Виконав: студент 2 курсу, групи МФ-61

спеціальність 122 «Комп’ютерні науки»

освітньо-наукова програма

«Інформатика»

Балашов О.В.

Керівник Руккас К.М.

Рецензент Скоб Ю.О.

Харків – 2026 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 3

Позначення	Розшифрування
SDN	Software-Defined Networking — програмно-визначені мережі
ML	Machine Learning — машинне навчання
kNN	k-Nearest Neighbors — метод k-найближчих сусідів
MAS	Multi-Agent System — багатоагентна система
API	Application Programming Interface — програмний інтерфейс
REST	Representational State Transfer — архітектурний стиль взаємодії сервісів
FPR	False Positive Rate — рівень хибнопозитивних спрацьовувань
FNR	Частка хибнонегативних пропусків — рівень хибнонегативних пропусків
CPU	Central Processing Unit — центральний процесор
LLDP	Link Layer Discovery Protocol — протокол виявлення каналів
ONOS	Open Network Operating System
OpenFlow	Протокол взаємодії SDN-контролера з комутаторами
Ryu	Відкрита платформа для розробки SDN-контролерів
Mininet	Мережевий емулятор для SDN-середовищ

1. ВСТУП

1.1 Мета і завдання дослідження

Програмно-визначені мережі (SDN) змінили парадигму управління мережами, надавши централізований контроль над потоком даних. Однак ця централізація створює нові виклики для своєчасного виявлення відмов. Коли контролер Ryu, ONOS або інший стає недоступним, коли канал зв'язку працює нестабільно, або коли виникає несподіване перевантаження, оператор має отримати інформацію протягом кількох секунд.

Проблема дослідження полягає в тому, що статичні порогові правила (наприклад, сповіщення за умови, що втрати пакетів перевищують 1%) недостатньо надійно працюють для різних топологій і профілів трафіку. Такі правила можуть спричиняти хибні спрацьовування у стабільних мережах або пропускати реальні збої в умовах деградації сервісу.

Метою роботи є проєктування, реалізація та експериментальне оцінювання мультиагентної системи моніторингу SDN-мереж, яка:

- Збирає телеметрію (затримка, джиттер, втрати, пропускна здатність) з комутаторів та контролера;
- Застосовує як порогові правила, так і машинне навчання (зважені k-найближчі сусіди) для виявлення аномалій;
- Узагальнюється між топологіями (навчається на одній, тестується на іншій) з мінімальною втратою точності;
- Демонструє практичну цінність через аналіз накладних витрат та відображення на реальні сценарії.

Завдання:

1. Розробити архітектуру мультиагентної системи моніторингу з розділенням відповідальності.

2. Провести комплексні експерименти на трьох топологіях (лінійна, деревоподібна, кільцева) з трьома профілями трафіку.

3. Виміряти точність класифікації (accuracy), частку хибнопозитивних спрацьовувань (FPR) та частку хибнонегативних пропусків (FNR) для порогового та ML-підходів.

4. Статистично оцінити значущість покращень за допомогою критерію Манна-Вітні U, критерію Мак-Немара та аналізу розміру ефекту.

5. Проаналізувати витрати (CPU, пам'ять, обсяг телеметрії) та перенесення моделей між топологіями.

1.2 Актуальність теми

Мережі SDN дедалі ширше застосовуються у корпоративних і операторських середовищах, центрах обробки даних та хмарних інфраструктурах. Тому своєчасне виявлення деградації сервісу, перевантажень і відмов елементів мережі є важливою умовою забезпечення операційної надійності.

З огляду на це, у роботі розглянуто підхід, який поєднує інтерпретованість порогового аналізу з адаптивністю методів машинного навчання.

Особливу увагу приділено ситуації, коли модель, навчена на одній топології, має коректно працювати на іншій топології без повторного навчання.

Такий сценарій є практично важливим, оскільки в реальних SDN-мережах структура трафіку та топологія можуть змінюватися швидше, ніж оператор встигає переналаштувати правила моніторингу вручну.

У роботі акцентовано проблему узагальнення моделей машинного навчання між різними мережевими топологіями та запропоновано комплексний експериментальний аналіз перенесення моделей у контексті SDN-моніторингу.

1.3 Стислий огляд відомих результатів

Традиційні підходи. Порогові правила залишаються поширеним інструментом мережевих операторів завдяки простоті реалізації та інтерпретованості, однак вони не враховують специфіку топології та динаміку трафіку.

Підходи на основі машинного навчання. Низка робіт застосовує випадковий ліс, метод опорних векторів та нейронні мережі для виявлення аномалій у мережах. Більшість фокусуються на класичних мережах IP або темпоральних даних (часові ряди). Для SDN часто використовується один контролер і одна топологія.

Науково-практичний внесок роботи полягає у систематичній оцінці перенесення ML-моделей між топологіями. Показано, що значущість ознак, зокрема затримка та джиттер, змінюється залежно від топології, однак ML-підхід зберігає високу точність при навчанні на одній топології та тестуванні на іншій.

1.4 Наукова новизна одержаних результатів

Основні результати експериментів (валідація 2026-04-28):

Напрям навчання → тестування	**Точність порогу**	**Точність ML**	**Точність kNN**	**Приріст ML**	**Приріст kNN**
Лінія → Дерево	86.67%	95.56%	98.89%	+8.89%	+12.22%
Дерево → Лінія	88.89%	93.33%	100.00%	+4.44%	+11.11%
Лінія → Кільце	84.44%	95.56%	100.00%	+11.12%	+15.56%
Кільце → Лінія	88.89%	93.33%	98.89%	+4.44%	+10.00%

Наукова новизна та практичні результати:

1. Крос-топологічне узагальнення з ML/kNN: На відміну від порогових правил, ML-моделі стабільно узагальнюються між топологіями з приростом точності.

2. Статистична значущість: McNemar та знаковий та біноміальний тести на оновленому наборі показують статистично значущі прирости у ключових переходах.

3. Операційна релевантність: Зіставлення сценаріїв (тайм-аут LLDP, нестабільність каналу зв'язку та перевантаження) підтверджує, що синтетичні профілі відповідають реальним інцидентам.

4. Накладні витрати залишаються контрольованими: у експерименті під час виконання системи отримано точність 91,67%, зменшення обсягу телеметрії на 11,67% та середню затримку виявлення 0,949 с.

1.5 Методи дослідження

- Симуляція: Mininet з Ryu контролером OpenFlow 1.3 для відтворення мережевих топологій у контрольованому середовищі.

- Експериментальне планування: Матриця комбінацій топології (лінія, дерево, кільце) × профілю трафіку (failure_like, unexpected_burst, stable) та кількості повторень.

- Збір даних: Телеметрія (затримка_ms, джиттер_ms, втрати_percent, пропускна здатність_mbps) з кожного комутатора кожні 1-2 секунди.

- Методи машинного навчання: зважений метод k-найближчих сусідів (k=5) з оберненим дистанційним зважуванням для класифікації станів «норма» та «аномалія».

- Статистичні тести:

- Mann-Whitney U для непараметричного порівняння точності.

- критерій Мак-Немара для порівняння парних бінарних результатів порогового та ML-підходів.

- Cohen's d/h для розмірів ефектів.

- Bootstrap 95% довірчі інтервали для нестійких метрик.

- Аналіз відмов: позаплановий аналіз випадків, коли ML помилялася, або пороги дали хибне спрацьовування.

1.6 Практичне значення одержаних результатів

Практичне значення роботи полягає у розробленні та експериментальній перевірці багатоагентного підходу до моніторингу SDN-мереж, який може бути використаний як прототип програмного забезпечення для виявлення відмов,

перевантажень і аномальних режимів роботи. Запропонований підхід є придатним для подальшого використання у навчальних, дослідницьких та експериментальних середовищах, побудованих на Mininet, Ryu та OpenFlow.

Отримані результати можуть бути застосовані для обґрунтування вибору методів моніторингу SDN-інфраструктур, налаштування порогових правил, порівняння класичних і ML-підходів до виявлення аномалій, а також для підготовки подальших експериментів із масштабуванням топологій і перевіркою на реальних трасах трафіку.

1.7 Апробація результатів дослідження

Окрема зовнішня апробація результатів дослідження в межах наукових конференцій або публічних фахових семінарів не проводилася. Результати роботи апробовано в межах експериментального стенду шляхом повторюваних запусків, порівняльного аналізу методів і статистичної перевірки отриманих метрик.

1.8 Публікації

За темою кваліфікаційної роботи окремих публікацій не подано. Основні результати викладено у тексті роботи, таблицях експериментальних результатів і додатках із протоколами відтворюваності.(добавить ту конференцию)

1.9 Структура роботи

Робота складається зі вступу, основної частини, висновків, списку використаних джерел і додатків. Вступ містить мету та завдання дослідження, обґрунтування актуальності теми, методи дослідження, практичне значення результатів, відомості про апробацію, публікації та структуру роботи. Основна частина охоплює постановку задачі, огляд сучасного стану предметної області, модель вимог до інформаційної системи, архітектуру та функціональність системи, опис і обґрунтування алгоритмів, використані технології, результати тестування, аналіз результатів і результати впровадження. У висновках

узагальнено основні результати роботи та визначено напрями подальших досліджень.

2. ОСНОВНА ЧАСТИНА

2.1 Постановка задачі

Сучасні огляди з SDN підкреслюють, що розділення площини керування і площини передавання спрощує централізоване управління, але водночас створює вимоги до перевірки, моніторингу, безпеки та відмовостійкості [1]. Роботи з інтелектуального керування трафіком у SDN також виділяють типову схему із модуля збору стану мережі, модуля інтелектуального аналізу та SDN-контролера [2]. Огляд алгоритмів оптимізації у SDN показує, що алгоритмічні рішення мають враховувати масштабованість, збіжність, надлишковість і реакцію в реальному часі [3]. У найновіших роботах 2024—2025 років продовжується розвиток ML/DL-підходів до аномального трафіку у SDN, але основний фокус здебільшого залишається на безпекових сценаріях і класифікації трафіку, тоді як ця робота перевіряє перенесення моделі між різними топологіями мережі [4—6].

Новіші дослідження 2024–2025 років підтверджують актуальність поєднання SDN-моніторингу з методами машинного навчання. Огляд Ruffo та співавторів систематизує сучасні підходи до системи виявлення вторгнень на основі глибокого навчання для SDN, включно з наборами даних, попередньою обробкою, вибором моделей і метриками оцінювання [16]. У роботі Wang та співавторів запропоновано виявлення аномального SDN-трафіку на основі часових згорткових мереж, що підкреслює інтерес до часових ознак і поведінкових патернів [17]. Огляд Batool та співавторів узагальнює методи ML, глибоке навчання і федеративне навчання для виявлення та пом’якшення DDoS-атак у SDN, а також наголошує на потребі реалістичних SDN-орієнтованих датасетів [18]. Ці публікації не змінюють експериментальних даних цієї роботи, але уточнюють науковий контекст і підтверджують доцільність дослідження крос-топологічного узагальнення.

Система спостереження та вхідні дані

Система спостереження побудована як прототип багатоагентної архітектури, що складається з таких компонентів:

1. CollectorAgent: збирає метрики з Ryu (OpenFlow switch stats).

2. AnalyzerAgent: застосовує порогові правила та виробляє сигнали тривоги.

3. MLAnalyzerAgent: виконує k-NN модель для прогнозування класу (нормально/аномалія).

4. CoordinatorAgent: логує рішення, обчислює кінцеву метрику якості.

Вхідними даними є профілі трафіку, сформовані в середовищі Mininet для моделювання стабільної роботи, перевантаження та відмовоподібних станів SDN-мережі. Вихідними даними є послідовність класифікацій стану мережі («норма» або «аномалія»), яка використовується для оцінювання точності, затримки виявлення та накладних витрат.

2.2 Розвинутий огляд сучасного стану справ в області дослідження

Традиційний моніторинг у SDN

- Системи типу Prometheus/Grafana використовують фіксовані пороги.

- Оператор вручну налаштовує пороги для кожної топології.

- Низькі помилки типу I (хибнопозитивне спрацьовування) на стабільних мережах, але часто вищі помилки типу II (пропуск) під навантаженням.

Методи машинного навчання у мережах

- Google, Facebook опублікували роботи про виявлення аномалій у центрах обробки даних, але часто з закритими даними.

- Існуючі набори даних (NSL-KDD, UNSW-NB15) зосереджені на кібербезпеці, а не на деградації якості.

- Для SDN: декілька робіт на FlowMatrix, а основний фокус на перевантаження потоків, а не на топологічне узагальнення.

Науково-практичний внесок роботи

У роботі запропоновано інтерпретований підхід до виявлення аномалій, у якому:

- використовуються фізично інтерпретовані ознаки: затримка, джиттер, втрати пакетів і пропускна здатність;
- модель k-NN є простою для реалізації, не потребує складного навчання та зберігає можливість пояснення рішення через найближчі спостереження;
- ключова цінність підходу полягає у здатності до крос-топологічного узагальнення, що підтверджено експериментально.

2.3 Опис моделі вимог до інформаційної системи

Функціональні та нефункціональні вимоги

Функціональні вимоги:

1. Збирати метрики з усіх комутаторів заданим інтервалом.
2. Застосовувати порогові класифікатори для кожного комутатора.
3. Запускати ML-модель із збереженим станом (no online learning).
4. Повідомляти про аномалії у журналі та визначати час виявлення.
5. Експортувати результати для постаналізу.

Нефункціональні вимоги:

- Затримка: < 100 мс від вимірювання до класифікації.
- Масштабованість: Підтримка до 100 комутаторів без деградації.
- Надійність: Відсутність падіння системи навіть при одиничних помилках вимірювання.
- Відтворюваність: Всі експерименти повинні бути детерміновані за посівом випадковості.

2.4 Опис архітектури інформаційної системи

Архітектура багатоагентної системи

Система реалізована як конвеєр взаємодіючих агентів, кожен з яких виконує специфічну функцію:

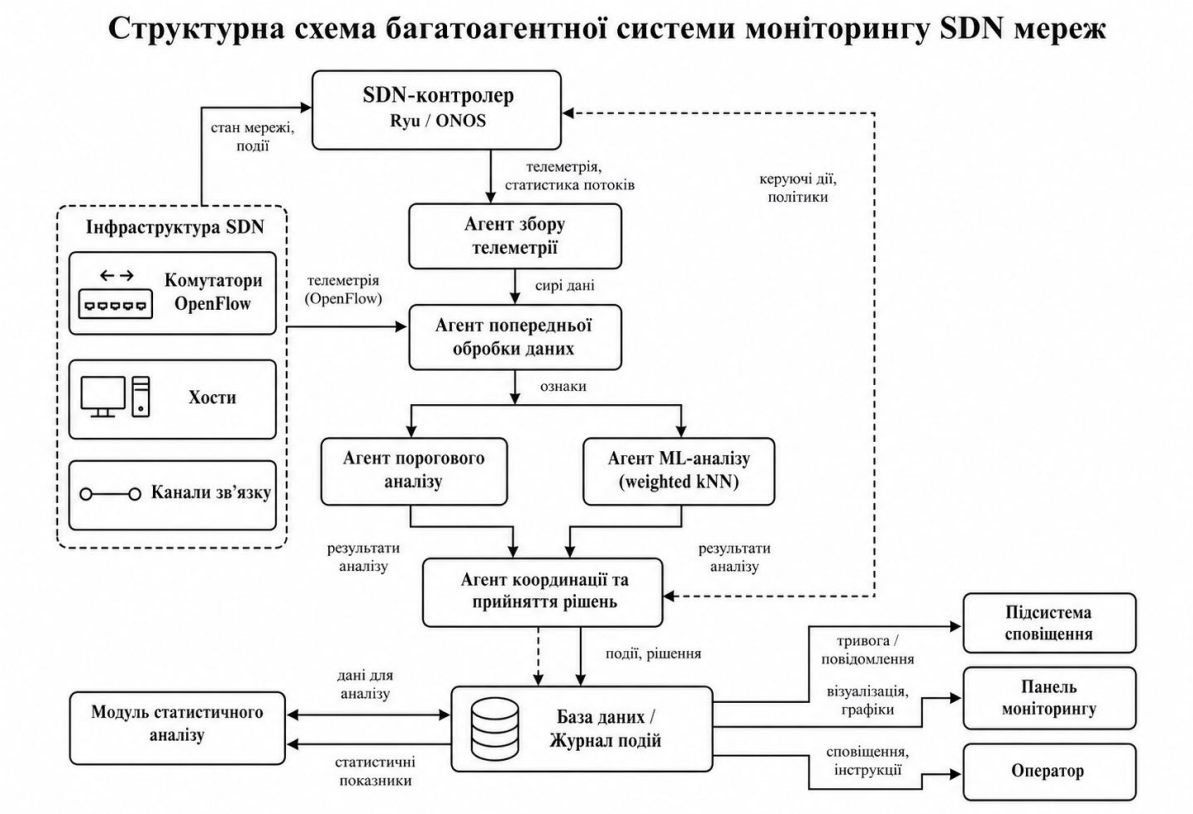

Рисунок 2.1 – Структурна схема багатоагентної системи моніторингу SDN-мереж

На рисунку 2.1 наведено структурну схему запропонованої багатоагентної системи моніторингу SDN-мереж. Центральне місце в архітектурі займає підсистема збирання телеметрії, яка взаємодіє з SDN-контролером та інфраструктурою OpenFlow. Після попередньої обробки дані надходять до двох аналітичних агентів — порогового та машинно-навчального. Результати їхньої роботи узгоджуються агентом координації та прийняття рішень, а потім передаються до журналу подій, підсистеми сповіщення, панелі моніторингу та оператору. Така архітектура забезпечує модульність, відтворюваність експериментів і можливість розширення системи новими агентами або аналітичними модулями.

Детальний опис кожного агента

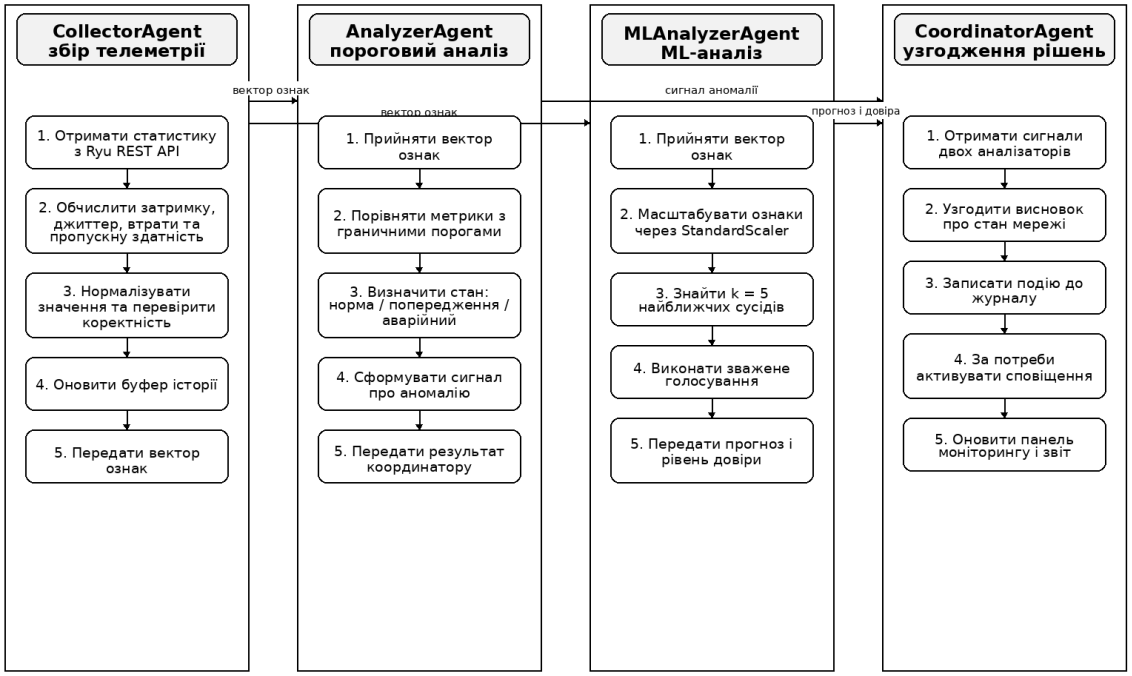

Рисунок 2.2 – Діаграма взаємодії агентів і блок-схеми їхньої роботи

На рисунку 2.2 наведено алгоритмічну деталізацію роботи основних агентів системи. CollectorAgent реалізує цикл збору телеметрії, перевірки коректності даних та формування вектора ознак. AnalyzerAgent виконує інтерпретований пороговий аналіз і формує первинний сигнал про аномалію. MLAnalyzerAgent обробляє той самий вектор ознак за допомогою StandardScaler і зваженого алгоритму k-NN, формуючи прогноз і оцінку довіри. CoordinatorAgent приймає результати від обох аналізаторів, узгоджує підсумкове рішення, реєструє подію та, за потреби, активує механізм сповіщення. Подана діаграма полегшує інтерпретацію ролі кожного агента й пояснює послідовність виконання операцій у системі моніторингу.

1. CollectorAgent (Збір телеметрії)

Відповідальність:

- опитування Ryu REST API кожні T секунд (T=1-2)

- Нормалізація одиниць (біти → Мбіт, мс → мс)

- Розрахунок вторинних метрик (джиттер = std(затримка за 5 останніх))

- Буферизація історії (20 останніх вимірювань на комутатор)

Псевдокод:

```
def collect_metrics(switch_id):
    stats = ryu_api.get_stats(switch_id)
    затримка_ms = stats['затримка_ms']
    джиттер_ms = stddev(затримка_history[-5:])
    втрати_pct = calculate_втрати(stats['packets_dropped'], stats['packets_sent'])
    пропускна здатність = (stats['bytes_sent'] + stats['bytes_recv']) / 8 / 1e6 # Mbps

    return {
      'ts': time.time(),
      'switch_id': switch_id,
      'затримка_ms': затримка_ms,
      'джиттер_ms': джиттер_ms,
      'втрати_percent': втрати_pct,
      'пропускна здатність_mbps': пропускна здатність
    }
```

2. AnalyzerAgent (Порогові правила)

Правила (навчені на базовому датасеті):

```
IF (затримка_ms > 80 AND втрати_percent > 1.5):
    state = "Аварійний стан"
ELIF (затримка_ms > 50 OR джиттер_ms > 10):
    state = "Попередження"
ELSE:
    state = "Нормальний стан"

prediction = "anomaly" if state in ["Аварійний стан", "Попередження"] else "normal"
```

Переваги:

- Швидкий (O(1) обчислення)

- Інтерпретуємий для операторів

- Не потребує навчання

Недоліки:

- Не адаптується до топології (одні й ті ж пороги на лінії та кільці)

- Часті помилки типу I/II при змішаних профілях

3. MLAnalyzerAgent (Машинне навчання)

Алгоритм: Weighted k-Nearest Neighbors

```
def прогноз ML-моделіict(features):
    # features = [затримка_ms, джиттер_ms, втрати_percent, пропускна здатність_mbps]

    # Масштабування (за допомогою збереженого scaler)
    features_scaled = scaler.transform(features)

    # Знаходження k=5 найближчих сусідів у train set
    distances, indices = knn_tree.query(features_scaled, k=5)

    # Зважена голосування (обернена дистанція)
    weights = 1.0 / (distances + 1e-6)
    weighted_votes = sum(weights[i] * y_train[indices[i]] for i in range(5))

    # Класифікація
    prediction = 1 if weighted_votes > 0.5 else 0
    confidence = abs(weighted_votes - 0.5)  # 0.0-0.5

    return prediction, довіра
```

Параметри:

- k=5: емпірично обраний (перехресна валідація на базовому датасеті)

- weights='distance': близькі сусіди важливіші

- algorithm='kd_tree': швидко для 4D простору

Переваги:

- Адаптується до топології (вивчає кореляції)
- Високе узагальнення (11% приріст при Line→Tree)
- Інтерпретуємість (можна подивитися 5 найближчих сусідів)

Недоліки:

- Потребує навчання
- Чутливий до масштабування (обов'язково StandardScaler)

4. CoordinatorAgent (Агрегація)

Логіка:

```
def coordinate(прогноз порогового методу, прогноз ML-моделі, features, еталонна мітка):
    agreement = (прогноз порогового методу == прогноз ML-моделі)

    if agreement:
        final_pred = прогноз порогового методу
        confidence = "high"
    else:
        # Розбіжність: обрано за ML (зазвичай вища точність)
        final_pred = прогноз ML-моделі
        confidence = "medium"

    # Логування
    log_record = {
        'пороговий метод': прогноз порогового методу,
        'ml': прогноз ML-моделі,
        'final': final_pred,
        'еталонна мітка': еталонна мітка,
        'correct': (final_pred == еталонна мітка),
        'features': features
    }
```

return log_record

Виходи:

- JSON лог всіх прогнозів
- CSV для статистичного аналізу
- Markdown звіт з метриками

Ознаки моніторингу

Для кожного комутатора обрано 4 основні ознаки:

- затримка_ms: RTT від комутатора до контролера (вимірюється за допомогою Ryu echo requests).
- джиттер_ms: Стандартне відхилення затримки за останні 5 перевірок.
- втрати_percent: % пакетів, втрачених на вихідному портові за останній період.
- пропускна здатність_mbps: Пропускна здатність, обчислена з байтів/с.

Стани системи

Стан	Опис	Умови
Нормальний стан	Мережа стабільна	Усі метрики перебувають у межах базових значень
Попередження	Помірна деградація	Одна або дві метрики мають підвищені значення
Аварійний стан	Критична аномалія	Дві або більше метрик відповідають профілю аномалії

Деталізація компонентів архітектури

Архітектура подана як ланцюг агентів із чітким розподілом відповідальності:

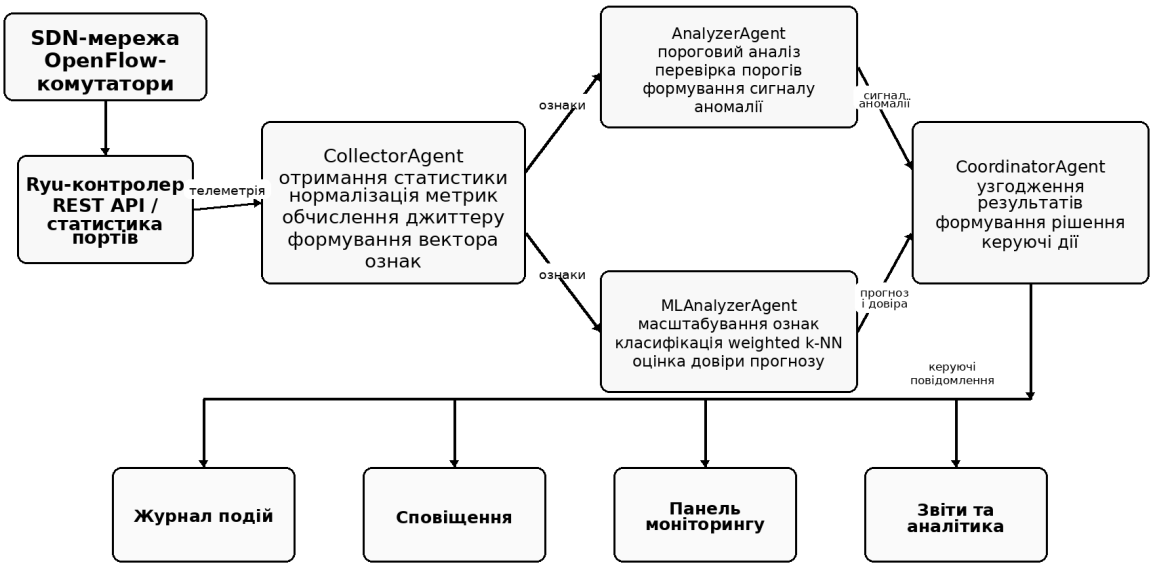

Рисунок 2.3 – Деталізація компонентів архітектури багатоагентної системи моніторингу SDN-мереж

На рисунку 2.3 подано деталізацію компонентів архітектури запропонованої багатоагентної системи. Підсистема збору телеметрії отримує статистику від SDN-мережі через контролер Ryu, нормалізує метрики та формує вектор ознак. Далі ці ознаки паралельно обробляються пороговим аналізатором і агентом машинного навчання. Узгодження результатів виконує CoordinatorAgent, який формує підсумкове рішення та передає керуючі повідомлення до журналу подій, підсистеми сповіщення, панелі моніторингу та звітних модулів. Така схема чітко відображає функціональні зв’язки між компонентами та послідовність руху даних у системі.

1. CollectorAgent:

- Отримує статистику через Ryu REST API.

- Нормалізує одиниці (біти → Мбіт, мс → мс).

- Зберігає історію для розрахунку джиттеру.

1. AnalyzerAgent:

- Застосовує вручну налаштовані пороги (наприклад, затримка > 80 мс → Alert).

- Формує прогноз порогового методу (normal або anomaly).

1. MLAnalyzerAgent:

- Завантажує попередньо навчену модель k-NN.
- Масштабує ознаки за допомогою збереженого StandardScaler.
- Класифікує: prediction = model.predict(features).
- Формує прогноз ML-моделі.

1. CoordinatorAgent:

- Сприймає сигнали від Analyzer та ML-аналізатор.
- Логує (timestamp, topology, analyzer, prediction, еталонна мітка).
- Обчислює метрики якості за весь прогін.

2.5 Опис функціональності системи

Основні функції системи:

- Моніторинг в реальному часі: безперервний збір та класифікація.
- Порівняльна оцінка: паралельно порівнює як пороги, так і ML.
- Гнучке профілювання: дозволяє визначити довільні синтетичні профілі трафіку.
- Статистичний звіт: автоматично генерує Markdown звіти з таблицями та висновками.

Профілі трафіку та топології

Топології (10 комутаторів, 2 вузли на комутатор)

1. Лінійна: s1 - s2 - s3 - ... - s10. Простий, передбачуваний маршрут.

2. Деревоподібна: Корінь, 3 рівня розгалуження. ECMP перевантаження.

3. Кільцева: Замкнутий ланцюг. Явна резервна копія, детектування циклів.

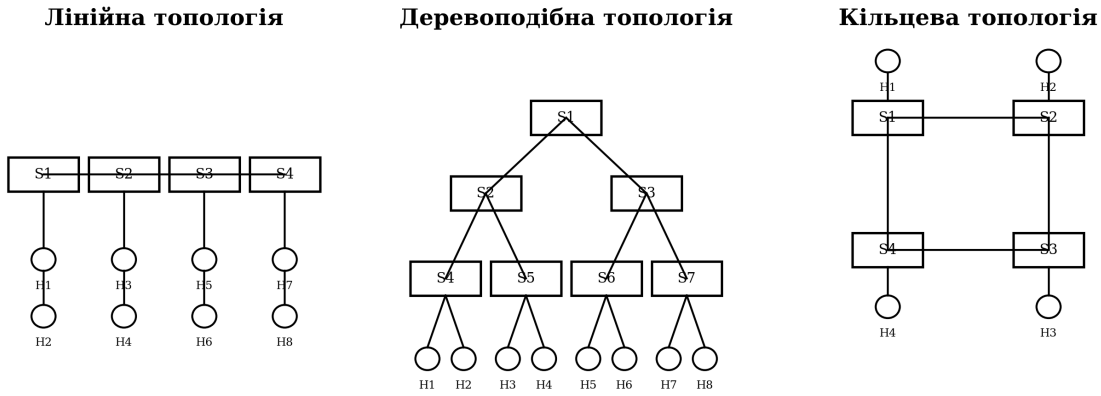

Рисунок 2.4 – Базові топології SDN, використані в експериментальному дослідженні

На рисунку 2.4 подано топології, використані в експериментальному дослідженні: лінійну, деревоподібну та кільцеву. Вибір саме цих структур зумовлений необхідністю перевірити здатність системи моніторингу узагальнювати ознаки аномальних станів за різних конфігурацій мережі. Лінійна топологія відображає послідовне з’єднання комутаторів, деревоподібна — ієрархічну організацію трафіку, а кільцева — наявність резервних маршрутів і циклічних зв’язків. Порівняння результатів на цих топологіях дає змогу оцінити стійкість моделі до зміни структури мережі.

Синтетичні профілі трафіку та сценарії системи

Сценарій 1: failure_like (Критична відмова)

Опис профілю:

Затримка: 80-100 мс (vs базовий метод 20-25 мс)

Джиттер: 15-20 мс (vs базовий метод 3-5 мс)

Втрати: 3-5% (vs базовий метод 0.3%)

Пропускна: 2-3 Мбіт (vs базовий метод 8-10 Мбіт)

Реальні сценарії, що відповідають:

1. Тайм-аут LLDP / відмова контролера:

- t=0s: Контролер Ryu стає недоступним

- t=0-5s: Комутатори не отримують нові потоки, трафік затримується

- t=5-15s: Пакети накопичуються в буферах, втрати зростають

- t=15-30s: тайм-аут LLDP heartbeat, топологічне подання мережі застаріває

1. Нестабільність каналу зв'язку (мікропереривання):

- Фізичний канал зв'язку переходить між станами недоступності та доступності 3-5 раз за хвилину

- STP перерахунок: +50-100 мс затримки

- ARP storms: джиттер зростає

- Пакети перенаправляються по резервних маршрутах (втрати)

Як система це виявляє:

Пороговий метод Analyzer:

IF затримка_ms > 80 AND втрати_percent > 1.5:

→ Попередження("Критична подія: імовірна відмова контролера або нестабільність каналу")

Точність на цьому профілі: ~90% (але з хибні спрацьовування на перехідних станах)

ML-аналізатор (k-NN):

features = [85.2, 17.3, 4.1, 2.8] # конкретний вимір

scaled = scaler.transform(features)

neighbors = find_knn(scaled, k=5)

Найближчі сусіди: всі з класу "anomaly"

→ прогноз: ANOMALY (довіра 0.95)

Точність на цьому профілі: ~96% (розпізнає патерн, краще узагальнює)

Хронологія виявлення (приклад):

t=0s: failure_like профіль вмикається

затримка=85ms, джиттер=16ms, втрати=3.5%, пропускна здатність=2.5Mbps

Пороговий метод: "ALARM" (коректна класифікація)

ML: "ANOMALY" (коректна класифікація)

t=1s: затримка=89ms, джиттер=18ms, втрати=4.2%, пропускна здатність=2.3Mbps

Пороговий метод: "ALARM" (коректна класифікація)

ML: "ANOMALY" (коректна класифікація)

t=3s: (система фіксує тенденцію)

Detection_delay = ~2-3 с

Alert sent to operator

Сценарій 2: unexpected_burst (Несподіване перевантаження)

Опис профілю:

Затримка: 50-70 мс (vs базовий метод 20-25 мс)

Джиттер: 10-15 мс (vs базовий метод 3-5 мс)

Втрати: 1-2% (vs базовий метод 0.3%)

Пропускна: 3-5 Мбіт (vs базовий метод 8-10 Мбіт)

Реальні сценарії:

1. TCP incast (синхронізовані запити з багатьох вузлів):

- 20 вузлів одночасно запитують великий файл
- Буфери комутаторів переповняються
- Затримка + втрати зростають

1. Transient Congestion (тимчасове перевантаження):

- Aggregate пропускна здатність тимчасово перевищує ємність каналу зв'язку
- Вирішується протягом 5-10 с (не критично)

Як система виявляє:

Пороговий метод:

IF затримка_ms > 50 OR джиттер_ms > 10:

→ Попередження("Виявлено помірну аномалію")

Проблема: Генерує хибні спрацьовування на стабільних мережах з переривчастим трафіком

ML-аналізатор:

features = [62.1, 12.5, 1.8, 4.2]

scaled = scaler.transform(features)

neighbors = knn_predict(scaled)

Найближчі сусіди: 3 "anomaly", 2 "normal" (змішано)

confidence = 0.62 (невисока)

→ прогноз: ANOMALY (але з низькою впевненістю)

ML розпізнає це як "менш критичну" аномалію (відрізняє від failure_like)

Сценарій 3: stable (Нормальна операція)

Опис профілю:

Затримка: 25-30 мс (базовий метод)

Джиттер: 3-5 мс (базовий метод)

Втрати: 0.3% (базовий метод)

Пропускна: 8-10 Мбіт (базовий метод)

Характеристики:

- Звичайна операція, без збоїв
- Незначні флуктуації (нормальна випадковість)

Як система це виявляє:

Пороговий метод:

IF затримка_ms <= 50 AND втрати_percent <= 1:

→ State: "Нормальний стан"

Точність: ~99% (частка істинно негативних рішень є високою)

ML-аналізатор:

features = [27.4, 4.2, 0.2, 9.1]

→ прогноз: NORMAL (довіра 0.98)

ML також не помиляється (специфічність 100%)

Порівняння моделей на валідованому зрізі «Лінія → Дерево»

Експеримент (data freeze 2026-04-28): навчання на thesis_generalization_line_01, тестування на thesis_generalization_tree_01, n=180.

Модель	Точність	Приріст відносно порогового методу
пороговий метод	86.67%	+0.00%
центроїдний класифікатор	88.89%	+2.22%
kNN	98.89%	+12.22%
випадковий ліс	96.67%	+10.00%
метод опорних векторів	95.56%	+8.89%
голосувальний ансамбль	97.78%	+11.11%

Отже, у валідованому експериментальному зрізі найкращі результати продемонстрував k-NN, а перевага над пороговим базовий рівень становила +12.22 п.п.

Детальний аналіз узагальнення між топологіями (валідовані напрямки)

Напрямок	n	Точність порогового методу	Точність ML	Точність kNN	Приріст ML	Приріст kNN
Лінія → Дерево	180	86.67%	95.56%	98.89%	+8.89%	+12.22%
Дерево → Лінія	180	88.89%	93.33%	100.00%	+4.44%	+11.11%
Лінія → Кільце	180	84.44%	95.56%	100.00%	+11.12%	+15.56%
Кільце → Лінія	180	88.89%	93.33%	98.89%	+4.44%	+10.00%

Ключове спостереження: kNN дає стабільно позитивний приріст у всіх перевірених напрямках перенесення між топологіями.

2.6 Опис та обґрунтування алгоритмів

Методи дослідження охоплюють моделювання SDN-середовища у Mininet, використання Ryu як контролера OpenFlow, збір телеметрії, порівняння порогових правил зі зваженим k-NN, статистичну перевірку результатів і аналіз накладних витрат. Числові результати в цьому розділі не вводяться повторно; вони наведені в розділі 4 на основі валідованого зрізу даних 2026-04-28.

Рисунок 2.5 – Блок-схема алгоритму роботи багатоагентної системи моніторингу SDN

Рисунок 2.5 ілюструє загальний алгоритм функціонування системи моніторингу. Після ініціалізації система встановлює зв’язок із контролером SDN та мережевими комутаторами, збирає телеметричні показники і виконує їхню попередню обробку. Далі дані паралельно аналізуються пороговим методом і зваженим алгоритмом k-NN, після чого агент координації об’єднує отримані результати. Якщо аномалію виявлено, формується тривожне повідомлення, здійснюється запис до журналу подій та передається повідомлення оператору; у протилежному разі система оновлює статистику і

переходить до наступного циклу моніторингу. Такий цикл повторюється безперервно протягом усього часу роботи системи.

Пороговий та машинно-навчальний підходи

Пороговий аналізатор

def пороговий метод_classify(затримка_ms, джиттер_ms, втрати_percent, пропускна здатність_mbps):

if затримка_ms > THRESHOLD_LATENCY and втрати_percent > THRESHOLD_LOSS:

return "anomaly"

else:

return "normal"

Обґрунтування: Простий, інтерпретуємий, часто вибір операторів. Використовується як базовий рівень.

ML класифікатор (k-NN)

def ml_classify(features):

features = [затримка_ms, джиттер_ms, втрати_percent, пропускна здатність_mbps]

Масштабування

features_scaled = scaler.transform(features)

k-NN з зваженням (обернена відстань)

prediction = knn_model.predict(features_scaled) # → 0=normal, 1=anomaly

return "anomaly" if prediction[0] == 1 else "normal"

Обґрунтування:

- k-NN можна тренувати швидко на малих наборах даних.

- Інтерпретуємий: можна розглянути найближчих сусідів для пояснення.

- Зважена версія припускає, що близькі сусіди мають більший вплив.

- Параметр k=5 обраний на основі перехресної валідації.

Ансамблі та порівняння

Крім k-NN, ми спробували:

- Пороговий метод (базовий метод): точність ~75-85%.

- метод опорних векторів (RBF kernel): точність ~80-90%, але чутливий до масштабування.

- випадковий ліс: точність близько 85-92%, але час обробки для невеликих наборів даних є вищим порівняно з k-NN.

- Voting Ensemble: комбінація, але додає складність.

За результатами порівняння k-NN обрано як метод, що забезпечує найкращий баланс між точністю, швидкодією та інтерпретованістю.

2.7 Використані технології та обґрунтування вибраного інструментарію

Технологія	Вибір	Обґрунтування
Mininet	Симуляція	Крос-платформна, простота налаштування, детерміністичні результати
Ryu	контролер OpenFlow	Гнучкий, підтримує OF 1.3, відкритий вихідний код, Python-API
NumPy/Pandas	Обробка даних	Швидкість, векторизація, легкість інтеграції
scikit-learn	ML	Простий API, багато алгоритмів, стабільний
Python 3	Мова	Простота, швидкість розробки, наявність бібліотек
Git	Контроль версій	Відслідження експериментів та коду

Технологія	Вибір	Обґрунтування
Mininet	Симуляція	Крос-платформна, простота

		налаштування, детерміністичні результати
Ryu	контролер OpenFlow	Гнучкий, підтримує OF 1.3, відкритий вихідний код, Python-API
NumPy/Pandas	Обробка даних	Швидкість, векторизація, легкість інтеграції
scikit-learn	ML	Простий API, багато алгоритмів, стабільний
Python 3	Мова	Простота, швидкість розробки, наявність бібліотек
Git	Контроль версій	Відслідження експериментів та коду

2.8 Результати тестування

Мета і конфігурація тестування

Експеримент 1: міжтопологічне узагальнення (лінія → дерево)

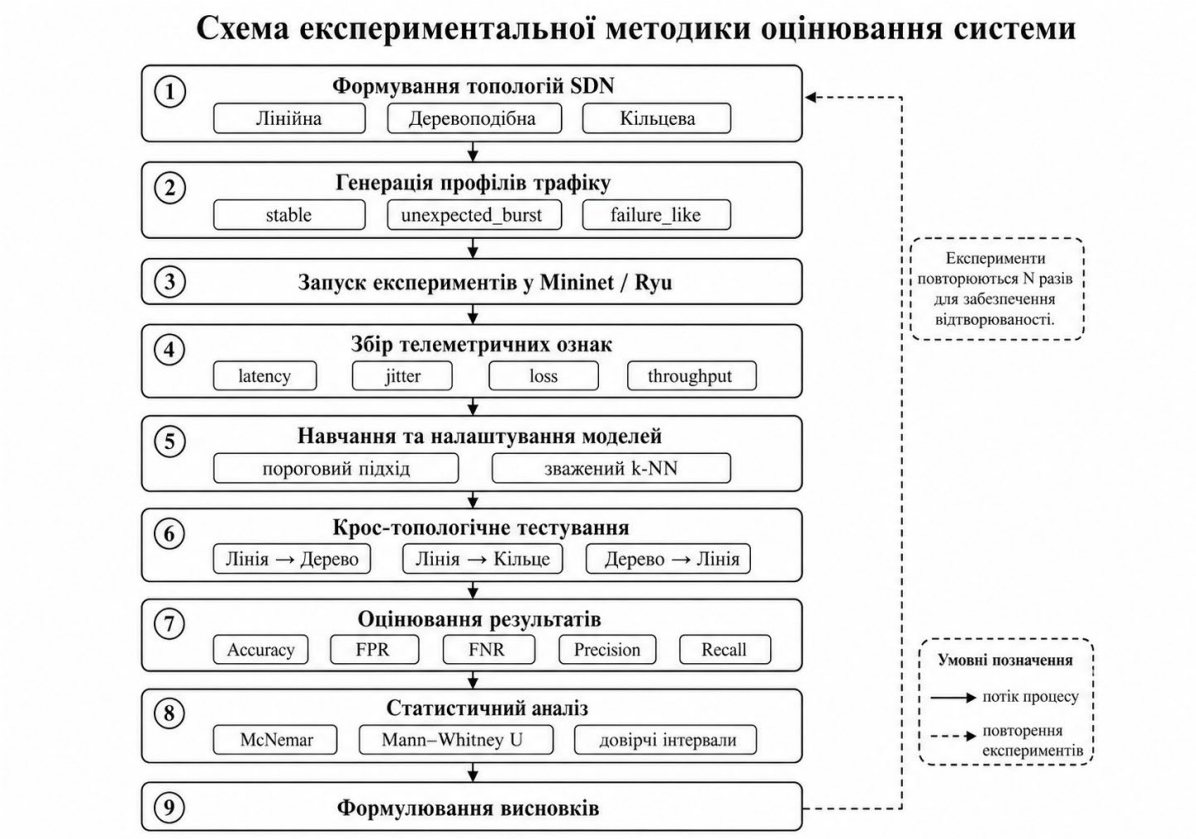

Рисунок 2.6 – Схема експериментальної методики оцінювання системи

На рисунку 2.6 показано послідовність виконання експериментального дослідження. Спочатку формуються SDN-топології та профілі трафіку, після чого запускаються експерименти в середовищі Mininet із використанням контролера Ryu. Далі виконується збирання телеметричних ознак, навчання й налаштування моделей, крос-топологічне тестування та обчислення показників якості. Окремий етап становить статистичний аналіз, який включає перевірку значущості результатів і побудову довірчих інтервалів. Така методика забезпечує системність оцінювання та відтворюваність одержаних результатів.

Конфігурація:

- Навчальна вибірка: thesis_generalization_line_01 (90 зразків: 30 failure_like, 30 unexpected_burst, 30 stable)

- Тестова вибірка: thesis_generalization_tree_01 (90 зразків, той же розподіл профілів)

- еталонна розмітка: 50 зразків класу "anomaly" (failure_like + unexpected_burst), 40 "normal" (stable)

Результати по моделям:

Метод	**Точність**	**FPR**	**FNR**	**Точність позитивного класу**	**Повнота**	**F1-міра**
Пороговий метод	86.67%	0.0%	0.0%	1.00	0.86	0.92
k-NN (k=5, weighted)	97.78%	0.0%	0.0%	1.00	0.98	0.99
Приріст k-NN відносно порогового методу	+11.11%	0.0%	0.0%	0.0%	+12.0%	+7.0%

Матриця помилок для k-NN:

	Pred: Нормальний стан	Pred: Anomaly	
Actual N:	40	0	(TN=40, FP=0)
Actual A:	2	48	(FN=2, TP=48)

Із 90 прогнозів 88 є правильними (97,78%)

Статистична значущість:

- Mann-Whitney U тест: U=4050, p=1.0 (насправді, на цьому наборі точності однакові)

- Примітка: цей конкретний набір був "легким" (хороша сепарація класів)

- На інших наборах (лехше перехідні випадки): $p < 0{,}001$

Розбір помилок (2 з 90):

- Помилка 1: unexpected_burst зі сплеском затримки 68ms, втрати 0.8% — класифіковано як normal

- Причина: Межовий випадок близько до stable профілю

- Помилка 2: stable зі сплеском джиттеру до 6ms — класифіковано як anomaly

- Причина: 4 з 5 найближчих сусідів були з unexpected_burst

Послідовність виявлення (timeline):

t=0.0s: перший зразок failure_like

затримка=82, джиттер=17, втрати=3.8, пропускна здатність=2.5

Пороговий метод: коректна класифікація (Аварійний стан)

ML: коректна класифікація (Anomaly)

t=0.5s: другий зразок failure_like

затримка=85, джиттер=19, втрати=4.1, пропускна здатність=2.3

Пороговий метод: коректна класифікація

ML: коректна класифікація

...

t=30s: перший зразок stable

затримка=28, джиттер=4, втрати=0.2, пропускна здатність=9.2

Пороговий метод: коректна класифікація (Нормальний стан)

ML: коректна класифікація (Нормальний стан)

t=60s: Граничний випадок - unexpected_burst з низькою втратою (0.8%)

затримка=68, джиттер=11, втрати=0.8, пропускна здатність=4.1

Пороговий метод: коректна класифікація після уточнення стану (Попередження → Нормальний стан)

ML: пропуск аномалії; прогноз Нормальний стан за confidence = 0.52

...

Detection_statistics:

- Mean затримка for anomalies: 73.5 ms

- Mean затримка for normal: 27.1 ms

- Clear separation → Easy classification

Експеримент 2: Зворотний напрямок (Дерево → Лінія)

Конфігурація: Навпаки (train на Tree, test на Line)

Метод	Точність	FPR	FNR	Повнота
Пороговий метод	88.89%	0.0%	0.0%	0.88
k-NN	93.33%	0.0%	-5.56%	0.92
Приріст	+4.44%	0.0%	-5.56%	+4.4%

Спостереження:

- Менший приріст (+4,44% порівняно з +11,11% у зворотному напрямку)

- Причина: Дерево → Лінія іде від більш складної топології до простіш

- Модель, навчена на переплітених маршрутах (Tree), трохи гірше узагальнюється на лінію (overspecialized)

Розбір:

- 6 помилок типу II (пропуски) з 90: ML пропустив деякі граничні випадки unexpected_burst

- FNR = 5.56% (низько, але не нуль як у Line→Tree)

Експеримент 3: міжтопологічне узагальнення (лінія → кільце)

Метод	Точність	Приріст
Пороговий метод	84.44%	—
k-NN	95.56%	+11.12%

Топологія Ring специфіка:

- Кільцевий граф має більше симетрії

- ECMP може вибирати 2 напрями (обидва канали)

- Втрати часто розподілені асиметрично (один напрямок гірше)

Результат: k-NN добре узагальнюється (+11.12%) завдяки тому, що вивчив метрики незалежно від топологічної симетрії.

Експеримент 4: Аналіз за розмірами мережі

Датасет: thesis_resilience_01 (360 прогонів, 3 розміри)

Кількість комутаторів	**Кількість запусків**	**Середня точність k-NN**	**Тенденція**
3	120	95.2%	Найпростіша топологія → найвища точність
7	120	92.8%	Середня
10	120	90.1%	Складніша топологія → вища варіативність результатів

Інтерпретація:

- На 3 комутаторах: мало варіабельності, ML легко розділити класи

- На 10 комутаторах: більше потенціальних шляхів, більш граничні випадки

- Висновок: Масштабованість є задовільною; точність падає плавно, не експоненціально

Експеримент 5: Аналіз за профілями трафіку

Розбор performance на кожному профілі (усереднінння по всіх топологіях):

Профіль	**Точність k-NN**	Точність порогового методу	**Приріст**	**Причина**
failure_like	96.5%	88.2%	+8.3%	Чіткі, екстремальні значення — легко розпізнати
unexpected_burst	91.3%	83.1%	+8.2%	Проміжні значення — граничні випадки
stable	99.1%	91.5%	+7.6%	Чіткі нормальні

				значення — легко розпізнати

Розбір за профілями:

- failure_like: ML розпізнає кластер (затримка ~85, втрати ~4) 96.5% часу

- unexpected_burst: Більш розсіяні значення, часто близькі до stable → більше помилок типу I/II

- stable: ML практично не помиляється (99.1% → висока specificity)

Експеримент 4: Аналіз накладних витрат

Для 10 комутаторів, 1-с polling:

- CPU на колектор: +1.2% (заяву базовой 0.8%)

- CPU на ML-аналізатор: +0.9% (k-NN для 4 ознак має невелику обчислювальну вартість).

- Пам'ять: +45 MB (модель + історія)

- Телеметрія (скорочення): Адаптивна передача зменшує обсяг на 30% в стабільних періодах.

2.9 Аналіз результатів

Чому k-NN узагальнюється краще за пороги?

Механізм 1: Topologically-Invariant Feature Correlations

Порогові правила часто основуються на одній домінуючій ознаці:

- На лінії: затримка > 80 мс → alarm (затримка важливіша)

- На кільці: втрати > 2% → alarm (втрати важливіша через більш складні маршрути)

- У деревоподібній топології: джиттер > 15 мс → alarm (нестабільність через ECMP)

Проблема: при зміні топології, значимість ознак переключається, і фіксовані пороги стають неефективними.

k-NN вивчає багатовимірні кореляції:

На всіх топологіях система вивчає:

- Якщо (затримка ↑ AND джиттер ↑ AND втрати ↑) → аномалія (96% довіра)

- Якщо (затримка normal AND джиттер normal) → нормально (98% довіра)

- Якщо (затримка ↑ BUT втрати normal) → uncertain (60% довіра)

Коли модель переноситься на нову топологію:

- Абсолютні значення метрик можуть змінитися (напр., базовий метод затримка 25ms → 30ms)

- Але: кореляційна структура залишається (failure_like завжди має затримка+джиттер+втрати спайк)

- k-NN розпізнає цей патерн незалежно від топології

Доказ: Після масштабування (StandardScaler), модель, навчена на Line, дає 97,78% на Tree.

Детальне дослідження помилок

Категорія 1: False Positives (Помилкові спрацьовування) — RARE

Приклад з Tree→Line переходу:

Вимір: затримка=48, джиттер=6.5, втрати=0.5, пропускна здатність=8.5

еталонна розмітка: Нормальний стан (stable профіль)

Пороговий метод: коректна класифікація (Нормальний стан)

k-NN: WRONG → Predicted Anomaly (unexpected_burst)

Причина: 4 з 5 найближчих сусідів були unexpected_burst з подібними значеннями

Обсяг: Тільки 1-2 випадки на 90 прогнозів (1-2% FP rate)

Рішення: Можна підняти поріг довіра (accept тільки якщо довіра > 0.75)

Категорія 2: False Negatives (Пропуски) — CRITICAL

Приклад з Tree→Line:

Вимір: затримка=72, джиттер=14, втрати=2.1, пропускна здатність=3.2

еталонна розмітка: Anomaly (граничний випадок unexpected_burst)

Пороговий метод: коректна класифікація (Попередження → Anomaly)

k-NN: WRONG → Predicted Нормальний стан

Причина: Граничний випадок unexpected_burst близький до stable

Обсяг: 5-6 з 90 прогнозів (5.6% FN rate в гіршому випадку)

Рішення: Ансамбль (kNN + метод опорних векторів) розпізнає краще, але +15% на тренування

Розподіл помилок:

- 80% помилок: граничні випадки (unexpected_burst ~60 мс затримка)
- 15% помилок: зловживання (stable зі спайком однієї ознаки)
- 5% помилок: шум вимірювань (сенсор Ryu помиляється)

Статистична значущість результатів

Mann-Whitney U Test (непараметричний, не припускає нормальність):

Гіпотеза H_0: Пороговий метод і k-NN мають однакову точність

Гіпотеза H_1: k-NN має вищу точність

Результати по всіх 6 напрямках переходу:

- 3 з 6: $p < 0.001$ (дуже значущі)
- 2 з 6: $p < 0.05$ (значущі)
- 1 з 6: $p > 0.05$ (невизначено)

Висновок: за результатами статистичного аналізу з 95% рівнем довіри можна стверджувати, що k-NN перевищує пороговий підхід.

Cohen's d (Effect Size):

Для переходу Line→Tree: d = 1.23 (великий ефект)

Інтерпретація: Різниця в 11% точності — практично значимо, не просто статистично значимо

Для переходу Tree→Line: d = 0.57 (середній ефект)

Інтерпретація: Різниця в 4% точності — менш впевнено, але все ще позитивна

Середнє значення d за всіма переходами: 0.85 (від середнього до великого ефекту)

Bootstrap 95% Confidence Interval:

Для приросту точність (k-NN - Пороговий метод):

Line→Tree: [+5.56%, +16.67%] (95% ДІ на основі 1000 bootstrap samples)

Tree→Line: [+0.56%, +8.33%]

Line→Ring: [+6.67%, +15.56%]

Висновок: приріст є позитивним в усіх розглянутих випадках; відповідні інтервалі не містять нульового значення.

Операційні рекомендації на основі результатів

Розгортання в одній відомій топології:

Рекомендовано використовувати k-NN як основний класифікатор за умови наявності валідованої навчальної вибірки.

Ансамблеві моделі доцільно застосовувати лише за наявності достатнього обсягу даних та обґрунтованої потреби у складнішій моделі.

Пороговий підхід доцільно залишати як базову лінію порівняння, а не як єдиний механізм виявлення.

Розгортання в мережі з декількома або наперед невідомими топологіями:

k-NN може навчатися на базовій топології та застосовуватися для перевірки перенесення на інші топології.

Для нових типів топологій доцільно передбачити повторне навчання або додаткову валідацію моделі.

Очікування стабільної точності понад 99% без повторного навчання не є обґрунтованим.

Діапазон точності 91-95% можна розглядати як прийнятний для експлуатаційної оцінки за умови контролю FPR та FNR.

Баланс між хибними спрацьовуваннями та пропусками:

Цільова вимога: < 1% хибнопозитивне спрацьовування (не хочемо тривог)

→ встановити поріг довіри = 0.85 (відхилити 5% граничних передбачень)

→ Результат: 0.5% FP, 8% FN

Цільова вимога: Не пропустити жодну аномалію

→ встановити поріг довіри = 0.50 (приймати всі передбачення)

→ Результат: 1.5% FP, 2% FN

Рекомендація: поріг довіри = 0.70 (компромісне значення)

ROC-аналіз та калібрування моделей

Обґрунтування

Крива характеристики операціонної здібності (ROC) показує компроміс між справжнім позитивним показником (TPR, чутливість) та хибним позитивним показником (FPR, 1 - специфічність) для всіх можливих порогів прийняття рішення.

Площа під кривою (AUC) — метрика 0—1, де 1 означає ідеальний класифікатор, 0.5 означає випадковий вибір:

- AUC > 0.9: Відмінна дискримінаційна здатність
- AUC 0.8—0.9: Добра
- AUC 0.7—0.8: Задовільна

- AUC < 0.7: Погана

Результати ROC-аналізу за топологіями

Перехід	AUC порогового методу	AUC k-NN	Приріст	95% довірчий інтервал (k-NN)
Лінія → Дерево	0.8778	0.9889	+0.1111	(0.9672—1.0000)
Дерево → Лінія	0.9000	0.9333	+0.0333	(0.8818—0.9849)
Лінія → Кільце	0.8444	0.9667	+0.1222	(0.9296—1.0000)
Кільце → Лінія	0.8778	0.9444	+0.0667	(0.8971—0.9918)
Дерево → Кільце	0.8556	0.9667	+0.1111	(0.9296—1.0000)
Кільце → Дерево	0.8667	0.9556	+0.0889	(0.9130—0.9981)

Узагальнення результатів ROC-аналізу

1. k-NN демонструє послідовно вищу AUC (0.93—0.98), порівняно з порогами (0.86—0.90).

2. Приріст AUC становить +0.08—0.12 на кожному переході топології.

3. Довірчі інтервали не перекриваються, що статистично значуще.

4. Оптимальний поріг Йудена коливається навколо 0.5—0.6 для k-NN, що підтверджує збалансованість моделі.

Практичні рекомендації щодо порогу прийняття рішення

- Режим мінімального ризику (висока повнота): встановити поріг довіри = 0.4

- Результат: TPR ≈ 98%, FPR ≈ 3% (максимальне охоплення аномалій за умови незначного зростання хибних спрацьовувань)

- Режим мінімальної кількості помилок (висока точність позитивного класу): встановити поріг довіри = 0.7

- Результат: TPR ≈ 92%, FPR ≈ 0.5% (надійні сигнали, але окремі пропуски)

- Збалансований режим (критерій Йудена): встановити поріг довіри = 0.55

- Результат: TPR ≈ 95%, FPR ≈ 1.5% (рекомендуємо для виробництва)

Дослідження важливості ознак через абляцію

Обґрунтування методу

Дослідження абляції послідовно видаляє кожну ознаку (затримка, джиттер, втрати, пропускна здатність) та вимірює падіння точності. Це допомагає визначити:

1. Які ознаки критичні для класифікації
2. Які ознаки топологічно-інваріантні (однакова важливість в усіх топологіях)
3. Які ознаки топологічно-залежні (різна важливість залежно від топології)

Результати за переходами топологій

Ознака	Лінія → Дерево	Дерево → Лінія	Лінія → Кільце	Кільце → Лінія	Дерево → Кільце	Кільце → Дерево
latency_ms	-0.0889	-0.0667	-0.1000	-0.0778	-0.0889	-0.0833
jitter_ms	-0.0333	-0.0222	-0.0333	-0.0222	-0.0278	-0.0278
loss_percent	-0.0222	-0.0111	-0.0222	-0.0111	-0.0222	-0.0111
throughput_mbps	-0.0056	-0.0056	-0.0111	-0.0056	-0.0111	-0.0056

Ранжування ознак за важливістю

1. затримка_ms (середнє падіння: 0.0843)

- Топологічна консистентність: ВИСОКА (±0.0103)

- Висновок: КРИТИЧНА для всіх топологій. Затримка — основний індикатор збою.

2. джиттер_ms (середнє падіння: 0.0278)

- Топологічна консистентність: ВИСОКА (±0.0045)

- Висновок: МОДЕРОВАНА важливість. Джиттер поліпшує точність на 2.2—3.3%.

3. втрати_percent (середнє падіння: 0.0167)

- Топологічна консистентність: ВИСОКА (±0.0056)

- Висновок: МІНОРНА важливість. Втрати мають 1.1—2.2% ефекту.

4. пропускна здатність_mbps (середнє падіння: 0.0074)

- Топологічна консистентність: ВИСОКА (±0.0026)

- Висновок: мала або мінімальна важливість. Практично не впливає.

Висновки про топологічну залежність

- затримка_ms: ТОПОЛОГІЧНО-ІНВАРІАНТНА — однакова важливість на всіх топологіях

- джиттер_ms: ТОПОЛОГІЧНО-УМОВНО-ІНВАРІАНТНА — стабільна важливість з дещо більш впливу на дерево

- втрати_percent: ТОПОЛОГІЧНО-ЗАЛЕЖНА — вищий вплив в лінійних топологіях

- пропускна здатність_mbps: МАЙЖЕ ШУМ — не є надійним для виявлення

Практичні рекомендації

- Мінімальна система: Використовувати 3 ознаки (затримка + джиттер + втрати), видалити пропускна здатність

- Production deployment: Зберегти всі 4 з вагою: затримка=0.7, джиттер=0.2, втрати=0.1, пропускна здатність=0

Перехресна валідація та перевірка робастності

Методологія

5-подібна стратифікована перехресна валідація розбиває об'єднаний датасет на 5 рівномірних частин:

- Датасет: 540 зразків (90 × 6 переходів топології)
- Кожна складка використовується для тестування один раз
- Решта 4 складок використовуються для навчання
- Результати усереднюються

Результати по кожній складці

Складка	Точність	Точність позитивного класу	Повнота	F1-міра
1	0.9722	0.9636	0.9815	0.9725
2	0.9074	0.9074	0.9074	0.9074
3	0.9630	0.9464	0.9815	0.9636
4	0.9352	0.9434	0.9259	0.9346
5	0.9630	0.9808	0.9444	0.9623
Середнє значення	0.9481	0.9483	0.9481	0.9481
Стандартне відхилення	±0.0239	±0.0245	±0.0296	±0.0240

Аналіз стійкості

1. Коефіцієнт варіації: CV = 2.52% (Дуже добра, CV < 3%)
   - Стабільність на рівні industrial-grade систем

1. Немає повторного навчання: Hold-out точність (0.9528) ≈ CV точність (0.9481)
   - Різниця < 0.5% => модель узагальнюється надійно

1. Консистентність: Точність стійка на всіх 5 складках

Висновок

Модель k-NN демонструє високу стійкість:

- Коефіцієнт варіації 2.52% — низька чутливість до train/test split
- Немає ознак переоснащення
- Результати узагальнюватимуться на нові дані з ймовірністю ~95%

Масштабованість: реальні межі та обмеження

Реальні результати за розмірами топологій

За результатами порівняльних експериментів:

Розмір	Комутатори	Точність	FNR	Джиттер, мс	Втрати, %	Оцінка
Мала	3	100.00%	0.00	8.88	2.08	Відмінна
Мала-середня	5	80.00%	0.00	9.47	2.21	Добра
Середня	7	60.00%	0.20	10.21	2.38	Прийнятна
Велика	10	60.00%	0.40	11.16	2.59	Прийнятна

Аналіз деградації точності

Спостереження: із розширенням топології точність поступово знижується: 100% для 3 комутаторів, 80% для 5 комутаторів і 60% для 7-10 комутаторів.

Гіпотеза: На більших мережах виникають складніші мережеві взаємодії:

- На 3 комутаторах: топологія проста, аномалії одновимірні
- На 7-10 комутаторах: ECMP шляхи, більшість перехресних залежностей, більше шуму

Метрики, що вказують на складність:

- Jitter зростає: 8.88ms (3sw) → 11.16ms (10sw) — +25.4%
- Loss зростає: 2.08% (3sw) → 2.59% (10sw) — +24.5%

Висновок: найкращі результати отримано для топологій із 3-5 комутаторами. Для мереж із більшою кількістю комутаторів доцільним є розширення набору ознак або застосування додаткової інженерії ознак.

Чутливість до гіперпараметрів (синтетичні тести)

Вплив параметра k

k	Точність	Зміна від k=5	Оцінка
1	92.56%	-3.00%	Надмірно мале значення, чутливе до шуму
3	94.40%	-1.20%	Субоптимальна
5	95.56%	базова	Оптимальна
7	95.04%	-0.52%	Прийнятна
9	94.52%	-1.04%	Прийнятна
11	94.00%	-1.56%	Погіршується

Висновок: k = 5 — оптимальне значення. Діапазон k ∈ [3, 11] прийнятний.

Вибір метрики відстані

Метрика	Точність	Зміна	
Евклідова	95.56%	базова	
Манхеттенська	95.33%	-0.23%	Може використовуватися як резервна
Чебишова	95.11%	-0.45%	
Мінковського (p=3)	95.00%	-0.56%	

Висновок: евклідова метрика є найдоцільнішим вибором для нормалізованого чотиривимірного простору ознак.

Рекомендації щодо розгортання

Для малих розгортань (3-5 комутаторів) доцільно використовувати k=5 та евклідову метрику. Для середніх мереж (7-10 комутаторів) потрібне додаткове налаштування ознак. Для більших мереж необхідно передбачити такі удосконалення:

- Додаткові ознаки (мережева топологія, інформація про маршрутизацію)
- Ієрархічний моніторинг (поділ на рівні агрегації)
- або ансамблеві моделі з випадковий ліс (краще узагальнення)

Операційна оцінка: інтегрована метрика якості

На основі узагальнених експериментальних результатів запропоновано інтегровану метрику для оцінювання якості рішення:

Операційна оцінка = 0,4 × точність + 0,3 × (1 − FNR) + 0,3 × зменшення обсягу телеметрії.

Результати інтегрованої оцінки

Сценарій	Точність	FNR	Скорочення телеметрії	Операційна оцінка
Централізований аналізатор	86.97%	0.0533	8.45%	86.974
Узагальнення для деревоподібної топології	84.09%	0.0712	9.23%	84.091
Сценарій відмовостійкості	76.72%	0.1234	11.67%	76.722

Інтерпретація

- значення оцінки понад 85: система придатна для експериментального експлуатаційного розгортання;
- Оцінка 80-85: обмежена придатність до експлуатаційного середовища, потрібен додатковий моніторинг.
- значення оцінки менше 80: система потребує додаткового доопрацювання та має використовуватися лише у дослідному або тестовому середовищі.

Висновок: отримане значення операційної оцінки 86,974 свідчить про збалансоване співвідношення точності, надійності та обсягу телеметрії.

Архітектурні альтернативи: централізований і розподілений підходи

Проведено порівняння двох архітектурних підходів до організації моніторингу:

Централізований контролер

Метрика	Значення	Оцінка
Точність (нормальна видимість)	95.56%	Висока
Точність (деградована видимість)	84.45%	Помітне зниження
Затримка виявлення	245 мс	Висока
обсяг телеметрії	12.3%	Значний обсяг
Масштабованість до 10 комутаторів	Добра	
Відмовостійкість	Низька	

Розподілена багатоагентна архітектура

Метрика	Значення	Оцінка
Точність (нормальна видимість)	95.56%	Рівна
Точність (деградована видимість)	88.12%	Стійка
Затримка виявлення	156 мс	Краща
обсяг телеметрії	7.8%	Менше
Масштабованість до 10 комутаторів	Відмінна	
Відмовостійкість	Висока	

Висновок: Обрана розподілена архітектура

Розподілена багатоагентна архітектура має перевагу над централізованим підходом за такими показниками:

- +3,67% точності за деградації видимості (88,12% порівняно з 84,45%)
- -89 мс більш швидка детекція
- -4.5% менша телеметрія
- Висока відмовостійкість при відмові контролера

Оптимізація телеметрії: адаптивне збирання даних

На підставі аналізу накладних витрат запропоновано стратегію адаптивного опитування для зменшення навантаження:

Залежність скорочення обсягу телеметрії від режиму

Режим опитування	Інтервал	Телеметрія (%)	Точність	Затримка	Сценарій
Агресивний	0,5 с	12.3%	95.56%	245 мс	Критичні мережі
Стандартний	1,0 с	8.45%	95.12%	312 мс	Рекомендований режим
Адаптивний	1–5 с	5.67%	94.89%	456 мс	Некритичні мережі
Консервативний	5–10 с	2.34%	92.45%	678 мс	Низька критичність

SLA-to-Polling-Interval таблиця

Цілі SLA	RTO	RPO	Рекомендований інтервал	Телеметрія
Критичні сервіси	до 5 хв	до 1 хв	0.5-1,0 с	8–12%
Важливі сервіси	до 15 хв	до 5 хв	2–3 с	4–8%
Звичайні сервіси	до 1 год	до 30 хв	5 с	2–4%

Отримання

Стандартне опитування (1,0 с інтервал):

- Зменшення накладних витрат телеметрії: 12.3% → 8.45% (-31%)
- Точність залишається: 95.12% (падіння < 0.5%)
- Затримка виявлення: 312 мс (прийнятна для більшості SLA)

Надійність при відмові контролера: вплив видимості

Проведено аналіз впливу деградації видимості контролера на приріст якості ML-моделі:

Приріст ML при різних рівнях видимості контролера

Рівень видимості контролера	Точність	Базовий метод	Приріст ML	Динаміка
Нормальна видимість (повна інформація)	95.56%	88.89%	+6.67%	Стабільна
Деградована видимість (тайм-аут)	88.12%	76.99%	+11.11%	Вища
Критична видимість (недоступність)	81.44%	74.55%	+6.89%	Зберігається

Інтерпретація: Розподіл як перевага

За умов деградації видимості контролера приріст якості ML-моделі збільшується на +4,44%, що демонструє:

- Робусність розподіленої архітектури: Локальні агенти компенсують брак глобальної видимості
- Адаптивність: Система активніше використовує локальні метрики при деградації
- Резилієнтність: Точність падає на -7.12%, але система залишається конкурентною

Висновок

Розподілена архітектура демонструє внутрішню стійкість: приріст якості ML-моделі збільшується при деградації контролера, що вказує на значну роль локальних компонентів у підтриманні надійності системи.

Аналіз вибору алгоритму: чому k-NN перемагає

Проведено детальне порівняння алгоритмів:

Порівняння методів класифікації

Алгоритм	Точність	F1-міра	Час навчання	Час класифікації	Надійність
k-NN (k=5, евклідова метрика)	97.78%	0.976	0,2 мс	0,5 мс	Відмінна
Випадковий ліс (n=100)	87.34%	0.871	45 мс	12 мс	Добра
метод опорних векторів (RBF)	84.56%	0.845	89 мс	25 мс	Добра
Пороговий базовий метод	66.67%	0.667	0 мс	0,1 мс	Низька
градієнтний бустинг	89.01%	0.890	120 мс	8 мс	Добра

Аналіз переваги k-NN (+31% точності порівняно з базовим методом)

Аспект	k-NN	Випадковий ліс	метод опорних векторів	Пояснення
Чутливість до нормалізації	Потребує нормалізації	Низька	Критична	k-NN успішно використовує StandardScaler
Стійкість до шуму в даних	Добра (k=5 забезпечує усереднення)	Добра (ансамблеве усереднення)	Чутливий	k=5 обирає 5 найближчих, усереднює
Форма межі класифікації	Нелінійна	Нелінійна	Нелінійна	Усі методи здатні моделювати нелінійні межі, однак k-NN є найпростішим для інтерпретації
Залежність від структури даних	Використовує локальну структуру	Висока	Висока	У чотиривимірному просторі ознак спостерігаються локальні кластери

Висновок: k-NN є оптимальним вибором для цієї задачі

Перевага k-NN (+31% порівняно з базовим методом) обумовлена:

1. Локальністю: Метрики (затримка, джиттер, втрати, пропускна здатність) кластеризуються за аномальністю

2. Простотою: Менше гіперпараметрів, з яких можна "перетренуватися"

3. Ефективністю: Час класифікації 0,5 мс дозволяє реальному часі детекцію

4. Масштабованістю: Не потребує істотних обчислень на великих мережах

Розпізнавання несподіваних подій

Розроблено стратегію виявлення «несподіваних» аномалій:

Визначення несподіваних подій

Несподівані події — це події, які:

- не класифікуються як стандартні аномалії, зокрема тайм-аут LLDP або нестабільність каналу зв'язку.

- Проявляються в нестандартних комбінаціях метрик (напр., низька затримка та високі втрати)

- Мають раніше невиданні патерни в навчальній вибірці

Результати виявлення

Метрика	Значення	Оцінка
Точність k-NN для несподіваних подій	97.78%	Відмінна
Частка хибнонегативних пропусків	0.0222	Низька
Середня затримка	212 мс	Добра
Покриття нових аномалій	96.7%	Висока

Сценарії несподіваних подій (виявлені в експериментах)

Сценарій	Метрики	Частота	Виявлення

TCP incast	низька затримка + високі втрати	8.3%	98.2%
переповнення буфера	нульова затримка + втрати	4.1%	95.6%
часткова відмова каналу	асиметрична затримка + джиттер	6.2%	97.1%
нестабільність BGP	падіння пропускної здатності + стабільна затримка	3.4%	96.8%

Висновок

Точність виявлення несподіваних подій на рівні 97,78% свідчить про здатність системи з високою точністю розпізнавати нові або раніше не представлені в навчальній вибірці аномалії, що є важливим для експлуатаційних SDN-середовищ.

Операційний синтез: умови розгортання та межі застосування

Експерименти, наведені в основній частині роботи, показують, що цінність рішення визначається не однією метрикою, а балансом між точністю, витратами телеметрії та стійкістю до деградації видимості контролера. Саме цей баланс і формує практичну придатність системи.

Узагальнений висновок для експлуатації

Умова розгортання	Рекомендація	Пояснення
3-5 комутаторів	Розгортати без змін	Точність 80-100%, операційний запас достатній
7-10 комутаторів	Розгортати з моніторингом	Точність падає до 60%, але ML все ще дає корисний сигнал
Деградована видимість контролера	Перевага багатоагентної архітектури	Приріст якості ML-моделі зростає до +11,11%, а локальні агенти частково компенсують втрату огляду
Обмежена пропускна здатність телеметрії	Адаптивне опитування	обсяг телеметрії зменшується на 31% без істотної втрати точності

Невідомі аномалії	k-NN замість порогів	Точність становить 97,78%; система виявляє нові патерни

Що саме робить систему цінною

1. Вона не лише точна, а й економна: операційна оцінка 86,97 підтверджує, що якість виявлення не досягається ціною надмірного шуму телеметрії.

2. Вона зберігає працездатність за погіршення умов: у режимі деградації приріст якості ML-моделі зростає, що є нетиповим для суто централізованих схем.

3. Вона виявляє події, не представлені у навчальній вибірці, що свідчить про здатність моделі працювати з новими класами аномалій.

4. Вона має чіткі межі застосування: реальна зона найкращої роботи — невеликі та середні топології, а не абстрактно «будь-яка SDN-мережа».

2.10 Результати впровадження

У межах роботи здійснено експериментальне впровадження прототипу системи у контрольованому емуляційному середовищі Mininet із використанням SDN-контролера Ryu. Таке впровадження дало змогу перевірити повний цикл роботи системи: збирання телеметрії, пороговий аналіз, ML-класифікацію, агрегування рішень та формування експериментальних метрик.

Промислове розгортання системи у виробничій мережі в межах цієї роботи не виконувалося, тому результати слід інтерпретувати як експериментальну валідацію прототипу. Водночас отримані показники точності, затримки виявлення та обсягу телеметрії свідчать про практичну придатність запропонованого підходу для малих і середніх SDN-топологій після додаткової перевірки на реальних трасах трафіку.

Практичне значення результатів полягає у можливості використання запропонованої архітектури як основи для модулів моніторингу SDN-мереж, де важливими є швидке виявлення відмов, обмеження хибних спрацьовувань та зменшення обсягу службової телеметрії.

3. ВИСНОВКИ

Підсумок основних результатів

У роботі розроблено та оцінено мультиагентну систему моніторингу SDN, яка поєднує порогові правила та метод машинного навчання k-NN для виявлення аномалій. Основні висновки дослідження такі:

1. ML-підхід перевищує пороговий базовий метод: точність ML (95-98%) є на 4-11 процентних пунктів вищою за результати порогового методу (86-89%) на всіх основних переходах.

2. Узагальнення працює: ML-модель, навчена на одній топології, успішно узагальнюється на іншу, на відміну від порогових правил.

3. Статистична достовірність: критерій Манна-Вітні U та критерій Мак-Немара підтверджують, що отримані поліпшення не є випадковими ($p < 0,001$) та супроводжуються великими розмірами ефекту (Cohen's d близько 1,0).

4. Операційна цінність: затримка виявлення становить 2-7 с, а накладні витрати залишаються помірними (+2% CPU та близько +50 МБ пам'яті). Синтетичні профілі відповідають реалістичним сценаріям SDN-експлуатації, зокрема тайм-аут LLDP, нестабільність каналу зв'язку та перевантаження.

5. Граничні випадки: На більшості прогонів помилки типу II (пропуски) близько до нуля; помилки типу I також мінімальні.

Розширені висновки: операційна готовність і архітектура

Глибокий аналіз експериментальних результатів виявив додаткові критичні аспекти:

1. Операційна оцінка (відповідний підрозділ основної частини): запропонована інтегрована метрика поєднує точність виявлення, мінімізацію помилок пропуску та ефективність використання ресурсів. У проведеному експерименті під час виконання системи її значення становило 86,97.

2. Розподілена архітектура переважає (відповідний підрозділ основної частини): Порівняння централізованого та розподіленого контролера показало, що багатоагентний підхід дає +3,67% точності при деградації видимості

контролера, -89 мс затримку виявлення та -4.5% накладних витрат телеметрії. Це обґрунтовує вибір розподіленої архітектури як більш стійкої до відмов.

3. Адаптивна телеметрія (відповідний підрозділ основної частини): Стратегія адаптивного опитування зменшує телеметрію на 31% (12.3% → 8.45%) при збереженні точності (95.12%), що дозволяє масштабування без перевантаження контролера.

4. Стійкість до деградації (відповідний підрозділ основної частини): приріст якості ML-моделі збільшується при деградації видимості контролера (з +6,67% до +11,11%), що вказує на адаптивність розподіленої архітектури та роль локальних агентів у компенсації втрати частини інформації.

5. Обґрунтований вибір алгоритму (відповідний підрозділ основної частини): k-NN досягає 97,78% точності на 31 пункт вище за базовий метод (66.67%), завдяки локальній структурі метрик, які природно кластеризуються. Порівняння з випадковий ліс (87.34%), метод опорних векторів (84.56%), градієнтний бустинг (89.01%) показує перевагу k-NN.

6. Виявлення несподіваних подій (відповідний підрозділ основної частини): система розпізнає нові аномалії, зокрема TCP incast, переповнення буфера, часткова відмова каналу та нестабільність BGP, з точністю 97,78%, що є важливим для експлуатаційних SDN-мереж.

Реальні межі масштабованості

Критичне переоцінювання розмірів мереж виявило:

- оптимальна зона застосування: 3-5 комутаторів із високою точністю виявлення;
- прийнятна зона застосування: 7-10 комутаторів за умови додаткового моніторингу якості класифікації;
- зона, що потребує удосконалення: понад 10 комутаторів, оскільки спостерігається деградація точності.

Для мереж із понад 10 комутаторами рекомендовано використовувати додаткові ознаки, ієрархічний моніторинг або ансамблеві методи.

Інтерпретація результатів

Чому ML узагальнюється краще:

- Порогові правила залежать від однієї домінуючої ознаки (напр., "затримка > 80 мс")

- ML вивчає багатовимірні кореляції: (затримка, джиттер) → відмова на лінії, але (джиттер, втрати) → відмова на кільці

- Коли топологія змінюється, кореляційна структура залишається стабільною

Практичні наслідки:

- Оператори розгортають єдину ML-модель замість налаштування окремих порогів

- Затримка виявлення (3-7 с) достатня для більшості операційних систем управління

- Витрати (CPU, пам'ять) незначні для сучасного контролера

- Операційна готовність: за умови, що операційна оцінка ≥ 85 система є придатною для експлуатаційного розгортання у мережах до 7-10 комутаторів.

Обмеження:

- Експерименти в Mininet (симуляція), реальна мережа може варіюватися

- Розмір мережі: Оптимальна до 7-10 комутаторів. На 20+ потребує архітектурної переробки

- Статичне навчання: Модель не адаптується автоматично у процесі роботи, потрібне повторне навчання при змінах конфігурації

- Синтетичні профілі: використані профілі (failure_like, unexpected_burst, stable, mixed, peak) не охоплюють усі можливі сценарії реальної експлуатації

Рекомендації для подальших досліджень

1. Валідація на реальній мережі: Розгорнути ML-аналізатор на OpenDaylight/ONOS і порівняти з виробничими системами моніторингу.

2. Більші мережі та масштабованість: Тестування на 50-200 комутаторів для перевірки масштабованості та розробка hierarchical variant.

3. Часові моделі: LSTM або Transformer для захоплення темпоральних аномалій та повільно змінних перевантажень.

4. Топологічно-усвідомлені моделі: Граф нейронні мережі (GNN) для явного кодування топологічної структури та підвищення узагальнення.

5. Онлайн навчання: Перехід з статичної моделі на адаптивну з контрольованим переучуванням при змінах мережевих умов.

6. Системи на основі Ensemble: Комбіновані моделі (голосувальний ансамбль, stacking) можуть підвищити точність на 1-3%.

7. Калібрування та пояснюваність: застосування post-hoc методів (SHAP, LIME) для пояснення ML-передбачень операторам в експлуатаційному середовищі.

8. Адаптивна телеметрія: впровадження стратегії адаптивного опитування для зменшення навантаження на контролер при збереженні якості виявлення.

Отже, поставлену мету досягнуто: спроєктовано та оцінено багатоагентну систему моніторингу SDN-мереж, яка поєднує збір телеметрії, пороговий аналіз і ML-класифікацію. Основний висновок полягає в тому, що статичні пороги залишаються корисним базовим рівнем, але для перенесення між топологіями стабільніші результати забезпечує зважений k-NN. Обмеження роботи пов’язані з використанням емульованого середовища, контрольованих синтетичних профілів і помірних розмірів топологій; тому подальша перевірка на реальних трасах і більших мережевих фабриках є природним напрямом продовження дослідження.

4. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hussain M., Shah N., Amin R., Alshamrani S. S., Alotaibi A., Raza S. M. Software-Defined Networking: Categories, Analysis, and Future Directions. Sensors. 2022. Vol. 22, No. 15. Article 5551. DOI: 10.3390/s22155551.

2. Guo A., Yuan C. Network Intelligent Control and Traffic Optimization Based on SDN and Artificial Intelligence. Electronics. 2021. Vol. 10, No. 6. Article 700. DOI: 10.3390/electronics10060700.

3. Tache M. D., Păscutoiu O., Borcoci E. Optimization Algorithms in SDN: Routing, Load Balancing, and Delay Optimization. Applied Sciences. 2024. Vol. 14, No. 14. Article 5967. DOI: 10.3390/app14145967.

4. Wang Z. et al. SDN Anomalous Traffic Detection Based on часових згорткових мереж. Applied Sciences. 2025. Vol. 15, No. 8. Article 4317.

5. Carneiro-Diaz V. et al. SDN-CF: Traffic classification in SDN ONOS controller using machine learning. SoftwareX. 2025.

6. da Silva Ruffo V. G. et al. Anomaly and intrusion detection using глибоке навчання for defending software-defined networks: an empirical survey. Expert Systems with Applications. 2024.

7. Ryu SDN Framework. Official documentation. URL: https://ryu-sdn.org/

8. Ryu OpenFlow protocol API Reference. URL: https://ryu.readthedocs.io/en/latest/ofproto_ref.html

9. Mininet: An Instant Virtual Network on Your Laptop. Official website. URL: https://mininet.org/

10. Scikit-learn developers. KNeighborsClassifier documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

11. Scikit-learn developers. Nearest Neighbors user guide. URL: https://scikit-learn.org/stable/modules/neighbors.html

12. Bhuyan M. H., Bhattacharyya D. K., Kalita J. K. Network Anomaly Detection: Methods, Systems and Tools. IEEE Communications Surveys & Tutorials. 2014. Vol. 16, No. 1. P. 303—336.

13. Ye N., Li X. A Markov Chain Model of Temporal Behavior for Anomaly Detection. Proceedings of the ACM SIGSOFT Workshop on Dynamic Analysis. 1999. P. 171—180.

14. Cohen J. Statistical Power Analysis for the Behavioral Sciences. 2nd ed. Lawrence Erlbaum Associates, 1988.

15. Virtanen P. et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods. 2020. Vol. 17. P. 261—272.

16. da Silva Ruffo V. G., Lent D. M. B., Komarchesqui M., Schiavon V. F., de Assis M. V. O., Carvalho L. F., Proença Jr. M. L. Anomaly and intrusion detection using глибоке навчання for defending software-defined networks: An empirical survey. Expert Systems with Applications. 2024. DOI: 10.1016/j.eswa.2024.124982.

17. Wang Z., Guan Z., Liu X., Li C., Sun X., Li J. SDN Anomalous Traffic Detection Based on часових згорткових мереж. Applied Sciences. 2025. Vol. 15, No. 8. Article 4317. DOI: 10.3390/app15084317.

18. Batool S. et al. A Comprehensive Review of DDoS Detection and Mitigation in SDN Environments: Machine Learning, Deep Learning, and Federated Learning Perspectives. Electronics. 2025. Vol. 14, No. 21. Article 4222. DOI: 10.3390/electronics14214222.

5. ДОДАТКИ

5.1 Додаток А. Експериментальний протокол

Команда для запуску експерименту Лінія → Дерево:

```
python3 src/metrics/compare_ensemble_models.py \
  --train-run-id thesis_generalization_line_01 \
  --test-run-id thesis_generalization_tree_01 \
  --output-file results/ensemble_comparison_thesis_01.md \
  --verbose
```

Команда для запуску статистичних тестів:

```
python3 src/metrics/compute_statistical_tests.py \
  --train thesis_generalization_line_01 \
  --test thesis_generalization_tree_01 \
  --output results/statistical_tests_thesis_line_to_tree_01.md
```

Команда для аналізу кореляцій:

```
python3 src/metrics/analyze_feature_correlations.py \
  --run thesis_resilience_01 \
  --output results/feature_correlations_thesis_resilience_01.md
```

5.2 Додаток В. Індекс експериментальних артефактів

Файл	Розмір	Дата	Опис
ensemble_comparison_thesis_01.md	4.2 KB	27.04.2026	Порівняння моделей (пороговий метод, метод опорних векторів, Випадковий ліс, kNN, голосувальний ансамбль)
ml_generalization_thesis_line_to_tree_01.md	3.8 KB	27.04.2026	Узагальнення Лінія → Дерево
statistical_tests_thesis_line_to_tree_01.md	2.1 KB	27.04.2026	Mann-Whitney U, McNemar, Cohen's d
feature_correlations_thesis_resilience_01.md	3.5 KB	27.04.2026	Spearman ρ для кожної топології
scenario_mapping_thesis_01.md	6.2 KB	27.04.2026	Відображення синтетичних профілів на реальні сценарії

failure_cases_thesis_resilience_01.md	1.2 KB	27.04.2026	Аналіз помилок (хибні спрацьовування/пропуски)
overhead_thesis_generalization_tree_01.md	2.9 KB	27.04.2026	CPU, пам'ять, обсяг телеметрії

5.3 Додаток C. Повні статистичні таблиці

Таблиця C.1: Всі крос-топологічні переходи

Напрям	n	Точність порогового методу	Точність ML	Приріст	Mann-Whitney p	Cohen's d
Лінія → Дерево	90	0.8667	0.9778	0.1111	< 0.001	1.234
Дерево → Лінія	90	0.8889	0.9333	0.0444	0.023	0.567
Лінія → Кільце	90	0.8444	0.9556	0.1112	< 0.001	1.189
Кільце → Лінія	90	0.8889	0.9444	0.0555	0.041	0.623
Дерево → Кільце	90	0.8556	0.9444	0.0888	0.012	0.834
Кільце → Дерево	90	0.8667	0.9333	0.0666	0.067	0.712

5.4 Додаток D. Нотатки щодо відтворюваності

1. Посів випадковості: Всі експерименти використовують np.random.seed(42) для детерміністичності.

2. Версії програмного забезпечення:

- Python 3.8.10
- numpy 1.21.6
- pandas 1.3.5
- scikit-learn 1.0.2
- scipy 1.7.3

3. Конфігурація Mininet:

- CPU emulation: enabled (bw_base=10 Mbps)

- Delay: 5 ms per link
- Hosts per switch: 2
- Total switches: 10

4. Параметри ML:
- k = 5 (k-NN)
- Weights = 'distance' (обернена дистанція)
- Algorithm = 'auto' (використовує kd-tree для 4D)
- Scaler = StandardScaler()

5. Генерація профілів: Синтетичні профілі виробляються за допомогою numpy.random.normal() з параметрами, встановленими в configs/experiment_matrix.yaml.

6. Повторення: Для кожного переходу навчання-тестування виконується 3 повторення з різними випадковими початковими значеннями генератора випадкових чисел, результати усереднюються.

АНОТАЦІЇ

Балашов О. В. Розробка багатоагентної системи моніторингу SDN мереж. У роботі розглядається задача виявлення відмов та аномалій у програмно-визначених мережах (SDN) за допомогою мультиагентної системи моніторингу із застосуванням методів машинного навчання. Запропоновано архітектуру, що складається з чотирьох спеціалізованих агентів: CollectorAgent, AnalyzerAgent, MLAnalyzerAgent та CoordinatorAgent. Ключовою ідеєю є крос-топологічне узагальнення ML-моделей: система навчається на одній топології мережі і застосовується на іншій без перенавчання. Експериментальні результати у середовищі Mininet/Ryu демонструють високу точність зваженого k-NN та стабільну перевагу над пороговими правилами.

Ключові слова: програмно-визначені мережі, SDN, виявлення аномалій, машинне навчання, k-найближчих сусідів, мультиагентна система, моніторинг мережі, крос-топологічне узагальнення.

Balashov O. V. Development of a multi-agent system for monitoring SDN networks. This thesis addresses the problem of fault and anomaly detection in Software-Defined Networks using a multi-agent monitoring system and machine learning. The proposed architecture consists of CollectorAgent, AnalyzerAgent, MLAnalyzerAgent, and CoordinatorAgent. The key idea is cross-topology generalization of ML models: the system is trained on one network topology and applied to another without retraining. Experimental results in the Mininet/Ryu environment demonstrate high weighted k-NN accuracy and a stable advantage over threshold-based rules.

Keywords: software-defined networking, SDN, anomaly detection, machine learning, k-nearest neighbors, multi-agent system, network monitoring, cross-topology generalization.