МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Кваліфікаційна наукова праця
на правах рукопису

БАНАР АНАТОЛІЙ ЮРІЙОВИЧ

УДК 004.8:621.39

ДИСЕРТАЦІЯ

МЕТОДИ І ТЕХНОЛОГІЇ ВИКОРИСТАННЯ АЛГОРИТМІВ ШТУЧНОГО ІНТЕЛЕКТУ В ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖАХ В УМОВАХ ОБМЕЖЕНИХ РЕСУРСІВ

172 – Телекомунікації та радіотехніка
17 – Електроніка та телекомунікації

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ А. Ю. Банар

Науковий керівник: Воробець Георгій Іванович, кандидат фізико-математичних наук, доцент

Чернівці – 2026

АНОТАЦІЯ

Банар А. Ю. Методи і технології використання алгоритмів штучного інтелекту в програмно-конфігурованих мережах в умовах обмежених ресурсів. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 172 - Телекомунікації та радіотехніка. – Чернівецький національний університет імені Юрія Федьковича, Чернівці, 2026.

Дисертаційна робота присвячена розв'язанню науково-практичного завдання покращення адаптивного керування доступом до обмежених ресурсів у програмно-конфігурованих мережах шляхом розроблення та експериментального обґрунтування методів і технологій застосування алгоритмів штучного інтелекту. Основну увагу зосереджено на інтеграції AI/ML-модулів у цикл керування SDN, коли деградація якості обслуговування спричинена не загальною нестачею пропускної здатності, а перевантаженням критичного прикладного ресурсу, зокрема API-сервісу, сервісу авторизації, шлюзу доступу до даних або іншого вузького місця сервісного рівня. Запропонований метод орієнтований на локалізацію керуючого впливу саме в точці дефіциту ресурсу, забезпечення рівномірного розподілу доступу між клієнтами та збереження стабільної роботи мережевої інфраструктури.

У **вступі** обґрунтовано актуальність теми дисертаційної роботи, сформульовано мету та завдання дослідження, визначено об'єкт, предмет і методи дослідження, наведено положення наукової новизни, практичне значення одержаних результатів, дані про апробацію та публікації за темою дисертації.

У **першому розділі** здійснено огляд літературних джерел, присвячених архітектурним принципам побудови програмно-конфігурованих мереж, типам локалізації SDN-контролерів, особливостям південних та північних інтерфейсів, проблемам управління ресурсами в сучасних SDN-мережах, а також підходам до інтеграції алгоритмів штучного інтелекту в SDN. Проаналізовано переваги SDN, що полягають у відокремленні площини керування від площини даних, централізації логіки прийняття рішень і програмованості мережевих політик.

Показано, що поряд із цим виникають нові виклики, пов’язані з обмеженнями масштабованості контролера, затримками керування, накладними витратами на телеметрію та ризиками перевантаження контрольної площини. Окрему увагу приділено порівнянню локальних, хмарних та ієрархічних edge-cloud архітектур SDN-контролерів, а також ролі REST/OpenFlow/MQTT у побудові інтегрованих AI-SDN середовищ.

Другий розділ присвячено методам побудови інтелектуальних SDN-мереж. У ньому розглянуто метрики оцінювання ефективності та справедливості розподілу ШІ-інтегрованих SDN-систем, зокрема класифікаційні метрики для ML-моделей та метрики розподілу ресурсів. Обґрунтовано вибір моделей машинного навчання для задач керування доступом до обмеженого ресурсу з урахуванням компромісу між якістю рішень і обчислювальними витратами. Також, проаналізовано платформи і реалізації SDN-контролерів та критерії інтеграції алгоритмів штучного інтелекту в їхню керуючу логіку. Результати розділу формують теоретичну та методичну основу для подальшого експериментального дослідження AI-керування в ресурсно-обмежених SDN-сценаріях.

У **третьому розділі** наведено моделювання та дослідження запропонованого Cloud-AI-SDN фреймворку в умовах обмежених ресурсів. Розроблено трирівневу архітектуру фреймворку, яка поєднує керованість мережі на рівні SDN-контролера, інтеграцію інтелектуальної аналітики в цикл прийняття рішень, а також засоби масштабованості, спостережуваності та аудиту керуючих впливів. Побудовано експериментальний стенд для відтворюваного дослідження доступу до обмеженого ресурсу в SDN із використанням AI-модуля, REST API та IoT-подій/телеметрії. Показано, що архітектура стенда реалізує взаємодію трьох основних типів інтерфейсів: північний (REST), південний (OpenFlow 1.3) та IoT-каналу подій (MQTT), формуючи наближену до практики модель гібридної AI-SDN-IoT екосистеми. У розділі також описано методику експериментального дослідження, етапи формування телеметричних даних, підготовку і навчання ШІ-модуля, а також процедури тестування та оцінювання результатів експерименту.

Четвертий розділ присвячено апаратним дослідженням та практичній реалізації запропонованого фреймворку. Проведено порівняльну оцінку моделей машинного навчання для ШІ-модуля SDN за показниками затримки прийняття рішень, справедливості розподілу обмежених ресурсів та загальної якості класифікації. На основі апаратної валідації в умовах обмежених ресурсів edge-пристрою показано, що модель Gradient Boosting забезпечує вищу якість прийняття рішень порівняно з Logistic Regression, зберігаючи при цьому прийнятні накладні витрати за CPU, RSS та енергоспоживанням. Зокрема, встановлено, що GB досягає accuracy ≈ 0.9988 та F1 ≈ 0.988, тоді як LR має accuracy ≈ 0.9768 та macro-F1 ≈ 0.843, при цьому різниця в енергоспоживанні між моделями є незначною. У розділі також реалізовано та описано прикладний веб-сервіс Cloud-AI-SDN фреймворку, який включає REST API, Web UI, засоби керування IoT-площиною експерименту, візуалізацію експериментальних даних і моніторинг системних метрик на базі Prometheus/Grafana. Це забезпечує відтворюваність сценаріїв, зручність керування експериментом і можливість аналізу стану системи під час роботи фреймворку. Отримані результати підтвердили ефективність запропонованих методів для покращення адаптивного керування доступом до обмежених ресурсів у програмно-конфігурованих мережах.

Наукова новизна результатів, отриманих у дисертаційній роботі, полягає у тому, що:

- ***Вперше*** запропоновано метод адаптивного керування доступом до обмеженого ресурсу прикладного рівня в програмно-конфігурованій мережі, який, на відміну від методів, що базуються на коригуванні загальних характеристик мережі (пропускної здатності, глобальних QoS-параметрів, пріоритизації трафіку в мережі загалом), передбачає локалізацію керуючого впливу безпосередньо в точці фактичного дефіциту ресурсу. Це дозволило підвищити середній показник рівномірності розподілу доступу до обмеженого ресурсу з 79,2% до 90,98% без необхідності коригування зазначених загальних характеристик мережі.

- ***Удосконалено*** механізм інтелектуального керування доступом до обмеженого ресурсу в SDN шляхом введення спеціалізованого AI-агента, який на відміну від підходів, що базуються на статичних правилах або враховують лише поточні мережеві параметри, формує керуюче рішення з урахуванням телеметричних ознак та параметрів, релевантних для конкретного типу обмеженого ресурсу, зокрема історії навантаження, часових характеристик звернень, частки клієнта в загальному навантаженні, факту попереднього блокування та кількості активних клієнтів. Це дало змогу локалізувати точку впливу в розподілені ресурси мережі, забезпечити адресний і адаптивний розподіл доступу до обмеженого ресурсу зі збереженням стабільного доступу до інших сервісів мережевої інфраструктури.

- ***Набув подальшого розвитку*** метод вибору моделі машинного навчання для інтелектуального керування в SDN в умовах обмежених ресурсів, який на відміну від підходів, що орієнтуються лише на загальну точність класифікації ML-моделі, передбачає оцінювання моделей за розширеними критеріями: класифікаційні показниками, стійкість до хибнопозитивних спрацювань, час прийняття рішення, завантаження CPU та RAM, енергоспоживання. Це дало змогу визначити Gradient Boosting (GB) як модель із максимальною точністю керування (Accuracy 0,9988 проти 0,9768 у Logistic Regression (LR); Precision 1,0 проти 0,54), а модель LR для сценаріїв, де важливішими є менші обчислювальні витрати, зокрема менший час прийняття рішення на 14,66%, нижче завантаження CPU на 13,4% та менше використання RAM на 6,33% порівняно з GB, за незначної різниці в енергоспоживанні між ними – 1,2%.

Удосконалено архітектуру інтелектуальної SDN-мережі для роботи в умовах обмежених ресурсів, яка, на відміну від рішень із окремим керуванням мережевими, прикладними та IoT-компонентами, забезпечує їх інтеграцію в єдиному контурі керування з включенням AI-підсистеми в цикл прийняття рішень. Особливістю архітектури є врахування типових обмежень централізованої SDN-архітектури, зокрема ризику компрометації контролера, стійкості до відновлення після збоїв, недостатньої прозорості змін політик та

складності інтеграції гетерогенних вузлів, шляхом поєднання засобів аудиту, автентифікації, резервного копіювання керуючих подій і змін політик, підтримки подієвої взаємодії та можливості адаптивного вибору й зміни ML-моделей залежно від ресурсних обмежень і вимог мережевого сценарію. Це дало змогу забезпечити централізоване та гнучке керування мережевою, прикладною та IoT-складовими системи, також забезпечити відтворюваність експериментальних сценаріїв і адаптацію режимів роботи до різних умов середовища розгортання та вимог до якості надання послуг.

Практичне значення одержаних результатів полягає в тому, що розроблені методи, програмно-апаратні компоненти та експериментально підтверджена архітектура можуть бути використані для підвищення ефективності функціонування SDN-орієнтованих інфраструктур у випадках наявності обмежених сервісних або прикладних ресурсів. Запропоновані рішення можуть бути впроваджені для адаптивного керування доступом до API, сервісів авторизації, систем зберігання даних, IoT-шлюзів та інших вузлів, критичних до перевантаження. Практична цінність результатів підтверджується апаратною апробацією запропонованої архітектури, реалізацією прикладного веб-сервісу керування і моніторингу, їх перевіркою в умовах реального ресурсного обмеження, а також впровадженням окремих результатів дисертаційної роботи в діяльність «Yukon Software Ltd» і ТОВ «ШАРПМАЙНДЗ ЮЕЙ», що підтверджено відповідними актами впровадження.

Ключові слова: зв’язок, інтернет речей, алгоритми, трафік, телекомунікаційна мережа, реалізація програмного забезпечення, машинне навчання, штучний інтелект (ШІ), виявлення, ефективність, оптимізація, програмно-конфігурована мережа, передача, методи.

ABSTRACT

Banar A. Methods and technologies for using artificial intelligence algorithms in software-defined networks under resource constraints – Qualifying scientific work in the form of a manuscript.

Dissertation for the degree of Doctor of Philosophy in specialty 172 - Telecommunications and Radio Engineering. - Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 2026.

The dissertation is devoted to addressing the scientific and practical task of improving adaptive access control to limited resources in software-defined networks by developing and experimentally substantiating methods and technologies for applying artificial intelligence algorithms. The main focus is placed on the integration of AI/ML modules into the SDN control loop when the degradation of service quality is caused not by a general shortage of bandwidth, but by the overload of a critical application resource, in particular, an API service, an authorization service, a data access gateway, or another bottleneck at the service level. The proposed approach aims to localize the control impact precisely at the point of resource shortage, ensure a fair distribution of access among clients, and maintain stable operation of the network infrastructure.

The **introduction** substantiates the relevance of the dissertation topic, formulates the aim and objectives of the research, defines the object, subject, and research methods, presents the scientific novelty, the practical significance of the obtained results, and provides information on approbation and publications related to the dissertation topic.

The **first section** reviews the literature on the architectural principles of software-defined networks, types of SDN controller placement, features of southbound and northbound interfaces, resource management problems in modern SDN networks, and approaches to integrating artificial intelligence algorithms into SDN. The advantages of SDN are analyzed, namely the separation of the control plane from the data plane, the centralization of decision-making logic, and the programmability of network policies. It is shown that, along with these advantages, new challenges arise, related to controller scalability limitations, control delays, telemetry overhead, and risks of overloading the control plane. Particular attention is paid to the comparison of local, cloud, and hierarchical edge-cloud architectures of SDN controllers, as well as to the role of REST/OpenFlow/MQTT in building integrated AI-SDN environments.

The **second section** is devoted to methods for building intelligent SDN networks. It considers metrics for evaluating the efficiency and fairness of AI-integrated SDN systems, in particular, classification metrics for ML models and resource allocation metrics. The choice of machine learning models for access control tasks under limited-resource constraints is substantiated by the trade-off between decision quality and computational cost. SDN controller platforms and implementations, as well as criteria for integrating artificial intelligence algorithms into their control logic, are also analyzed. The results of the chapter form the theoretical and methodological basis for further experimental research on AI-based control in resource-constrained SDN scenarios.

The **third section** presents the modeling and investigation of the proposed Cloud-AI-SDN framework under resource-constrained conditions. A three-level framework architecture has been developed, combining network controllability at the SDN controller level, the integration of intelligent analytics into the decision-making cycle, and mechanisms for scalability, observability, and control-impact auditing. An experimental testbed was built for reproducible investigation of access to a limited resource in SDN using an AI module, REST API, and IoT events/telemetry. It is shown that the testbed's architecture implements the interaction among three main interface types: Southbound (OpenFlow 1.3), Northbound (REST), and the IoT event channel (MQTT), thereby forming a practice-oriented model of a hybrid AI-SDN-IoT ecosystem. The chapter also describes the experimental methodology, the stages of telemetry data generation, the preparation and training of the AI module, and the procedures for testing and evaluating the experimental results.

The **fourth section** is devoted to hardware-based research and the practical implementation of the proposed framework. A comparative evaluation of machine learning models for the SDN AI module was conducted based on decision-making latency, fairness in limited-resource allocation, and overall classification quality. Based on hardware validation under constrained resources of an edge device, it is shown that the Gradient Boosting model achieves higher decision quality than Logistic Regression while maintaining acceptable overhead in CPU, RSS, and energy consumption. It was

established that GB achieves an accuracy ≈ of 0.9988 and an F1 ≈ of 0.988, whereas LR has an accuracy ≈ of 0.9768 and a macro-F1 ≈ of 0.843. The difference in energy consumption between the models is insignificant. The chapter also implements and describes an application-level web service within the Cloud-AI-SDN framework, which includes a REST API, a Web UI, tools for controlling the IoT plane of the experiment, experimental data visualization, and monitoring of system metrics using Prometheus/Grafana. This ensures the reproducibility of scenarios, the convenience of experiment management, and the ability to analyze the system state during framework operation. The obtained results confirmed the effectiveness of the proposed methods for improving adaptive access control to limited resources in software-defined networks.

The **scientific novelty** of the results obtained in the dissertation lies in the following:

- ***For the first time,*** a method for adaptive control of access to a limited application-layer resource in a software-defined network has been proposed. Unlike methods that adjust general network characteristics, such as bandwidth, global quality of service parameters, or overall traffic prioritization, the proposed method localizes the control action directly at the point of resource shortage. This made it possible to increase the average fairness of access distribution to the limited resource from 79.2% to 90.98% without adjusting the general network characteristics.

- The mechanism of intelligent access control to a limited resource in SDN ***has been improved*** by introducing a specialized AI agent which, unlike approaches based on static rules or considering only current network parameters, forms control decisions taking into account telemetry features and parameters relevant to a specific type of limited resource, in particular load history, temporal characteristics of requests, the client's share in the total load, the fact of previous blocking, and the number of active clients. This made it possible to localize the point of control action within distributed network resources, ensure targeted and adaptive allocation of access to the limited resource, and maintain stable access to other network infrastructure services.

- The method for selecting a machine learning model for intelligent control in SDN under resource constraints has been ***further developed***. Unlike approaches that focus only on the overall classification accuracy of an ML model, the proposed method evaluates models using extended criteria, including classification metrics, robustness to false-positive decisions, decision-making time, CPU and RAM utilization, and energy consumption. This made it possible to identify Gradient Boosting (GB) as the model with the highest control accuracy (Accuracy 0.9988 versus 0.9768 for Logistic Regression (LR); Precision 1.0 versus 0.54), while LR was identified as preferable for scenarios where lower computational cost is more important, particularly due to its 14.66% shorter decision-making time, 13.4% lower CPU utilization, and 6.33% lower RAM usage compared to GB, with only a minor difference in energy consumption between them of 1.2%..

- The architecture of an intelligent SDN network for operation under resource-constrained conditions ***has been improved***. Unlike solutions that separate control of network, application, and IoT components, the proposed architecture integrates them into a unified control loop, including an AI subsystem in the decision-making cycle. A distinctive feature of the architecture is that it accounts for typical limitations of centralized SDN architectures, in particular, the risk of controller compromise, resilience in post-failure recovery, insufficient transparency of policy changes, and the complexity of integrating heterogeneous nodes. This is achieved by combining audit mechanisms, authentication, backup of control events and policy changes, support for event-driven interaction, and the ability to adaptively select and replace ML models based on resource constraints and the requirements of a specific network scenario. This made it possible to ensure centralized, flexible management of the system's network, application, and IoT components, as well as the reproducibility of experimental scenarios and the adaptation of operating modes to different deployment environments and service quality requirements.

The ***practical significance*** of the results is that the developed methods, hardware-software components, and experimentally validated architecture can be used to improve the efficiency of SDN-oriented infrastructures when service or application

resources are limited. The proposed solutions can be implemented for adaptive access control across APIs, authorization services, data storage systems, IoT gateways, and other nodes critical to overload protection. The practical value of the results is confirmed by the hardware validation of the proposed architecture, the implementation of an application-level web service for control and monitoring, its verification under real resource-constrained conditions, as well as by the implementation of selected dissertation results in the activities of "Yukon Software Ltd" and "SHARPMINDS UA LLC", as confirmed by the relevant implementation acts.

Keywords: connection, internet of things (IoT), algorithm, traffic, telecommunication network, software system implementation, machine learning, artificial intelligence (AI), detection, efficiency, optimization, software-defined network (SDN), transmitting, methods.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації:

***Наукові праці у виданнях, включених до переліку наукових фахових видань України та проіндексованих у наукометричній базі даних Scopus*:**

1. **Banar A.,** Vorobets H. AI-based adaptive management of limited resources in SDN-IoT ecosystems. *Radioelectronic and Computer Systems*. 2025. Vol. 2025. № 4. P. 154-170. DOI: 10.32620/reks.2025.4.11 (Scopus, Q3, https://www.scimagojr.com/journalsearch.php?q=21101038702&tip=sid&clean=0). (*Внески авторів:* ***Банар А.Ю****. - огляд літератури, розробка методу управління SDN-мережею на базі інтелектуальних агентів, побудова топології мережі та архітектури фреймворку, налаштування інтелектуального управління, програмна реалізація алгоритмів керування, підготовка тексту роботи. Воробець Г.І. - формулювання концепції інтелектуального управління мережею, визначення вимог до системи управління, наукове редагування тексту.)*

Наукові праці у виданнях, що входять до переліку наукових фахових видань України:

2. **Банар А. Ю.,** Воробець Г. І. Перспективні напрями розвитку, удосконалення і застосувань мережі SDN на основі методів штучного інтелекту. *Вісник Хмельницького національного університету. Серія: Технічні науки*. 2025. Т. 355. № 4. С. 15-21. DOI: 10.31891/2307-5732-2025-355-1 *(Внески авторів: **Банар А.Ю.** - аналіз архітектурних особливостей SDN та викликів безпеки, систематизація методів штучного інтелекту для завдань балансування трафіку та забезпечення QoS/QoE, розробка класифікації загроз та методів їх протидії у програмно-конфігурованих мережах, підготовка тексту статті та візуалізацій. Воробець Г.І. - постановка завдання дослідження, визначення концепції інтеграції ШІ в SDN, аналіз огляду літератури та наукове редагування рукопису.)*

3. **Банар А. Ю.,** Воробець Г. І. Алгоритми штучного інтелекту для оптимізації функціонування SDN: сучасні підходи та перспективи. *Зв'язок*. 2025. № 4. С. 11-18. DOI: 10.31673/2412-9070.2025.041241 *(Внески авторів: **Банар А. Ю.** - дослідження та порівняльний аналіз ефективності алгоритмів машинного навчання для завдань SDN, розробка класифікації напрямів інтеграції ШІ (безпека, маршрутизація, енергоефективність), формування порівняльної таблиці ефективності методів, написання основного тексту. Воробець Г.І. - формування структури дослідження, перевірка результатів порівняльного аналізу, формулювання висновків щодо обмежень використання ШІ, науковий огляд рукопису перед друком.)*

4. **Банар А. Ю.,** Воробець Г. І. Хмарні SDN-контролери з підтримкою ШІ: архітектура, масштабованість та безпека (порівняльне дослідження). *Безпека інфокомунікаційних систем та Інтернету речей*. 2025. Т. 3. № 1. С. 01011:1-6. DOI: 10.31861/sisiot2025.1.01011 *(Внески авторів: **Банар А.Ю**. - дослідження архітектур локальних та хмарних SDN-контролерів (монолітна, мікросервісна, ієрархічна), аналіз проблем масштабованості та затримок у хмарному середовищі, аналіз сценаріїв інтеграції AI-модулів у хмарну інфраструктуру,*

підготовка тексту та графічних матеріалів. Воробець Г.І. - концептуалізація порівняльного дослідження, визначення критеріїв оцінки архітектур, наукове редагування статті.)

Наукові праці, які засвідчують апробацію матеріалів дисертації:

5. **Банар А. Ю**. Перспективні напрями оптимізації і захисту даних в SDN мережі з використанням штучного інтелекту. *Проблеми інформатики та комп'ютерної техніки (ПІКТ - 2023):* праці XII Міжнар. наук.-практ. конф. (м. Чернівці, 10 - 12 лист. 2023 р.). С. 91-92.

6. **Банар А. Ю.**, Воробець Г. І. Особливості і переваги застосування штучного інтелекту у хмарній реалізації SDN контролера. *Physical and technological problems of transmission, processing and storage of information in infocommunication systems (PREDT - 2025):* Proceedings of Xth International Scientific-Practical Conference. (Чернівці, 15-17 травня 2025). С. 189-190. *(Внески авторів: **Банар А.Ю.** - проведення порівняльного аналізу архітектур хмарних SDN-контролерів та оцінка їх масштабованості, дослідження аспектів інтеграції ШІ-модулів для керування трафіком, редагування тексту тез, розробка презентаційних матеріалів та очний виступ із доповіддю на конференції. Воробець Г.І. - постановка завдання дослідження, науковий огляд питань безпеки хмарних рішень, підготовка тексту тез, перевірка матеріалів доповіді.)*

7. **Банар А. Ю**., Воробець Г. І. Open-source фреймворк для моделювання та оптимізації SDN-мереж із використанням алгоритмів штучного інтелекту. *Проблеми інформатики та комп'ютерної техніки (ПІКТ - 2025):* праці XIV Міжнар. наук.-практ. конф. (м. Чернівці, 13 - 15 лист. 2025 р.). С. 22-24. *(Внески авторів: **Банар А.Ю.** - обґрунтування вибору open-source інструментів для реалізації SDN-фреймворку, програмна реалізація компонентів архітектури та проведення експериментального моделювання, написання тез, підготовка презентаційних матеріалів та очний виступ із пленарною доповіддю. Воробець*

Г.І. - огляд використаного відкритого програмного забезпечення для досліджень в SDN, формулювання завдань роботи, перевірка матеріалів доповіді.)

8. **Банар А. Ю**., Воробець Г. І. Побудова та управління SDN мереж з використанням методів штучного інтелекту. *Інтелектуальні інформаційні системи*: Матеріали всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів. (м. Миколаїв, 4-5 груд. 2025 р.). С. 10-12. *(Внески авторів: **Банар А.Ю.** - розробка архітектури та побудова експериментальної інтелектуальної SDN-платформи, реалізація алгоритмів для адаптивного розподілу ресурсів, підготовка тексту тез та виступ із доповіддю. Воробець Г.І. - формулювання концепції об'єднання SDN та ШІ для гетерогенних систем, визначення вимог до відтворюваності експериментів, перевірка матеріалів доповіді та підготовка до виступу.)*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 17
ВСТУП 18
РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ 26
1.1. Архітектурні принципи програмно-конфігурованих мереж 26
1.2. Типові архітектури локалізації SDN-контролерів 29
1.3. SDN інтерфейси: Northbound та Southbound 33
1.4. Проблеми управління ресурсами в сучасних SDN-мережах 39
1.5. Аналіз існуючих підходів до автоматизації управління ресурсами 41
1.6. Застосування та обмеження використання алгоритмів ШІ в SDN 43
1.7. Аналіз ML моделей та реалізацій для SDN 45
1.8. Застосування MQTT у програмно-конфігурованих мережах 46
Висновки до розділу 1 49
РОЗДІЛ 2. МЕТОДИ ПОБУДОВИ ІНТЕЛЕКТУАЛЬНИХ SDN-МЕРЕЖ ... 51
2.1. Метрики оцінювання ефективності та справедливості розподілу доступу в ШІ-інтегрованих SDN-системах 51
2.1.1. Класифікаційні метрики ефективності ML-алгоритмів 51
2.1.2. Метрики розподілу ресурсів в SDN 55
2.1.3. Взаємозв’язок класифікаційних та метрик справедливості 56
2.2. Обґрунтування вибору моделей машинного навчання для задач SDN 57
2.3. Платформи та реалізації SDN-контролерів 64
2.3.1. Класи SDN-контролерів 65
2.3.2. Аналіз критеріїв інтеграції алгоритмів ШІ в SDN 68
Висновки до розділу 2 72
РОЗДІЛ 3. МОДЕЛЮВАННЯ ТА ДОСЛІДЖЕННЯ CLOUD-AI-SDN ФРЕЙМВОРКУ В УМОВАХ ОБМЕЖЕНИХ РЕСУРСІВ 74
3.1. Архітектура запропонованого Cloud-AI-SDN фреймворку 74
3.2. Структура експериментального стенда та взаємодія його компонентів ... 78
3.3. Візуалізація та верифікація функціонування стенду 82
3.4. Методика та етапи експериментального дослідження інтелектуального керування доступом до обмеженого ресурсу 87
3.4.1. Формування експериментальних даних та попередня обробка мережевої телеметрії 88

3.4.2. Підготовка, навчання та розгортання ШІ-модуля інтелектуального керування доступом 90
3.4.3. Тестування та оцінювання результатів експерименту 93
Висновки до розділу 3 97
РОЗДІЛ 4. АПАРАТНІ ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНОГО ФРЕЙМВОРКУ 99
4.1. Порівняльна оцінка моделей машинного навчання для ШІ-модуля SDN . 99
4.1.1. Аналіз затримки прийняття рішень 100
4.1.2. Аналіз справедливості розподілу обмежених ресурсів обраних ML-моделей 101
4.1.3. Аналіз результатів обраних ML-моделей 102
4.2. Апаратна валідація інтелектуального SDN-модуля в умовах обмежених ресурсів 105
4.2.1. Результати апаратної валідації: накладні витрати (LR та GB) 107
4.2.2. Результати апаратної валідації: енергоспоживання системи 109
4.3. Розробка та інтеграція веб-сервісу прикладного рівня Cloud-AI-SDN фреймворку 111
4.3.1. Вимоги до прикладного веб-сервісу та постановка завдання 112
4.3.2. Архітектура веб-сервісу і принципи інтеграції з компонентами фреймворку 113
4.3.3. REST API, сценарії керування та приклади запитів 115
4.3.4. Web UI та візуалізація експериментальних даних 118
4.4. Апаратна апробація запропонованого Cloud-AI-SDN фреймворку 125
Висновки до розділу 4 133
ВИСНОВКИ 135
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 137
ДОДАТКИ 152

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI – (Artificial Intelligence) - штучний інтелект;

API – (Application Programming Interface) - прикладний програмний інтерфейс;

CSV – (Comma-Separated Values) - формат даних, значення розділені комами;

DRL – (Deep Reinforcement Learning) - глибоке навчання з підкріпленням;

EDGE – (Edge Computing / Edge Devices) - периферійні (крайові) обчислення чи пристрої;

HTTP – (HyperText Transfer Protocol) - протокол передавання гіпертексту;

IDS – (Intrusion Detection System) - система виявлення вторгнень;

IoT – (Internet of Things) - Інтернет речей;

JFI – (Jain's Fairness Index) - індекс справедливості Джейна;

JSON – (JavaScript Object Notation) - формат обміну даними;

LAN – (Local Area Network) - локальна мережа;

LSTM – (Long Short-Term Memory) - рекурентна нейромережа типу LSTM;

ML – (Machine Learning) - машинне навчання;

MQTT – (Message Queuing Telemetry Transport) - легковаговий подійний протокол;

NBI – (Northbound Interface) - північний інтерфейс SDN;

NN – (Neural Network) - нейронна мережа;

QoE – (Quality of Experience) - якість сприйняття послуг;

QoS – (Quality of Service) - якість обслуговування;

RPS – (Requests Per Second) - запитів за секунду;

RSS – (Resident Set Size) - резидентний обсяг пам'яті процесу;

SBI – (Southbound Interface) - південний інтерфейс SDN;

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

WAN – (Wide Area Network) - глобальна мережа.

ВСТУП

Актуальність та обґрунтування теми дисертаційного дослідження. Сучасні телекомунікаційні мережі функціонують в умовах стрімкого зростання інтенсивності трафіку, урізноманітнення сервісів і прикладних програм (хмарні застосунки, IoT/edge, мультимедійні платформи), а також, відповідно, ускладнення політик безпеки та вимог до якості обслуговування. Ці фактори обумовлюють підвищену потребу в гнучкому та адаптивному управлінні мережевими ресурсами, оскільки традиційні підходи до конфігурації мережі (статичні правила, ручне адміністрування, ізольовані механізми QoS) недостатньо ефективні в динамічних середовищах з нерівномірним навантаженням і неоднорідними потоками даних. Концепція програмно-конфігурованих мереж (SDN) створює технологічне підґрунтя для автоматизації управління мережею за рахунок відокремлення площини керування від площини даних, централізованої логіки прийняття рішень та програмованості політик управління мережею. Водночас, практичне впровадження SDN у реальних середовищах висуває нові виклики – контролер повинен обробляти значні обсяги телеметрії, оперативно інсталювати правила, враховувати поведінку і підтримувати працездатність при обмежених обчислювальних і мережевих ресурсах.

Особливу актуальність набуває ситуація, коли деградація якості сервісу спричинена не стільки загальною пропускною здатністю мережі, скільки наявністю критичного вузького місця на рівні сервісу або прикладного ресурсу (наприклад, сервіс авторизації, API з лімітом запитів, шлюз доступу до сховища даних, компонент хмарної інфраструктури з лімітами доступу, тощо). У таких випадках класичні методи управління мережею (наприклад коригування пропускної здатності) можуть погіршувати показники навіть для трафіку, який не спричиняє перевантаження критичного ресурсу, що негативно впливає на загальний показник якості обслуговування в мережі (QoS).

Отже, актуальною є науково-прикладна задача розроблення методів і технологій інтеграції алгоритмів штучного інтелекту в SDN для адаптивного

керування доступом до обмежених ресурсів з урахуванням вимог щодо їх рівномірного розподілу між користувачами, стабільності сервісу та придатності до роботи в ресурсо-обмежених середовищах.

Метою дисертаційної роботи є покращення адаптивного керування доступом до обмежених ресурсів із забезпеченням рівномірного розподілу доступу та збереженням стабільної роботи мережевої інфраструктури в умовах обмежених ресурсів телекомунікаційної мережі за рахунок розроблення та експериментального обґрунтування методів і технологій застосування алгоритмів штучного інтелекту в програмно-конфігурованих мережах.

Для досягнення поставленої мети в роботі визначено такі **завдання**:

1. Провести огляд наукових джерел щодо сучасного стану дослідження питань, які стосуються програмно-конфігурованих мереж, моделей локалізації SDN-контролерів та підходів до управління ресурсами в SDN з акцентом на ресурсні обмеження контролера, каналів керування і прикладних сервісів.

2. Проаналізувати існуючі технології автоматизації управління ресурсами (статичні політики, механізми на основі правил, традиційні QoS-підходи) та обґрунтувати доцільність використання методів штучного інтелекту для прийняття керуючих рішень у динамічних сценаріях роботи мережі.

3. Сформувати постановку задачі керування доступом до обмеженого прикладного ресурсу в SDN-середовищі, визначити релевантні вхідні ознаки (телеметрія навантаження, інтенсивність запитів клієнтів, стан ресурсу, історія запитів), критерії якості рішень і можливі обмеження щодо покращення функціоналу мережі на рівні апаратно – програмних ресурсів (час прийняття рішення моделлю машинного навчання, обчислювальна потужність мережевого обладнання, стабільність мережі, енергоспоживання).

4. Розробити метод інтелектуального керування доступом до обмеженого ресурсу на основі алгоритмів машинного навчання, який забезпечує:

- адаптивну зміну політики доступу в часі залежно від поточного та прогнозованого навантаження;

- локалізацію впливу керування на рівні критичного ресурсу (вузького місця) без обмежень доступу до інших сервісів;

5. Запропонувати архітектуру SDN мережі та побудувати експериментальний стенд для дослідження інтеграції AI-модуля з SDN-контролером, реалізувати механізми збору телеметрії та протоколи взаємодії компонентів екосистеми AI/SDN.

6. Провести експериментальну перевірку ефективності запропонованих рішень, зокрема оцінити якість роботи моделей керування та показники рівномірного розподілу доступу, проаналізувати затримку прийняття рішень AI-модулем і накладні витрати (на рівні обчислювальних ресурсів мережевого обладнання, енергоспоживання), виконати апаратну реалізацію запропонованого підходу в ресурсо-обмеженому середовищі, а також реалізувати прикладний веб-сервіс (API та Web UI) для керування експериментальним середовищем і візуалізації даних із забезпеченням відтворюваного запуску сценаріїв, збору результатів та аналізу показників.

Об’єктом дослідження є процеси керування в умовах обмеженого доступу до обчислювальних та сервісних ресурсів інфраструктури у програмно-конфігурованих мережах.

Предметом дослідження є методи, моделі та технології застосування алгоритмів штучного інтелекту в SDN для адаптивного регулювання доступу клієнтів до обмежених ресурсів прикладного рівня, а також критерії оцінювання ефективності та якості такого регулювання.

Методи досліджень. У роботі використано методи імітаційного моделювання та експериментального дослідження для перевірки працездатності запропонованих рішень, методи аналізу для порівняння існуючих підходів до AI-інтеграції в SDN, методи теорії ймовірностей і математичної статистики для аналізу телеметрії та експериментальних результатів, методи машинного навчання для побудови моделей прийняття рішень, методи програмної інженерії та об’єктно-орієнтованого програмування для реалізації фреймворку, API та

інструментів моніторингу, методи інструментального вимірювання для оцінки накладних витрат і можливості апаратної реалізації.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконувалася відповідно до тематики наукових досліджень кафедри радіотехніки та інформаційної безпеки в межах науково-дослідної роботи «Методи формування сигнальних конструкцій та інформаційні процеси програмно-апаратної взаємодії широкосмугових телекомунікаційних систем та Інтернету речей» (державний реєстраційний номер 121U112870, 2021-2025 рр.).

Наукова новизна отриманих результатів. При виконанні дисертаційної роботи отримано такі пункти наукової новизни:

- ***Вперше*** запропоновано метод адаптивного керування доступом до обмеженого ресурсу прикладного рівня в програмно-конфігурованій мережі, який, на відміну від методів, що базуються на коригуванні загальних характеристик мережі (пропускної здатності, глобальних QoS-параметрів, пріоритизації трафіку в мережі загалом), передбачає локалізацію керуючого впливу безпосередньо в точці фактичного дефіциту ресурсу. Це дозволило підвищити середній показник рівномірності розподілу доступу до обмеженого ресурсу з 79,2% до 90,98% без необхідності коригування зазначених загальних характеристик мережі.
- ***Удосконалено*** механізм інтелектуального керування доступом до обмеженого ресурсу в SDN шляхом введення спеціалізованого AI-агента, який на відміну від підходів, що базуються на статичних правилах або враховують лише поточні мережеві параметри, формує керуюче рішення з урахуванням телеметричних ознак та параметрів, релевантних для конкретного типу обмеженого ресурсу, зокрема історії навантаження, часових характеристик звернень, частки клієнта в загальному навантаженні, факту попереднього блокування та кількості активних клієнтів. Це дало змогу локалізувати точку впливу в розподілені ресурси мережі, забезпечити адресний і адаптивний

розподіл доступу до обмеженого ресурсу зі збереженням стабільного доступу до інших сервісів мережевої інфраструктури.

- ***Набув подальшого розвитку*** метод вибору моделі машинного навчання для інтелектуального керування в SDN в умовах обмежених ресурсів, який на відміну від підходів, що орієнтуються лише на загальну точність класифікації ML-моделі, передбачає оцінювання моделей за розширеними критеріями: класифікаційні показниками, стійкість до хибнопозитивних спрацювань, час прийняття рішення, завантаження CPU та RAM, енергоспоживання. Це дало змогу визначити Gradient Boosting (GB) як модель із максимальною точністю керування (Accuracy 0,9988 проти 0,9768 у Logistic Regression (LR); Precision 1,0 проти 0,54), а модель LR для сценаріїв, де важливішими є менші обчислювальні витрати, зокрема менший час прийняття рішення на 14,66%, нижче завантаження CPU на 13,4% та менше використання RAM на 6,33% порівняно з GB, за незначної різниці в енергоспоживанні між ними - 1,2%.
- ***Удосконалено*** архітектуру інтелектуальної SDN-мережі для роботи в умовах обмежених ресурсів, яка, на відміну від рішень із окремим керуванням мережевими, прикладними та IoT-компонентами, забезпечує їх інтеграцію в єдиному контурі керування з включенням AI-підсистеми в цикл прийняття рішень. Особливістю архітектури є врахування типових обмежень централізованої SDN-архітектури, зокрема ризику компрометації контролера, стійкості до відновлення після збоїв, недостатньої прозорості змін політик та складності інтеграції гетерогенних вузлів, шляхом поєднання засобів аудиту, автентифікації, резервного копіювання керуючих подій і змін політик, підтримки подієвої взаємодії та можливості адаптивного вибору й зміни ML-моделей залежно від ресурсних обмежень і вимог мережевого сценарію. Це дало змогу забезпечити централізоване та гнучке керування мережевою, прикладною та IoT-складовими системи, а також

забезпечити відтворюваність експериментальних сценаріїв і адаптацію режимів роботи до різних умов середовища розгортання та вимог до якості надання послуг.

Практичне значення отриманих результатів полягає в тому, що запропоновані методи і реалізовані технологічні компоненти можуть бути використані для побудови та впровадження механізмів адаптивного керування доступом у SDN-орієнтованих інфраструктурах, де інтегровано дефіцитний ресурс прикладного або сервісного рівня.

Запропонований підхід є придатним для регулювання доступу до сховищ даних, API-сервісів, сервісів авторизації, компонентів хмарних платформ та інших вузлів, де перевантаження призводить до погіршення якості обслуговування, а традиційне «глобальне» обмеження мережевих параметрів погіршує показники навіть для некритичних потоків. Система дозволяє ідентифікувати моделі поведінки клієнтів, запобігати витісненню легітимних транзакцій активними споживачами ресурсу та підтримувати стабільність сервісу шляхом адресного квотування доступу. Результати роботи підтверджені реалізацією експериментального стенда та фреймворку, що включає SDN-контролер (у приватній хмарній конфігурації), інструменти моніторингу, уніфіковані програмні засоби керування (скрипти/модулі) та прикладний API/Web UI для керування експериментами і візуалізації даних.

Результати дисертаційної роботи використано при формуванні навчальних дисциплін кафедри комп’ютерних систем та мереж, та кафедри радіотехніки та інформаційної безпеки Чернівецького національного університету імені Юрія Федьковича. Основні результати дисертаційної роботи використано і впроваджено з метою підвищення ефективності керування доступом до обмежених сервісних ресурсів у програмно-конфігурованих мережах, забезпечення рівномірного розподілу доступу між клієнтами та зменшення впливу пікових навантажень на критичні вузли інфраструктури у «Yukon Software Ltd» і ТОВ «ШАРПМАЙНДЗ ЮЕЙ», що підтверджено актами впровадження.

Достовірність отриманих результатів підтверджується високим рівнем відповідності між теоретичними розрахунками, імітаційним моделюванням, експериментальними даними та апаратною апробацією. Така узгодженість свідчить про правильність обраних підходів і підтверджує достовірність проведених досліджень.

Особистий внесок здобувача. Усі результати, представлені в дисертаційній роботі, здобувач отримав самостійно шляхом проведення теоретичних досліджень, комп'ютерного моделювання, експериментальних вимірювань, апаратної реалізації та аналізу отриманих даних. У роботах, опублікованих у співавторстві, особистий внесок автора конкретизовано у списку публікацій за темою дисертації.

Апробація результатів дисертації. Основні результати отримані в дисертаційній роботі були предметом обговорень на:

- Науковий семінар кафедри радіотехніки та інформаційної безпеки Чернівецького національного університету імені Юрія Федьковича (м. Чернівці, 13 березня 2026), форма участі - очна.
- Міжнародний науково-технічний семінар «Критичні комп'ютерні технології та системи (КриКТехС)» (м. Харків, 29 січня 2026), форма участі - дистанційна.
- П'ятнадцята міжнародна конференція «Dependable Systems, Services and Technologies (DESSERT'2025)» (Греція, Афіни, 19-21 грудня 2025), форма участі - дистанційна.
- Всеукраїнська науково-практична конференція молодих вчених, аспірантів і студентів «Інтелектуальні інформаційні системи» (м. Миколаїв, 4-5 грудня 2025), форма участі - дистанційна з публікацією тез.
- Чотирнадцята міжнародна конференція «Проблеми інформатики та комп'ютерної техніки (ПІКТ-2025)» (м. Чернівці, 13-15 листопада 2025), форма участі - очна з публікацією тез.

- Десята міжнародна науково-практична конференція «Physical and technological problems of transmission, processing and storage of information in infocommunication systems (PREDT - 2025)» (м. Чернівці, 15 травня 2025), форма участі - очна з публікацією тез.
- Дванадцята міжнародна конференція «Проблеми інформатики та комп'ютерної техніки (ПІКТ-2023)» (м. Чернівці, 10 - 12 листопада 2023), форма участі - очна з публікацією тез.

Публікації. Основні результати дисертаційного дослідження відображено у 9 наукових працях, з яких 1 у виданнях, включених до переліку наукових фахових видань України та проіндексованих у наукометричній базі даних Scopus (Q3), 3 у виданнях, включених до переліку наукових фахових видань України, 4 праці, які засвідчують апробацію матеріалів дисертації.

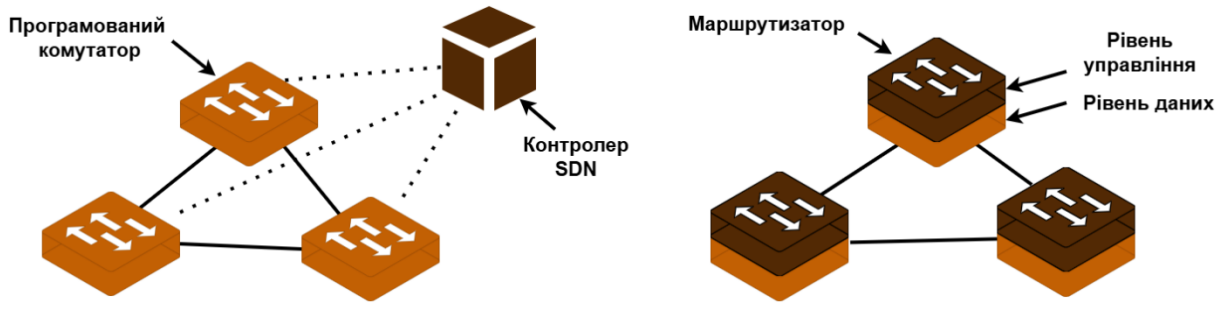
Структура і обсяг дисертації. Дисертаційна робота відповідає встановленим нормативним вимогам. Вона включає вступ, чотири основні розділи, підсумкові висновки, список використаних джерел та додатки. Загальний обсяг роботи становить 159 сторінок, містить 57 рисунків, бібліографічний список із 118 джерел та додатки.

РОЗДІЛ 1. ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1.1. Архітектурні принципи програмно-конфігурованих мереж

Сучасні телекомунікаційні мережі характеризуються стрімким зростанням кількості сервісів, користувачів та підключених пристроїв, зумовлюючи підвищені вимоги до гнучкості, масштабованості та ефективності управління мережевими ресурсами. Традиційні мережеві архітектури побудовані на основі децентралізованої моделі керування, де функції маршрутизації, комутації та прийняття рішень жорстко пов’язані з мережевим обладнанням. Такий підхід ускладнює оперативну зміну політик управління, адаптацію до динамічних умов та автоматизацію процесів конфігурування мережі.

Програмно-конфігуровані мережі (Software-Defined Networks, SDN) - це концепція побудови мереж, що передбачає відокремлення площини управління від площини передачі даних, завдяки чому забезпечується централізоване управління мережевими ресурсами (рис. 1.1). У традиційних мережах, пристрої (комутатори, маршрутизатори) самостійно приймають рішення щодо маршрутизації чи пересилання даних. На противагу цьому, в архітектурі SDN використовується централізований контролер, який через програмні інтерфейси здійснює управління всіма мережевими вузлами [1].



а) б)

Рис. 1.1. Структура мережі: а) SDN, б) класична

Особливість SDN полягає у відокремленні площини пересилання від площини керування, причому як концепція цей підхід уперше був застосований у телефонних мережах загального користування, задовго до його перенесення в телекомунікаційні мережі [2]. Дослідження, зроблені на факультеті комп'ютерної інженерії Стенфордського університету, сформували прецедент [3] використання відкритого програмного забезпечення в проєктах із розділеними площинами керування та даних. Наслідком цих робіт стала поява протоколу OpenFlow, який завдяки спеціалізованому дизайну комутаторів надавав мережевому контролеру можливість задавати шлях руху пакетів у мережі [4].

Архітектуру SDN умовно поділяють на три функціональні рівні: прикладний, управління та інфраструктурний, які взаємодіють між собою за допомогою стандартизованих інтерфейсів. Ця концепція забезпечує централізоване керування мережевими ресурсами та їх програмну конфігурацію засобами ПЗ [3, 5]. Типова схема мережі SDN зображена на рис. 1.2.

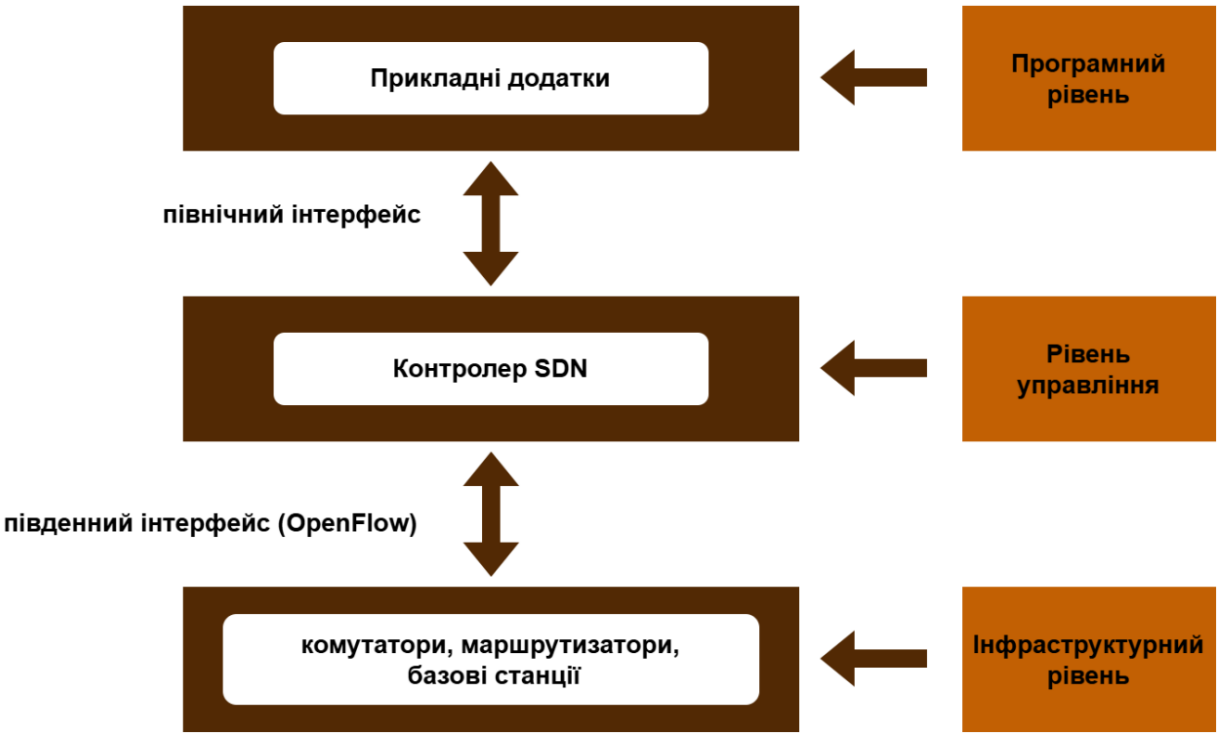


Рис. 1.2. Структурна схема SDN мережі

Прикладний рівень розташований на верхньому рівні SDN-архітектури, де реалізуються мережеві функції й сервіси, зокрема маршрутизація, інженерія трафіку, балансування навантаження, моніторинг та механізми безпеки. Взаємодія з контролерами для реалізації цих функцій відбувається через північний інтерфейс, що дозволяє здійснювати управління якістю обслуговування (QoS) та політиками керування трафіком уникаючи втручання у мережеве обладнання.

Рівень управління центральним елементом архітектури, на якому функціонують SDN-контролер і мережева операційна система. На цьому рівні фізично розподілена топологія мережі маскується, натомість для застосунків верхнього рівня надається абстрактне подання елементів пересилання у вигляді графа мережі [6]. Конфігурацію комутаторів забезпечує SDN-контролер, який через південний інтерфейс інсталює правила пересилання/маршрутизації. Основним стандартом для цієї взаємодії є протокол OpenFlow [4]. Контролер SDN володіє цілісною картиною про поточний стан мережі, що дозволяє здійснювати ефективний розподіл ресурсів, на відміну від традиційних мереж, де інформація про стан мережі є фрагментованою поміж окремих вузлів.

Інфраструктурний рівень охоплює сукупність взаємопов’язаних елементів пересилання (SDN-комутаторів), які виконують передавання/маршрутизацію пакетів і фактично формують площину даних. На відміну від традиційної архітектури комп’ютерних мереж, інфраструктурні комутатори тут реалізують переважно функцію пересилання трафіку відповідно до правил, що задаються рівнем керування. Вузли не приймають самостійних рішень щодо маршрутизації.

Така архітектурна модель забезпечує низку практичних переваг. SDN надає можливість динамічно коригувати параметри мережі через програмні інтерфейси, що дозволяє оперативно адаптувати її до змінних вимог у реальному часі та оптимізувати доступ до ресурсів із обмеженою пропускною здатністю. Крім того, SDN істотно полегшує масштабування мережі - додавання нових

комутаторів не потребує індивідуального переналаштування кожного пристрою, оскільки налаштування здійснюється централізовано.

Водночас, враховуючи централізовану концепцію, архітектура стикається з викликами. Головним недоліком цієї структури можна виділити залежність ефективності кожного шару від попереднього. Централізоване управління призводить до зростання складності обробки великих обсягів мережевої інформації, а компрометація контролера може надати зловмисникам можливість захопити контроль над усією мережею [7].

Таким чином, архітектура SDN трансформує підхід до побудови телекомунікаційних систем, пропонуючи програмованість та абстракцію від апаратного забезпечення, що відкриває широкі можливості для впровадження інтелектуальних методів управління.

1.2. Типові архітектури локалізації SDN-контролерів

У сучасних програмно-конфігурованих мережах SDN-контролер виступає центральним елементом управління, забезпечуючи логічно централізований контроль над мережею та абстрагування фізичної інфраструктури. У класичних реалізаціях контролер розгортається локально та, як правило, має монолітну архітектуру, у межах якої всі функції керування, збору телеметрії та прийняття рішень реалізовані в єдиному програмному компоненті [8]. Відомими прикладами таких реалізацій є рішення з відкритим вихідним кодом, такі як OpenDaylight(ODL), ONOS та Ryu. Ці контролери використовують південні інтерфейси (наприклад, OpenFlow) для зв’язку з мережевими пристроями та північні інтерфейси, що дозволяють керувати політиками програмованих комутаторів [9].

Масштабування таких контролерів може бути реалізоване вертикально, шляхом збільшення апаратних ресурсів одного сервера, або горизонтально, шляхом створення кластерів, що складаються з декількох екземплярів контролера, які синхронізують стани мережі між собою. Проте горизонтальна кластеризація вносить додаткову складність у систему [10], вимагає механізмів

синхронізації станів і зазвичай ефективно підтримує лише обмежену кількість вузлів. Як наслідок, традиційна архітектура SDN-контролера стикається з обмеженнями масштабованості в умовах значних навантажень, де обмежені обчислювальні ресурси можуть призвести до затримок у обробці запитів та оновленні таблиць потоків.

З розвитком хмарних технологій і зростання масштабу мереж, з'явилася тенденція до перенесення SDN-контролерів у хмарну інфраструктуру (приватну, публічну або гібридну). Хмарна реалізація SDN-контролера дозволяє використовувати еластичні обчислювальні ресурси, спрощує централізоване управління географічно розподіленими сегментами мережі та забезпечує зручну інтеграцію з зовнішніми сервісами моніторингу, телеметрії та автоматизації DevOps-процесів [11].

Аналіз сучасних досліджень дозволяє виділити декілька основних архітектурних підходів до реалізації такого контролера:

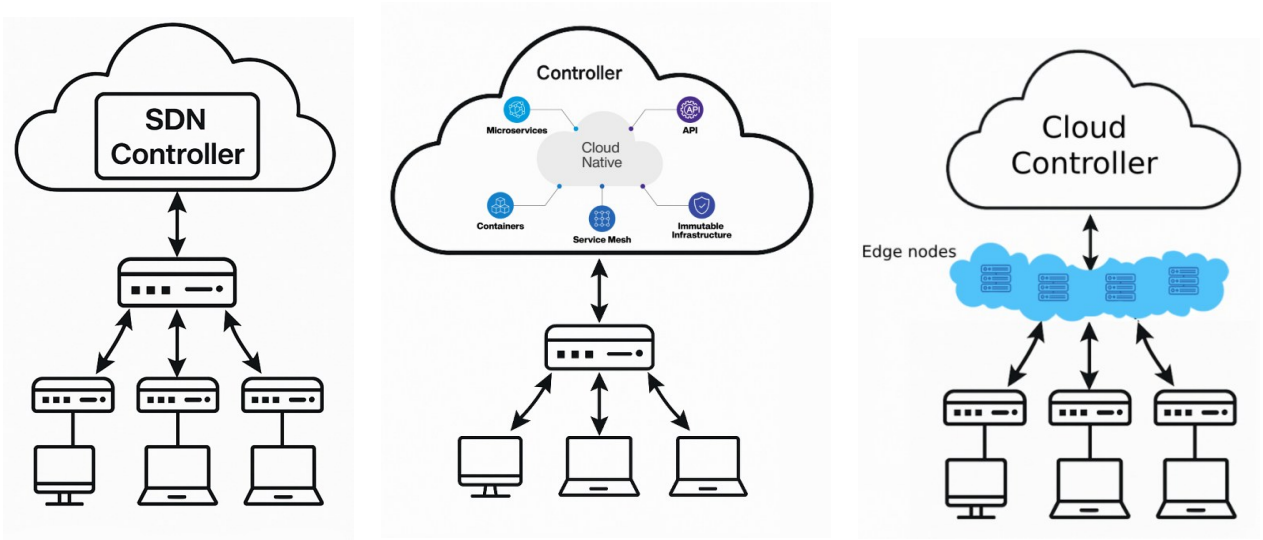
- Хмарна монолітна архітектура передбачає розгортання того ж програмного забезпечення SDN-контролера на хмарному сервері, що й у локальному сценарії. Це дозволяє централізовано керувати географічно розподіленими мережами (наприклад, корпоративними мережами) без необхідності утримувати власні центри обробки даних. Комерційні приклади цього підходу включають платформи Cisco Meraki та TP-Link Omada, які виконують функції SDN-контролерів для клієнтської інфраструктури.

- Мікросервісна архітектура. Рішення, такі як ETSI TeraFlowSDN, перепроектовують SDN-контролер спеціально для хмарного розгортання, декомпозуючи його на множину мікросервісів. Кожен мікросервіс відповідає за окремий функціональний аспект, такий як управління топологією, безпека, маршрутизація або аналітика трафіку. Ці мікросервіси оркеструються за допомогою Kubernetes [12], що дозволяє автоматичне масштабування, безшовні оновлення без порушення роботи мережі.

- Ієрархічна архітектура -поєднує переваги локальних та хмарних SDN-контролерів. На верхньому рівні цієї ієрархії знаходиться глобальний хмарний

контролер, який має повне уявлення про всю мережу та керує високорівневими операціями, такими як політики безпеки та міждоменна маршрутизація [13]. На нижньому рівні розташовані локальні контролери, зазвичай реалізовані як компактні пристрої або віртуальні машини на краю мережі. Кожен локальний контролер керує певним сегментом мережі (наприклад, локальною мережею банку або підмережею IoT в межах підприємства) і обробляє оперативні завдання в реальному часі. Локальні контролери взаємодіють з глобальним, передаючи агреговані дані та отримуючи глобальні інструкції.

Узагальнену структуру типових архітектур SDN-контролерів відображено на рис. 1.3.



а) б) в)

Рис. 1.3. Типові архітектури SDN-контролерів: а) хмарна монолітна, б) мікросервісна, в) ієрархічна

Важливою характеристикою хмарних SDN-контролерів є можливість природньої інтеграції алгоритмів ШІ в площину управління [14]. Використання обчислювальних ресурсів хмари дозволяє виконувати роботу з моделями

машинного навчання та глибокого навчання без впливу на базові функції контролера [15].

ШІ-модулі можуть працювати паралельно з основними сервісами керування, виконуючи задачі прогнозування трафіку, виявлення аномалій, адаптивного налаштування маршрутів та автоматизованого управління якістю обслуговування [16, 17].

Разом з тим, перенесення SDN-контролера у хмарну інфраструктуру породжує низку викликів, пов’язаних із затримками керування та безпекою обміну даними між контролером і мережевими пристроями. Передача керуючих повідомлень через зовнішні канали вимагає застосування захищених протоколів, шифрування та жорстких механізмів автентифікації [18]. Тому у багатьох архітектурах застосовується гібридний підхід, який поєднує локальні edge-контролери для критичних сегментів та хмарний контролер для глобальної оптимізації.

За умови коректного проєктування, хмарні SDN-контролери здатні забезпечити рівень продуктивності, порівнянний з локальними рішеннями, зберігаючи затримки та пропускну здатність у допустимих межах [15]. Водночас вони забезпечують значно вищу гнучкість розвитку архітектури та готовність до інтеграції інтелектуальних сервісів управління мережею.

Локальна реалізація добре підходить для мереж малого та середнього розміру з високими вимогами до затримок, але їй бракує масштабованості для великих або глобально розподілених систем. Натомість хмарна архітектура вносить додатковий рівень абстракції, що може призвести до збільшення затримок, але значно спрощує управління розподіленими інфраструктурами та вирішує проблеми масштабованості [19].

Для наочного зіставлення ключових характеристик розглянутих підходів наведено порівняльний аналіз архітектурних відмінностей між локально розгорнутими та хмарними SDN-контролерами (таблиця 1.1)

Таблиця 1.1

Порівняння архітектур SDN-контролерів

Критерій	Хмарна	Локальна
Архітектура	Віддалена (мікросервіси, контейнери, оркестрація).	Монолітна або кластерна в межах локальної мережі.
Обслуговування	Поетапні, незалежні оновлення без зупинки сервісу.	Простої під час оновлень.
Відмовостійкість	Вбудована за допомогою механізмів хмарної інфраструктури.	Вимагає наявності резервного контролера на місці.
Масштабованість	Гнучка, автоматичне горизонтальне масштабування через реплікацію сервісів.	Обмежена, ручна (вертикальне масштабування або малі кластери).
Затримка	Вища, залежить від інтернет-з'єднання.	Мінімальна (завдяки близькості до мережевих вузлів).
Розгортання	Хмарна інфраструктура (приватна/публічна).	Локально (в межах організації).

1.3. SDN інтерфейси: Northbound та Southbound

Відкриті інтерфейси між площинами SDN є умовою сумісності керуючих застосунків, автоматизації та еволюції мережевих сервісів у бік кращої програмованості [20]. З позиції мережевої інженерії SDN-інтерфейси зазвичай групують на північні інтерфейси (northbound interface, NBI), які пов'язують прикладні сервіси з контролером, та південні інтерфейси (southbound interface, SBI), які пов'язують контролер із комутаторами чи маршрутизаторами площини даних. Виділяють також східні-західні (east/westbound) інтерфейси для взаємодії між контролерами або доменами керування, однак у контексті інтеграції AI/ML за умов обмежених ресурсів найбільш навантаженими є саме NBI (для телеметрії, політик і керуючих викликів) та SBI (для інсталяції правил і подій у площині даних) [20, 21].

Northbound Interface в SDN виступає як інтерфейсний шар, через який прикладні сервіси (керування політиками, оркестрація сервісів, аналітика)

взаємодіють із контролером, отримуючи доступ до абстракцій мережевого стану (топології, статистики, конфігурацій) та впливають на поведінку мережі через високорівневі політики. Саме NBI перетворює контролер на платформу, де мережеві функції програмуються як застосунки. Для ресурсно-обмежених середовищ (edge/IoT, лабораторні стенди, невеликі домени керування) NBI важливий ще й тим, що визначає рівень складності для інтеграції зовнішніх сервісів - інженерні рішення щодо формату даних, моделі запитів/відповідей та підтримки подій напряму впливають на CPU-витрати, обсяг трафіку керування й відтворюваність експериментів. [22, 23]. Типові реалізації NBI у сучасних SDN-екосистемах описують як:

- веб-орієнтовані API (REST)
- RPC-орієнтовані API
- SDK механізми або вбудовані застосунки контролера
- політико-орієнтовані API, шо піднімають рівень абстракції над низькорівневими мережевими правилами.

Водночас у існує проблема стандартизації NBI. На відміну від SBI, де історично домінували стандартизовані протоколи, NBI часто є контролеро-специфічним, що ускладнює сумісність застосунків, підвищує ризик «чорного ящику виробника» [24] і змушує переносити бізнес-логіку між різними моделями даних та абстракціями. Практично значущим напрямом є розвиток NBI, де REST порівнюють із gRPC-подібними стрімінговими підходами [25] для зменшення накладних витрат і підвищення ефективності обміну між застосунками та SDN-контролером.

З погляду інтеграції AI/ML-модулів NBI виступає каналом:

- отримання телеметрії/статистики для навчання або прийняття рішень моделлю;

- виклику сервісів прийняття рішень (класифікація трафіку, виявлення аномалій, прогноз навантаження);

- застосування політик у контролері та подальшого перетворення політик у правила площини даних через SBI.

Наявність централізованого погляду та доступ до мережевих даних розглядаються як передумова для впровадження інтелектуальних механізмів керування [26, 27].

Вибір REST як методологічної основи NBI в SDN/NFV-контексті (рис. 1.4) історично домінує у веб-орієнтованих інтерфейсах і широко застосовується для керування сервісами та оркестрації [23]. Водночас у ресурсно-обмежених умовах критичними стають компроміси між простотою інтеграції та ефективністю протоколу, тому поряд із REST розглядаються gRPC-подібні підходи, які уточнюють межі його доцільності [25].

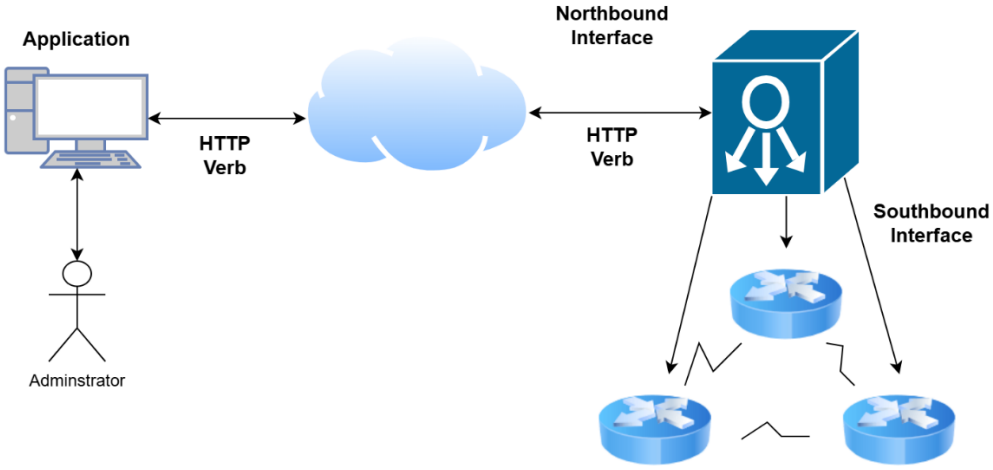


Рис. 1.4. Схема взаємодії з SDN-контролером через REST northbound API

Роботи з проєктування REST-орієнтованих northbound API показують, що системне застосування REST-принципів покращує узгодженість інтерфейсу взаємодії застосунків із контролером і зменшує типові вади дизайну, що важливо для інтероперабельності та повторюваності інтеграцій [23]. REST природно поєднується з HTTP-екосистемою та форматами на кшталт JSON, знижуючи поріг входу для підключення зовнішніх сервісів аналітики й оркестрації [5], а для дослідницьких SDN-стендів додатково вагомим є фактор швидкого прототипування та спільної відтворюваності, підкреслений у дослідницьких роботах про Mininet [28]. Для інтеграції AI/ML-модулів REST-NBI типово використовується для доступу до телеметрії, статистики, керування політиками

та тригерів застосування рішень у контролері, що узгоджується з моделлю SDN-контролера як платформи керуючих застосунків [26, 27].

Разом із тим, компроміси REST пов’язані з накладними витратами текстових форматів і серіалізації. Порівняння JSON із бінарними форматами (наприклад, Protocol Buffers) показують вплив вибору формату на продуктивність і енергоспоживання [29], тому практично релевантним є доповнення REST-інтерфейсів RPC механізмами (зокрема gRPC) для зменшення затримки і підтримки ефективнішої подієвої моделі обміну.

Сукупно джерела дозволяють зробити висновок, що REST є раціональною основою NBI для інтеграції прикладних сервісів (включно з AI/ML-агентами) завдяки інтероперабельності та низькому бар’єру інтеграції, тоді як для «важкої» телеметрії або подій доцільною є комбінована модель, що поєднує REST для керування/політик і gRPC для продуктивного обміну в умовах обмежених ресурсів.

Southbound interface (SBI) у SDN визначає, яким чином контролер керує площиною даних - встановлює правила пересилання, отримує події та статистику, реагує на зміни стану портів/лінків. Ефективність SBI впливає на масштабованість SDN як системи керування - надмірна кількість подій або часті модифікації правил здатні перевантажувати площину керування й погіршувати затримки застосування політик. Найбільш репрезентативним прикладом SBI у класичних SDN-сценаріях є протокол OpenFlow. OpenFlow ґрунтується на наявності таблиць потоків (flow table) у комутаторі та відкритому протоколі, через який контролер модифікує її записи, а еволюція від 1.0 до 1.3+ розширила цю модель від схеми з однією таблицею, до мультитаблиць з набором дій, інструкцій та групових таблиць для складнішої логіки пересилання та meter tables для QoS-лімітування і керування навантаженням [30].

Базова концепція OpenFlow-керування полягає в тому, що комутатор зіставляє (вираз-співпадіння) поля заголовків пакета з записами таблиці потоків, та виконує дії, визначаючи пересилання, модифікацію заголовків або інші операції (рис. 1.5).

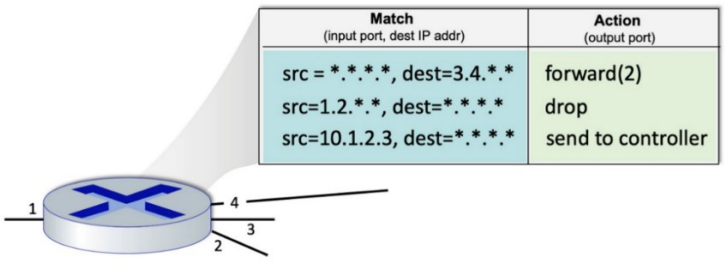


Рис. 1.5. Flow-Table протоколу OpenFlow

У багатотабличній моделі (OpenFlow 1.1+) обробка починається з першої таблиці, для кожної таблиці вибирається запис з найвищим пріоритетом, що відповідає пакету, а потім виконуються інструкції, де goto-table дозволяє перейти до таблиці з більшим ID і тим самим формувати детермінований багатокроковий процес обробки [31]. Записи таблиці потоків, окрім match-умов і дій/інструкцій, включають лічильники (counters) та службові параметри (зокрема, timeouts), які використовують для моніторингу активності правил і керування часом життя записів. Це створює основу для статистичного моніторингу й керуючих циклів, але водночас формує накладні витрати на обробку правил і збору статистики [32].

Групи таблиць Group tables розглядають як механізм, що дозволяє визначати спільні групові дії через buckets та застосовувати їх у різних правилах [33]. У контексті QoS важливим доповненням OpenFlow 1.3 стали meter tables, що реалізують лімітування трафіку та маркування пакетів і, роблять meter tables базовим механізмом балансування навантаженням у сценаріях ресурсного контролю [34].

Типи повідомлень OpenFlow та вплив на площину керування визначає обмін між контролером і комутатором у OpenFlow, включає реактивні та асинхронні повідомлення, основні з них:

- Packet-In/Packet-Out (інкапсуляція пакета/вказівки на обробку)
- Flow-Mod (додавання/оновлення/видалення flow правил)
- Stats Request/Reply (збір статистики/відповідей)
- асинхронні події - PortStatus чи FlowRemoved

Приклад системи моніторингу OpenFlow описує наступну логіку - у разі відсутності відповідного запису в flow-таблиці комутатор передає Packet-In до контролера, а контролер відповідає Flow-Mod для інсталяції правила. Для моніторингу можуть періодично надсилатися Flow-Stats запити і отримуватися відповіді, також використовується Flow-Removed для відстеження видалення правил за timeout чи іншими причинами.

Покрокове per-flow керування в реактивній моделі збільшує навантаження на площину керування і може створювати вузькі місця на контролері, і призводити до проблем з масштабуванням за зростання кількості «малих» потоків і відповідних накладних витрат [18], тоді як компроміс (reactive/proactive) у OpenFlow полягає в тому, що reactive мінімізує попередню конфігурацію, але додає затримку «першого пакета» через цикл Packet-In - Flow-Mod, а proactive зменшує цю затримку ціною наперед встановлених правил, більших витрат пам’яті таблиць і складнішого узгодження, тому гібридні підходи фокусуються на зниженні саме цієї реактивної затримки [35].

Обмеження OpenFlow в ресурсно-обмежених SDN-сценаріях проявляються на двох рівнях:

1) вартість реактивного режиму роботи може різко зростати за високої інтенсивності нових потоків/подій. Рішенням може бути зменшення кількості взаємодій з площиною керування, переносячи певні функції у площиною даних, навіть у межах match-action парадигми можна скорочувати потребу в постійних викликах контролера, підвищуючи стабільність;

2) масштабованість обмежується ресурсами таблиць пересилання. Дослідження ефективного керування flow-таблицями пропонують методи компресії або заміни правил [36].

Окремо слід виділити проблему переповнення flow tables і необхідність компромісу між точністю правил і використанням пам’яті комутатора [37]. Моніторинг OpenFlow і телеметрія зазвичай базуються на статистичних повідомленнях, що збільшує кількість контрольних повідомлень, тому в умовах

обмежених ресурсів потрібен компроміс між точністю спостереження та накладними витратами моніторингу.

Поява альтернативних північних інтерфейсів, зумовлена тим, що програмованість мережі еволюціонує за межі OpenFlow-моделі через інші механізми конфігурації та програмування площини даних, зокрема через конфігураційні протоколи з модельованими даними (NETCONF/YANG-підходи). Такі інтерфейси важливі як доповнення, оскільки також дозволяють відокремлювати «конфігурацію» пристрою (модель-орієнтоване керування) від «правил пересилання», а також підвищувати програмованість площини даних, порівняно з match-action в OpenFlow [38].

1.4. Проблеми управління ресурсами в сучасних SDN-мережах

В умовах обмежених ресурсів, таких як обчислювальна потужність, пам’ять або пропускна здатність, традиційні підходи SDN стикаються зі проблемами у підтримці продуктивності, безпеки та адаптивності. Аналіз сучасного стану розвитку SDN дозволяє виділити ряд ключових проблем управління ресурсами, які потребують дослідження.

Однією з головних проблем є обмежена адаптивність класичних механізмів управління ресурсами, які здебільшого ґрунтуються на фіксованих правилах та статичних політиках. У таких умовах контролер SDN реагує на зміну стану мережі постфактум, що може призводити до перевантаження каналів, збільшення затримок та деградації якості обслуговування [39, 40]. Особливо це проявляється в умовах пікових навантажень, характерних для хмарних сервісів, IoT-інфраструктур та мультимедійних застосунків [41, 42].

Централізована архітектура SDN, з одного боку, спрощує управління мережею, а з іншого - створює додаткове навантаження на SDN-контролер. Контролер повинен обробляти великі обсяги телеметричних даних, приймати рішення щодо маршрутизації та політик доступу в реальному часі. За відсутності інтелектуальних механізмів агрегації та аналізу даних це призводить до

зростання затримки прийняття рішень і негативного впливу на стабільність функціонування мережі [6, 43].

Ще однією суттєвою проблемою є обмеженість ресурсів на рівні мережевих вузлів та контролера, зокрема обчислювальних потужностей, оперативної пам’яті та пропускної здатності каналів керування. Це особливо актуально для гібридних SDN-архітектур, що поєднують хмарні та edge-компоненти, де ресурси вузлів на краю мережі є доволі обмеженими [2, 44].

Проблема забезпечення якості обслуговування (QoS) в SDN-мережах також залишається актуальною. Класичні метрики QoS, такі як пропускна здатність, затримка, джиттер та втрати пакетів, не завжди адекватно відображають реальне сприйняття якості сервісу з боку кінцевого користувача [45]. У результаті навіть формальне дотримання параметрів якості надання послуг не гарантує задовільної якості користувацького досвідуї, особливо для мультимедійних і інтерактивних сервісів [45, 40].

Додатковим ускладненням є зростання гетерогенності мережевого трафіку, зумовлене одночасною передачею даних різних типів даних - від критично важливих IoT-повідомлень до важкого відеоконтенту. Це вимагає динамічного перерозподілу ресурсів і пріоритезації потоків, що складно реалізувати за допомогою традиційних правил маршрутизації та статичних політик управління [46].

Також слід відзначити проблему безпеки в SDN. Централізований контролер є критично важливим елементом мережі, атаки на який можуть призвести до порушення роботи всієї інфраструктури. Зловмисні впливи, такі як DDoS-атаки на контролер або маніпуляції з правилами пересилання, можуть спричинити неефективне використання ресурсів або повну втрату керованості мережею [47, 48].

У сукупності зазначені проблеми свідчать про те, що традиційні підходи до управління ресурсами в SDN-мережах мають складнощі із забезпеченням необхідного рівня адаптивності, масштабованості та ефективності в умовах динамічного мережевого середовища та обмежених ресурсів. Це обґрунтовує

необхідність подальших досліджень у напрямі автоматизації процесів управління, застосування інтелектуальних методів аналізу даних та інтеграції алгоритмів штучного інтелекту в цикл керування SDN-контролера [49, 50].

1.5. Аналіз існуючих підходів до автоматизації управління ресурсами

Автоматизація управління ресурсами в програмно-конфігурованих мережах базується на концепції централізованого контролю та програмної реконфігурації мережевих елементів у режимі реального часу. Ця властивість дає змогу програмно й динамічно змінювати конфігурацію мережі через відповідний інтерфейс, забезпечуючи її адаптацію в реальному часі до нових вимог [50]. Аналіз літературних джерел показує, що існуючі підходи до автоматизації управління ресурсами можна класифікувати за функціональними завданнями: забезпечення якості обслуговування, балансування навантаження, управління енергоефективністю та інтеграція з хмарними/периферійними обчисленнями.

Однією з найбільш актуальних задач є гарантування якості обслуговування (QoS), особливо з огляду на зростання різноманітності мережевого трафіку. У роботах [40, 41] підкреслюється потреба в гнучкому та масштабованому підході до керування й налаштування мережевого трафіку, де SDN дозволяє централізовано керувати трафіком через контролер. Для розв’язання завдань QoS/QoE широко застосовуються методи класифікації та пріоритезації. Наприклад, у дослідженні [51] запропоновано підхід до автоматичного виявлення «важких» потоків для запобігання перевантаженню мережі та оптимізації QoS за допомогою алгоритмів глибокого навчання. Інші автори [52] запропонували метод глибокого навчання, на основі якого контролер автоматично коригує пріоритети і маршрути для критичних потоків, чутливих до затримок.

Балансування навантаження в SDN охоплює як розподіл трафіку між різними маршрутами, так і розподіл контролю між декількома контролерами. Автори в [53] провели розгорнутий огляд алгоритмів балансування, класифікувавши їх на підходи, натхненні природними процесами, методи

машинного навчання та математичні моделі. Вагомим напрямком є використання алгоритмів оптимізації. Зокрема, у дослідженні [54] представлено стратегію DRL-SMS, яка визначає, коли і які комутатори слід мігрувати до іншого контролера, щоб вирівняти завантаження площини контролю. Такий підхід досягає швидшого часу відгуку на сплески трафіку, ніж статичні алгоритми. Для розв'язання задачі запобігання перевантаженням також застосовують гібридний підхід до розподілу трафіку - із поєднанням традиційних маршрутизаторів та SDN-вузлів [44]. Така схема може бути результативною, оскільки здатна підвищувати продуктивність мережі та водночас зменшувати експлуатаційні витрати.

Управління ресурсами в середовищі IoT та Cloud-SDN ускладнюється специфікою Інтернету речей, яка накладає додаткові обмеження на обчислювальну потужність, енергоспоживання та пропускну здатність мережі. Інтеграція IoT та SDN, відома як SDN-IoT [55], створює єдину архітектуру для координації взаємодії вузлів. Для управління великими обсягами даних IoT ефективним є використання хмарних обчислень [56, 57]. Хмарна реалізація SDN забезпечує кращі можливості до автоматичного масштабування та централізованого управління розподіленими мережами [11]. Однак, як зазначено в [58], для задач із жорсткими вимогами до затримок доцільно використовувати дворівневу архітектуру (Edge-Cloud), що поєднує потужність хмари зі швидкістю периферійних контролерів. У роботі [13] запропоновано ієрархічну систему контролерів HECSDN, яка зменшує ризик перевантаження центрального контролера шляхом розподілу обчислювальних завдань. Такий децентралізований підхід [59] дозволяє наблизити прийняття рішень до джерел даних, зберігаючи при цьому глобальний нагляд.

Енергоефективність і безпека формують окремий клас задач у сфері управління ресурсами в SDN, зокрема пов'язаних з оптимізацією енергоспоживання. У роботі [60] пропонується алгоритм, що передбачає оптимальну конфігурацію мережі, дозволяючи вимикати або переводити у режим енергозбереження частини обладнання під час низького навантаження. Для

великих промислових IoT-мереж у [61] запропоновано використання еволюційних алгоритмів (рою часток та бджолиного рою) для енергоефективної маршрутизації. Безпека також розглядається як ресурсна проблема, оскільки атаки (наприклад, DDoS) виснажують ресурси мережі. У [62] представлено модульну архітектуру для виявлення DDoS атак, де моделі ШІ забезпечили високу точність виявлення (до 95%), що дозволяє своєчасно вивільняти ресурси, зайняті шкідливим трафіком.

Аналіз існуючих підходів свідчить, що автоматизація управління ресурсами в SDN розвивається в напрямку інтеграції інтелектуальних методів [63]. Існуючі рішення, такі як ETSI TeraFlowSDN [64] або гібридні моделі [65], демонструють ефективність у специфічних завданнях. Проте, більшість розглянутих підходів фокусуються на окремих аспектах (тільки QoS, або тільки безпека, або тільки енергоефективність). Спостерігається нестача комплексних фреймворків, які б дозволяли адаптивно керувати доступом до обмежених ресурсів в гетерогенних середовищах (Cloud-SDN-IoT) з урахуванням багатьох факторів одночасно. Це підтверджує актуальність розробки нових методів і технологій використання алгоритмів ШІ в умовах ресурсних обмежень.

1.6. Застосування та обмеження використання алгоритмів ШІ в SDN

Програмно-конфігуровані мережі надають гнучку централізовану інфраструктуру, а штучний інтелект - інструменти для розумного прийняття рішень [66]. Така комбінація дає змогу прогнозувати мережеве навантаження, виявляти аномалії, оптимізувати маршрути і розподіляти ресурси автоматично. Впровадження методів штучного інтелекту істотно розширює функціональні можливості SDN-мереж. У літературі [67] відзначають успішне використання ML/DL-алгоритмів для підвищення пропускної здатності, зниження затримок та своєчасного виявлення атак.

Одним із найбільш поширених напрямів застосування ШІ в SDN є інтелектуальне управління трафіком та оптимізація маршрутизації. Такі підходи особливо ефективні для сервісів реального часу та мереж з високою мінливістю

трафіку, зокрема в середовищах IoT та edge computing [61]. Виявлення несправностей та відновлення після збоїв у SDN полягає в ідентифікації відмов на різних рівнях мережі та автоматичному відновленні її працездатності. Дослідники [68] запропонували ігрову модель для автоматичного відновлення після збоїв контролера SDN. Інший підхід - аналітика логів контролера за допомогою ML, у роботі [69] запропоновано метод, де ML модель навчена розпізнавати у послідовності подій характерні патерни, що передують збою контролера.

Попри значні переваги, інтеграція ШІ в SDN супроводжується низкою обмежень та викликів, які особливо критичні в умовах обмежених ресурсів:

- Проблема «чорного ящика». Моделі ШІ, особливо глибокі нейронні мережі, часто працюють як «чорні ящики», і їх рішення важко інтерпретувати. Це є проблемою для мережевих адміністраторів, яким необхідно розуміти причини прийняття конкретних рішень щодо блокування трафіку чи зміни маршрутів.

- Вимоги до обчислювальних ресурсів. Деякі алгоритми (особливо DL) потребують значних обчислювальних потужностей і часу на навчання. Це може обмежувати їх застосування у реальному часі безпосередньо на контролері або вимагати попереднього тренування в емуляторах чи використання хмарних ресурсів. В умовах обмежених ресурсів традиційні підходи SDN можуть стикатися з проблемами продуктивності [70], тому акцент зміщується на легковагові моделі.

- Залежність від даних - ефективність моделей прямо залежить від якості та обсягу навчальних даних. Також, модель, навчена на одній топології чи наборі даних, може втрачати ефективність при зміні умов функціонування мережі.

- Стійкість та безпека самих моделей. Необхідно гарантувати надійність моделей ШІ, зокрема їх стійкість до атак на навчальні дані.

Розробка методів і технологій використання ШІ в SDN вимагає ретельного балансування між точністю алгоритмів та витратами ресурсів на їх роботу [71], що є предметом подальших досліджень.

1.7. Аналіз ML моделей та реалізацій для SDN

У SDN-системах ML/DL-модулі зазвичай включають у цикл керування як засоби класифікації, прогнозування та оптимізації, використовуючи телеметрію та логічно централізований збір даних. Однак ресурсо-обмежені сценарії (edge/IoT, малі контролери, лабораторні стенди з обмеженим CPU/RAM) вимагають порівнювати моделі не лише за точністю, а й за часом прийняття рішень, вимогами до даних і витратами на інтеграцію в керуючий контур [26, 27]. Вибір моделі та реалізації має відповідати доступним даним, вимогам реального часу та обчислювальним обмеженням середовища виконання. В умовах обмежених ресурсів додатково оцінюють можливість впливу виконання моделі на CPU, стабільність затримки в межах керуючого циклу та співвідношення накладних витрат моніторингу й очікуваного ефекту від ML-керування.

В оглядах застосувань ML у SDN зазначено, що більшість рішень базується на класичних алгоритмах для задач керування та безпеки, що обґрунтовує використання «легких» моделей, особливо за обмежених даних або потреби в інтерпретованості. У ресурсно-чутливому SDN-дизайні показовими є підходи з reinforcement learning для зменшення додаткових витрат часу моніторингу, де ML оптимізує не лише мережеву політику, а й системні витрати.

У виборі бібліотек для реалізації ML/DL-компонентів у SDN, підхід має враховувати не лише підтримувані алгоритми, а й експлуатаційні фактори - залежності, складність розгортання, відтворюваність експериментів, інтеграцію з мовою/стеком, у якому реалізовано SDN-керування та прикладні сервіси. Для дослідницьких стендів додатково важливо, щоб інструментальна база підтримувала швидке повторення експериментів у контрольованих умовах [28]. Бібліотеку scikit-learn розглядається як інструмент, орієнтований на доступність машинного навчання та зручність використання через узгоджений API. В роботі [107`72] підкреслено ідею зниження бар’єру входу та прикладну спрямованість бібліотеки, що особливо актуально для інтеграції ML у SDN, де модуль має

працювати в межах керуючого циклу без надмірного ускладнення експлуатації. У рамках даного дослідження (ресурсні обмеження, edge/IoT-сценарії) стек на основі scikit-learn доцільно пов’язувати з класичними моделями, які простіше виконуються на CPU та легше інтегруються в Python-орієнтований сервіс обробки телеметрії й прийняття рішень.

TensorFlow/PyTorch бібліотеки у розглядаються як репрезентативні DL-фреймворки, що на практиці використовують для нейромережних моделей і reinforcement learning, однак їх застосування у ресурсно-обмежених середовищах часто вимагає додаткових оптимізацій, про що свідчить систематизація технік «efficient deep learning» для зменшення моделей та підвищення часу прийняття рішень [73].

Якщо ціллю є побудова відтворюваного, відносно простого керуючого контуру SDN+AI за обмежених ресурсів, то використання scikit-learn і класичних моделей може виступати як обґрунтована практична реалізація, тоді як DL-стек доцільно розглядати як розширення, що потребує окремого шару інженерних оптимізацій і оцінювання накладних витрат [26, 72, 73].

1.8. Застосування MQTT у програмно-конфігурованих мережах

У поєднанні програмно-конфігурованих мереж та IoT очікуваною є здатність мережі одночасно працювати як транспорт телеметрії та як середовище доставки керувальних дій, з прийнятними затримками й обмеженими накладними витратами. IoT-пристрої генерують потоки подій і показників, які часто мають малий розмір, сплесковий характер та нерегулярність через енергозбереження й нестабільні канали зв’язку. За таких умов доцільно використовувати подієво-орієнтований обмін повідомленнями, здатний підтримувати асинхронну взаємодію та розсилання одного повідомлення багатьом отримувачам без перетворення кожної операції на «важкий» клієнт-серверний виклик [74]. Саме тому MQTT системно розглядають як практичний компроміс між мінімізацією трафіку та обсягу стану на вузлах і достатньо гнучким механізмом доставки, який можна додатково підсилювати засобами

SDN на рівні мережевих політик (пріоритезація потоків, сегментація, керування пропускною здатністю).

MQTT належить до брокер-центричних publish/subscribe протоколів. Видавці (publishers) відправляють повідомлення в теми (topics), підписники (subscribers) декларують інтерес до тем, а брокер виконує роль посередника, який приймає публікації та пересилає їх за політиками підписок [75]. Така модель дозволяє роз'єднати у часі та просторі джерела даних і споживачів, що особливо важливо для IoT-вузлів, які можуть бути періодично недоступними або мати обмежений ресурс енергії. У прикладному сенсі теми формують «простір імен» для телеметрії та команд, окремі гілки тем можуть відображати тип сенсора, місце, вузол, сервіс або клас керуючих дій, і саме через цю абстракцію часто будують керування IoT у SDN-керованих системах.

Властивістю MQTT є підтримка рівнів якості обслуговування (QoS 0/1/2), які задають семантику доставки (рис. 1.6) «щонайбільше один раз», «щонайменше один раз» та «рівно один раз».

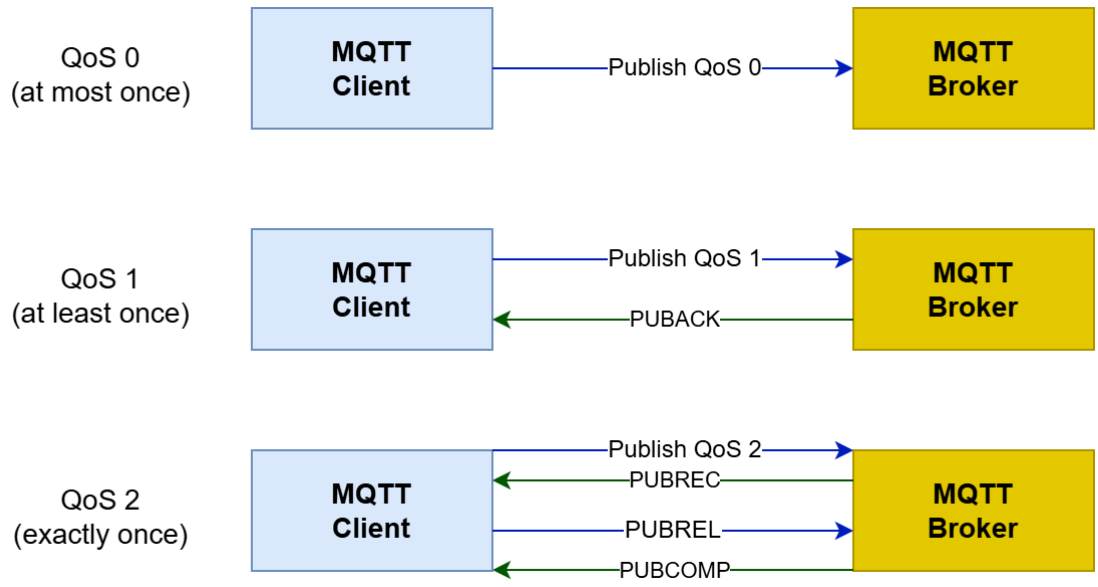


Рис. 1.6. Послідовності обміну повідомленнями MQTT для рівнів QoS 0, 1 та 2

Практичний наслідок полягає в тому, що підвищення надійності доставки супроводжується зростанням службових обмінів і обсягу стану в мережі та на брокері, а отже збільшенням затримки, витрат процесорного часу та

енерговитрат. У роботі [76] підкреслено, що для QoS 2 потрібна багатокрокова процедура узгодження (підтвердження доставки), яка ускладнює обмін і підвищує енергоспоживання, тоді як QoS 0 має найменші накладні витрати, але не передбачає підтверджень. Для нестабільних з'єднань існують механізми сесій і відновлення після розривів з'єднання, вони дозволяють зберігати контекст взаємодії та продовжувати доставку повідомлень після тимчасової відмови вузла [77].

На рівні загроз для брокера суттєвою проблемою є відкритість MQTT-інфраструктури та уразливості реалізацій. Брокери як центральний компонент MQTT-систем [78] стають цілями для атак на доступність (DoS/DDoS) і для експлуатації вразливостей обробки пакетів, а також підкреслюють дефіцит IDS-інструментів, які «розуміють» MQTT на рівні парсингу.

Застосування SDN відкриває можливість узгоджувати політики безпеки між прикладним рівнем (MQTT-автентифікація/авторизація і контроль доступу за темами) та мережевим рівнем (ізоляція сегментів, пріоритезація потоків керування, обмеження швидкості, блокування підозрілих джерел). Хоча конкретні механізми керування зазвичай реалізуються в брокері або в шлюзі, SDN-контролер може виступати єдиною точкою розгортання політик і переналаштування маршрутів/черг, що особливо важливо в розподілених edge-сценаріях. У роботі [79] SDN-складова описується як керування маршрутизацією потоків і роботою edge-контейнерів під контролем SDN-контролера, що закладає технічні передумови для дотримання політик на інфраструктурному рівні. Дослідження [74] обґрунтовує доцільність поєднання SDN і MQTT потребою динамічної реконфігурації розподілених брокерних інфраструктур. У безпековому плані це можна трактувати як здатність швидко «перемикати» топологію взаємодії та правила доступу/пересилання у відповідь на зміну ризиків, появу аномалій або розвиток атаки, не зупиняючи систему.

Перетин MQTT з AI-автоматизацією зазвичай формулюють як необхідність динамічних політик, що перебудовуються на основі спостережуваних метрик

навантаження та якості каналу. До них належать – архітектурне розміщення брокера, адаптація рівня QoS, планування повідомлень, застосування захисних правил. Оптимізація планування на edge та керування пам’яттю брокера у хмарі використовуються для зменшення затримок і скорочення споживання пам’яті [80], тобто для підтримання працездатності системи в умовах обмежених ресурсів.

Висновки до розділу 1

У розділі проведено огляд літературних джерел за тематикою програмно-конфігурованих мереж та застосування методів штучного інтелекту для керування ними в умовах обмежених ресурсів. Розглянуто архітектурні принципи SDN як підхід із розділенням площин керування та пересилання, що забезпечує програмованість мережі, централізоване керування ресурсами й гнучку реконфігурацію, також визначено типові ризики такого типу мереж, що проявляються у складності обробки мережевої інформації та критичність порушень безпеки контролера. Проаналізовано підходи до організації SDN-контролера (локальні, хмарні та гібридні edge–cloud архітектури) та показано компроміси між масштабованістю, затримками керування і безпекою. Визначено, що хмарні та гібридні підходи створюють сприятливі умови для інтеграції сервісів AI/ML, однак потребують контролю затримок і захищеної взаємодії з площиною даних.

Узагальнено роль інтерфейсів SDN (NBI/SBI) в контурі керування та підкреслено їх вплив на накладні витрати телеметрії й застосування політик у ресурсно-обмежених сценаріях, включно з проблемами стандартизації NBI та необхідністю зниження «важкої» телеметрії без втрати керованості. Виявлено виклики управління ресурсами в SDN за умов обмежень CPU/RAM/каналів керування - недостатня адаптивність статичних політик, зростання навантаження на контролер через потоки телеметрії та потреба прийняття рішень у реальному часі, підтримка QoS/QoE для гетерогенного трафіку, а також посилення безпекових ризиків через критичність контролера.

Систематизація існуючих підходів автоматизації (балансування, QoS/QoE-керування, енергоефективність, інтеграція з edge/cloud, безпека) показала, що значна частина рішень орієнтована на окремі задачі, тоді як комплексні підходи до адаптивного керування доступом до обмежених ресурсів у гетерогенних середовищах потребують розвитку. Показано перспективність застосування ШІ в SDN для прогнозування навантаження, оптимізації маршрутів, виявлення аномалій та автоматизованого розподілу ресурсів, але також виділено обмеження впровадження, пов’язані з вимогами до якості даних.

Розробка методів і технологій використання ШІ в SDN вимагає ретельного балансування між точністю алгоритмів та витратами ресурсів на їх роботу, оскільки в ресурсно-обмеженому контурі керування визначальними стають не лише метрики якості класифікації чи прогнозу, а й часу прийняття рішень, стабільності затримки та накладні витрати інтеграції й моніторингу. Також розглянуто роль протоколів IoT-взаємодії у формуванні керованих політик обміну повідомленнями та забезпеченні працездатності системи за умов обмежених ресурсів.

Результати огляду підтверджують актуальність і доцільність розроблення методів та технологій інтеграції алгоритмів штучного інтелекту в контур керування SDN, які одночасно підвищують адаптивність керування ресурсами та забезпечують контроль накладних витрат у ресурсно-обмежених середовищах.

РОЗДІЛ 2. МЕТОДИ ПОБУДОВИ ІНТЕЛЕКТУАЛЬНИХ SDN-МЕРЕЖ

2.1. Метрики оцінювання ефективності та справедливості розподілу доступу в ШІ-інтегрованих SDN-системах

В контексті ШІ-інтегрованих SDN-систем, важливо оцінювати як ефективність моделі машинного навчання, так і справедливість розподілу мережевих ресурсів. Для цього застосовуються метрики двох типів: 1) класифікаційні метрики ML, які відображають точність та якість роботи алгоритму [81], та 2) метрики справедливості, що характеризують рівномірність і неупередженість розподілу обмежених ресурсів між користувачами або вузлами мережі. Нижче розглянуто основні показники кожного типу, їх визначення, формули та доцільність використання в умовах обмежених ресурсів, дисбалансу класів та вимог до QoS.

2.1.1. Класифікаційні метрики ефективності ML-алгоритмів

При оцінюванні класифікаторів у задачах SDN (наприклад, для класифікації трафіку чи виявлення атак) найчастіше використовуються класичні метрики: Accuracy (точність), Precision (прецизійність), Recall (повнота), F1-score (F1-міра) та ROC-AUC. Кожна з цих метрик має наступні особливості:

- Точність (Accuracy) - частка правильних класифікацій серед усіх випадків. Визначається як (2.1):

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (2.1)$$

де *TP, TN, FP, FN* - кількість істинно позитивних, істинно негативних, хибнопозитивних та хибнонегативних спрацювань відповідно.

Accuracy інтерпретується як загальна ймовірність правильного рішення класифікатора. Вона доцільна для збалансованих наборів даних і некритичних задач, оскільки високий відсоток правильних відповідей свідчить про загальну ефективність моделі. Втім, у випадку сильно незбалансованих класів Accuracy може вводити в оману - наприклад, при 1% аномалій класифікатор, що завжди видає «норму», матиме 99% точності [82]. Отже, в умовах рідкісних подій чи

критичних QoS показників (для безперебійного моніторингу критичного трафіку) одну лише метрику Accuracy використовувати недоцільно.

- Прецизійність (Precision) - частка коректно передбачених позитивних випадків (*TP*) серед усіх, класифікованих моделлю як позитивні (*TP* + *FP*), наприклад атака або пріоритетний трафік. Формула Precision (2.2):

$$Precision = \frac{TP}{TP+FP} \tag{2.2}$$

Дана метрика показує, наскільки «чистими» є позитивні спрацювання моделі, тобто який відсоток її сповіщень є істинними. Високі значення наприклад означають, що модель рідко помилково позначає безпечний трафік як шкідливий або низькоприторитетний як високопріоритетний. Precision особливо важлива, коли хибні тривоги (False Positives) небажані – наприклад, при виділенні обмежених ресурсів критично, щоб не призначити їх помилково непотрібному потоку. Однак надмірна оптимізація Precision може знизити Recall (модель буде надто консервативною). У задачах з дисбалансом класів Precision є кориснішою за Accuracy, бо фокусується саме на позитивному класі.

- Повнота (Recall) - частка правильно виявлених позитивних випадків серед усіх реальних позитивних прикладів. Обчислюється як (2.3):

$$Recall = \frac{TP}{TP+FN} \tag{2.3}$$

Recall (чутливість або True Positive Rate) відображає здатність моделі знаходити всі важливі випадки. Високе значення Recall означає, що модель виявляє майже всі пріоритетні або аномальні потоки, хоча це може бути ціною появи деяких хибних спрацювань. Recall є критичною метрикою, коли пропуск (False Negative) недопустимий. Наприклад, у системах виявлення вторгнень або при гарантуванні QoS для чутливих до затримок застосунків. У мережах з обмеженими ресурсами високий Recall гарантує, що всі потоки, які дійсно потребують ресурсу, його отримають, але без урахування Precision може виникнути перевиділення (нецільове витрачання) ресурсу.

- F1-score - гармонійне середнє Precision та Recall, що обчислюється як (2.4):

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (2.4)$$

F1-score поєднує обидва аспекти - і точність позитивних класифікацій, і повноту їх виявлення. Високий F1 свідчить, що модель збалансовано добре ловить позитивні випадки і при цьому мало помиляється в хибних тривогах. Ця метрика особливо корисна коли важливий баланс між пропусками і хибними спрацюваннями. У SDN-завданнях (скажімо, класифікація типів трафіку) F1 дає узагальнену оцінку якості, що актуально для практичного використання (наприклад, якщо F1 високий, то і Precision, і Recall достатньо добрі для забезпечення QoS). В умовах обмежених ресурсів F1 є зручним критерієм, бо не дозволяє оптимізувати лише один аспект (як-от Precision) ціною погіршення іншого.

- ROC-AUC - площа під ROC-кривою (Receiver Operating Characteristic). ROC-крива будується за рахунок варіювання порогу класифікації та відображає залежність чутливості (True Positive Rate) від частоти хибних спрацювань (False Positive Rate). Показник AUC (Area Under Curve) дорівнює ймовірності, що модель присвоїть вищу оцінку (ранг) випадковому позитивному зразку, ніж випадковому негативному. На практиці ROC-крива формується не як неперервна функція, а як впорядкований набір дискретних точок, отриманих при зміні порогового значення класифікації. У такому випадку площа під ROC-кривою (ROC-AUC) обчислюється чисельно, найчастіше за допомогою методу трапецій. ROC-AUC для дискретного набору точок визначається як (2.5):

$$AUC \approx \sum_{i=1}^{N} \frac{(TPR_i + TPR_{i-1})}{2} \times (FPR_i - FPR_{i-1}) \quad (2.5)$$

де: - TPR_i - значення True Positive Rate у i-й точці ROC-кривої;

- FPR_i -значення False Positive Rate у i-й точці ROC-кривої;

- i - індекс точки, що відповідає певному значенню порогу класифікації;

- N - загальна кількість дискретних порогових значень (точок ROC-кривої), впорядкованих за зростанням FPR.

З математичної точки зору, формула є чисельним наближенням інтеграла (2.6):

$$AUC = \int_0^1 TPR(FPR)d(FPR) \tag{2.6}$$

Оскільки ROC-крива в експериментальних дослідженнях формується як дискретна множина точок, отриманих при зміні порогу класифікації, площа під кривою обчислюється чисельно методом трапецій, що є стандартним підходом у задачах оцінювання якості класифікації. Значення AUC (рис. 2.1) варіюється від 0.5 (модель не краща за випадкове гадання) до 1.0 (ідеальний класифікатор). Перевага ROC-AUC - інваріантність до вибору порогу та цілісна оцінка моделі на всьому діапазоні чутливості/специфічності. В задачах SDN (наприклад, виявлення аномалій) високий AUC свідчить, що модель добре розрізняє класи при різних налаштуваннях порогу. Втім, при сильному дисбалансі класів AUC може давати завищену оцінку, коли негативних прикладів дуже багато, модель може отримати високий AUC переважно за рахунок правильних негативів, навіть пропускаючи значну частку позитивів. Тому в умовах рідкісних, але критичних подій до ROC-AUC варто додатково аналізувати Precision-Recall криві.

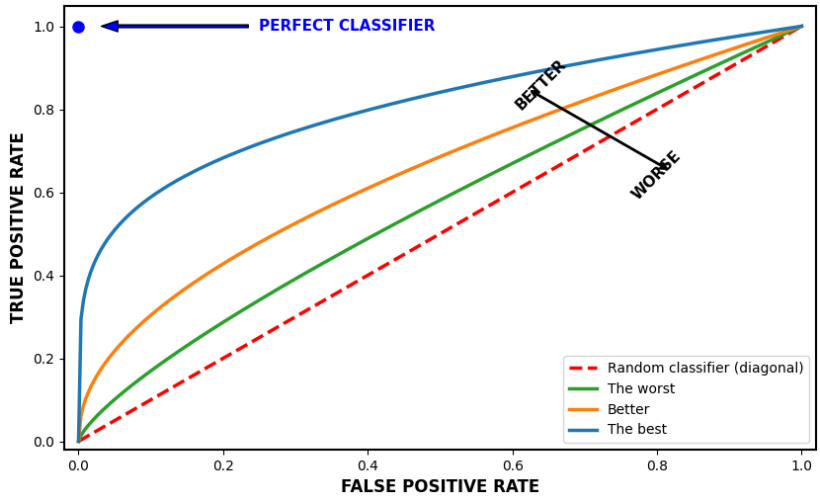


Рис. 2.1. Приклад порівняння якості роботи класифікаторів за допомогою ROC-кривої

2.1.2. Метрики розподілу ресурсів в SDN

У SDN-системах, які динамічно розподіляють обмежену пропускну здатність, час обробки або інші ресурси між потоками, надзвичайно важливою є справедливість цього розподілу. Алгоритми ML можуть приймати рішення про пріоритизацію трафіку, проте це не гарантує, що всі користувачі отримають чесну частку ресурсу. Тому вводяться спеціальні метрики справедливості, які дозволяють кількісно оцінити рівномірність розподілу. Розглянемо дві ключові метрики: індекс справедливості Джейна та критерії на кшталт справедливості Max-Min, і порівняємо їх.

Індекс справедливості Джейна (Jain's Fairness Index) - широко використовувана метрика, що дає скалярну оцінку рівномірності розподілу ресурсу між n учасниками. Для значень розподіленого ресурсу $x_1, x_2, \ldots, x_n$ індекс Джейна визначається формулою (2.7):

$$J = \frac{(\sum_{i=1}^{n} x_i)^2}{n \times \sum_{i=1}^{n} x_i^2} \tag{2.7}$$

Значення J, близькі до 1 (рис. 2.2), вказують на високий рівень справедливості, тоді як зменшення показника означає зростаючу нерівність. Важливо, що індекс Джейна не залежить від абсолютної величини ресурсу чи кількості користувачів - він відображає лише баланс розподілу.

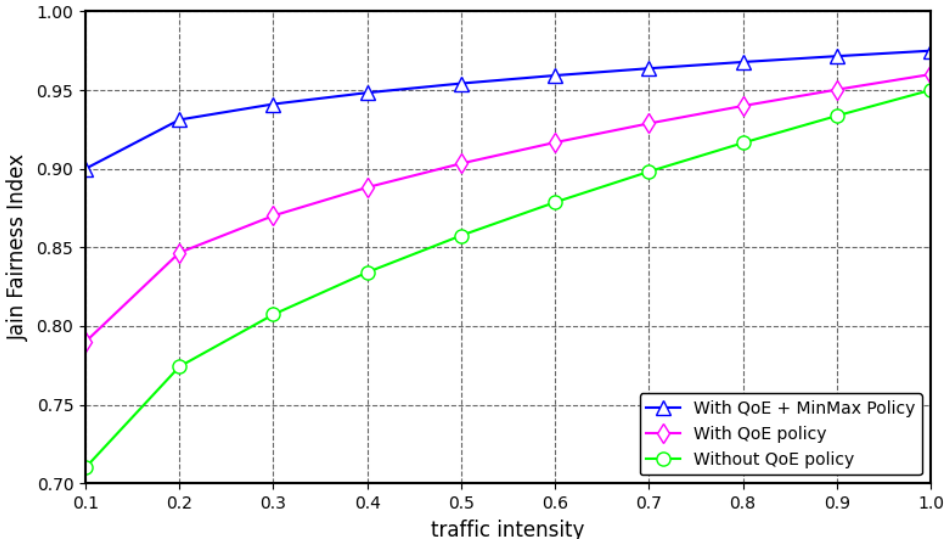


Рис. 2.2. Приклад індексу Джейна для різних політик керування трафіком

У SDN його доцільно застосовувати для оцінки алгоритмів виділення смуги пропускання, часу процесора на мережевих функціях тощо, особливо коли потрібно гарантувати, що жоден з потоків не голодує. В умовах критичних QoS, JFI допомагає виявити, чи не страждає якийсь клас трафіку - значення, нижчі за певний поріг (наприклад, <0.8), можуть слугувати сигналом, що деякі користувачі отримують недостатньо ресурсу [83].

Max-Min Fairness - критерій розподілу, згідно з яким ресурси надаються так, щоб максимізувати найменшу долю серед усіх учасників. Формально, розподіл називають max-min справедливим, якщо неможливо збільшити виділення ресурсу будь-якому учаснику без зменшення частки іншого, що має таку ж або меншу долю [84]. Іншими словами, спочатку максимально задовольняється користувач з найменшим запитом (або можливістю), потім наступний і т.д. Такий принцип гарантує, що жоден потік не залишиться обділеним - спершу всі отримують принаймні мінімально прийнятний рівень обслуговування, і лише після насичення мінімумів залишковий ресурс розподіляється між менш критичними потребами.

Порівнюючи з індексом Джейна - індекс дає числову оцінку існуючого розподілу, тоді як Max-Min Fairness це принцип або алгоритм досягнення справедливого розподілу. Max-min розподіл завжди приводить до високого значення JFI (близького до 1), але при цьому може жертвувати загальною ефективністю. Така політика забезпечує більш стабільну якість обслуговування для всіх користувачів, проте загальна пропускна здатність мережі може бути менша, ніж при агресивному виділенні ресурсів найсильнішим потокам.

2.1.3. Взаємозв’язок класифікаційних та метрик справедливості

Незважаючи на високу точність або інші ML-метрики, алгоритм керування мережею не завжди гарантуватиме якісне обслуговування користувачів, якщо не враховано справедливість розподілу. Висока Accuracy класифікатора не означає справедливого розподілу ресурсу. Наприклад, модель може правильно визначати пріоритетні потоки з точністю 95%, але якщо решта 5% постійно залишаються

без ресурсу, мережа буде несправедливою. Механізми, зосереджені лише на QoS або продуктивності, можуть ігнорувати справедливість, і навпаки - алгоритми, що оптимізують лише справедливість, можуть жертвувати затримкою чи пропускною здатністю [85].

У сучасних SDN-рішеннях є сенс інтегровувати обидва типи метрик. Під час оцінювання AI/ML-алгоритму для мережі слід аналізувати як показники точності прогнозування/класифікації, так і показники рівності обслуговування користувачів. Такий підхід гарантує, що ШІ-агент, навіть навчившись ефективно класифікувати чи прогнозувати трафік, не буде монополізувати ресурси для окремих потоків ціною деградації інших. Як наслідок, лише поєднання ML-метрик (Accuracy, F1 тощо) зі метриками розподілу дозволяє всебічно оцінити якість рішення в SDN, алгоритм повинен не тільки правильно приймати рішення, а й робити це без упередження у виділенні ресурсів. Це особливо важливо для гарантування QoS у сценаріях обмежених ресурсів, де система повинна дотримуватися балансу між ефективністю (використання пропускної здатності, мінімізація затримок) та справедливістю (рівноправний доступ до ресурсів для різних користувачів і сервісів).

Таким чином, оцінювання рішень AI-SDN вимагає багатогранного підходу. Показники на кшталт Precision/Recall інформують про точність і надійність алгоритму прийняття рішень, а індекс Джейна чи подібні критерії про те, як ці рішення впливають на розподіл ресурсів між усіма учасниками мережі. Комбінований аналіз забезпечує більш надійне та справедливе управління мережею, де висока ефективність не досягається за рахунок дискримінації окремих користувачів.

2.2. Обґрунтування вибору моделей машинного навчання для задач SDN

Враховуючи типові можливості SDN щодо централізованого збору статистик потоків і керованості площиною даних, ML-підходи широко використовуються для виявлення аномалій [86, 87], класифікації трафіку та

прогнозування навантаження/станів мережі [88]. Централізація керування та програмованість мережі SDN створюють передумови для збору телеметрії/статистик і побудови моделей, які приймають рішення щодо трафіку, політик або безпеки. З погляду методології, вибір моделей для SDN доцільно формалізувати через критерії:

- Тип задачі та вимоги до затримки (режим реального часу проти офлайн-аналізу), оскільки SDN-контролер може бути «вузьким місцем», а моделі детекції/класифікації мають працювати з мінімальними накладними витратами.

- Інтерпретованість і керування ризиками. У мережевій безпеці та управлінні QoS важлива можливість пояснення рішень (особливо для політик блокувань чи обмежень [89]). Моделі на кшталт LR забезпечують прозору інтерпретацію ваг, тоді як ансамблі та нейромережі потребують додаткових засобів інтерпретації .

- Стійкість до нелінійностей і взаємодій ознак. Потоки SDN часто описуються статистичними ознаками (пакети, часові характеристики тощо) з нелінійними залежностями. Ансамблі дерев (RF/GB) та MLP краще моделюють складні взаємодії [90], ніж лінійні моделі.

- Дисбаланс класів та рідкісні події. В системах виявлення вторгнень чи DDoS часто трапляється дисбаланс [91], тому бажано включити процедури корекції дисбалансу (ваги класів, ресемплінг), а також метрики, чутливі до дисбалансу (PR-AUC).

- Обчислювальна вартість навчання та час прийняття рішень. В SDN практично важлива швидкість обробки потоків [89], тому в набір моделей варто включити як «легкі» (LR), так і більш потужні (GB, MLP) з подальшим порівнянням точності проти затримки.

Застосування розширеного набору методів машинного навчання є методологічно обґрунтованим кроком [92], оскільки дозволяє сформувати репрезентативне покриття основних класів сучасних алгоритмів [90]. Охоплюючи лінійні ймовірнісні методи, ансамблі дерев, нейромережеві апроксиматори, такий підхід забезпечує можливість об’єктивного порівняння за

ключовими для SDN-сценаріїв критеріями - точністю детекції, рівнем інтерпретованості результатів та обчислювальною складністю. Це дозволяє не лише знайти найбільш ефективне рішення, а й оцінити компроміс між якістю класифікації та витратами ресурсів контролера. З огляду на це, як основні моделі-кандидати для подальшого огляду обрані: Logistic Regression (LR), Random Forest (RF), Gradient Boosting (GB), Support Vector Machine (SVM) та Multilayer Perceptron (MLP), що забезпечують репрезентативне порівняння для SDN-задач при різних вимогах до точності та часу прийняття рішень.

Logistic Regression належить до базових методів керованого машинного навчання для задач бінарної класифікації [93]. Модель описує ймовірність належності об'єкта до класу $y = 1$ через логістичну функцію від лінійної комбінації вхідних ознак x (2.8). У статистичній інтерпретації параметри w та b оцінюються за принципом максимальної правдоподібності, що забезпечує формальний зв'язок між даними спостережень і параметрами моделі. Для задач управління в SDN така модель є доцільною як легковаговий класифікатор, який забезпечує низькі обчислювальні витрати та достатню точність у сценаріях, де рішення повинні прийматися швидко та прозоро.

$$p\left[p(y = 1 \mid x) = \sigma(w^{\top x} + b), \quad \sigma(z) = \frac{1}{1+e^{-z}}\right] \tag{2.8}$$

Навчання LR у прикладних задачах зазвичай здійснюється шляхом мінімізації негативної лог-правдоподібності, з можливим додаванням регуляції для підвищення узагальнювальної здатності та стійкості моделі.

Логістична регресія відзначається високою швидкодією як у процесі навчання, так і низьким часом прийняття рішення, що є критичним для SDN-сценаріїв з обмеженим часом реагування контролера або обмеженими ресурсами вузла. Додатковою перевагою є базова інтерпретованість, знак та модуль коефіцієнтів

w дозволяють оцінити внесок ключових ознак у зміну шансів класу, що полегшує аналіз причин прийняття рішення. У контексті SDN це робить логістичну

регресію доцільною для «сигнальних» бінарних рішень (наприклад, первинний скринінг станів потоку чи клієнта, груба класифікація трафіку, визначення ризику перевищення квоти), а також як нижню межу для порівняння з більш складними моделями.

Основним недоліком LR є припущення лінійної роздільності у просторі ознак. Якщо мережеві показники та їх взаємодії мають виражену нелінійну природу (комбінації затримки, джитера, втрат, інтенсивності запитів, складні залежності між контекстними та історичними ознаками), то LR може поступатися ансамблевим методам на деревах рішень або MLP-моделям.

Random Forest - ансамбль дерев рішень, де кожне дерево навчається на вибірці даних, а на кожному розбитті розглядається випадкова підмножина ознак. Рішення ансамблю агрегується [94] (наприклад, голосуванням у класифікації). Кожне дерево $T_m(\cdot)$ (2.9) навчається на вибірці навчальних даних, а під час побудови дерева на кожному кроці розглядається лише випадково відібраний піднабір ознак, що зменшує кореляцію між деревами та підвищує узагальнювальну здатність ансамблю. Для задач SDN, де використовуються табличні ознаки потоків і станів мережі, RF є практичним варіантом завдяки здатності моделювати нелінійні залежності та взаємодії ознак без явного конструювання складних перетворень.

$$[\hat{y} = mode\{T_m(x)\}_{m=1}^{M}] \tag{2.9}$$

RF добре працює на табличних описах потоків SDN і є стійким до «шуму» та часткової нестабільності ознак, що типово для мережевих вимірювань (лічильники, статистики пакетів, часові агрегації). Завдяки ансамблевій природі RF часто демонструє високу точність у задачах класифікації мережевих подій (виявлення аномалій чи атак), оскільки здатен ефективно відтворювати нелінійні межі рішень і комбіновані патерни у просторі ознак. Додатковою практичною перевагою є можливість оцінювання важливості ознак, що корисно для аналізу

які саме параметри потоку чи стану мережі найбільше впливають на рішення моделі.

Порівняно з легковаговими моделями (зокрема LR), Random Forest зазвичай має більшу латентність часу прийняття рішення, оскільки для прогнозу необхідно обчислити вихід багатьох дерев. При значній кількості дерев або великій глибині це може бути критичним у сценаріях SDN з жорсткими вимогами до часу реакції та в умовах обмежених обчислювальних ресурсів (edge-контролер чи окремий вузол). Крім того, хоча окреме дерево є інтерпретованим, великий ансамбль глибоких дерев знижує інтерпретацію правил у порівнянні з простішими моделями, практична інтерпретація часто зводиться до аналізу важливості ознак та локальних пояснень.

Gradient Boosting - це ансамблевий метод, у якому підсумкова модель будується як адитивна композиція слабких базових учнів (найчастіше дерев рішень), що додаються послідовно [95]. Він формує ансамбль так, щоб кожна наступна базова модель компенсувала помилки попередньої, тобто реалізує ітеративне зменшення функції втрат шляхом «градієнтного кроку» у функціональному просторі (2.10).

$$[F_M(x) = \sum_{m=1}^{M} \nu f_m(x)] \tag{2.10}$$

де (f_m) базовий учень (зазвичай дерево рішень), (ν) коефіцієнт навчання

Побудова кожного наступного базового учня виконується так, щоб мінімізувати задану функцію втрат відносно поточного ансамблю, тобто «донавчити» модель у напрямку зменшення помилки.

Gradient Boosting належить до найбільш ефективних підходів для табличних ознак, що є типовим форматом даних у SDN (ознаки потоків, агреговані статистики, часові індикатори перевантаження/ризику). Практична цінність для SDN полягає у здатності моделювати складні нелінійні межі рішень, відтворювати комбіновані патерни в даних та забезпечувати високу якість класифікації/прогнозування при коректному налаштуванні [96]. Це робить GB

доцільним кандидатом для задач інтелектуального управління в мережі (наприклад, класифікація станів, ризику, прогнозування перевищення лімітів, підтримка рішень щодо квот та пріоритезації).

Недоліком GB є підвищена чутливість до налаштувань, коефіцієнт навчання, кількість ітерацій, складність базових дерев та параметри підвибірки істотно впливають на результат. Без регулювання або контролю збіжності бустинг має схильність до перенавчання. Також навчання може бути повільнішим, ніж у деяких альтернативних ансамблів, оскільки базові моделі додаються послідовно (на відміну від паралельної побудови дерев у Random Forest).

Support Vector Machine - є маржинальним класифікатором, який будує розділювальну гіперплощину з максимальною маржею між класами [97]. Інтуїтивно SVM намагається знайти таку межу, яка не лише розділяє класи, але й забезпечує найбільший «запас» відстані до найближчих навчальних точок, що підвищує стійкість до шуму та покращує узагальнення (2.11).

$$\left[\min_{w,b,\xi} \frac{1}{2}|w|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{s.t.} \quad y_i(w^{\top}\phi(x_i) + b) \geq 1 - \xi_i,\ \xi_i \geq 0\right] \quad (2.11)$$

де w та b задають розділяючу гіперплощину, ξ_i - штрафи за порушення маржі, C - параметр жорсткості м’якої маржі $\phi(\cdot)$ - відображення у простір ознак.

SVM доцільно включати як «маржинальну» базову модель для табличних ознак потоків SDN, оскільки вона часто демонструє конкурентну якість у задачах класифікації трафіку чи станів мережі [98], особливо у випадках, коли межа між класами добре описується гіперплощиною (лінійний SVM) або помірно нелінійною поверхнею (RBF-ядро). Практично SVM часто використовується як еталонний метод поряд із LR та ансамблями, що дозволяє об’єктивніше порівнювати приріст якості від складніших моделей.

SVM з нелінійними ядрами може бути обчислювально дорогим у навчанні на великих вибірках (n), що стає критичним, якщо передбачається часте перенавчання моделі або обробка великих обсягів поточних мережевих потоків [97]. Крім того, SVM чутливий до масштабу ознак, тому потребує обов'язкої попередньої обробки (нормалізації/стандартизації), інакше вибір параметрів C та γ може бути некоректним або нестабільним. У ресурсно-обмежених SDN-сценаріях ці фактори впливають на практичну придатність SVM у режимі близькому до реального часу.

Multilayer Perceptron - це нейронна мережа прямого поширення, що складається з одного або кількох прихованих шарів та вихідного шару [99]. Мережа виконує послідовні афінні перетворення вхідного вектора ознак із застосуванням нелінійних функцій активації, а навчання здійснюється шляхом мінімізації функції втрат методом градієнтного спуску з використанням алгоритму зворотного поширення похибки (2.12).

$$\left[h^{(l)} = g\left(W^{(l)}h^{(l-1)} + b^{(l)}\right)\right] \tag{2.12}$$

де $h^{(l-1)}$- вектор активацій попереднього шару (для $l = 1$ це вхід x), $W^{(l)}$ та $b^{(l)}$ - матриця ваг і вектор зміщень шару l, $g(\cdot)$ - нелінійна функція активації

MLP забезпечує високу гнучкість у моделюванні нелінійних залежностей в ознаках мережевих потоків і може бути застосована [51] як у задачах класифікації (наприклад, аномалії, класи трафіку, ризик перевищення ліміту), так і у задачах регресії (наприклад, прогноз навантаження або трафіку) за коректної постановки виходу та функції втрат. Для SDN-сценаріїв, де важливо враховувати взаємодії між показниками та історичний контекст, нейромережевий підхід може виступати сильним предиктором за умови достатньої якості даних та правильного налаштування моделі.

Практичне застосування MLP потребує нормалізації, масштабування ознак і є чутливим до вибору архітектури (кількість шарів/нейронів) та параметрів навчання. Порівняно з лінійними моделями й деревами, MLP є менш прозорою з точки зору інтерпретації (без додаткових методів пояснення).

Узагальнимо моделі за практичними критеріями для SDN в умовах обмежених ресурсів (таблиця 2.1): інтерпретованість, витрати на навчання, затримка прийняття рішення, стійкість до дисбалансу класів і вимоги до конструювання ознак

Таблиця 2.1

Порівняльна характеристика ML-моделей за критеріями придатності для задач SDN в умовах обмежених ресурсів

Модель	Інтерпретованість	Час навчання	Затримка прийняття рішення	Стійкість до дисбалансу	Потреба у конструюванні ознак
Logistic Regression	Висока;	Низький;	Дуже низька;	Середня;	Висока (масштабування, взаємодії);
Random Forest	Середня;	Середній;	Середня;	Середньо-висока;	Низька/середня;
Gradient Boosting	Середня;	Середній/ високий;	Середня;	Середньо-висока (ваги/пороги);	Низька/середня;
SVM	Низька/середня;	Середній/ високий;	Низька/серед-ня;	Середня;	Середня/висока (масштабування);
MLP	Низька;	Середній/ високий;	Низька/серед-ня;	Середня (ваги/ресемплінг);	Середня (нормалізація);

2.3. Платформи та реалізації SDN-контролерів

В SDN контролер виступає централізованою точкою прийняття рішень, що на основі телеметрії формує та застосовує політики примусу в мережі [100], а ефективність інтеграції ШІ в цикл керування визначається насамперед якістю і доступністю даних і затримками та пропускною здатністю площини керування. В зв’язці з OpenFlow комутатор взаємодіє з контролером через захищений канал, а контролер керує таблицями потоків [101]. У розподілених контролерах додається ще й міжконтролерна синхронізація стану [102] (узгоджені сховища, консенсус), що прямо впливає на роботу застосунків та вартість інтеграції обчислювально важких ШІ-модулів. Саме тому вибір конкретної платформи SDN-контролера визначає не лише «можливість» інтеграції ШІ, а й її «вартість»

у сенсі накладних витрат, складності підтримки та гарантованої продуктивності [103].

2.3.1. Класи SDN-контролерів

SDN-контролери доцільно класифікувати за архітектурними, функціональними та експлуатаційними ознаками, що дає змогу виокремити їхні основні класи з погляду придатності до інтеграції алгоритмів штучного інтелекту в умовах обмежених ресурсів.

- Легковагі контролери дослідницько-прототипного класу орієнтовані на швидкий цикл «ідея - реалізація - експеримент». У цьому сегменті Ryu [104] виступає компонентно-орієнтованим SDN-фреймворком, надаючи бібліотеки та модулі з визначеними API для розробки мережевих застосунків і підтримує протоколи керування, зокрема OpenFlow. Його модель програмування є подійною, OpenFlow-повідомлення декодуються та доставляються застосункам як події, обробники яких реєструються декларативно. Аналогічно, POX [105] реалізує подійний підхід, де взаємодія з компонентами мережі організована через publish/subscribe-механізм.

Для задач ШІ на edge та в умовах обмежених ресурсів такі контролери привабливі низьким бар’єром інтеграції й швидким прототипуванням завдяки Python-екосистемі. Водночас подійна модель має критичне обмеження - блокування обробника затримує оброблення наступних подій, а отже погіршує реактивність площини керування (Ryu визначає як ризик). Тому ресурсомістке виконання моделей всередині контролера або зовнішні синхронні виклики можуть спричинити деградацію площини керування. Для крайових-сценаріїв це обґрунтовує архітектурну ізоляцію ШІ у вигляді зовнішнього сервісу та уникнення вбудовування ШІ логіки в і так критичний шлях оброблення OpenFlow-подій.

- Середній клас контролерів орієнтований на ширше коло експериментів і прикладних задач, поєднуючи модульність із відносно стабільними північними інтерфейсами. Floodlight зокрема, просуває REST як практичний механізм

створення «північних» застосунків, що споживають функції контролера [106]. Історично він пов’язаний з ідеями Beacon, у роботах орієнованих на нього [107] підкреслюються рішення для підвищення продуктивності розробки, динамічного керування застосунками та продуктивності контролера, а Floodlight розглядається як платформа цієї технічної лінії Java-контролерів.

З позицій інтеграції ШІ Floodlight є компромісом. Перевагою є північний REST-інтерфейс, який спрощує зв’язування з зовнішнім ШІ-сервісом (мовонезалежна взаємодія), а Java-екосистема дозволяє реалізовувати продуктивні модулі всередині контролера. Водночас вбудований ШІ у Java-контролері ускладнює розгортання й діагностику на edge-вузлах (JVM, залежності, профілювання), тоді як зовнішній Python-сервіс потребує асинхронної інтеграції, щоб не посилювати тиск на площину керування. Тому Floodlight доцільний для сценаріїв «між прототипом і готовим продуктом», а придатність для жорстко ресурсних вузлів визначається тим, наскільки телеметрію та обчислення можна винести з критичного шляху.

- Розподілені контролери проєктуються для середовищ із вимогами до високої доступності, масштабованості та кластерних розгортань. ONOS позиціонується як розподілена SDN-платформа для великих мереж і провайдерських сценаріїв, у класичних роботах він описується як «open, distributed SDN OS» [108]. Практично кластеризація ONOS спирається на узгоджені розподілені сценарії, у типовій конфігурації очікується Atomix-кластер з Raft-групами для консенсусу. Поряд із REST в екосистемі ONOS застосовуються й підходи на базі gRPC (HTTP/2, ефективніше кодування), що зменшує накладні витрати взаємодії.

Для інтеграції ШІ ONOS привабливий у масштабі реальної мережі, але «вартість» інтеграції зміщується в площину розподілених систем - міжвузлова узгодженість, мережеві затримки між контролерами та ефекти лідерства/консенсусу [109] впливають на час реакції прикладних клієнтів. В умовах edge це означає, що ONOS доцільний лише тоді, коли сценарій об’єктивно

вимагає кластерної відмовостійкості та наявні ресурси для підтримки узгодженого сховища й сервісних компонентів.

- Модельно-керовані контролери, такі як OpenDaylight є прикладом платформи з глибокою модульністю та модельно-керованим підходом [110]. Центральним механізмом виступає MD-SAL, використання YANG-моделей дозволяє формалізувати структури даних, RPC та нотифікації, а згенеровані API поширюються через RESTCONF (і суміжні північні інтерфейси). Типові північні інтерфейси ODL (RESTCONF/NETCONF) забезпечують стандартизований доступ для прикладних застосунків та інтеграційних сервісів.

З погляду інтеграції ШІ, ODL привабливий саме як production-клас - його інтерфейси спрощують уніфікований доступ до стану мережі й політик, роблять взаємодію передбачуваною та зменшують «ручну» підтримку API при еволюції моделі даних. Водночас ці переваги мають експлуатаційну «ціну» - у ресурсо-обмежених середовищах зростають накладні витрати на підтримку контейнера, модулів, сховищ стану, а у кластері на реплікацію, узгодженість і координоване відновлення [111]. Тому, OpenDaylight є природним кандидатом для масштабованих розгортань, але не завжди оптимальним для edge-прототипування, де критично мінімізувати споживання CPU/RAM при простому життєвому циклі компонентів.

- Компактні контролери (операційно-орієнтований клас) займають нішу між прототипами та готовими до інтеграцій платформами, вони розраховані на експлуатаційні мережі де потрібна передбачуваність конфігурації, спостережуваність і DevOps-інтеграція. Faucet є типовим прикладом такого підходу як «компактний» OpenFlow-контролер, що переносить контрольні функції у кероване серверне ПЗ [112]. Проєкт декларує підтримку OpenFlow 1.3, експорт статистики і метрик системи через Prometheus/Grafana, а також Ryu як базовий фреймворк.

Для задач ШІ на краю мережі, Faucet [113] цікавий тим, що спостережуваність закладена «з коробки» - компонент Gauge-моніторингу періодично опитує комутатор і робить дані доступними для Prometheus,

спрощуючи формування ознак із телеметрії. Водночас, частота опитування має узгоджуватися з ресурсним бюджетом площини керування, інакше зростають накладні витрати навантаження.

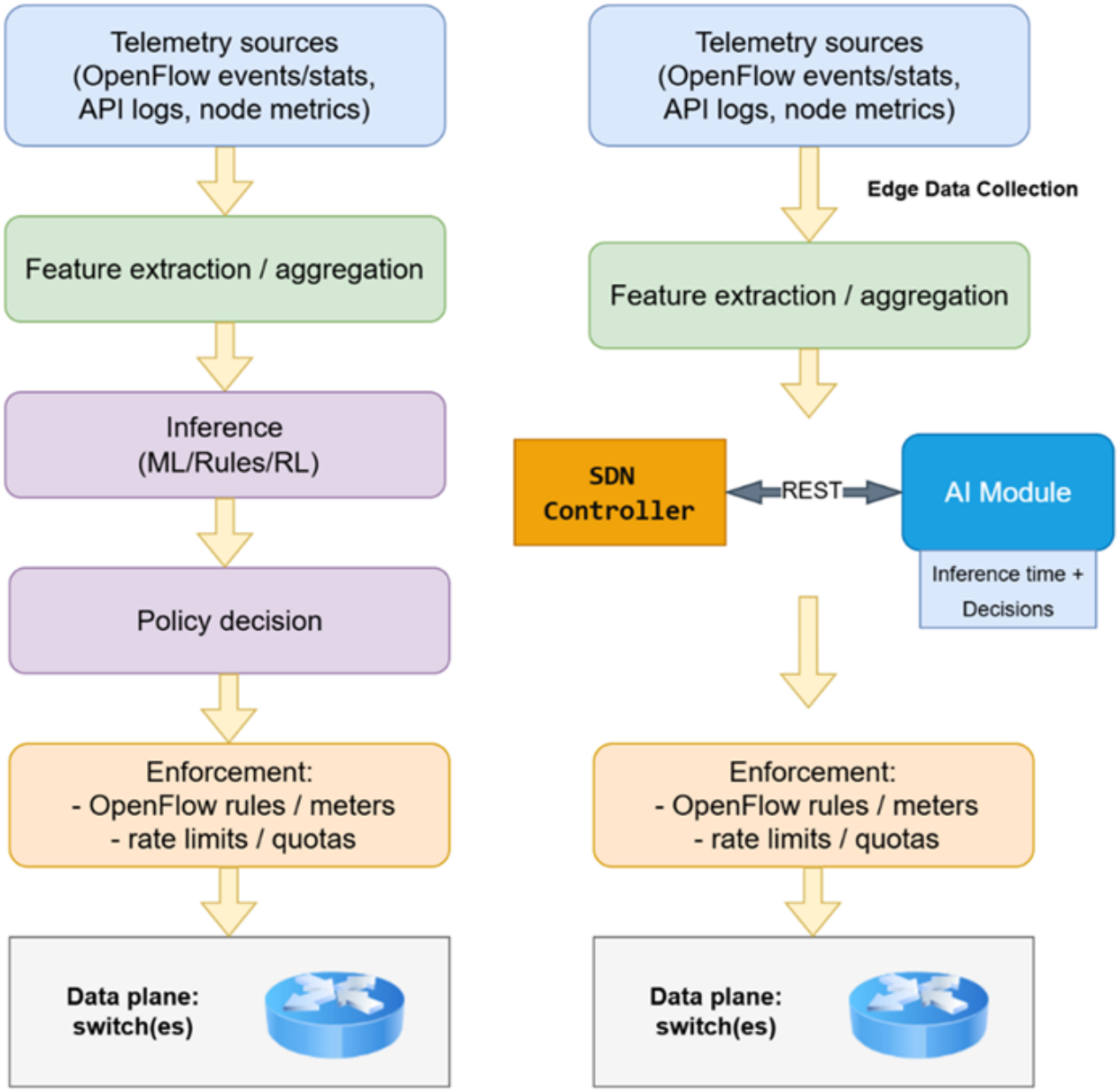
2.3.2. Аналіз критеріїв інтеграції алгоритмів ШІ в SDN

Цикл керування «телеметрія-ознаки-прийняття рішення-політика-дія» у SDN реалізується через конкретні механізми контролера та протоколу керування комутатором. OpenFlow задає модель, у якій контролер керує пересиланням через таблиці потоків і взаємодіє з комутаторами по контрольному каналу, а також надає статистику портів/потоків як базу телеметрії. Звідси випливає причинно-наслідковий зв'язок - чим «щільніше» ШІ інтегрований у контролер, тим більше його обчислювальні витрати та можливі блокування знижують реактивність встановлення правил, і тим вищою стає системна ціна помилки або перевантаження.

- Модель програмування та подійність. У Ryu подієва-орієнтована обробка організована через черги та потоки, офіційна документація застерігає [114], що блокування обробника події призводить до зупинки подальшого оброблення подій у межах застосунку. Це вимагає архітектурної дисципліни - ШІ-виклики мають бути асинхронними або винесеними за межі подійного контуру. POX так само базується на publish/subscribe-механізмі і трактує повідомлення від комутатора як події, що означає аналогічний ризик блокувань. Java-контролери (Floodlight, ONOS, ODL) зазвичай мають складніше багатопотокове середовище та сервісні шини, однак це не знімає ключову вимогу - «важкі» обчислення не повинні потрапляти в критичний шлях реактивного встановлення політик, особливо в edge-середовищах з обмеженим CPU/RAM.

- Накладні витрати та тиск на площину керування. Вбудований ШІ в контролері (рис. 2.3.а) посилює конкуренцію за CPU/RAM між: а) обробкою мережевих подій, б) збором/агрегацією телеметрії та в) часом прийняття рішення. Для Ryu це особливо критично через ризик блокування подійного обробника, що безпосередньо знижує реактивність керування. Винесення ШІ в

окремий сервіс (рис. 2.3.б) зменшує цей ризик завдяки ізоляції процесів і спрощує контроль накладних витрат на крайовому вузлі, але додає затримку «виклик - відповідь», яку потрібно закладати в бюджет реактивності. Роботи, що розглядають мікросервісну декомпозицію контролера або його функцій [28], підкреслює комунікаційні наслідки міжкомпонентної взаємодії та розглядає важливість компромісу при виборі підходу інтеграції керуючих алгоритмів.



а) б)

Рис. 2.3. Інтеграції ШІ-модуля з SDN-контролером: а) вбудований, б) зовнішній сервіс

- Телеметрія та формування ознак у SDN для задач штучного інтелекту здебільшого ґрунтуються на даних OpenFlow-статистики та подіях площини керування (зокрема, packet-in, port, status тощо). Для ресурсно-обмежених вузлів критичною є частота збору даних і тиск на площину керування, часті запити статистики збільшують контрольний трафік і обчислювальні витрати на обробку. На концептуальному рівні це підкреслено в роботі Kandoo [103], де показано, що обмеження накладних витрат контрольної площини є умовою масштабованості SDN. Практичний компроміс демонструє Faucet, де статистика виводиться у зовнішню систему спостереження.

- Практичність для відтворюваних експериментів. Для дисертаційних експериментів критичною є відтворюваність, швидке розгортання топологій, керування OpenFlow-потоками та повторюваність умов навантаження. Mininet забезпечує саме це, реалізуючи швидке прототипування SDN на одному хості з реальним Linux-стеком і механізмами віртуалізації ОС [28]. На рівні площини даних є Open vSwitch [115] як програмний віртуальний комутатор. У такій зв’язці легковагові контролери (Ryu/POX) зазвичай мають найменшу інженерну вартість для запуску повторюваних сценаріїв (простий стек, швидкий цикл змін та перезапусків).

Натомість модельно-керовані платформи (ONOS/OpenDaylight) доцільні тоді, коли експериментальна мета прямо включає дослідження кластерних ефектів та масштабування.

Наведене узагальнене порівняння (таблиця 2.2) показує, що для експериментів у ресурсно-обмежених умовах найбільш раціональними є легковагові та компактні контролери з простим стеком технологій і зовнішньою інтеграцією ШІ, тоді як кластерні та модельно-керовані платформи доцільніші для масштабування, але мають вищу «вартість» телеметрії та експлуатації.

Таблиця 2.2

Порівняння SDN-контролерів за архітектурою, інтерфейсами та ресурсними вимогами.

Контролер	Мова/архітектура	Northbound API	Телеметрія	Ресурсні вимоги
Ryu	Python; компонентний, подійно-орієнтований фреймворк;	Вбудовується через застосунки; типово REST через WSGI/власні кінцеві точки;	OpenFlow події + статистика (окремі запити застосунком);	Низькі
POX	Python; подійно-орієнтований (publish/subscribe);	Залежить від сценарію; здебільшого власні інтеграції;	Події з площини даних та базова статистика через повідомлення;	Низькі
Floodlight	Java; модульний OpenFlow-контролер;	REST API як рекомендований NB шлях;	Модулі контролера, flow/port stats та події (залежить від реалізації модулів);	Середні
ONOS	Java; розподілений контролер;	REST, gRPC; застосунки в екосистемі ONOS;	Події/статистика, узгодження стану в кластері;	Високі (особливо в кластері)
OpenDaylight	Java; Модельно-керований підхід;	RESTCONF/NETCONF (генерується з моделей);	статистика через плагіни; інтеграція з мережевими моделями;	Високі
Faucet	Python (на базі Ryu-екосистеми); компактний, OpenFlow-контролер з конфігураційним підходом;	Немає «єдиного» NB; інтеграція через інструменти оператора;	Експорт метрик (Prometheus), інтеграція з Grafana;	Низькі–середні

Підсумовуючи, в нашому випадку вибір SDN-контролера визначає «вартість» інтеграції ШІ через подійну архітектуру, northbound-інтерфейси, механізми телеметрії та здатність підтримувати керувальний цикл із прогнозованою латентністю. У нашому дослідженні зв’язка Ryu+Mininet+Open-vSwitch є методично обґрунтованою [116], Mininet забезпечує відтворюваність SDN-сценаріїв на одному хості, а OVS є зрілим програмним комутатором з підтримкою OpenFlow і добре вивченою архітектурою. Винесення AI у окремий

Python-сервіс відповідає вимозі обмежених ресурсів, оскільки ізолює обчислювально ресурсомістке виконання моделі від контуру подій контролера та зменшує ризик блокування обробників, на що прямо звертає увагу документація Ryu [118]. Потребує уваги додаткова затримка міжкомпонентної взаємодії між контролером і ШІ-сервісом, які потрібно закласти в бюджет реактивності площини керування.

Висновки до розділу 2

У розділі сформовано методичну основу побудови інтелектуальних SDN-мереж, орієнтованих на роботу в умовах обмежених ресурсів, з акцентом на коректне оцінювання якості AI-керування, вибір придатних моделей машинного навчання та вимоги до платформ контролерів і сценаріїв інтеграції ШІ.

Оцінювання рішень інтеграції алгоритмів ШІ в SDN є багатовимірною задачею та не може зводитися лише до класичних класифікаційних показників. Для реалістичної оцінки потрібне поєднання метрик якості ML-алгоритмів (Accuracy, F1, Precision, Recall тощо) з метриками розподілу ресурсів в мережі, оскільки алгоритм має не тільки коректно приймати рішення, а й забезпечувати відсутність упередженості при виділенні обмеженого ресурсу для гарантування QoS у сценаріях, де необхідно одночасно підтримувати ефективність (мінімізація затримок, максимальне використання пропускної здатності) та справедливість доступу. Вибір ML-підходів для задач SDN в умовах обмежених ресурсів має здійснюватися за принципом «достатньої складності» - забезпечення якості керування, але при цьому бути сумісною з ресурсним бюджетом площини керування та вимогами для роботи в режимі реального часу.

Вибір SDN-контролера визначає «вартість» інтеграції ШІ через подійну архітектуру, північні інтерфейси, механізми телеметрії та здатність підтримувати цикл керування із прогнозованою латентністю. У дослідженні зв’язка Ryu - Mininet – Open-vSwitch є методично обґрунтованою: Mininet забезпечує відтворюваність SDN-сценаріїв на одному хості, а OVS є програмним комутатором із підтримкою OpenFlow та добре вивченою архітектурою.

Винесення логіки інтелектуального керування в окремий сервіс ізолює обчислювально затратну логіку від контуру подій контролера та зменшує ризик блокування обробників, водночас, потребує уваги додаткова затримка міжкомпонентної взаємодії між контролером і ШІ-сервісом, яку необхідно закладати в бюджет реактивності площини керування.

Результати розділу 2 формують підґрунтя для подальшої розробки й експериментальної перевірки методів AI-керування в SDN, визначено вимоги до метрик оцінювання, обґрунтовано критерії вибору ML-моделей з урахуванням затримок і ресурсоємності, а також систематизовано практичні вимоги до інтеграції ШІ в контур керування SDN у ресурсно-обмежених сценаріях.

РОЗДІЛ 3. МОДЕЛЮВАННЯ ТА ДОСЛІДЖЕННЯ CLOUD-AI-SDN ФРЕЙМВОРКУ В УМОВАХ ОБМЕЖЕНИХ РЕСУРСІВ

3.1. Архітектура запропонованого Cloud-AI-SDN фреймворку

Архітектурне рішення Cloud-AI-SDN фреймворку, запропоноване в межах даного дослідження, розглядається як структурний каркас для реалізації методів інтелектуального керування програмно-конфігурованою мережею в умовах обмежених ресурсів. На практиці, такі обмеження можуть проявляються на кількох рівнях: у площині даних (вузькі місця каналів передачі даних, черги, втрати), у керуючій площині (час обробки подій, навантаження SDN-контролера, безпека, відновлення після збоїв), а також на рівні зовнішніх ресурсів і сервісів (квоти/ліміти API, сховища, служби автентифікації, хмарні обчислювальні ліміти). Тому архітектура проєктується так, щоб підтримувати проактивне прийняття рішень, контроль накладних витрат моніторингу та керованість доступу до обмежених ресурсів у єдиному контурі керування.

Концептуально фреймворк (рис. 3.1.) організовано у вигляді трирівневої SDN структури (інфраструктурний рівень, рівень керування та хмарний прикладний рівень), де між рівнями визначені канали обміну даними та командами управління. Перенесення частини функцій у хмарний рівень використовується не як «замінник» SDN, а як спосіб розвантажити контролер, забезпечити масштабування та децентралізоване управління мережею.

- Інфраструктурний рівень є базою фреймворку та включає розподілені крайові домени (наприклад, сегмент філії, факультету або виробничої зони), у яких функціонують кінцеві вузли (клієнти, пристрої, сервери сервісів або інші елементи даних) і мережеві елементи доступу. Залежно від сценарію розгортання домен може бути організований як: а) «прямий» сегмент без проміжних агрегаторів або б) сегмент із виділеною точкою агрегації (gateway, edge hub, IoT controller тощо). У другому випадку хаб використовується як інфраструктурний компонент, який спрощує приєднання гетерогенних підсистем та взаємодію з механізмами обміну подій (рис. 3.1). Крайовий хаб може виконувати роль агрегатора підключень і точки інтеграції з механізмом обміну подій - публікувати

стан підключених вузлів, приймати керуючі події та застосовувати локальні дії (наприклад, зміна режиму роботи вузла, тимчасове обмеження доступу, виконання політики реагування на інцидент). Водночас у доменах без хаба ці функції можуть бути реалізовані безпосередньо кінцевими вузлами, які взаємодіють із верхніми рівнями через доступні інтерфейси та визначені політики.

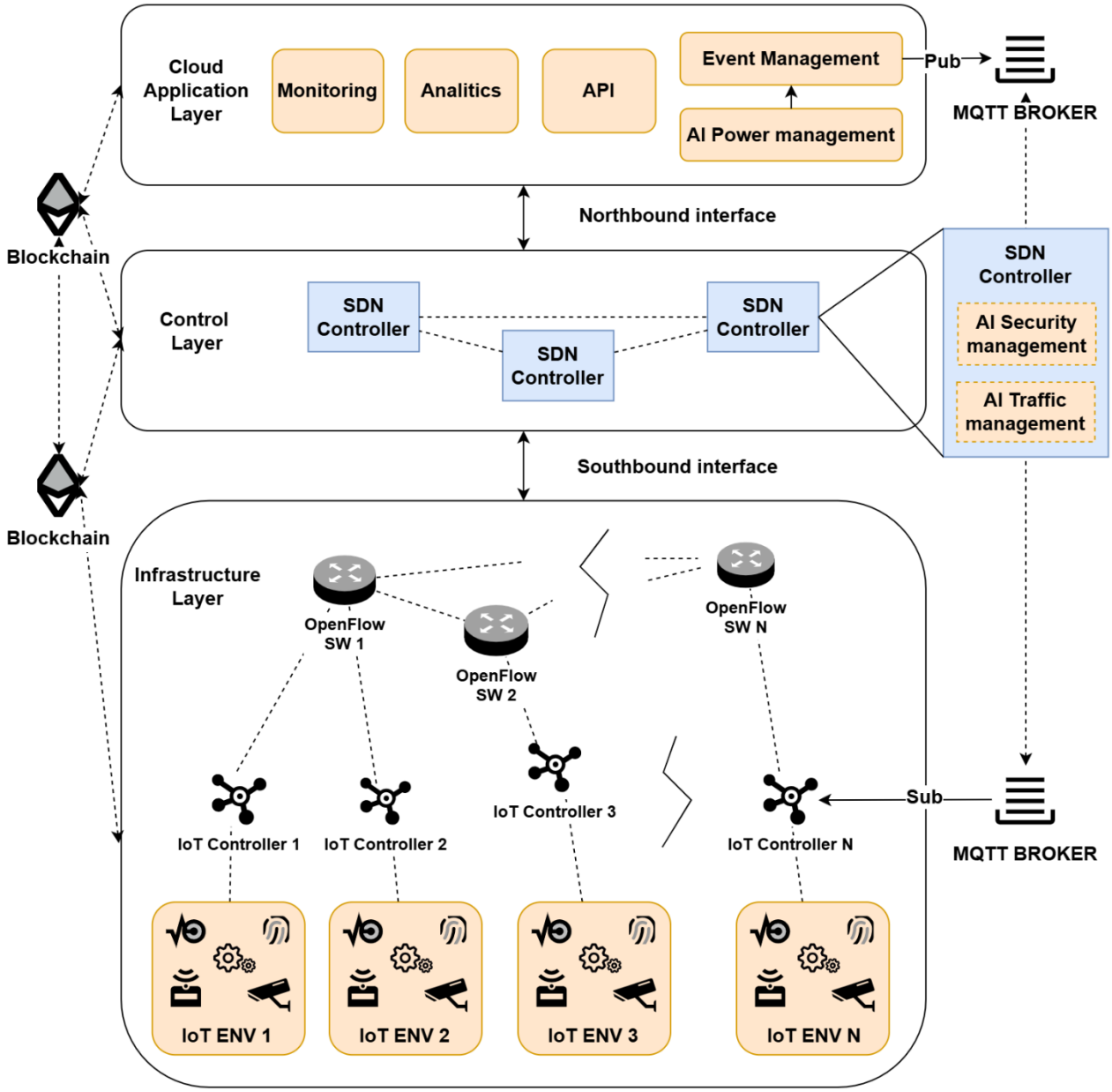


Рис. 3.1. Архітектура Cloud-AI-SDN фреймворку (три рівні та крос-рівнева взаємодія).

Для маршрутизації та інтеграції з SDN-мережею крайові домени підключаються до програмованих комутаторів (OpenFlow SDN switch), які виконують правила потоків, встановлені SDN-контролером. Взаємодія інфраструктурного рівня з рівнем керування здійснюється через SDN Southbound Interface, що забезпечує централізоване керування мережевими ресурсами на рівні інфраструктури. У результаті інфраструктурний рівень забезпечує збір первинних даних про стан підключень і трафіку та реалізацію керуючих дій «на місці», тоді як логіка прийняття рішень і глобальна координація залишаються на рівні керування та у хмарному прикладному рівні (рис. 3.1).

- Рівень керування забезпечує централізоване керування трафіком і політиками, а також інтеграцію інтелектуальної аналітики в цикл прийняття рішень. Функціонально він включає один або декілька SDN-контролерів, які мають глобальний огляд топології/стану та застосовують керуючі дії до комутаторів через південні інтерфейси. У трирівневій архітектурі саме цей рівень виконує роль «операційного ядра» SDN і забезпечує узгодженість політик між доменами, підтримує актуальність мережевого стану та реалізує швидкі реакції на зміну навантаження або інциденти.

Особливістю запропонованої архітектури є наявність інтегрованої ШІ-підсистеми у контурі SDN-контролера, яка працює з телеметрією та журналами подій, перетворюючи їх на керуючі рішення. На відміну від «ручного» або суто керування орієнтованого на правила, ШІ-підсистема дозволяє реалізувати проактивні та адаптивні реакції на основі класифікації станів та прогнозування тенденцій, що є критичним для мереж із динамічним трафіком і ресурсними обмеженнями. Узагальнено, AI-підсистема рівня керування виконує такі функції:

- аналітика мережевого трафіку та виявлення аномальних/потенційно шкідливих патернів на основі моделей машинного навчання, що дозволяє формувати реакції у вигляді блокування, ізоляції або зміни маршрутів;

- підтримка рішень щодо оптимізації маршрутизації та балансування навантаження з урахуванням прогнозу стану мережі та QoS-вимог;

- адаптивне керування доступом до лімітованих ресурсів (наприклад, сервісів із квотами, зовнішніх API, сховищ або елементів хмарної інфраструктури), що реалізується через політики контролера та правила потоків.

Для підвищення цілісності керування та захисту від компрометації керуючих дій у фреймворку передбачено використання blockchain-механізмів на рівні керування - фіксація транзакцій змін політик, збереження ключових конфігурацій, можливість відновлення до останнього коректного стану, а також підтримка безпечної автентифікації при взаємодії з верхнім (хмарним) рівнем. У практичному сенсі blockchain-компонента розглядається як контур автентифікації та аудиту, який підвищує спостережуваність керування і забезпечує відтворюваність станів керування у випадку збоїв або підозрілих змін.

- Хмарний прикладний рівень виконує роль центра управління та аналітики, і забезпечує моніторинг, агрегування даних із нижніх рівнів, інтерфейси адміністратора та можливість масштабування обчислень без ускладнення крайової інфраструктури. Вибір хмарної реалізації обґрунтовується гнучкістю оновлення чи заміни аналітичних компонентів і масштабуванням обчислювальних ресурсів без необхідності розширення локального обладнання. У контексті дисертаційного дослідження цей рівень важливий тим, що дозволяє винести ресурсоємні операції (довготривала аналітика, централізоване накопичення даних, навчання/перенавчання моделей) у середовище з керованим масштабуванням, зменшуючи навантаження на контролер та вузли на краю мережі (рис. 3.1).

На цьому рівні доцільно виділити такі функціональні складові. По-перше, модуль моніторингу та аналітики, що формує візуалізацію стану мережі, статистику використання ресурсів та підтримує виявлення аномалій на рівні оператора. По-друге, зовнішнє SDN Network REST API, яке надає прикладним системам і операторським інструментам уніфікований доступ до функцій SDN-мережі. Це API може використовуватися у двох режимах. У базовому режимі воно забезпечує моніторинг стану мережі - отримання топології, статистики портів/лінків, інформації про потоки, події та ключові метрики продуктивності.

У розширеному (керуючому) режимі через це API можливо реалізувати приклад керування мережею (SDN застосунки) - задавати або змінювати політики, ініціювати конфігураційні операції, запускати сценарії реакції на події, а також застосовувати правила/обмеження, пов'язані з використанням ресурсів. По-третє, оптимізаційні механізми на основі ШІ, які можуть формувати керуючі команди на основі телеметрії та публікувати їх у чергу подій (MQTT), на яку підписані хаби/контролери на краю мережі, що виконують відповідні дії. Такий підхід забезпечує гнучкий зворотний зв'язок і дозволяє реалізувати узгоджену політику управління ресурсами у мультидоменному середовищі (рис. 3.1).

Взаємодія з рівнем керування здійснюється через Northbound Interface, який передає інформацію про поточний стан трафіку, статус хабів та загальне навантаження. На хмарному рівні blockchain може використовуватися як механізм авторизації доступу до брокера подій та перевірки керуючих команд з метою запобігання їх підробленню. У результаті хмарний рівень доповнює SDN-керування інструментами керованої масштабованості, централізованого аналізу та координації команд.

У підсумку, запропонована трирівнева архітектура Cloud-AI-SDN фреймворку забезпечує: 1) керованість та програмованість мережі на рівні SDN-контролера, 2) інтеграцію інтелектуальної аналітики в цикл прийняття рішень без жорсткої прив'язки до конкретного домену кінцевих вузлів, 3) масштабованість та спостережуваність через хмарні сервіси, а також захист і аудит керуючих впливів через поєднання черг подій та blockchain-механізмів.

3.2. Структура експериментального стенда та взаємодія його компонентів

Експериментальний стенд створено для верифікації працездатності запропонованого Cloud-AI-SDN фреймворку в контрольованих умовах та для відтворюваного моделювання сценаріїв доступу до обмежених ресурсів в SDN, із залученням ШІ-модуля прийняття рішень, API для зовнішнього керування та IoT-подій/телеметрії. На відміну від спрощених SDN-тестів, у даному стенді

акцент зроблено на розділенні функцій за площинами та наявності трьох типів інтерфейсів взаємодії: Southbound (OpenFlow 1.3), Northbound (REST) і каналу подій IoT (MQTT), що в сукупності формує наближену до практики модель гібридної AI-SDN-IoT екосистеми.

Архітектурна організація стенда відображає класичну SDN-парадигму (три площини), розширену компонентами IoT та моніторингу (рис. 3.2).

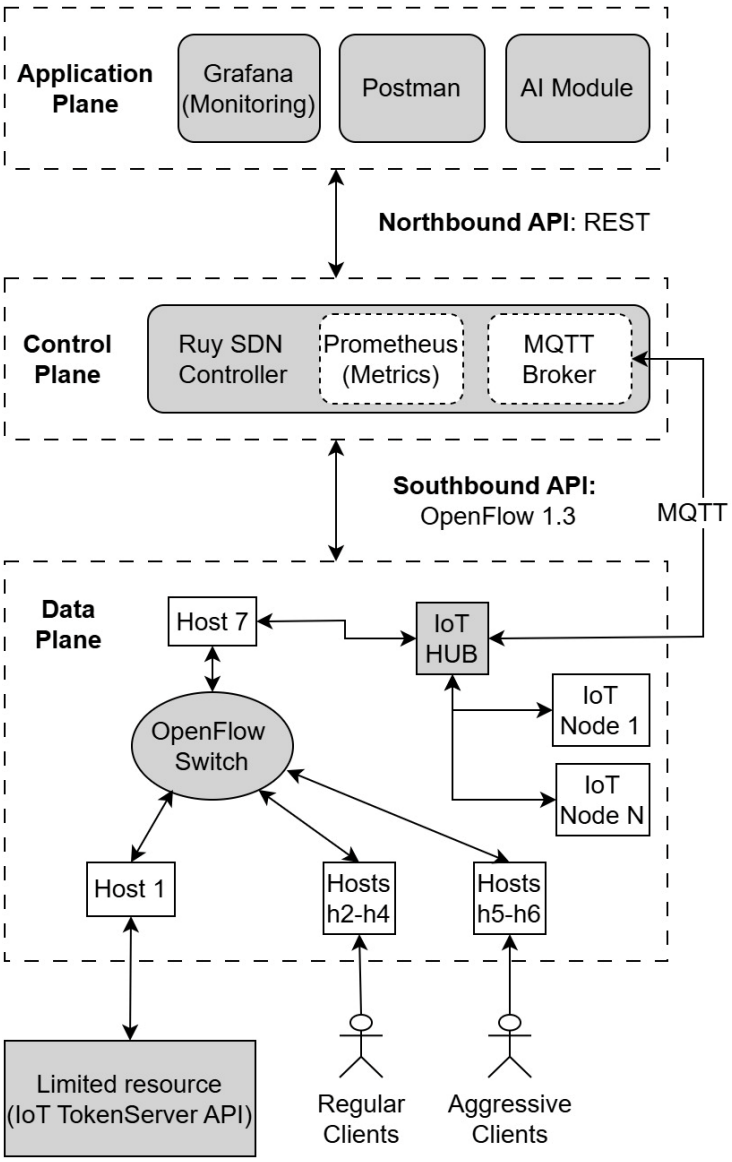


Рис. 3.2. Архітектура експериментального стенду

- Площина даних (Data Plane) реалізує емульовану мережеву інфраструктуру - комутатор Open-vSwitch (OVS) та хости Mininet (h1 - h7), а також логічний IoT-сегмент (IoT Hub та емуляція під'єднаних пристроїв IoT Nodes). Саме в цій площині генерується користувацький трафік до обмеженого ресурсу і формується реальне навантаження на систему.

- Площина керування (Control Plane) містить SDN-контролер Ryu з підтримкою OpenFlow 1.3, модулі збору статистики/метрик та інфраструктуру подій IoT (MQTT broker). Вона забезпечує централізований контроль потоків, збір телеметрії та надає зовнішній інтерфейс керування (Northbound API).

- Площина застосунків (Application Plane) включає інструменти експериментального керування та спостереження (Postman, Grafana), а також ШІ-модуль, який аналізує ознаки/метрики, класифікує поведінку клієнтів і ініціює політики доступу (квоти/обмеження) до ресурсу.

У таблиці 3.1 наведено компоненти експериментального стенда та їх функції відповідно до практичної конфігурації, реалізованої на Linux-платформі з використанням Mininet-емуляції та Python-оточення.

Таблиця 3.1

Компоненти стенда та їх функції

Компонент	**Опис**
Ubuntu 22.04 LTS	Linux-середовище для розгортання компонентів.
Mininet 2.3 (емулятор мережі SDN)	Формує топологію з OVS-комутатором і наборами хостів (h1..h7), дозволяє відтворювати сценарії навантаження (нормальні/гіперактивні клієнти) та ізолювати експеримент.
Open vSwitch (OVS)	Програмний OpenFlow-комутатор, який виконує правила потоків, встановлені контролером, і забезпечує пересилання трафіку між хостами.
Ryu SDN Controller (OpenFlow 1.3)	Центральний компонент керування. Збирає статистику потоків/портів, підтримує керування правилами, та розширюється прикладним модулем/додатком, що забезпечує Northbound REST API для зовнішніх сервісів.
AI module (Python + scikit-learn)	Реалізує модель керованого навчання для класифікації поведінки клієнтів та прогнозування ризику перевищення квоти/перевантаження ресурсу. На основі телеметрії від контролера формує рішення щодо корекції квот або застосування політик доступу.

MQTT broker (Eclipse Mosquitto)	Контур подій IoT. Забезпечує публікацію телеметрії та доставку команд керування до IoT Hub/IoT Nodes, а також підтвердження виконання (feedback) у разі застосування режимів енергозбереження/активації пристроїв.
IoT Hub (на хості h7)	Агрегує IoT-вузли, керує їхнім станом і взаємодіє з брокером MQTT. У рамках експерименту може відтворювати сценарії «on/off/sleep/awake» як реакцію на політики керування та дефіцит ресурсів.
Limited resource: IoT TokenServer API (на хості h1)	Сервіс-ресурс із квотою запитів (req/min), який виступає обмеженим зовнішнім ресурсом, доступ до якого має бути справедливо розподілений між клієнтами.
Prometheus + Grafana	Стек моніторингу. Prometheus збирає метрики з експортера/endpoint-ів контролера або допоміжних сервісів; Grafana візуалізує часові ряди, стани та ефекти політик ШІ.
Postman	Інструмент тестування Northbound REST API (ручне керування сценаріями, валідація відповідей, відтворення типових запитів керування/моніторингу).
Скрипти генерації трафіку (Python)	Реалізація профілів поведінки клієнтів, що дозволяє відтворювати контрольовані перевантаження та перевіряти стійкість механізму квот/обмежень.
Git	Система контролю версій. Відстеження змін конфігурацій і коду експериментів та відокремлення сценаріїв (окремі гілки/теги для різних топологій, моделей ШІ та режимів навантаження).

Взаємодія компонентів відбувається чотирма основними каналами, кожен з яких відповідає окремому класу функцій:

1) Southbound API: OpenFlow 1.3 (площини керування та даних). Контролер Ryu керує OVS-комутатором через OpenFlow - встановлює/видаляє правила потоків, отримує події (packet-in) та статистику (flow/port stats). Саме цей канал забезпечує програмованість площини даних та реалізацію політик доступу на рівні мережевих правил.

2) Northbound API: REST (рівень застосунків і даних). REST-інтерфейс використовується двома класами споживачів:

- інструменти керування/тестування (Postman) - ініціація експериментальних дій (запуск/зупинка політик, налаштування порогів, запит станів);

- ШІ-модуль - отримання агрегованої телеметрії/ознак та передачі рішень (оновлення квот/правил доступу) у контрольну площину.

3) MQTT (для взаємодії на рівні застосунків та IoT-сегменту). MQTT використовується як шина подій для обміну між IoT-хабом та IoT-пристроями, через яку передаються дані телеметрії стану, а також команди керування живленням і режимами функціонування пристроїв. Важливо, що цей контур дозволяє розширити сценарій «обмеженого ресурсу» не лише до API-квот, а й до керованого управління ресурсами IoT-сегмента (sleep/awake) під час перевантаження або вичерпання квот.

4) Моніторинг (площина застосунків та Prometheus + Grafana). Статистика та метрики (мережеві і прикладні) експортуються у форматі, придатному для збору Prometheus, після чого в Grafana формуються дашборди для аналізу - динаміка запитів, частка клієнтів, факти перевищення квот, реакції політик та побічні ефекти (затримки, відмови, блокування тощо).

Запропонована структура стенда забезпечує одночасну перевірку коректності SDN-керування потоками через OpenFlow, інтеграції AI-модуля через Northbound REST API, наявності контуру подій IoT через MQTT та відтворюваного моніторингу через Prometheus/Grafana. Це створює методично обґрунтовану базу для подальшої роботи, зокрема для розширеного моделювання SDN.

3.3. Візуалізація та верифікація функціонування стенду

Експериментальний стенд розгорнуто на Linux-платформі, що забезпечує відтворюваність досліджень у незалежних умовах і узгоджується з типовими практиками проведення експериментів у програмно-конфігурованих мережах. Базовими компонентами стенда є емулятор мережі Mininet та SDN-контролер Ryu, які обрано з огляду на їх відкритість, широке використання в наукових

роботах [117, 88], а також через можливість розширення власними модулями (генерація трафіку, збір статистики, ШІ-агент, емуляція обмеженого ресурсу). Запуск SDN-топології та ініціалізація вузлів у середовищі Mininet наведено на рис. 3.3.

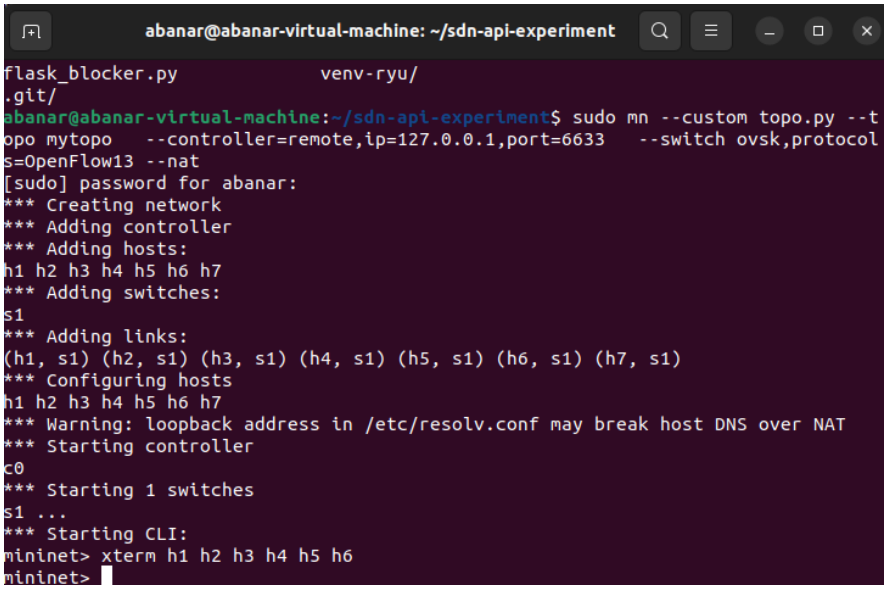


Рис. 3.3. Вікно терміналу Mininet із запуском експериментальної SDN-топології стенда

Подальша верифікація функціонування стенда виконується послідовно, від перевірки запуску мережевої топології до підтвердження коректної роботи площини керування та прикладних сервісів і моніторингу. Зокрема, у середовищі Mininet контролюється доступність вузлів, зв’язок між вузлами та коректність ініціалізації віртуального комутатора/лінків (рис. 3.3). На рівні контролера Ryu (рис. 3.4) підтверджується встановлення керуючих з’єднань, обробка подій та коректний збір статистики, необхідної для подальшого аналізу та роботи ШІ-модуля.

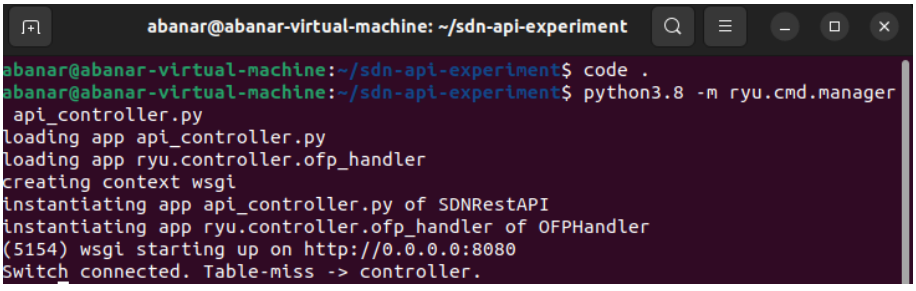


Рис. 3.4. Вікно терміналу SDN-контролера Ryu під час роботи в експериментальному середовищі

Для оперативної перевірки коректної роботи стенду та спостереження за динамікою навантаження, використано ланцюг збору й візуалізації метрик на основі Prometheus та Grafana. Prometheus виконує збір метрик мережі та прикладних компонентів, тоді як Grafana (рис. 3.5) забезпечує інтерактивні панелі моніторингу, що дозволяють візуально підтверджувати наявність трафіку, активність клієнтів та зміну інтенсивності запитів у часі, а також частку успішних/неуспішних звернень до сервісів.

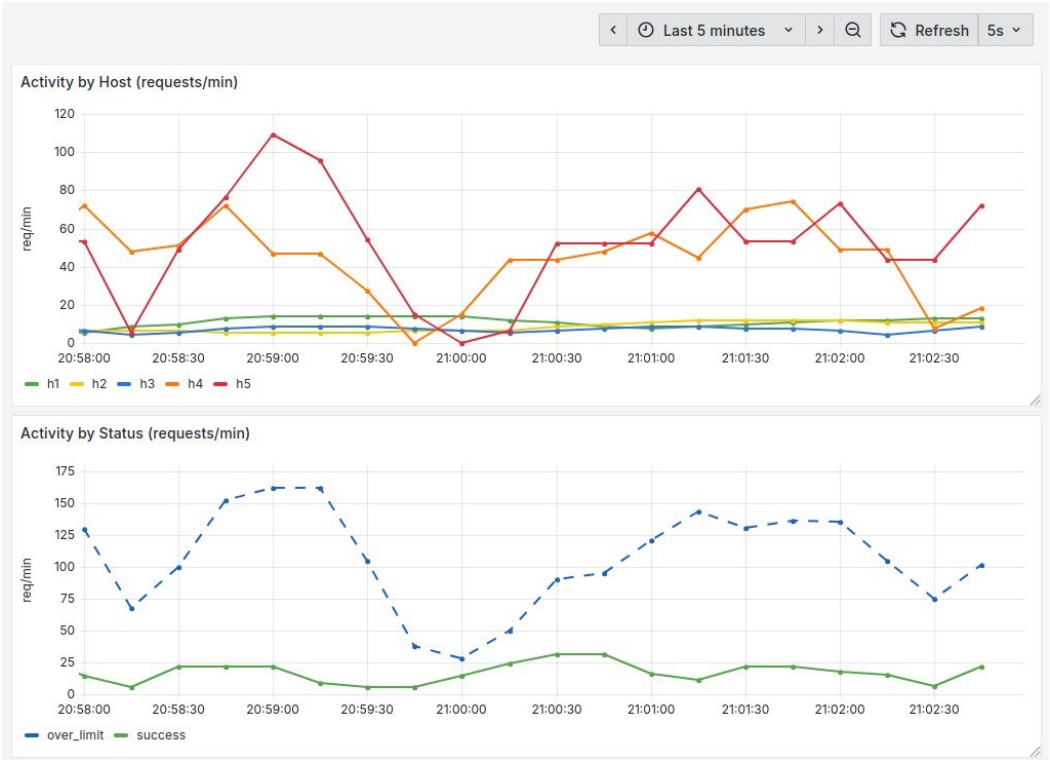


Рис. 3.5. Інтерактивні графіки моніторингу мережевої активності в Grafana

Перевірку доступності та коректності зовнішнього керуючого інтерфейсу виконано за допомогою тестування REST API використовуючи Postman (рис. 3.6). Це дозволяє підтвердити працездатність ендпоінтів, керування/моніторингу, валідувати структуру відповіді сервісу та коди стану, а також відтворювати запити в межах серій експериментів.

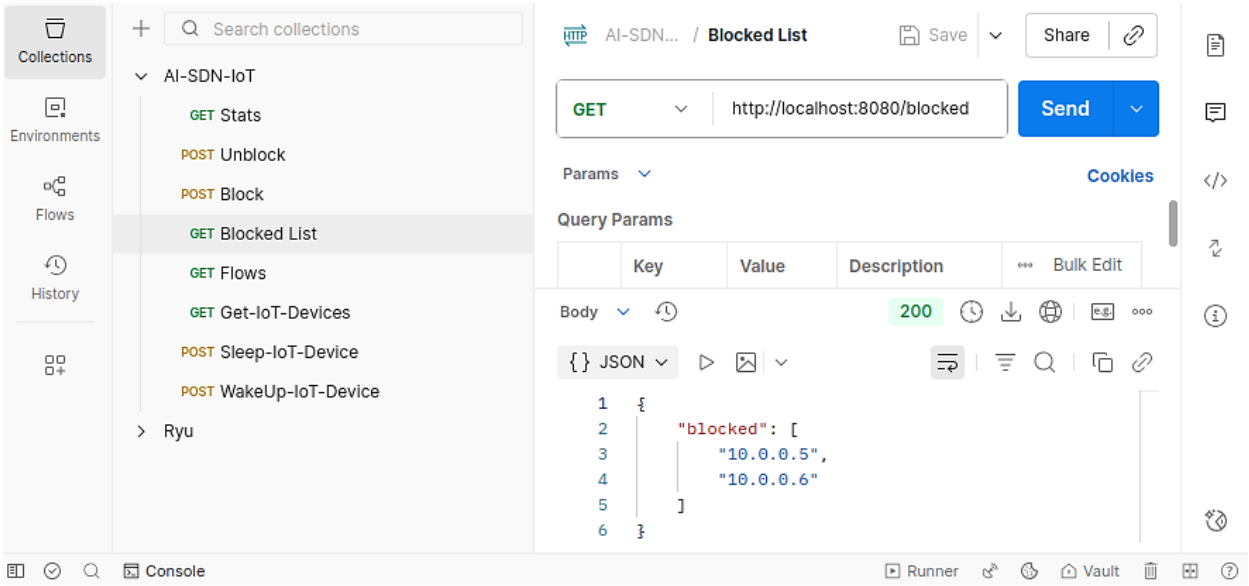


Рис. 3.6. Вікно формування запиту до мережевого REST API в Postman

Для моделювання площини IoT у стенді використано MQTT-брокер та IoT-компоненти, що забезпечують типову для IoT взаємодію (публікація/підписка, передавання телеметрії) і дозволяють емулювати присутність IoT-інфраструктури як джерела та споживача трафіку. Стан IoT-компонентів у межах експериментального стенда показано на рис. 3.7.

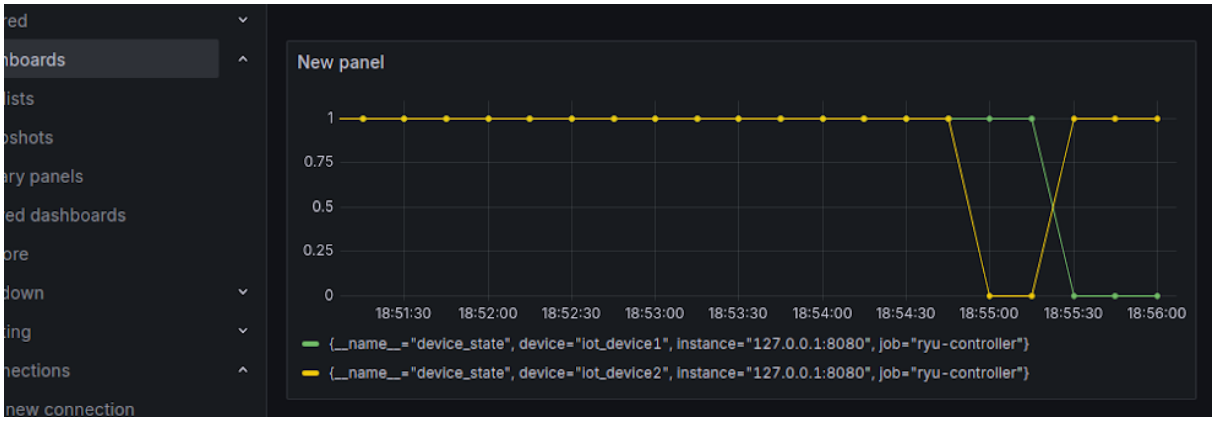


Рис. 3.7. Статус підключених IoT-компонентів у процесі експерименту

Додатково реалізовано сценарій конкурентного доступу до обмеженого ресурсу, який моделюється як сервіс із лімітом запитів за одиницю часу. У такому сценарії частина клієнтів генерує рівномірне навантаження, тоді як інша частина створює пікові запити, що призводить до перевищення ліміту та появи відмов/деградації сервісу, формуючи умови дефіциту ресурсу та необхідність адаптивного керування доступом.

Практична перевірка працездатності стенда виконується через демонстрацію повного циклу роботи - запущена мережева топологія, активні клієнти, обмежений ресурс приймає/відхиляє запити, модуль ШІ аналізує телеметрію та формує керуючі рішення. Загальний вигляд запущеного експериментального стенду наведено на рис. 3.8. Процес виконання керуючої логіки ШІ-модуля (обробка даних, формування рішень у відповідь на зміну навантаження) наведено на рис. 3.9.

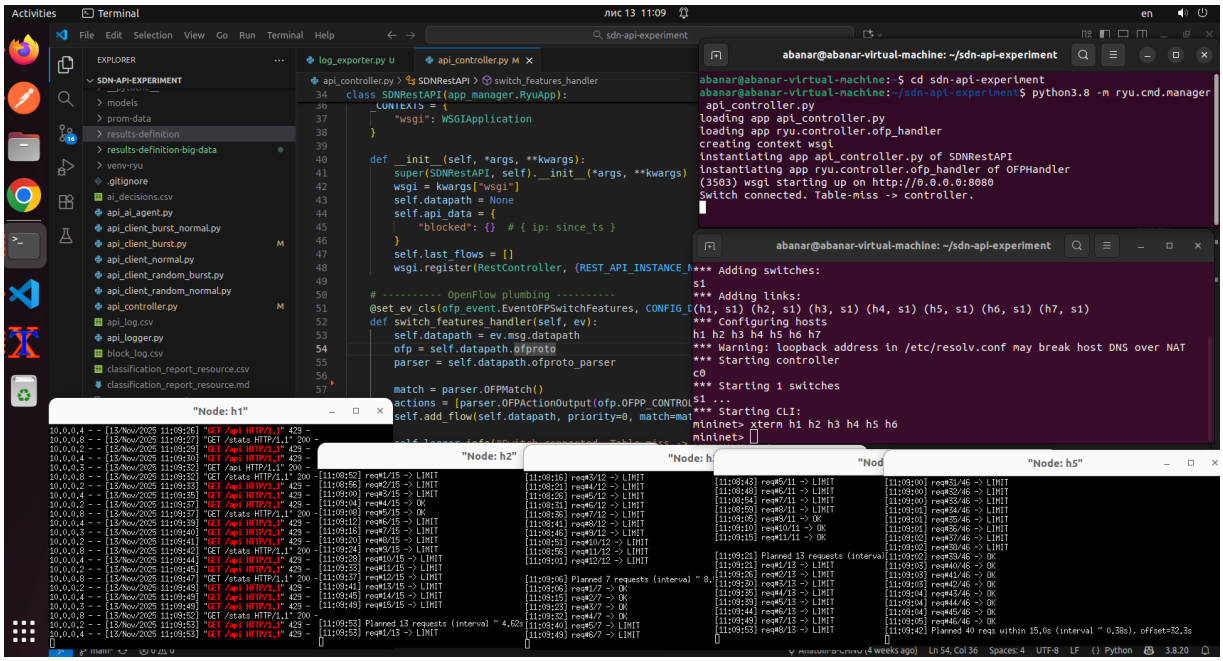


Рис. 3.8. Загальний вигляд запущеного експериментального сетапу стенда (активні компоненти та процеси)

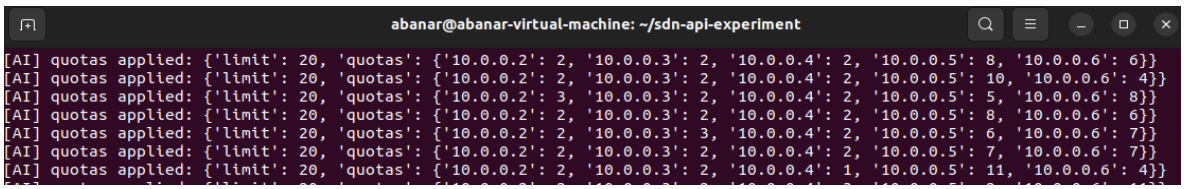
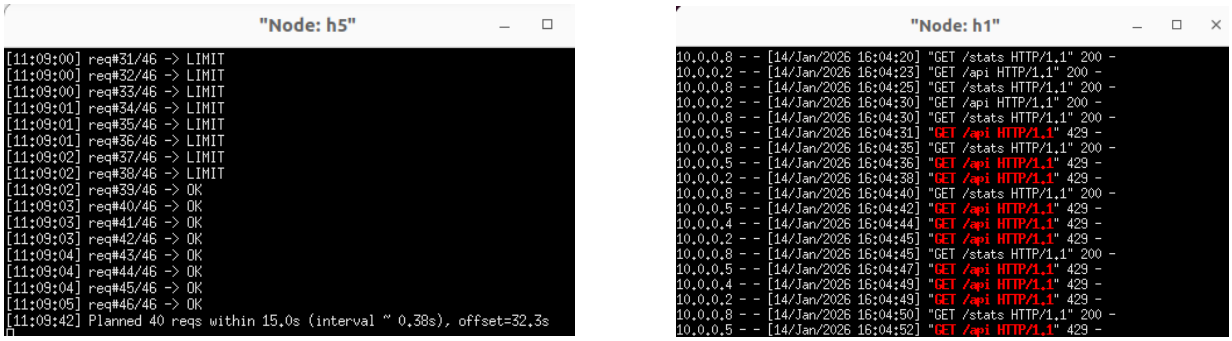


Рис. 3.9. Термінал роботи ШІ-модуля під час формування керуючих рішень

На рис. 3.10.а наведено приклад терміналу клієнтського хоста під час генерації тестового трафіку, що ілюструє факт формування запитів у межах заданого сценарію навантаження. На рис. 3.10.б показано журнал роботи вузла з підключеним обмеженим ресурсом, де відображено обробку запитів і реакцію сервісу на перевищення ліміту.



а) б)

Рис. 3.10. Вікно терміналу: а) клієнтського хоста під час генерації тестового трафіку, б) хоста з під’єднаним обмеженим ресурсом

Загалом, стенд успішно розгорнуто та запущено - мережева топологія, контролер, моніторинг і прикладні модулі працюють узгоджено й стабільно. Отримана візуалізація підтверджує коректну роботу всіх компонентів, що забезпечує можливість проведення експериментів, фіксації вимірювань та відтворення результатів.

3.4. Методика та етапи експериментального дослідження інтелектуального керування доступом до обмеженого ресурсу

Метою експериментального дослідження є застосування алгоритмів штучного інтелекту для динамічного розподілу доступу до обмеженого ресурсу та експериментальна перевірка функціональності й придатності запропонованого фреймворку для задач цього класу. У межах дослідження IoT-систему емульовано підключеним IoT TokenServer API із обмеженням кількості запитів за хвилину, тоді як клієнтські хости можуть інтерпретуватися як IoT-пристрої або IoT-хаби/шлюзи, що генерують запити до зазначеного сервісу. Кроки проведення експерименту наведено на рис. 3.11, які відображають логіку виконання трьох основних етапів: 1) підготовка даних, 2) підготовка ШІ-модуля, 3) тестування та оцінювання результатів.

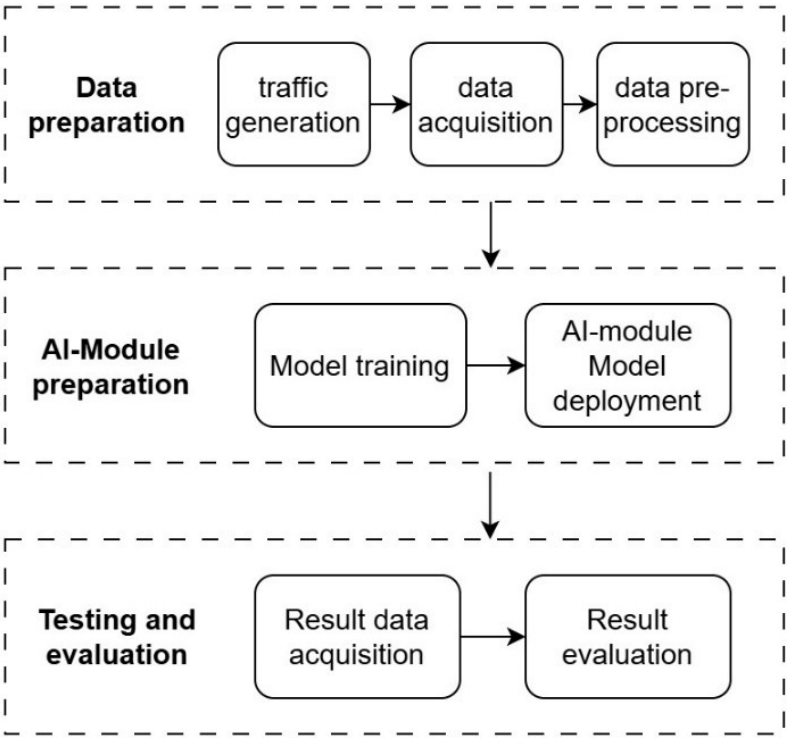


Рис. 3.11. Етапи експериментального дослідження

3.4.1. Формування експериментальних даних та попередня обробка мережевої телеметрії

На етапі підготовки даних здійснюється генерація клієнтського трафіку та збір мережевої телеметрії, необхідної для побудови навчальної вибірки ШІ-модуля. Трафік генерується з боку клієнтських вузлів, які імітують різні профілі поведінки (регулярний рівномірний потік запитів та інтенсивні навантаження), що дозволяє відтворити конкуренцію за ресурс у спільному середовищі. Обмежений ресурс представляється сервісом IoT TokenServer API із фіксованою глобальною квотою на кількість запитів на хвилину, а часовий крок керування та агрегації даних задається вікном $\Delta t = 60$ с, що відповідає періодичному циклу контролю доступу.

Збір даних організовано так, щоб одночасно фіксувати як локальну активність кожного клієнта, так і глобальний стан навантаження на ресурс у відповідному часовому вікні. Для цього в ході експерименту формується агрегований журнал «training_log.csv», у якому кожний рядок відповідає

конкретному клієнту в межах поточного 60-секундного інтервалу та містить ознаки, необхідні для подальшого навчання моделі (таблиця 3.2).

Таблиця 3.2

Опис полів журналу training_log.csv для навчання моделі

Поле	Опис
ts	Часова мітка (мс, Unix epoch) зафіксованого запису;
src_ip	IP-адреса джерела трафіку (клієнта);
status	HTTP-код відповіді сервера (наприклад, 200, 429);
count_60s	Кількість запитів від клієнта за останню хвилину;
global_limit	Поточний ліміт ресурсу (запитів за хвилину);
total_60s	Загальна кількість усіх запитів до ресурсу за останню хвилину;
share_prev	Частка використання ресурсу клієнтом у попередньому інтервалі;
active_clients	Кількість активних клієнтів;
label_fair_prev	Еталонна мітка , що відображає коректність попереднього розподілу ресурсу;
reset_in_sec	Час (у секундах), що залишився до наступного скидання глобального ліміту;

До таких ознак належать кількість запитів від клієнта за останню хвилину, загальна кількість запитів до ресурсу за той самий інтервал, оцінка частки споживання ресурсу клієнтом, а також допоміжні параметри, що характеризують режим роботи ресурсу і підготовку до наступного циклу (наприклад, час до скидання глобального ліміту). Окремо фіксується результат звернень до ресурсу на рівні HTTP-відповідей (успішне обслуговування або відмова через перевищення ліміту), що дозволяє поєднати параметри навантаження з фактичними наслідками для клієнтів. Таким чином, набір даних відображає як поведінку клієнтів, так і реакцію обмеженого ресурсу в умовах конкуренції, що є необхідним для навчання ШІ-модуля для прогнозування ризику некоректного розподілу доступу та підтримувати адаптивне керування квотами в наступних

етапах методики. Локації формування журналів та приклад структури лог-файлів до агрегації наведено на рис. 3.12.

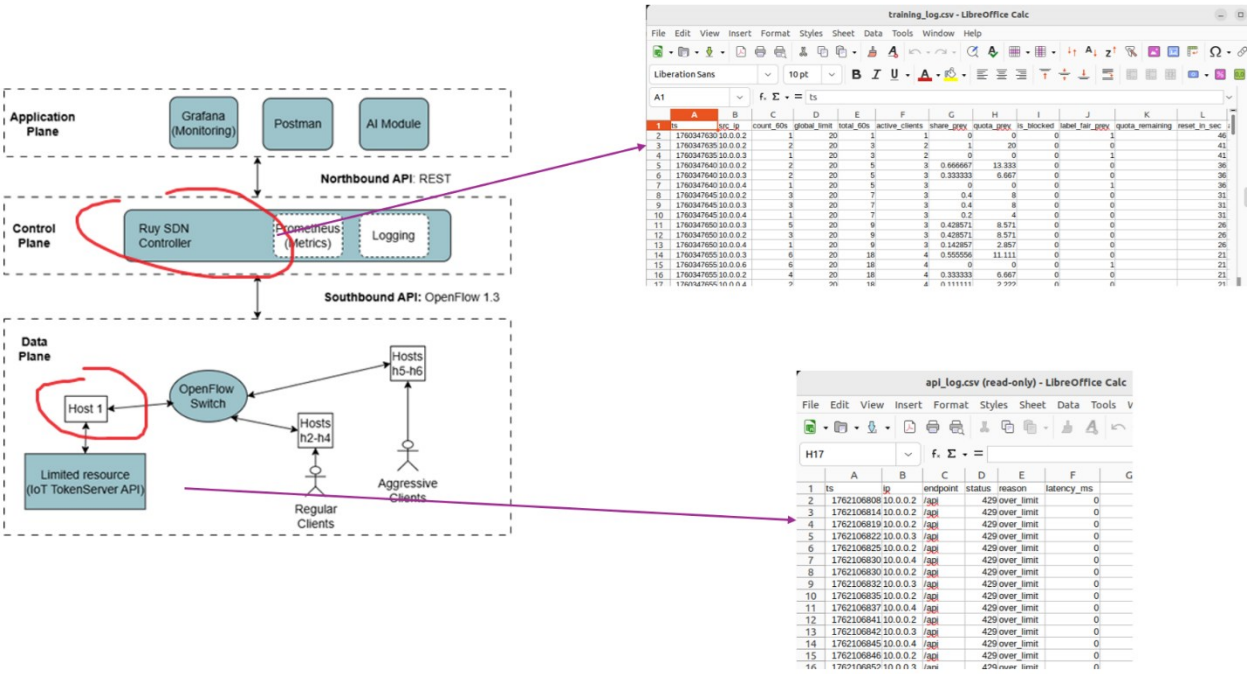


Рис. 3.12. Локації формування журналів експерименту

Попередня обробка даних включає приведення форматів полів до уніфікованого вигляду, контроль пропусків, а також узгодження часових інтервалів агрегації, щоб кожний запис однозначно відповідав конкретному вікну Δt.

3.4.2. Підготовка, навчання та розгортання ШІ-модуля інтелектуального керування доступом

У межах експериментального стенда Cloud-SDN-AI середовища ШІ-модуль реалізує цикл прогнозування перевитрати обмеженого ресурсу та адаптації квот доступу для клієнтів. Операційна логіка побудована як замкнений контур керування (рис. 3.13) і включає: збір телеметрії та логів, формування ознак, оцінювання ризику перевитрати, коригування квот/політик та верифікація ефекту за логами і метриками ефективності. Що дозволяє інтегрувати машинне навчання безпосередньо в цикл прийняття рішень SDN-контролера.

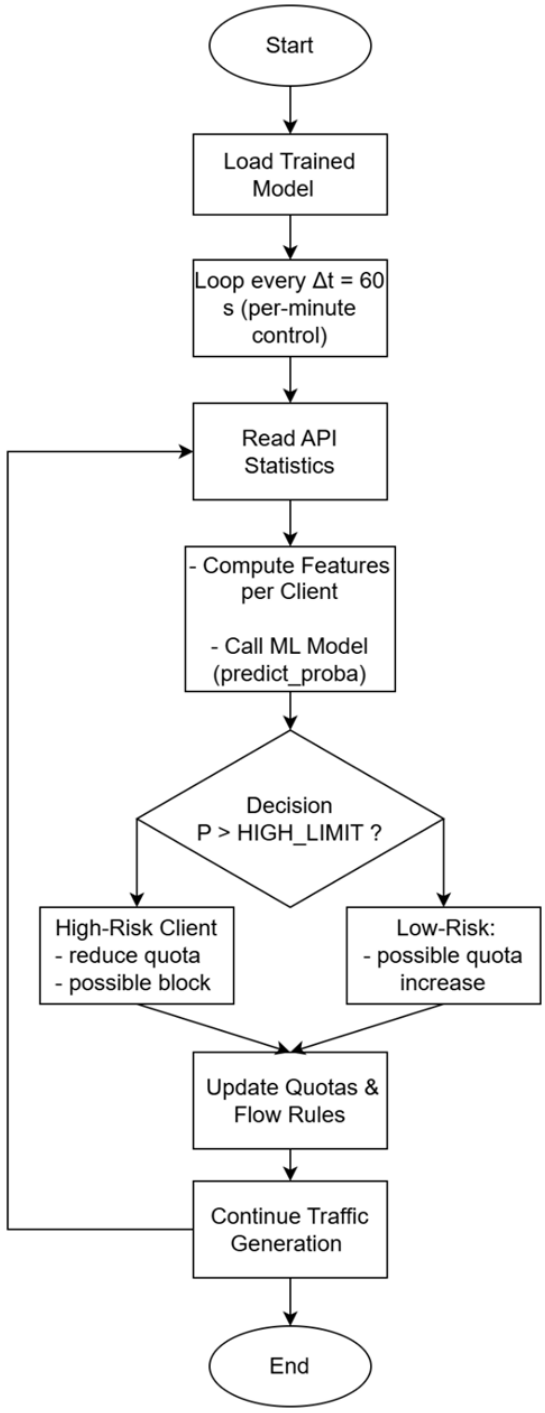


Рис. 3.13. Цикл роботи ШІ-модуля інтелектуального керування доступом

Первинні дані формуються на основі логів (CSV), які генеруються експортером статистики, у процесі роботи стенда він реєструє мережеві характеристики та агрегує їх за контрольними інтервалами. На основі чого формується навчальний набір даних для прогнозування стану перевитрати ресурсу в SDN-середовищі. Для зменшення розмірності та усунення дублікатів,

виконується відбір ознак (3.1), у підсумку зберігаються лише часові та динамічні параметри трафіку:

$$x_i(t) = \begin{Bmatrix} count_60s, global_imit, active_clients, \\ share_prev, status, total_60s \end{Bmatrix} \quad (3.1)$$

де $x_i(t)$ вектор ознак для клієнта i у поточному вікні керування t, $count_60s$ та $total_60s$ - характеристики запитів за останню хвилину, $global_limit$ - глобальне обмеження ресурсу, $active_clients$ - кількість активних клієнтів, $share_prev$ - попередня частка клієнта відповідно до всього трафіку, $status$ - чи був клієнт заблокований.

Для задачі бінарної класифікації стану ресурсу (в межах ресурсу / перевищення) навчається модель логістичної регресії з бібліотеки scikit-learn. Розбиття вибірки: 80% - навчальні дані, 20% - тестові дані. Вибір логістичної регресії обґрунтовується стійкістю на малих наборах даних і мінімальними обчислювальними витратами, що критично для роботи в реальному часі на рівні керування SDN. Після отримання стабільних результатів модель експортується у формат .joblib для подальшої інтеграції в Python-скрипт ШІ-модуля. Додатково проводиться перевірка на нових експериментальних даних повторного прогону трафіку, результати прогнозу зберігаються в CSV для порівняння з фактичними мітками.

Під час ініціалізації ШІ-модуль завантажує навчений класифікатор із файлу .joblib. Далі на кожному інтервалі моніторингу формується вектор $x_i(t)$, який подається в модель для отримання класу:

0 - within_resource (в межах ресурсу, перевитрата не прогнозується)

1- over_resource (перевищення, прогнозується перевитрата).

Модель формує оцінку ризику $p_i(t) \in [0,1]$ - ймовірність того, що клієнт i у наступному часовому вікні перевищить виділену квоту за умови збереження поточної політики доступу. Отримане значення $p_i(t)$ порівнюється з заданим порогом ризику t_{high}, якщо $p_i(t) \geq t_{high}$, активується обмеження (перехід у

режим блокування чи застосування штрафних політик), а якщо $p_i(t) < t_{high}$ - політика доступу зберігається або коригується без жорсткого блокування.

У випадку прогнозу перевитрати, модуль здійснює керування через SDN-контролер, де динамічно додається drop-flow (рис. 3.14) правило для відповідного джерела трафіку, що тимчасово обмежує доступ до ресурсу на заданий інтервал (timeout), до моменту «оновлення» обмеженого ресурсу.

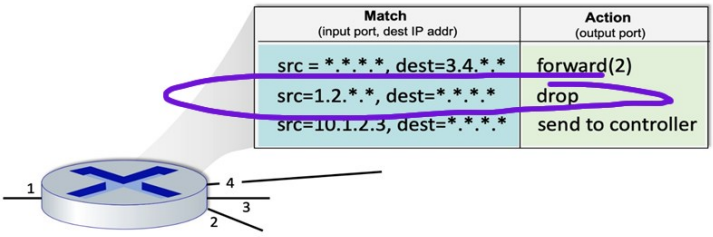


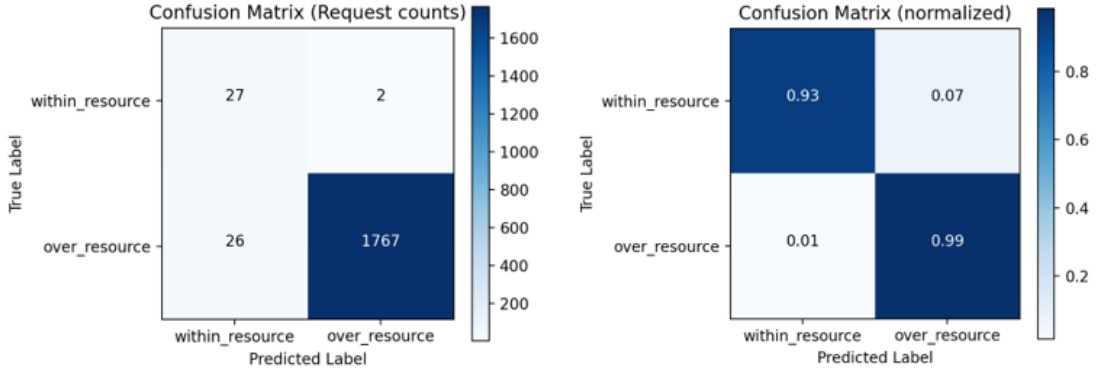
Рис. 3.14. Додавання drop-flow правила в таблицю потоків OpenFlow

Після завершення інтервалу блокування правило автоматично видаляється, що забезпечує адаптивне керування доступом у реальному часі на рівні комутатора OpenFlow. В ході експерименту ШІ-модуль у вигляді Python-скрипта (AI-agent) запускається як окремий процес у середовищі стенда.

3.4.3. Тестування та оцінювання результатів експерименту

Оцінювання ефективності інтеграції ШІ-модуля виконувалося у двох сценаріях - без застосування ШІ-модуля та з активованим ШІ-модулем керування квотами доступу. Для забезпечення порівняння експериментів, лог-файли з боку обмеженого ресурсу та SDN-контролера агрегувалися за часовими вікнами тривалістю 5 хвилин. Для кожного клієнта у межах кожного вікна обчислювалися узагальнені показники - кількість спроб доступу до ресурсу, кількість успішних запитів, частка спроб відносно сумарного трафіку та частка успішних відповідей, а також похідні величини. Отримані агреговані CSV-файли використовувалися для побудови порівняльних графіків і для розрахунку метрик продуктивності та справедливості доступу.

Окремою складовою оцінювання було підтвердження придатності навченої моделі (у цьому дослідженні - логістична регресія) до роботи на рівні керування SDN. Для моделі прогнозування ризику перевищення доступу до ресурсу застосовувалися стандартні метрики якості класифікації: Accuracy (частка правильних рішень), Precision (точність для позитивного класу), Recall (повнота для позитивного класу) та F1-score (гармонійне середнє Precision і Recall). Розрахунки виконувалися засобами бібліотеки scikit-learn із використанням функцій classification_report() та confusion_matrix(). Результати наведено у вигляді матриць помилок в абсолютних значеннях (рис. 3.15.а) та у нормалізованій формі (рис. 3.15.б), що дозволяє наочно оцінити частки правильних/хибних рішень для кожного класу.



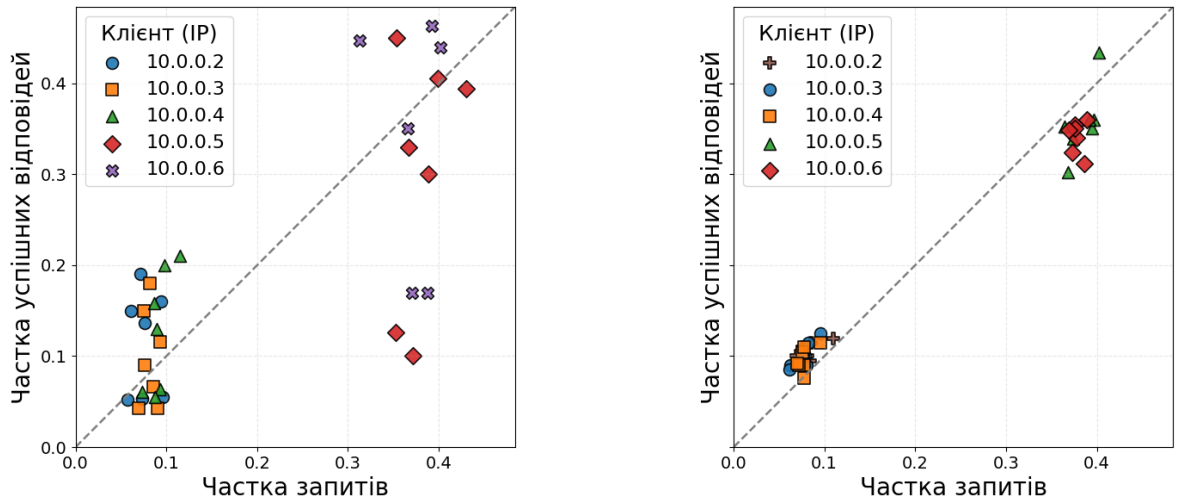
а) б)

Рис. 3.15. Матриця плутанини моделі логістичної регресії: а) абсолютні значення, 2) нормалізована форма

Інтерпретація нормалізованої матриці помилок підтверджує, що модель коректно розпізнає переважну більшість випадків: приблизно 93% класу «within_resource» та близько 99% класу «over_resource», що є достатнім рівнем для використання моделі як компонента підтримки рішень у контурі керування SDN.

Вплив ШІ-модуля на справедливість доступу та стабільність розподілу ресурсу ілюструється порівнянням залежності «частка спроб - частка успішних запитів» (рис. 3.16). У цій візуалізації по осі *x* відкладається частка спроб клієнта

в загальному потоці, а по осі y - частка успішних запитів. Штрих пунктирна лінія $y = x$ відповідає ідеальному пропорційному розподілу, коли клієнт із більшою часткою навантаження отримує пропорційно більшу частку успішних відповідей без витіснення інших клієнтів.



а) б)

Рис. 3.16. Співвідношення спроб та успішних запитів у сценаріях: а) без ШІ-модуля, б) з ШІ-модулем

У сценарії без ШІ (рис. 3.16.а) спостерігається помітне розсіювання точок відносно лінії $y = x$, тобто клієнти з високою часткою спроб не завжди отримують пропорційну частку успішних відповідей, а менш активні клієнти в пікових режимах можуть втрачати доступ до ресурсу. Натомість у сценарії з ШІ (рис. 3.16. б) хмара точок концентрується ближче до лінії $y = x$, що вказує на вирівнювання пропорційності - клієнти, які формують більший попит, отримують співмірну частку успішних запитів, а клієнти з меншим попитом зберігають гарантований доступ за рахунок адаптивного коригування квот.

Динаміка справедливості в часі додатково оцінювалася показником рівномірного розподілу, який у даному дослідженні визначається на основі індексу справедливості Джейна, описаного в розділі 2.1. Для кожного часового вікна узгодженість фактичного розподілу успішних відповідей між активними клієнтами з пропорційною (очікуваною) поведінкою кількісно відображається значенням індексу Джейна, поданим у відсотковій формі (для зручності

інтерпретації). Результати (рис. 3.17) демонструють, що за наявності ШІ-модуля показник fairness accuracy утримується у високому діапазоні (88-95%), тоді як у базовому сценарії без ШІ-модуля він має суттєві коливання (74-88%), що вказує на меншу передбачуваність доступу та більшу чутливість до пікових навантажень.

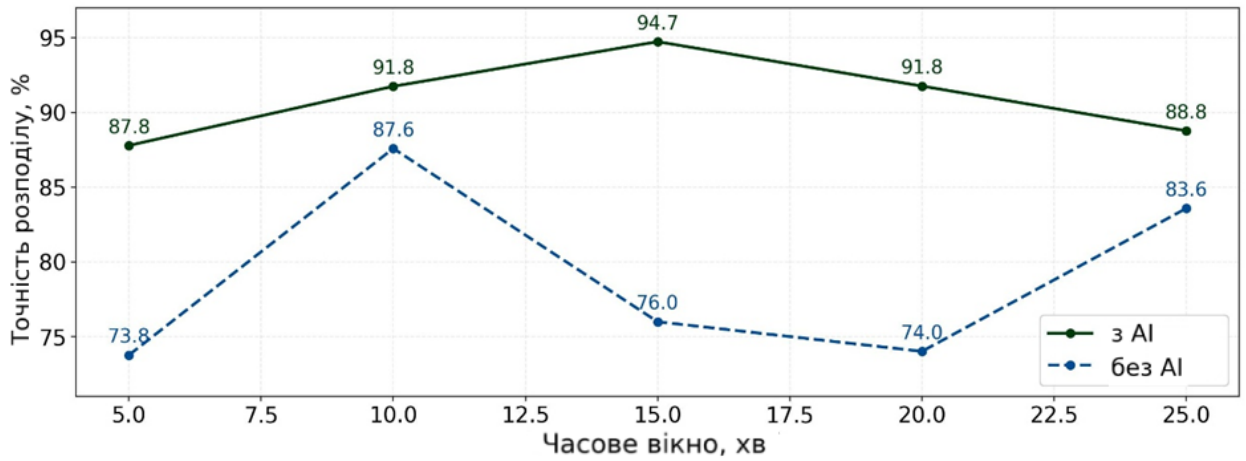


Рис. 3.17. Динаміка справедливості розподілу у часі для базового сценарію та з використанням ШІ-модуля

З погляду інтерпретованості та довіри до рішень інтелектуального модуля керування, застосування методів машинного навчання на рівні керування SDN є практично доцільним, оскільки такі методи дають змогу формувати прогноз ризику перевищення доступу до ресурсу на основі вимірюваних ознак трафіку та стану мережі (наприклад, інтенсивності запитів у часовому вікні, загального навантаження та кількості активних клієнтів). Достовірність отриманих результатів забезпечується відтворюваністю процедури формування ознак, кількісною валідацією якості прогнозування за стандартними метриками класифікації та наявністю логування рішень для кожного циклу керування. Для підвищення операційної безпеки модуль ML вбудовується в політику керування з обмеженнями - застосовуються мінімально допустимі квоти, а коригування параметрів доступу виконується поступово, що знижує ризик різких рішень у відповідь на короткочасні флуктуації трафіку.

Висновки до розділу 3

У розділі запропоновано архітектуру Cloud-AI-SDN фреймворку та виконано її моделювання і дослідження в умовах обмежених ресурсів, зосередивши увагу на побудові відтворюваного експериментального стенда, формалізації методики проведення експерименту та перевірці працездатності інтелектуального циклу керування.

Показано, що запропонована трирівнева архітектура Cloud-AI-SDN фреймворку забезпечує: 1) керованість та програмованість мережі на рівні SDN-контролера, 2) інтеграцію інтелектуальної аналітики в цикл прийняття рішень без жорсткої прив'язки до конкретного домену кінцевих вузлів, 3) масштабованість та спостережуваність через хмарні сервіси. На основі чого сформовано модель відтворюваної гібридної архітектури, яка забезпечує адаптивне керування доступом до обмежених ресурсів IoT-вузлів шляхом поєднання централізованого SDN-контролю з алгоритмами ШІ. Така інтеграція дозволила визначити та реалізувати підходи, спрямовані на досягнення ключових критеріїв функціонування SDN-мережі в гібридному середовищі, зокрема підтримку масштабованості, балансування навантаження, підвищення рівня безпеки та забезпечення інтероперабельності, а також врахування енергетичних аспектів для вузлів із обмеженими енергоресурсами.

Для верифікації працездатності підходу сформовано експериментальний стенд, який відтворює класичну архітектуру SDN та розширює її компонентами IoT і моніторингу, що дозволяє наблизити модель до практичних гібридних AI-SDN-IoT сценаріїв. У складі стенда акцентовано розділення функцій за площинами та використання різних каналів взаємодії, для коректного аналізу накладних витрат, реактивності керування та поведінки системи в умовах ресурсних обмежень.

Обґрунтовано необхідність візуалізації та верифікації функціонування стенда як складової відтворюваного експерименту - контроль стану компонентів і реєстрація рішень у кожному циклі керування підвищують інтерпретованість та

довіру до рішень інтелектуального модуля, а також забезпечують зв’язок між поведінкою мережі та вимірюваними ознаками стану системи.

З практичної точки зору, показано доцільність інтеграції ML-модуля у політику керування з обмеженнями (мінімальні квоти, контроль різких змін), що знижує ризик дестабілізації керування через короткочасні флуктуації та підвищує операційну стійкість системи.

Сформовано та структуровано методику експериментального дослідження інтелектуального керування доступом до обмеженого ресурсу як послідовність етапів - підготовка даних, підготовка ШІ-модуля та тестування з оцінюванням результатів. Така декомпозиція забезпечує відтворюваність експерименту, та прозоре трасування переходу від телеметрії до інтеграції навченої моделі в SDN-середовище. У підсумку розділ формує цілісне методичне й інженерне підґрунтя для наступних розділів дисертації, де здійснюється порівняння моделей, аналіз накладних витрат та апаратна апробація запропонованого підходу.

РОЗДІЛ 4. АПАРАТНІ ДОСЛІДЖЕННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНОГО ФРЕЙМВОРКУ

4.1. Порівняльна оцінка моделей машинного навчання для ШІ-модуля SDN

У попередніх розділах обґрунтовано архітектуру запропонованого Cloud-AI-SDN фреймворку, описано структуру експериментального стенда, механізми збору телеметрії та методику проведення досліджень для задачі адаптивного керування доступом клієнтів до обмеженого ресурсу. Наступним логічним кроком є вибір моделі машинного навчання для ШІ-модуля, який у циклі керування формує рішення щодо коригування квот/політик доступу до обмеженого API-ресурсу в реальному часі. Ключова вимога до вибору моделі в контурі керування SDN - забезпечити достатню точність прогнозування ризику перевищення квоти при мінімальному обчислювальному навантаженні та затримці формування рішення. Це є принциповим для SDN-середовищ з обмеженими ресурсами, де збільшення часу прийняття рішень прямо впливає на оперативність реакції контролера та стабільність циклу керування і мережі в цілому. Для відтворюваного порівняння в єдиному експериментальному середовищі підготовлено та оцінено такі моделі-кандидати для класифікації:

- Logistic Regression - базова лінійна модель з високою інтерпретованістю та низькою обчислювальною складністю.

- Random Forest - ансамблева модель на основі паралельних дерев рішень, ефективна для виявлення нелінійних залежностей та стійка до перенавчання.

- Gradient Boosting - ітеративний підхід, що забезпечує високу точність прогнозування шляхом послідовної мінімізації помилок попередніх моделей.

- Support Vector Machine - метод пошуку оптимальної розділової гіперплощини з можливістю імовірнісної оцінки приналежності до класів.

- Multi-Layer Perceptron - повнозв’язна нейронна мережа, здатна апроксимувати складні закономірності у даних.

Зазначені моделі розглядаються як кандидати на роль ядра прогнозування у ШІ-модулі, який на основі агрегованої телеметрії та ознак за вікно часу оцінює імовірність перевищення квоти та ініціює адаптацію політики доступу.

4.1.1. Аналіз затримки прийняття рішень

Одним із важливих параметрів для SDN-контролерів, що працюють у режимі реального часу, є затримка прийняття рішення (inference latency) - середній час обчислення прогнозу для одного вектора ознак у межах одного циклу керування потоками даних. Порівняння середнього часу обробки запиту для різних моделей показує (рис. 4.1), що Logistic Regression, SVM та Gradient Boosting характеризуються високою реактивністю із затримкою в межах 0.6-1.4мс. Такі показники є прийнятними для оперативного керування доступом до ресурсу без суттєвого навантаження на керуючу площину мережі.

Натомість модель Random Forest, попри високу якість прогнозування, демонструє середній час прийняття рішень понад 20мс, що формує «вузьке місце» при високій інтенсивності циклів керування або при зростанні кількості активних клієнтів від мережевих вузлів. У такому режимі алгоритм RF доцільніше розглядати як інструмент офлайн-аналітики, а не як модель для реального часу в SDN-контурі прийняття рішень.

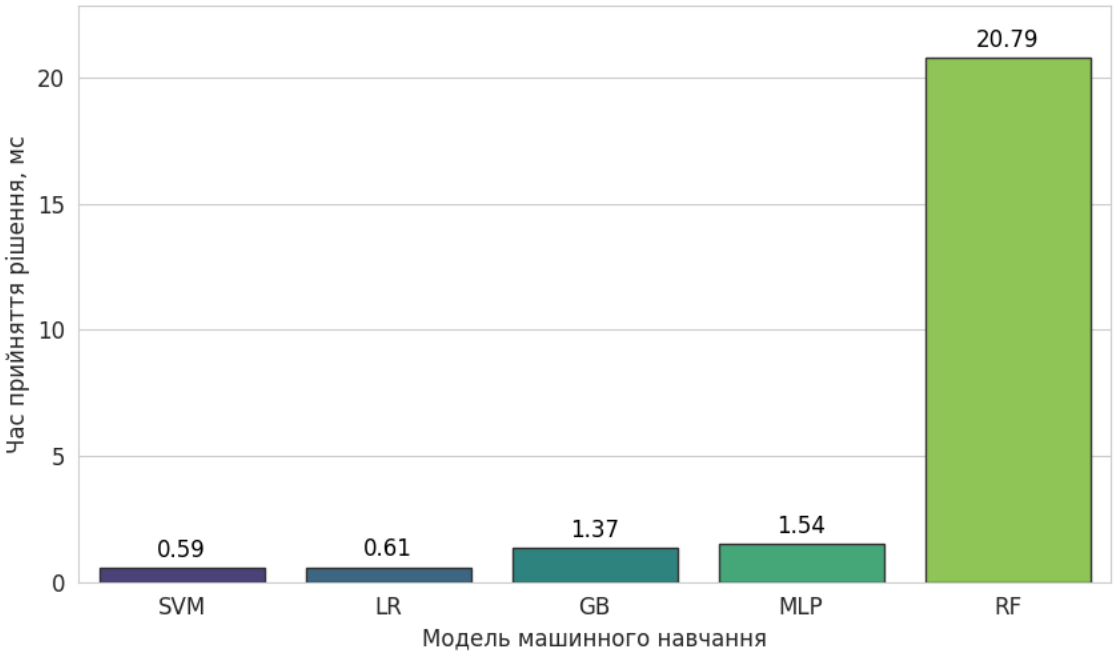


Рис. 4.1. Середній час прийняття рішень моделей в експерименті

4.1.2. Аналіз справедливості розподілу обмежених ресурсів обраних ML-моделей

Для кількісного оцінювання справедливості розподілу доступу до обмеженого ресурсу між користувачами (потоками даних) використано індекс справедливості Джейна (Jain's Fairness Index), теоретичне обґрунтування якого розглядалося у розділі 2. Нагадаємо, що показник *J* розраховується як (4.1):

$$J = \frac{(\sum_{i=1}^{n} x_i)^2}{n \times \sum_{i=1}^{n} x_i^2} \quad (4.1)$$

де n - кількість мережевих вузлів, а x_i - нормалізована частка виділеного ресурсу для *i*-го вузла.

Використання нормалізованих часток дозволяє порівнювати справедливість розподілу незалежно від абсолютного обсягу трафіку або інтенсивності запитів. Аналіз розподілу значень індексу Джейна (рис. 4.2) показує, що алгоритми GB та LR забезпечують найвищі медіанні значення справедливості на рівні $J \approx 0.98$, зберігаючи пропорційний доступ до API навіть за умов критичного перевантаження каналів управління.

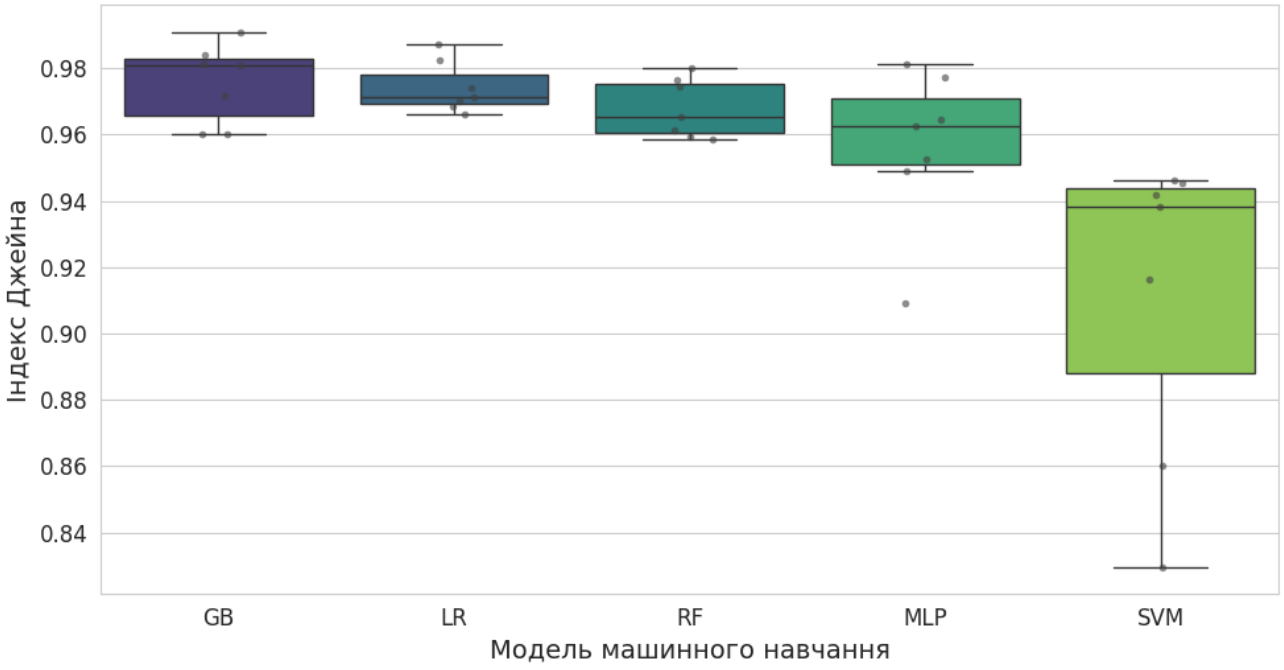


Рис. 4.2. Розподіл значень індексу Джейна

Водночас SVM демонструє гірші показники з широким розкидом значень, що опускаються до $J \approx 0.83$. Це свідчить про часті випадки необґрунтованого обмеження окремих потоків (дискримінації) внаслідок помилок класифікації станів мережі. Такі коливання індексу для SVM підтверджують його низьку стійкість до незбалансованого вхідного навантаження в SDN.

4.1.3. Аналіз результатів обраних ML-моделей

Оцінювання ефективності проведено за сукупністю метрик якості класифікації та показників часової ефективності:

- Accuracy - загальна частка правильно класифікованих екземплярів;

- ROC-AUC - інтегральна метрика якості розділення класів, що оцінює здатність моделі до ранжування ймовірностей;

- Inference time (мс) - середній час, необхідний для прийняття рішення щодо одного вектора ознак (обчислювальна затримка).

Такий набір показників дозволяє проаналізувати компроміс між розпізнавальною здатністю та обчислювальною складністю моделей (таблиця 4.1), що є важливим критерієм для забезпечення роботи SDN-контролера в реальному часі та в умовах обмежених ресурсів.

Таблиця 4.1

Порівняння моделей машинного навчання для регулювання політики доступу

Модель	***Accuracy***	***ROC-AUC***	***Inference time, мс***	***Коментар***
LR	0.9768	0.997	~ 0.6	Висока швидкодія та інтерпретованість;
RF	0.9939	0.999	~ 21	Дуже висока точність, але значна затримка;
GB	0.9988	0.973	~ 1.4	Максимальна точність; прийнятна затримка
SVM	0.9532	0.973	~ 0.6	Низька чутливість до перевантажень; миттєве рішення;
MLP	0.9821	0.982	~ 1.5	Висока точність; схильність до хибнопозитивних спрацювань;

Logistic Regression - за результатами розширеного експерименту модель забезпечила близько 97.7% точності класифікації та ROC-AUC = 0.997, демонструючи здатність коректно ідентифікувати стани перевищення квоти. Середній час затримки становив близько 0.6 мс, що робить LR однією з найшвидших моделей. Обмеженням є лінійна межа прийняття рішень, через що модель слабше відображає складні нелінійні патерни поведінки клієнтів. Водночас, завдяки низьким обчислювальним витратам, дана модуль придатна для роботи в реальному часі та розгортання на малопотужних edge-пристроях (зокрема Raspberry Pi).

Random Forest - модель показала дуже високі значення accuracy ≈ 99.4% та ROC-AUC ≈ 0.999, фактично забезпечуючи майже безпомилкове виявлення ситуацій перевищення ресурсу. Однак ці переваги досягаються ціною відчутних витрат, середній час затримки становив близько 21 мс (приблизно у 30 разів повільніше за LR), а також зростають вимоги до пам'яті та CPU через ансамблеву природу моделі. Тому пряме розгортання RF у контурі керування SDN може бути обмеженим у сценаріях з великою кількістю клієнтів та жорсткими часовими вимогами.

Gradient Boosting - продемонстрував accuracy ≈ 99.9% із повним виявленням подій перевищення квоти (100% recall для позитивного класу), при цьому середній час затримки становив близько 1.4 мс. Значення ROC-AUC ≈ 0.973 дещо нижче очікуваного (ймовірно через розподіл даних або помірне перенавчання), однак у цілому модель забезпечує майже оптимальний компроміс між прогнозною здатністю та прийнятною затримкою.

Support Vector Machine - показав найнижчу точність серед розглянутих моделей (accuracy ≈ 95.3%, ROC-AUC ≈ 0.973). Хоча точність для позитивного класу (перевищення ресурсу) була високою (precision ≈ 99.7%), модель пропускала помітну частку реальних перевищень (recall ≈ 95.4%), що важливо для завдання превентивного керування доступом, оскільки пропуски можуть погіршувати стабільність та справедливість розподілу ресурсу в телекомунікаційній мережі. З урахуванням чутливості до дисбалансу класів та

додаткової складності калібрування ймовірностей, практична доцільність SVM у даному SDN-сценарії є обмеженою.

Multi-Layer Perceptron - забезпечив accuracy ≈ 98.2% та ROC-AUC ≈ 0.982, майже повністю виявляючи перевищення (recall ≈ 99.6%), однак із підвищеною кількістю хибнопозитивних спрацювань для нормального трафіку. Це означає ризик необґрунтованого «посилення» політик, що потенційно може зменшувати справедливість для частини клієнтів і призводити до зниження якості досвіду користування. За рівнем затримки модель залишалась прийнятною (~1.5 мс), але підвищений рівень хибних спрацювань робить її менш бажаною для експлуатаційного контуру керування доступом.

Аналіз отриманих результатів свідчить про те, що оптимальна модель визначається не лише показником абсолютної точності, а вимогами конкретного сценарію управління в SDN-мережі:

- Сценарії Edge-обчислень, реального часу та суворих обмежень обчислювальних ресурсів - доцільно використовувати модель LR як мінімально затратну з погляду використання CPU/RAM та з мінімальною часовою затримкою.

- Забезпечення максимальної ефективності ідентифікації станів мережі при помірно допустимих витратах ресурсів контролера - найбільш збалансованим вибором є алгоритм GB (що забезпечує високу точність при прийнятному часі прийняття рішення).

- Ретроспективний аналіз (офлайн-аналітика) або сценарії з невеликою кількістю потоків даних і без жорстких часових обмежень - можливе застосування RF як референтного за якістю класифікатора, але з обмеженою придатністю для оперативного онлайн-управління через високу обчислювальну складність.

Використання моделей SVM та MLP у даному експериментальному контурі є менш доцільним. SVM через високу ймовірність пропуску подій перевищення лімітів (низький recall). MLP через схильність до хибнопозитивних рішень та ризик необґрунтованих обмежень трафіку користувачів.

4.2. Апаратна валідація інтелектуального SDN-модуля в умовах обмежених ресурсів

У попередньому пункті виконано порівняльну оцінку моделей машинного навчання для ШІ-модуля SDN і обґрунтовано доцільність використання двох моделей-кандидатів, які продемонстрували найкращі результати у попередніх тестах: Logistic Regression - як еталон мінімальної обчислювальної складності та Gradient Boosting - як моделі з вищою якістю прогнозування при прийнятних часових витратах.

Метою даного етапу є перехід від віртуалізованого середовища до моделювання на фізичному обладнанні. Це дозволяє врахувати реальні затримки системних викликів, обмеження архітектури ARM та фізичні показники енергоспоживання, що неможливо повноцінно відтворити у віртуальних умовах.

Для проведення експерименту було розгорнуто стенд, конфігурація якого максимально наближена до реального SDN-контролера периферійного сегмента мережі (рис. 4.3.а).



а) б)

Рис. 4.3. Апаратна конфігурація експерименту:

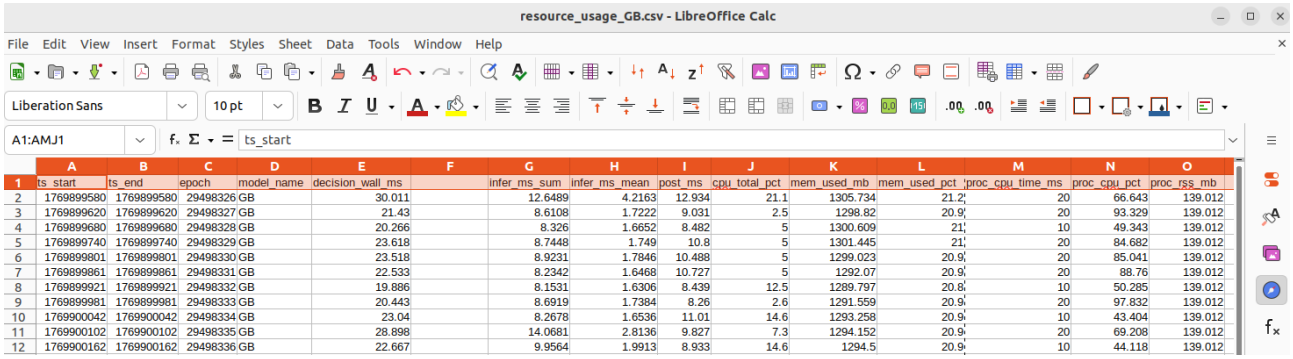
а) запуск стенда на Raspberry Pi 5, б) підключення USB-ватметру

Апаратну основу стенда становить одноплатний комп’ютер Raspberry Pi 5 (8 GB RAM) на базі 64-бітного 4-ядерного процесора ARM Cortex-A76 з тактовою частотою 2.4 ГГц. Вибір даної платформи обумовлений її широким використанням як основи для побудови IoT-шлюзів та локальних SDN-контролерів. Для контролю енергоспоживання використано USB-ватметр Atorch J7-C (USB-C), який вмикається в ланцюг живлення пристрою (рис. 4.3.б) та фіксує накопичену енергію (Wh) і спожитий заряд (mAh).

З метою підтвердження практичної придатності ШІ-модуля в умовах обмежених ресурсів навчання та первинну валідацію моделей LR і GB виконано безпосередньо на Raspberry Pi, щоб уникнути прихованих відмінностей між x86/ARM-реалізаціями бібліотек, а також отримати репрезентативні показники якості моделі в умовах подальшого.

За результатами навчання GB забезпечує вищу узгодженість метрик на обох класах: для класу «у межах ресурсу» отримано precision = 1.000 та recall = 0.955 (F1 = 0.977), тоді як LR при recall = 1.000 демонструє суттєво нижчу precision = 0.537 (F1 = 0.698). З огляду на те, що помилкові «дозволи» напряму знижують ефективність визначення квоти, вища точність GB у класі «у межах ресурсу» є більш прийнятною для задачі керування доступом до обмеженого API.

Щоб порівняння LR та GB моделі було коректним у середовищі з фоновими подіями ОС, акцент зроблено на метриках процесу в якому виконується робота ШІ-агента. Вони найкраще відображають накладні витрати саме модуля прийняття рішень (а не всієї системи загалом) і зменшують вплив сторонніх процесів. Лог формується у вигляді CSV-файлу (рис. 4.4) та містить часові мітки, ідентифікацію моделі, тривалість циклу прийняття рішення, усереднені показники часу прийняття рішення, а також процесні метрики CPU та оперативної пам’яті.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	ts_start	ts_end	epoch	model_name	decision_wall_ms		infer_ms_sum	infer_ms_mean	post_ms	cpu_total_pct	mem_used_mb	mem_used_pct	proc_cpu_time_ms	proc_cpu_pct	proc_rss_mb
2	1769899580	1769899580	29498326	GB	30.011		12.6489	4.2163	12.934	21.1	1305.734	21.2	20	66.643	139.012
3	1769899620	1769899620	29498327	GB	21.43		8.6108	1.7222	9.031	2.5	1298.82	20.9	20	93.329	139.012
4	1769899680	1769899680	29498328	GB	20.266		8.326	1.6652	8.482	5	1300.609	21	10	49.343	139.012
5	1769899740	1769899740	29498329	GB	23.618		8.7448	1.749	10.8	5	1301.445	21	20	84.682	139.012
6	1769899801	1769899801	29498330	GB	23.518		8.9231	1.7846	10.488	5	1299.023	20.9	20	85.041	139.012
7	1769899861	1769899861	29498331	GB	22.533		8.2342	1.6468	10.727	5	1292.07	20.9	20	88.76	139.012
8	1769899921	1769899921	29498332	GB	19.886		8.1531	1.6306	8.439	12.5	1289.797	20.8	10	50.285	139.012
9	1769899981	1769899981	29498333	GB	20.443		8.6919	1.7384	8.26	2.6	1291.559	20.9	20	97.832	139.012
10	1769900042	1769900042	29498334	GB	23.04		8.2678	1.6536	11.01	14.6	1293.258	20.9	10	43.404	139.012
11	1769900102	1769900102	29498335	GB	28.898		14.0681	2.8136	9.827	7.3	1294.152	20.9	20	69.208	139.012
12	1769900162	1769900162	29498336	GB	22.667		9.9564	1.9913	8.933	14.6	1294.5	20.9	10	44.118	139.012

Рис. 4.4. Приклад CSV лог-файлу з показниками роботи ШІ-агента

4.2.1. Результати апаратної валідації: накладні витрати (LR та GB)

Для підвищення наочності дані було агреговано з кроком 5 хв. . Порівняння завантаження CPU процесом ШІ-агента для LR та GB наведено на рис. 4.5. Візуально спостерігається, що GB у середньому демонструє вищий рівень використання CPU (66.5%), тоді як LR частіше має нижчі значення (57.6%) і глибші просідання в окремих інтервалах часу, що відповідає меншому на 13.4% використанню CPU. При цьому обидві моделі залишаються працездатними в межах обраного циклу керування.

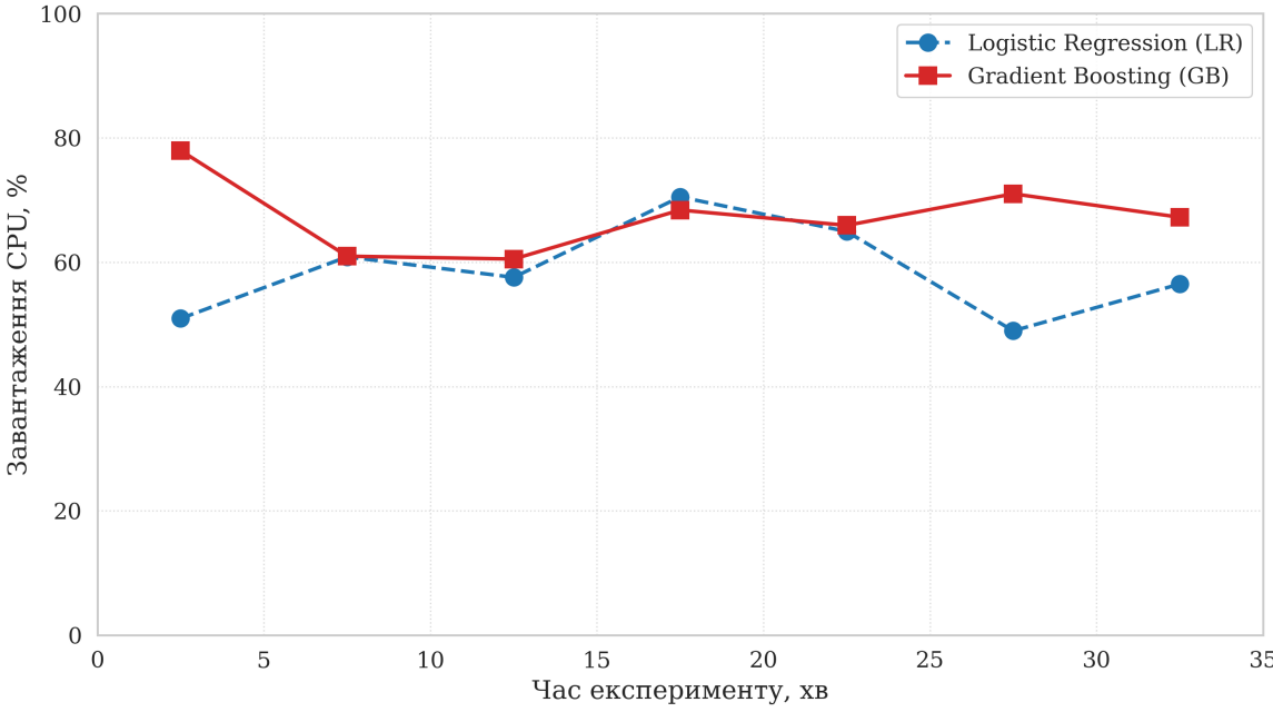
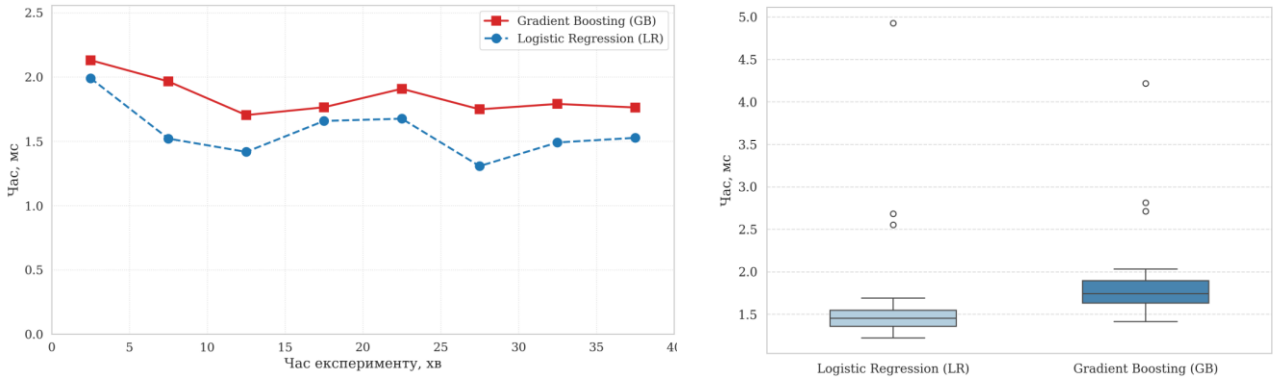


Рис. 4.5. Завантаження CPU процесом ШІ-агента для LR та GB

Час прийняття рішення є принциповим для застосування ML-моделі в контурі керування, оскільки збільшення латентності безпосередньо впливає на реактивність керуючої площини. На рис. 4.6.а показано середній час прийняття рішення в часі (агрегація 5 хв), а на рис. 4.6.б розподіл відповідних значень.



а) б)

Рис. 4.6. Час прийняття рішення (LR та GB): а) середнє значення в часі, б) розподіл значень

На графіках можна спостерігати, що LR має менший середній час прийняття рішення, тоді як GB характеризується помірно більшими значеннями - 1.86мс проти 1.59мс LR (LR на 14.66% швидша за GB), проте обидва алгоритми залишаються в низькому часовому діапазоні (мілісекнуди), що є прийнятним для реалізації циклу керування в експериментальному стенді.

Умови обмежених ресурсів вимагають контролю не лише CPU-витрат, але й пам’яті, оскільки на edge-пристроях саме RAM часто є лімітуючим фактором при масштабуванні кількості сервісів. На рис. 4.7 наведено середнє споживання оперативної пам’яті для двох реалізацій ШІ-агента. За результатами вимірювань GB використовує більше пам’яті, ніж LR (різниця помірна та очікувана для більш складної моделі), а саме 139.0 МБ проти 130.2 МБ, що відповідає збільшенню на 6.33%. Однак, різниця залишається помірною і не впливає на стабільність у межах доступного об’єму оперативної пам'яті Raspberry Pi 5.

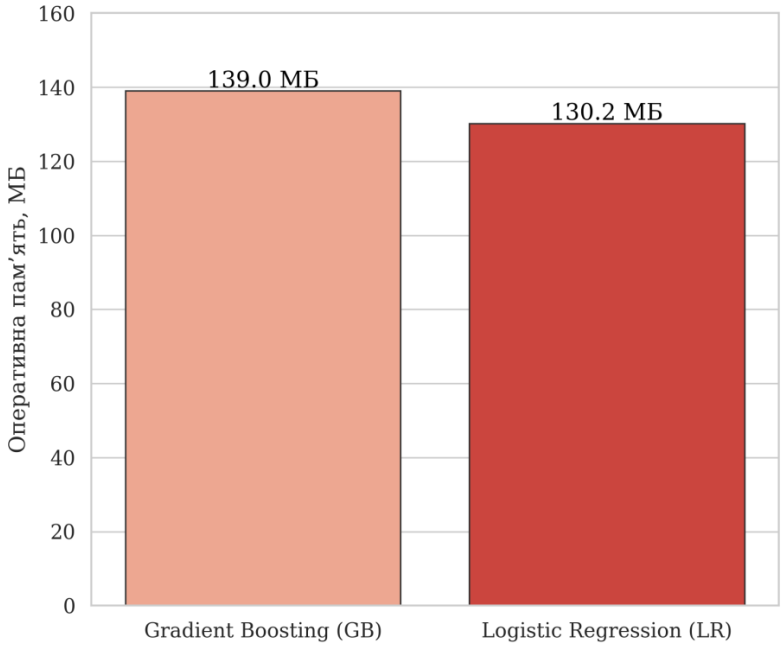


Рис. 4.7. Середнє використання оперативної пам'яті процесом ШІ-агента

4.2.2. Результати апаратної валідації: енергоспоживання системи

Енергоспоживання оцінювалось за показами Atorch J7-C при виконанні ідентичного експерименту тривалістю 35 хв для кожної моделі. Для порівняння використано значення спожитої енергії, Вт·год, та ємності, мА·год. Підсумкове порівняння наведено на рис. 4.8.

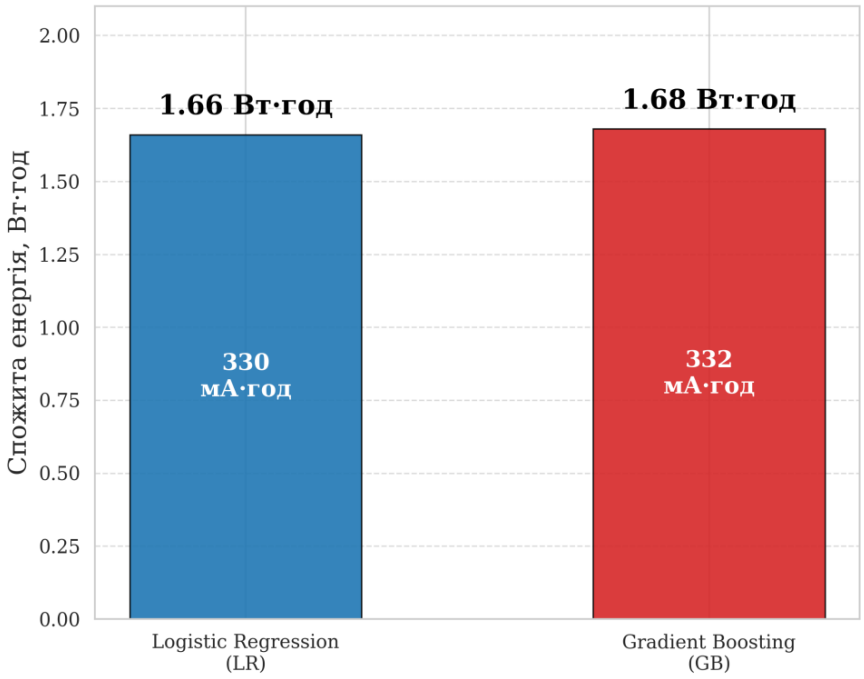


Рис. 4.8. Енергоспоживання стенда в ході експерименту

Різниця у спожитій енергії між режимами роботи з LR та GB становить лише 1,2%, тобто підвищення обчислювальної складності (перехід від LR до GB) на ARM-платформі практично не впливає на автономність пристрою у межах даної конфігурації. Зафіксовані відмінності проявляються переважно в процесних показниках ШІ-агента: GB споживає більше RSS-пам'яті (+6.33%; 139.0 МБ проти 130.2 МБ), і характеризується більшим середнім часом прийняття рішення (LR швидша на 14.66%, 1.59 мс проти 1.86 мс) та характеризується вищим середнім завантаженням CPU, тоді як LR використовує CPU менше приблизно на 13.4% (57.6% проти 66.5%). Отже, з погляду енергоспоживання вибір моделі не є критичним, а ключовими критеріями стають якість керування та прийнятні накладні витрати процесу ШІ-агента.

Перенесення експериментального стенда на Raspberry Pi 5 підтвердило практичну здійсненність реалізації AI-керування доступом до обмеженого ресурсу в умовах обмежених ресурсів edge-пристроїв. Попри те, що LR є «легшою» за процесними метриками (нижча латентність, менше RSS та CPU), вибір GB є більш обґрунтованим за показниками точності прийняття рішень - GB досягає accuracy ≈ 0.9988 та F1 ≈ 0.988, тоді як LR має accuracy ≈ 0.9768 та macro-F1 ≈ 0.843. Важливо, що для класу «у межах ресурсу» GB забезпечує precision = 1.0 (тобто, відсутні помилкові дозволи для випадків перевищення ресурсу), тоді як LR при precision = 0.5366 формує значно більшу кількість ризикових рішень, які знижують ефективність квотування в реальному часі. З урахуванням мінімальної різниці в енергоспоживанні (1.2%), прийнятних процесних накладних витрат та кращої якості класифікації в критичному режимі «доступ дозволено», доцільно надалі використовувати GB як основну модель у наступних експериментах, залишаючи LR як спрощений базовий варіант для сценаріїв із жорсткішими обмеженнями CPU/RAM.

4.3. Розробка та інтеграція веб-сервісу прикладного рівня Cloud-AI-SDN фреймворку

В ході експериментальної перевірки методів ML-керування доступом до обмежених ресурсів у програмно-конфігурованих мережах виникає потреба у розробці прикладної підсистеми, яка забезпечує керований доступ до функцій фреймворку, а також формалізований збір результатів та візуалізації дослідів в реальному часі. У реальному середовищі подібні функції виконує система мережевого керування, однак у рамках дисертаційного дослідження необхідним є полегшений, відтворюваний та прозорий інструмент, який:

- надає єдину точку керування експериментом;
- стандартизує взаємодію між компонентами фреймворку;
- забезпечує журналювання роботи мережі та метрик для подальшого аналізу;
- мінімізує вплив людського фактору на результати та хід експерименту.

З цією метою у складі запропонованого Cloud-AI-SDN фреймворку розроблено веб-сервіс прикладного рівня, який об’єднує функції адміністративного керування (керування клієнтами та політиками доступу), інтерфейси інтеграції (REST API із документуванням), засоби спостереження за експериментом (Web додаток з візуалізацією та статистикою), а також інструменти моніторингу ресурсів вузла розгортання SDN мережі (Prometheus/Grafana). Веб-сервіс дозволяє запускати та повторювати однакові сценарії, швидко перемикати активну ML-модель AI-агента, контролювати стан клієнтів, переглядати агреговані показники звернень до обмеженого ресурсу та аналізувати часові й ресурсні характеристики прийняття рішень AI-модуля.

Особливістю розробки є орієнтація на умови обмежених ресурсів (edge-вузол, зокрема платформи класу Raspberry Pi), тому архітектура сервісу має бути мінімалістичною, зі зниженими накладними витратами, відсутністю зайвих залежностей та можливістю розгортання на одному вузлі без складних зовнішніх компонентів. Водночас сервіс повинен забезпечувати достатню

функціональність для керованого порівняння ML-моделей і верифікації поведінки системи.

4.3.1. Вимоги до прикладного веб-сервісу та постановка завдання

В ході проєктування веб-сервісу прикладного рівня сформульовано такі функціональні вимоги:

1) Керування клієнтами та політиками доступу.

- отримання списку клієнтів та їхнього поточного стану в мережі (активний/заблокований);

- можливість примусового блокування/розблокування;

- доступ до агрегованої статистики звернень за часовим вікном та можливість очищення статистики для нових серій експериментів.

2) Керування AI-агентом і ML-моделями:

- перегляд переліку доступних моделей та їх статусу;

- активація обраної моделі в реальному часі шляхом зміни конфігурації фреймворку;

- доступ до журналів рішень AI-агента (вектори ознак, розраховані квоти/ризики, час прийняття рішення).

3) Управління IoT-площиною експерименту:

- кероване переведення IoT-вузлів у різні стани енергоспоживання (on/power-saving/off), що дозволяє відтворювано змінювати навантаження та умови для AI-модуля.

4) Моніторинг та відлагодження системи:

- Web UI для швидкої діагностики та візуалізації (графіки за агрегацією по часу, статуси, статистика);

- формалізовані API-інтерфейси (Swagger/OpenAPI), що дозволяють відтворювати експериментальні сценарії через Postman чи користувацькі скрипти.

Окремо визначено нефункціональні вимоги, критичні саме для умов обмежених ресурсів:

- Низьке споживання ресурсів. Сервіс не повинен істотно впливати на результати вимірювань (CPU/RAM/latency).

- Надійність журналювання. Дані повинні записуватись атомарно та зчитуватись без спотворень при паралельних зверненнях.

- Відтворюваність еквпериментів. Можливість відновити послідовність дій та умови експерименту (конфігурації, активну модель, параметри лімітів, періоди агрегації).

- Простота розгортання. Запустити сервіс на одному вузлі (edge) або у зв'язці вузлів (контролер/генератори/IoT) без складного операційного ланцюга.

4.3.2. Архітектура веб-сервісу і принципи інтеграції з компонентами фреймворку

З метою зменшення зв'язаності компонентів і підвищення керованості, веб-сервіс побудовано за принципом розділення зовнішнього інтерфейсу та внутрішнього (виконання операційної логіки). У практичній реалізації використаний двошаровий підхід:

- Зовнішній шар (API gateway) надає єдиний REST API та Swagger UI, здійснює базову валідацію параметрів і маршрутизацію запитів.

- Внутрішній шар (експериментальний сервіс керування доступом) - реалізує механізми обліку звернень до обмеженого ресурсу, політики блокування, агрегацію статистики за часовим вікном, а також інтеграцію з AI-агентом (читання активної моделі, запис рішень, часу обробки тощо) та IoT площиною.

Такий поділ дає наступні переваги:

- Усі операції, що можуть впливати на вимірювані характеристики (логування, агрегація, рішення щодо допуску/відхилення), ізольовані в одному компоненті.

- Зовнішній інтерфейс залишається стабільним навіть при зміні внутрішніх механізмів.

- За потреби можна обмежувати зовнішній шар (авторизація доступу) не торкаючись внутрішнього, що відповідає практикам побудови керуючих контурів у мережах.

- У розширених сценаріях зовнішній і внутрішній шари можуть бути рознесені по вузлах (наприклад, UI на робочій станції, експериментальна логіка на edge-вузлі), не змінюючи логіки взаємодії.

З точки зору площин SDN, веб-сервіс функціонує як компонент прикладної площини, який взаємодіє з площиною керування (SDN-контролер/AI-модуль) опосередковано - через задані API, конфігураційні файли, журнали рішень та MQTT (черги подій). Сервіс не втручається в механізми OpenFlow напряму (рис. 4.9), а керує стендом на рівні сервісних політик (допуск/відхилення звернень до обмеженого ресурсу, блокування IP, зміна активної моделі, стани IoT-навантаження).

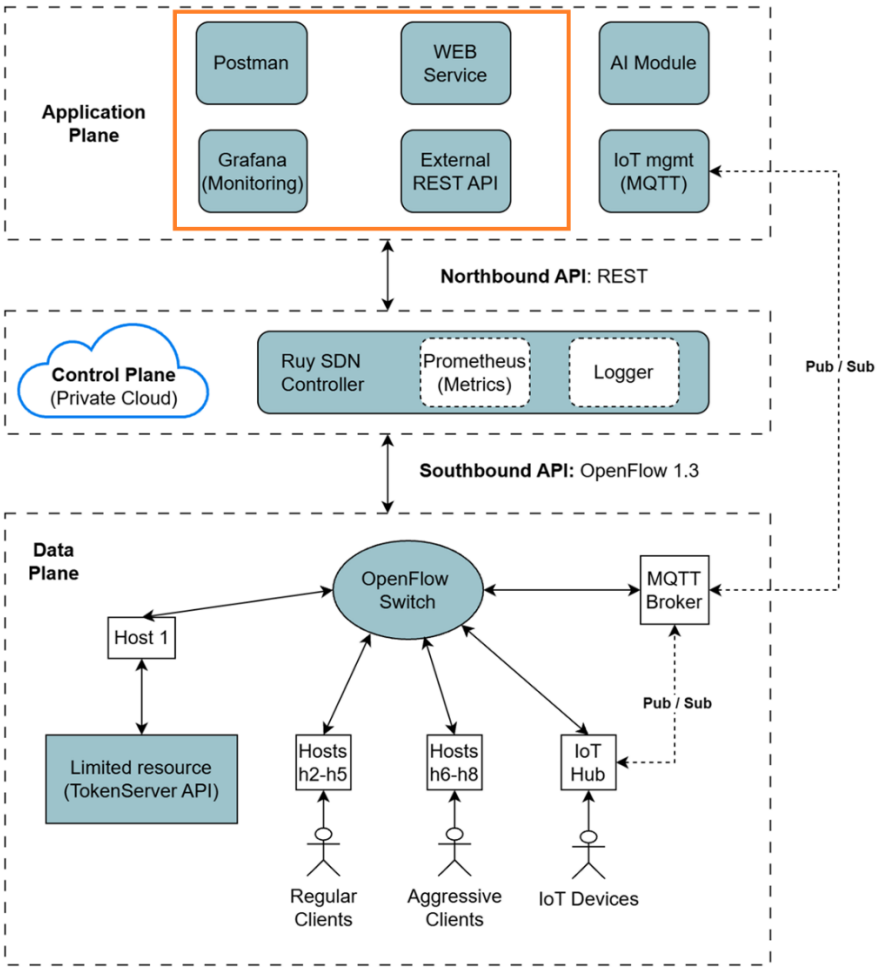


Рис. 4.9. Схема інтеграції компонентів веб-сервісу в SDN

4.3.3. REST API, сценарії керування та приклади запитів

Інтерфейси API структуровано (рис. 4.10) за функціональними доменами, що відповідають підсистемам Cloud-AI-SDN фреймворку - керування клієнтами та статистикою доступу до обмеженого ресурсу, керування AI-агентом і ML-моделями, керування IoT-площиною (через MQTT-шлюз), а також системна діагностика.

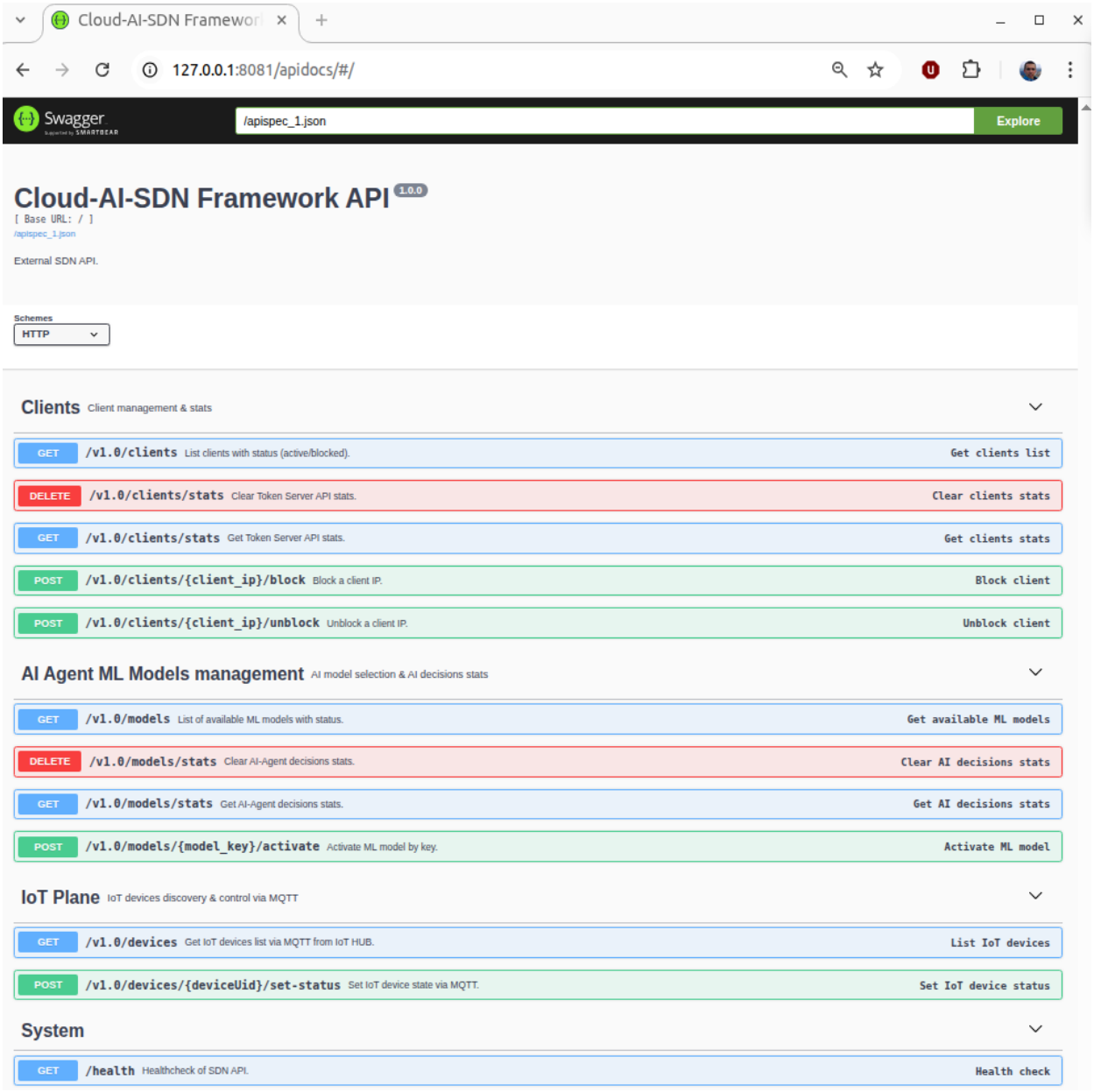


Рис. 4.10. Swagger інтерфейс побудованого REST-API

- Група Clients (керування клієнтами та доступом). API-методи цієї групи націлені на керування станом клієнтів та аналіз динаміки звернень. Типові

операції - отримати список клієнтів (рис. 4.11), заблокувати / розблокувати IP, отримати агреговану статистику й очистити її.

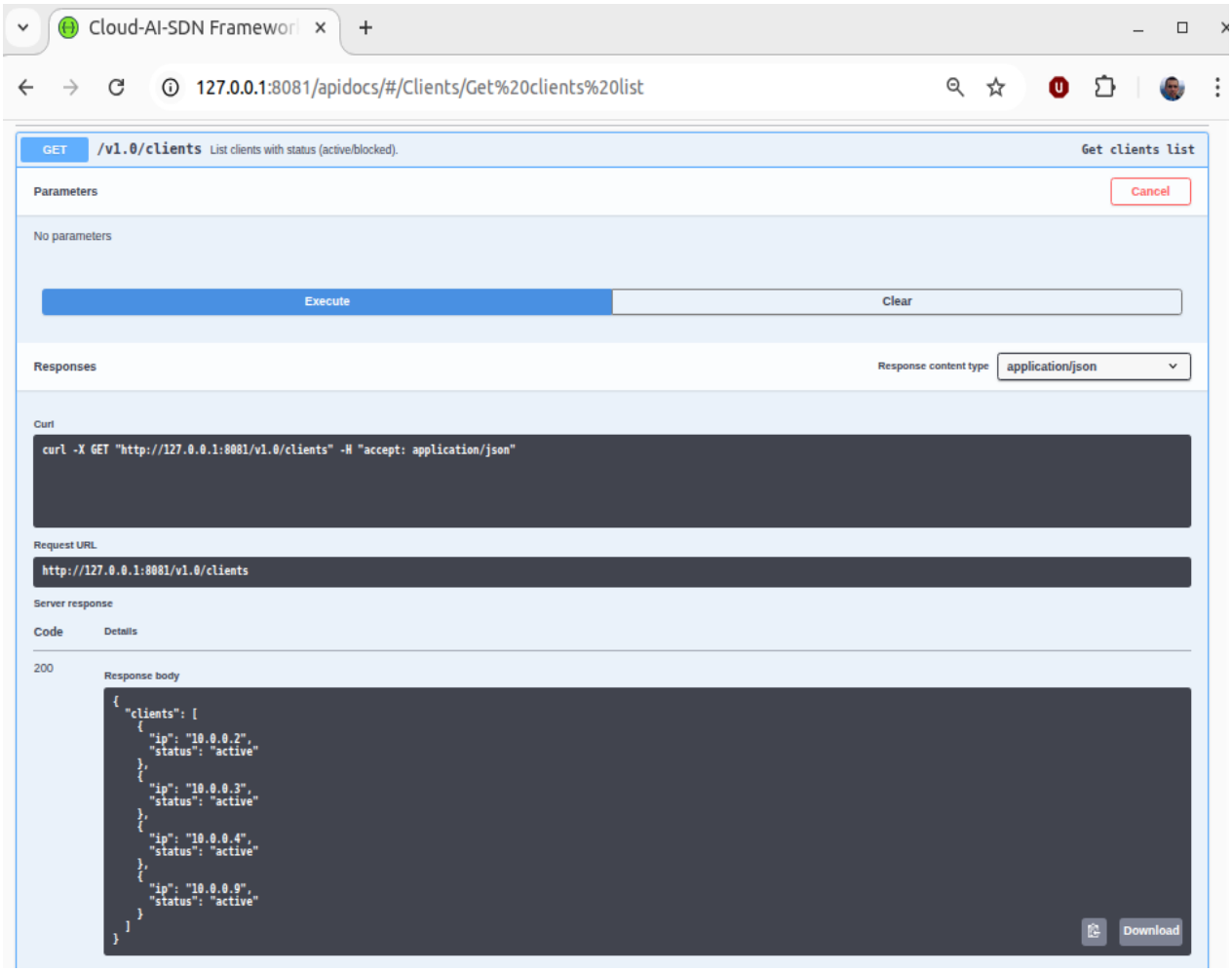


Рис. 4.11. Приклад виклику ендпоінту через Swagger UI

- Група AI Agent ML Models (керування моделями машинного навчання). Група забезпечує доступ до переліку моделей моделі (рис. 4.12), активацію обраної та отримання статистики рішень. Саме через цю групу реалізується експериментальна логіка порівняння моделей у режимі реального часу. Активація моделі формалізована як API-операція із валідацією ключа, напротивагу редагуванню конфігурації фреймворку в ручному режимі, що знижує ризик помилки та підвищує повторюваність експериментів.

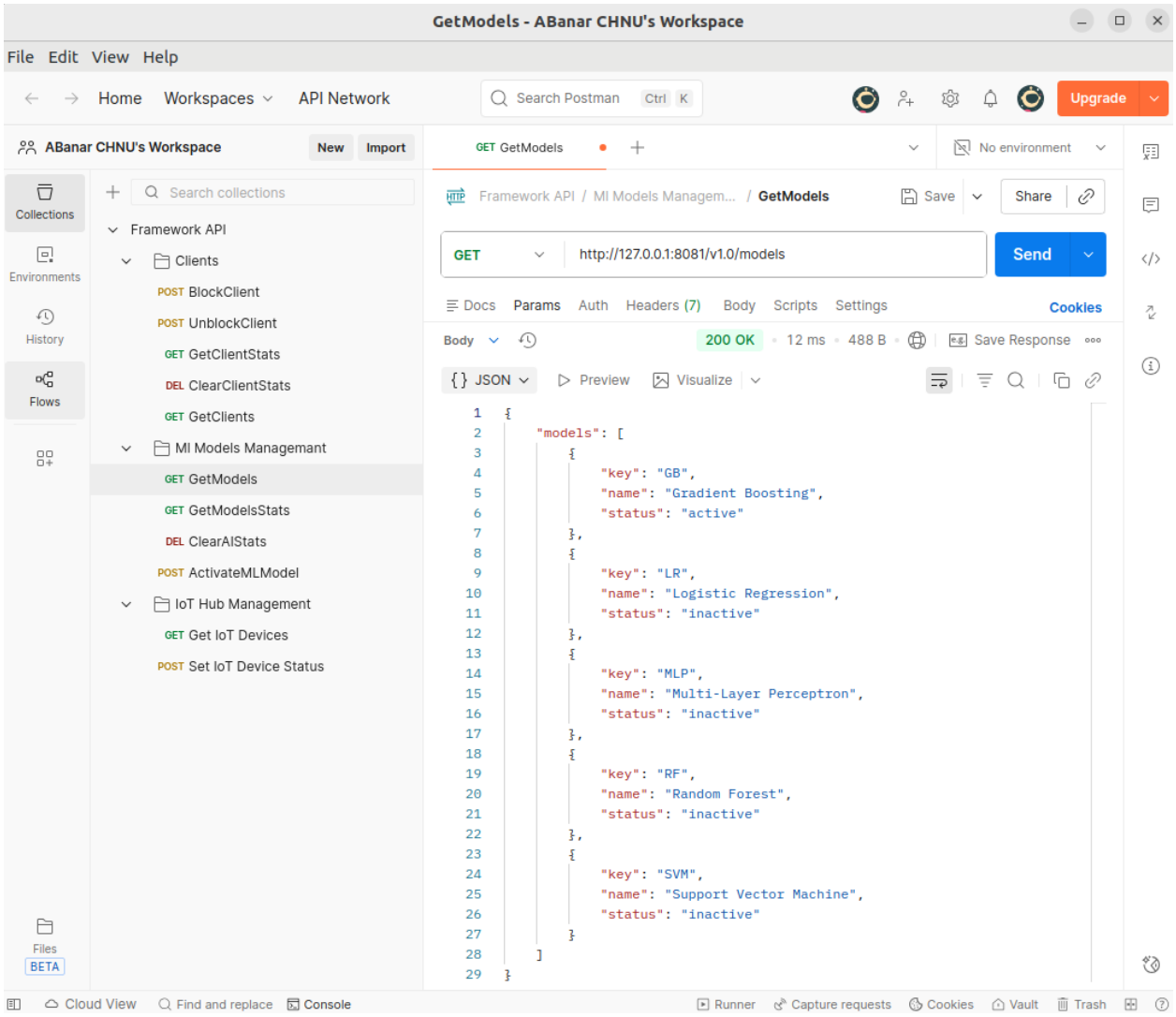


Рис. 4.12. Приклад виконання сценарію ручного керування через Postman - отримання доступних ML моделей

- Група IoT Plane (керування IoT-вузлами через MQTT). API група націлена на зміну профілю навантаження в експериментальному середовищі за рахунок керування станами енергоспоживання IoT-пристроїв. Типові операції - отримати перелік IoT-пристроїв з IoT HUB та встановити стан конкретного пристрою (on / power_saving / off), що дозволяє відтворювано формувати фази навантаження і досліджувати реакцію AI-модуля та політик доступу без прямого втручання в логіку генераторів трафіку.

- Група System. Наявність діагностичних ендпоінтів є необхідною для відтворюваного запуску стенда та перевірки готовності компонентів та стану системи.

У таблиці 4.2 наведено перелік REST ендпоінтів веб-сервісу за групами Swagger/OpenAPI.

Таблиця 4.2

Перелік REST ендпоінтів веб-сервісу та призначення

Група	HTTP-Метод	Url	Призначення
Clients	**GET**	*/v1.0/clients*	Отримання списку клієнтів зі статусом (active/blocked);
Clients	**GET**	*/v1.0/clients/stats*	Отримання статистики TokenServer AP;
Clients	**DELETE**	*/v1.0/clients/stats*	Очищення статистики TokenServer API;
Clients	**POST**	*/v1.0/clients/{client_ip}/block*	Блокування клієнта за IP;
Clients	**POST**	*/v1.0/clients/{client_ip}/unblock*	Розблокування клієнта за IP;
Models	**GET**	*/v1.0/models*	Перелік доступних ML-моделей та їх статус;
Models	**GET**	*/v1.0/models/stats*	Отримання статистики рішень AI-агента;
Models	**DELETE**	*/v1.0/models/stats*	Очищення статистики рішень AI-агента;
Models	**POST**	*/v1.0/models/{model_key}/activate*	Активація ML-моделі за ключем;
IoT	**GET**	*/v1.0/devices*	Отримання переліку IoT-пристроїв (з IoT HUB через MQTT);
IoT	**POST**	*/v1.0/devices/{deviceUid}/set-status*	Встановлення стану IoT-пристрою (напрмклад, on/power_saving/off);
System	**GET**	*/health*	Перевірка працездатності веб-сервісу (healthcheck);

4.3.4. Web UI та візуалізація експериментальних даних

Для керування стендом і контролем перебігу експерименту в прикладній площині Cloud-AI-SDN фреймворку (рис. 4.13) реалізовано Web-сервіс, який працює поверх описаного REST API та виконує роль єдиного робочого середовища адміністратора. На відміну від взаємодії через запити вручну, Web-сервіс зменшує кількість таких дій, уніфікує типові операції (перемикання станів, оновлення статистики, запуск/зупинка сценаріїв) і забезпечує наочне відображення поточного стану системи. Web-сервіс не обходить

експериментальну логіку, усі дії виконуються через стандартні REST ендпоінти, а отже кожен крок може бути відтворений та задокументований.

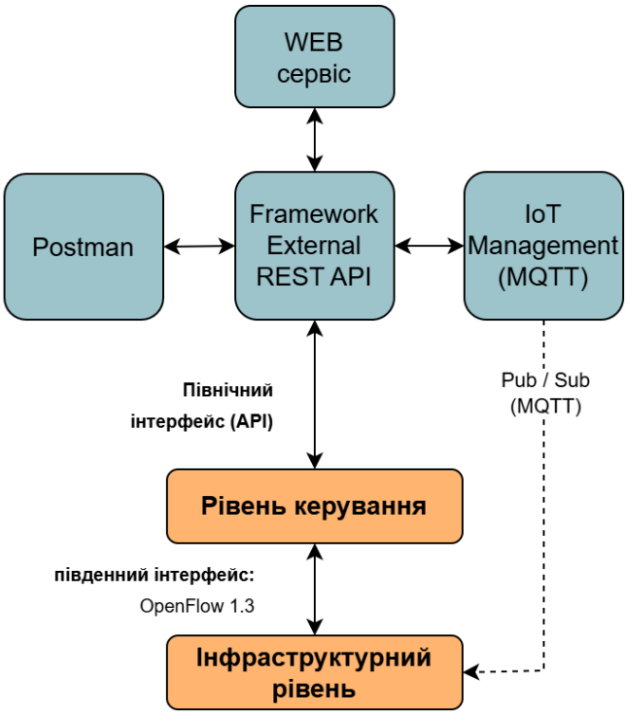


Рис. 4.13. Схема інтеграції веб-сервісу в Cloud-AI-SDN

Інтерфейс побудовано у вигляді трьох основних панелей, що відповідають функціональним доменам фреймворку:

1) Панель Clients призначена для контролю стану клієнтів і політик доступу до обмеженого ресурсу. У панелі відображається список IP-клієнтів зі статусом (active/blocked) та елементи керування блокуванням/розблокуванням. Окремо відображаються агреговані часові показники звернень (зокрема, за хвилинним вікном), що дозволяє швидко оцінити інтенсивність трафіку та наслідки керуючих дій. Практично це використовується як інструмент введення керованих змін трафіку (наприклад, блокування частини IP) та спостереження, чи коректно система припиняє обслуговування небажаного трафіку та як змінюється загальна динаміка звернень.

2) Панель ML Models забезпечує керування ML-моделями AI-агента в режимі реального часу, а саме - перегляд доступних і поточної активної моделі, перемикання моделі, отримання результатів роботи через статистику рішень. Ця

панель корисна для експериментальної методики порівняння, за однакових умов стенда (топологія, ліміти, профіль навантаження) можна змінювати лише активну модель і одразу бачити, як змінюються характеристики прийняття рішень (формування квот/обмежень, часові показники прийняття рішення, індекс справедливості розподілу).

3) Панель IoT Plane використовується для керування IoT-вузлами, що формують навантаження на стенд, через REST та MQTT. У панелі відображається перелік IoT-пристроїв, їх ідентифікатори та стан пристроїв, а також надаються інструменти зміни режимів роботи (on, power_saving, off). Це дозволяє формувати відтворювані фази навантаження без модифікації генераторів трафіку в ручному режимі та спостерігати реакцію AI-модуля і політик доступу в контрольованих умовах.

Розглянемо приклади використання Web UI для виконання типових керуючих операцій і верифікації результатів. На рисунку рис. 4.14 показано результат застосування політики блокування до клієна (10.0.0.9).

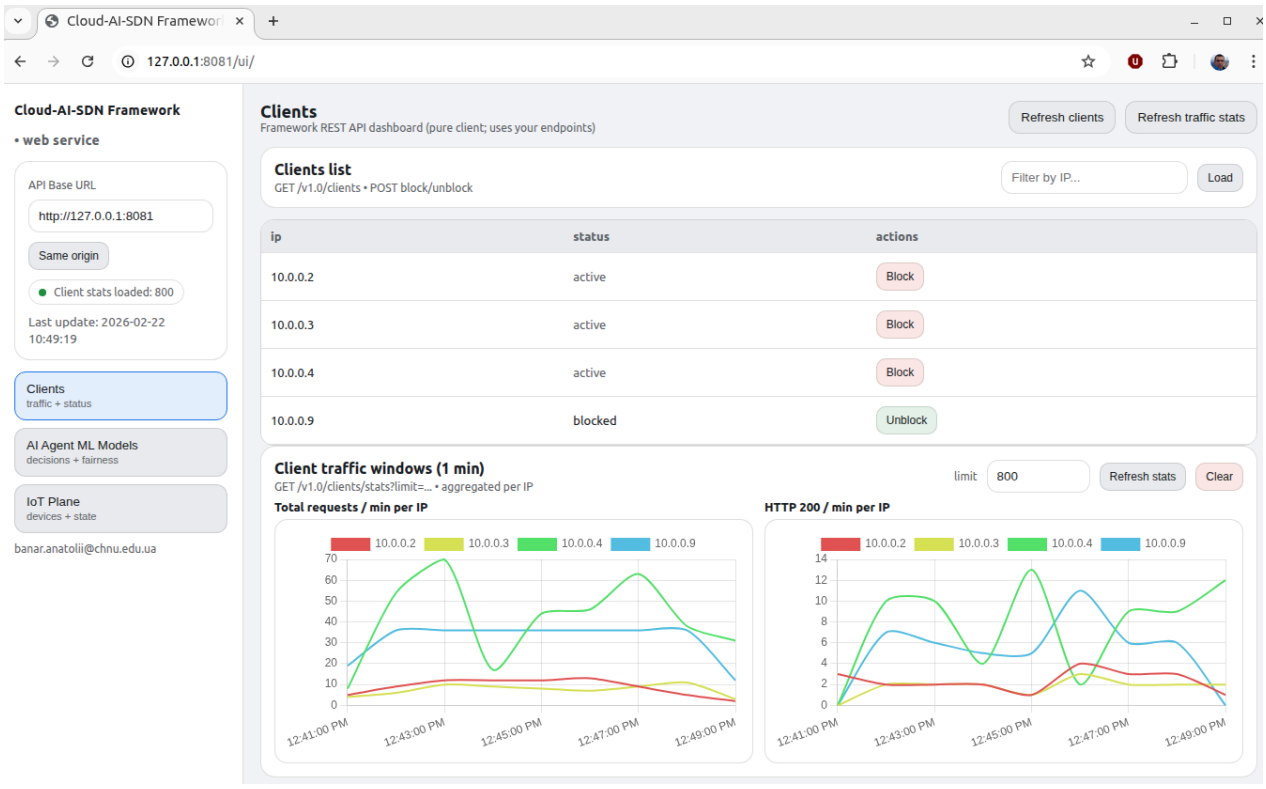


Рис. 4.14. Приклад блокування IP (10.0.0.9): трафік не надходить

Після блокування запити з відповідної адреси не проходять до обмеженого ресурсу, що візуально підтверджує коректність виконання керуючої дії та її миттєвий вплив на трафік та статистику.

У Web UI також передбачено граничний сценарій повного блокування клієнтів (рис. 4.15), який використовується як контрольна перевірка механізмів доступу та як швидкий спосіб заморозити стенд під час переналаштування експерименту (зміна активної ML-моделі, очищення логів, перебудова профілю навантаження).

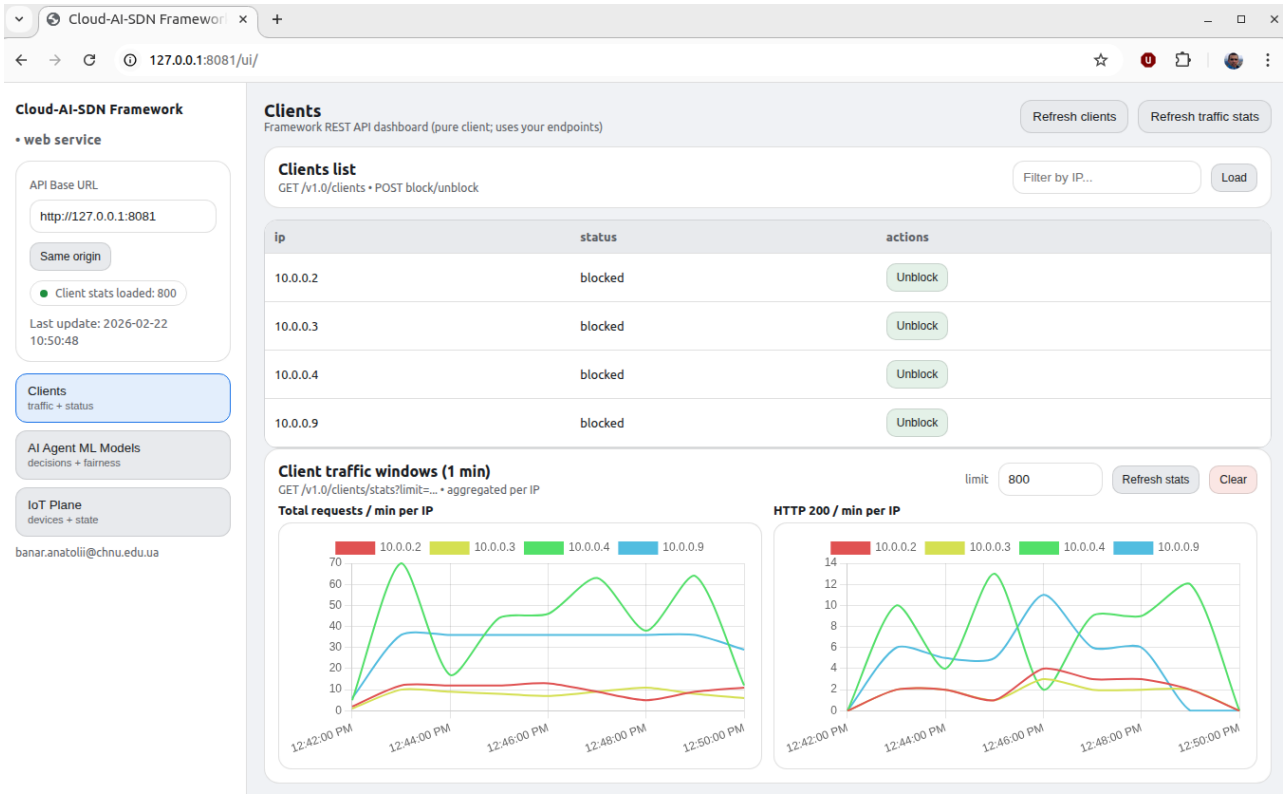


Рис. 4.15. Блокування всіх клієнтів

Перемикання активної ML-моделі AI-агента реалізовано як керуюча операція через Web UI, що необхідно для порівняння моделей у режимі реального часу (рис. 4.16). Після активації стенд продовжує роботу вже з новою моделлю машинного навчання для прийняття рішень.

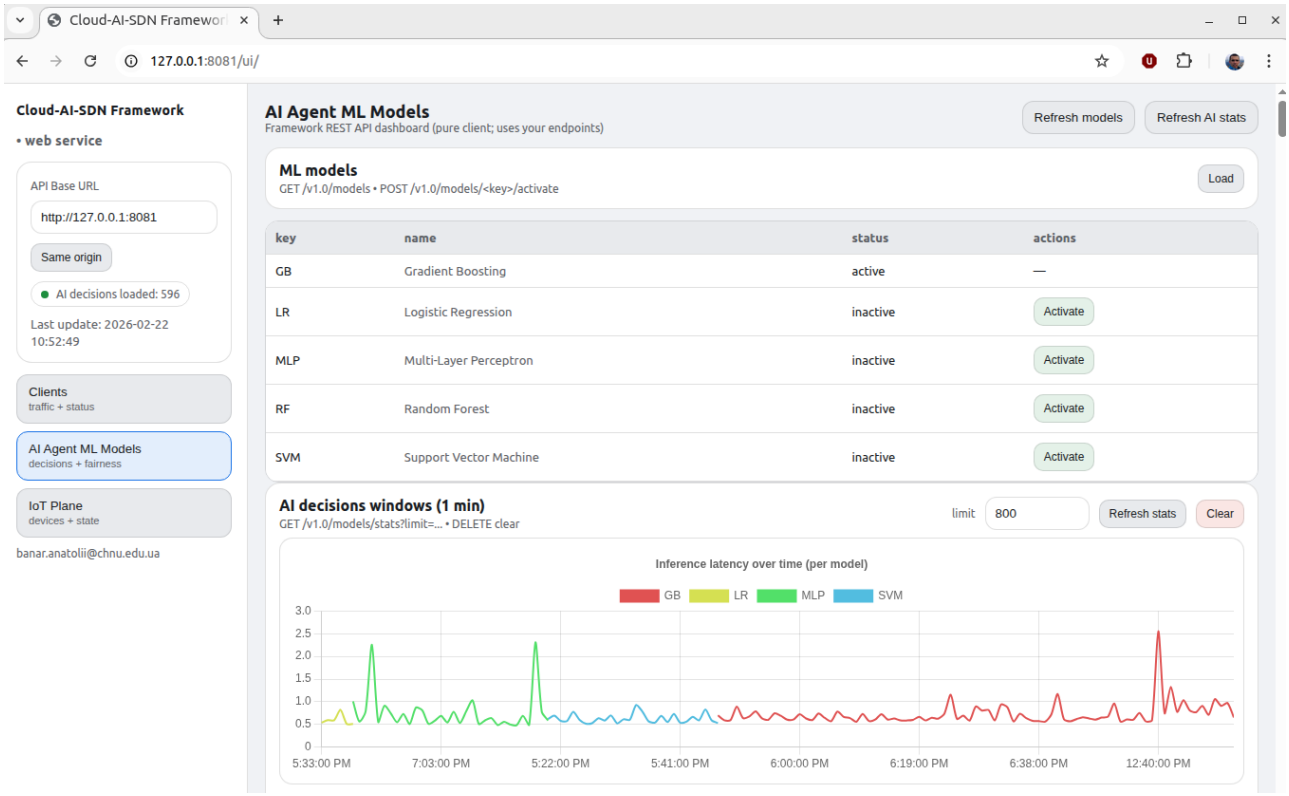


Рис. 4.16. Активація нової моделі (GB)

Після виконання керуючої дії (перемикання моделі або зміни статусу) інтерфейс відображає оновлений стан системи (рис. 4.17).



Рис. 4.17. Інтерфейс після зміни статусу ML моделі

У панелі IoT Plane (рис. 4.18) відображається перелік IoT-пристроїв, їх ідентифікатори (UID) та поточні стани енергоспоживання. Такий вигляд панелі використовується для швидкої верифікації, що IoT HUB доступний, пристрої видимі у системs, а їхні стани узгоджені з поточним експериментальним сценарієм навантаження.

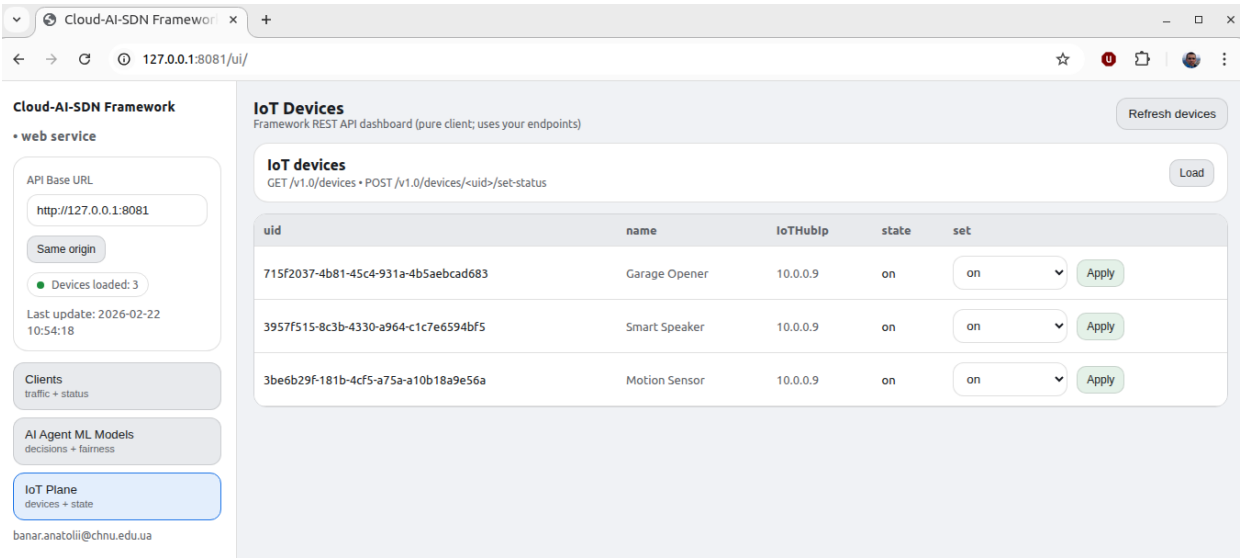


Рис. 4.18. Панель IoT Plane: перелік IoT-пристроїв та їх поточні стани

Зміна стану вибраного IoT-пристрою (рис. 4.19) виконується безпосередньо через Web UI як керуюча операція, що транслюється у виклик REST API (далі - у MQTT-команду до IoT HUB). Це дозволяє формувати фази навантаження без ручного втручання в генератори трафіку та без перезапуску компонентів стенда.

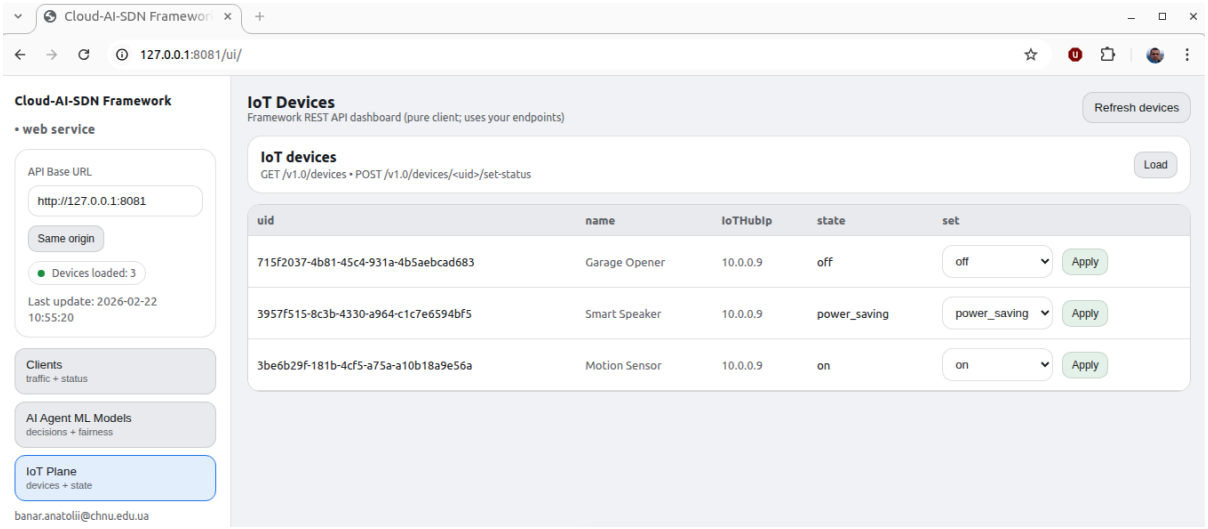


Рис. 4.19. Керування IoT-пристроєм: зміна стану через Web UI

Після виконання керуючої дії інтерфейс відображає оновлений стан IoT-пристрою, що використовується для підтвердження коректного застосування команди та актуальності даних у системі. Після зміни станів IoT-пристроїв у Web UI (наприклад, переведення Garage Opener у off, а Smart Speaker у power_saving) ці зміни негайно відображаються у консолі IoT Hub як оновлені значення поля state (рис. 4.20). У поєднанні з панелями Clients і ML Models, зміна стану IoT-площини (профілю навантаження) може бути одразу співставлена зі змінами в статистиці звернень та рішеннях AI-агента.

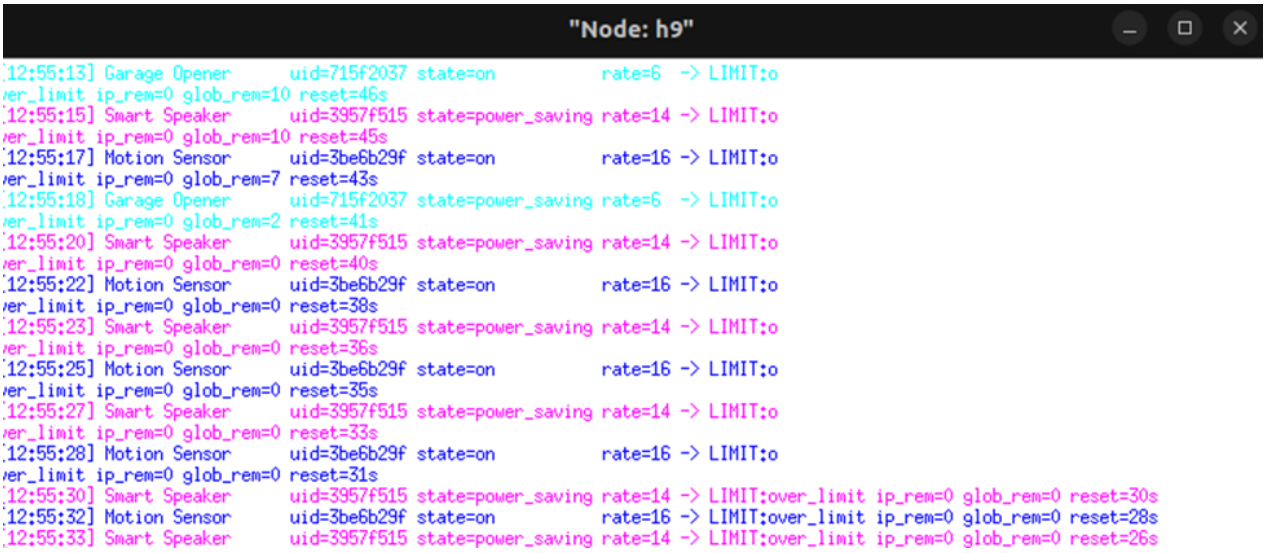


Рис. 4.20. Вікно терміналу IoT Hub:
стани IoT-пристроїв після керування через Web UI

У фреймворк інтегровано моніторинг системних метрик на базі Prometheus/Grafana для спостереження за станом вузла, на якому запущено експериментальний стенд. Prometheus здійснює регулярний збір показників (завантаження CPU, використання RAM, температура та ін.), а Grafana відображає їх у вигляді часових графіків (рис. 4.21).

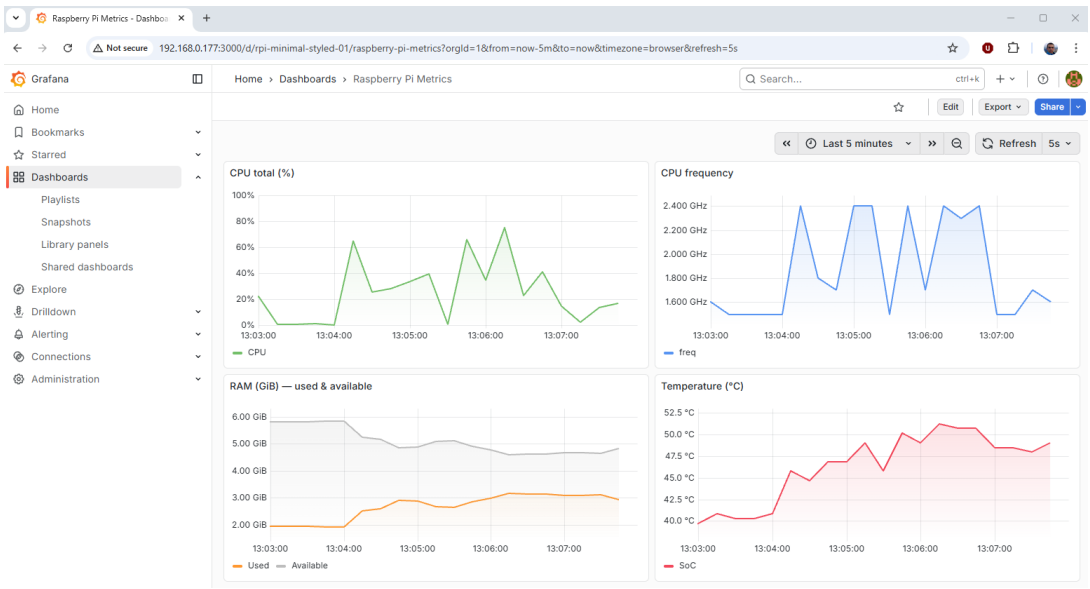


Рис. 4.21. Системні метрики під час роботи фреймворку

Такий моніторинг дозволяє контролювати, як змінюються системні ресурси під час роботи компонентів фреймворку, а також зіставляти ці зміни з подіями експерименту (зокрема фазами навантаження, перемиканням ML-моделей та моментами прийняття рішень AI-агентом). Це дає змогу пояснити коливання часу обробки або поведінки системи через зовнішні фактори (перегрів/тротлінг CPU, фонове навантаження, піки використання пам’яті).

4.4. Апаратна апробація запропонованого Cloud-AI-SDN фреймворку

У попередніх підпунктах розділу було підтверджено доцільність вибору ML-моделей для модуля прийняття рішень, апаратну здійсненність інтелектуального керування в умовах обмежених ресурсів, а також реалізацію прикладного веб-сервісу для керування та візуалізації. Наступним етапом є апаратна апробація, метою якої є підтвердження працездатності фреймворку як цілісної системи, де компоненти інфраструктурного, управління та прикладного рівнів працюють узгоджено у спільному циклі керування. Це дозволить перейти від оцінювання окремих метрик ML-моделей до аналізу працездатності всієї архітектури. Саме цілісна інтеграція всіх рівнів фреймворку дозволяє підтримувати необхідну якість обслуговування при динамічних змінах навантаження в гібридному середовищі.

Апробацію виконано на експериментальному стенді, узагальнену структуру якого наведено на (рис. 4.22). В якості центрального елементу апаратного стенду використано edge-вузол на базі Raspberry Pi, що виконує роль SDN-комутатора, вузла площини даних, та забезпечує підключення мережевих сегментів через кілька фізичних інтерфейсів - вбудований гігабітний Ethernet, дві USB-мережеві карти та Wi-Fi інтерфейс, який використовується як канал доступу до прикладного веб-сервісу.

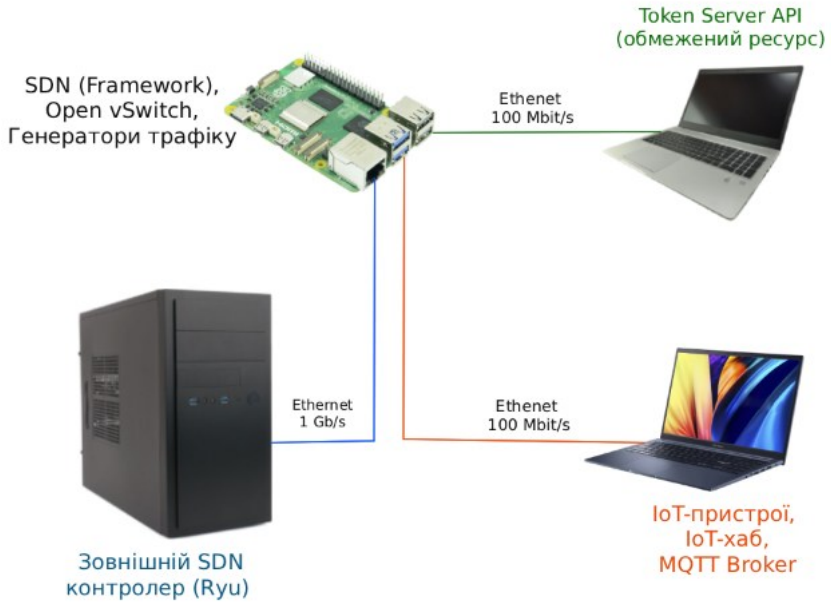


Рис. 4.22. Експериментальна схема для дослідження розробленого Cloud-AI-SDN фреймворку.

Така конфігурація дозволяє відокремити експериментальний трафік від трафіку адміністрування та спростити контроль для точок входу політик доступу. Загальний вигляд прототипу SDN-комутатора показано на рис. 4.23.

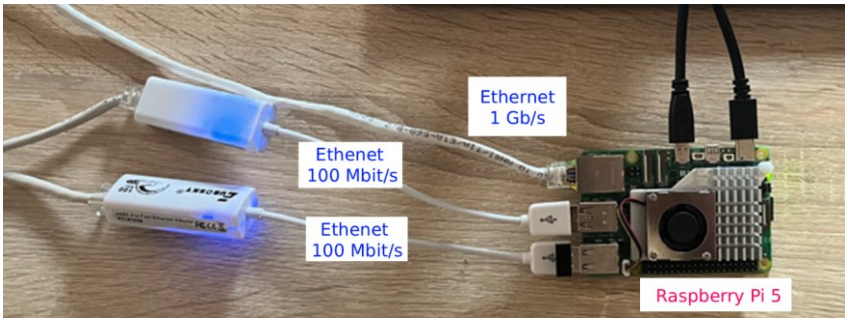
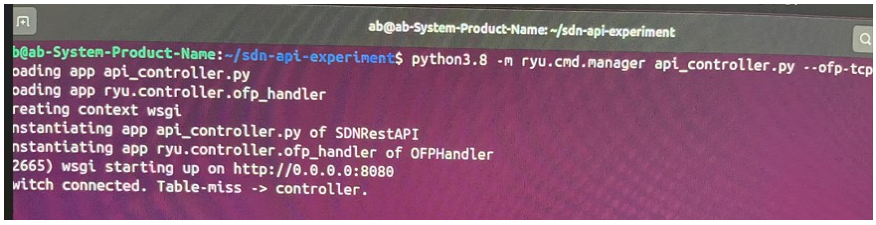
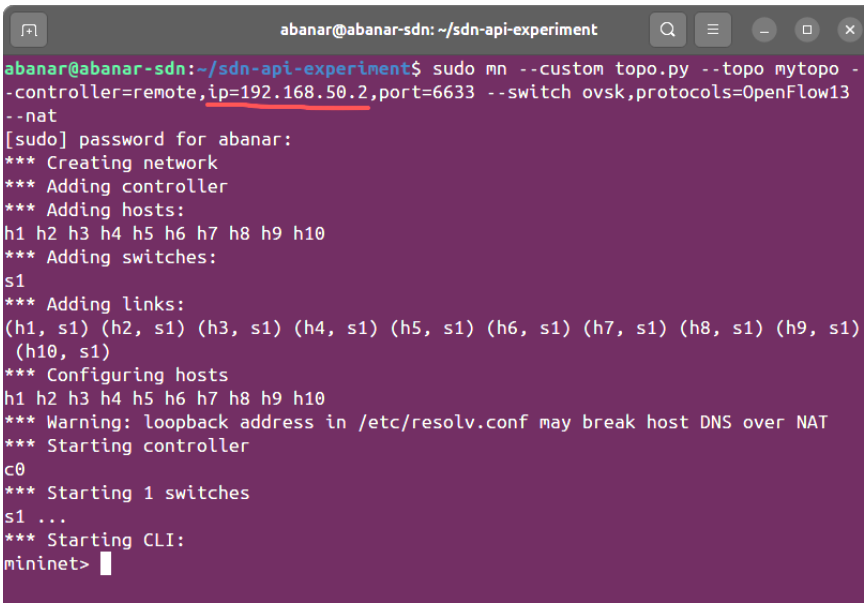


Рис. 4.23. Розроблений прототип SDN-комутатора на Raspberry Pi

SDN-контролер розміщено віддалено у середовищі приватної віртуальної хмари (в експерименті це окремий ПК у локальній мережі), для винесення площини керування за межі edge-вузла. Запуск віддаленого контролера наведено на рис. 4.24.а). Далі під час старту мережі (рис. 4.24.б) фіксується встановлення з'єднання між SDN-вузлом та контролером (switch connected). У SDN-системах на базі OpenFlow встановлення такого з'єднання супроводжується службовим обміном повідомленнями, після чого контролер ініціює запит конфігурації, а комутатор відповідає інформацією з актуальною конфігурацію на даних час. Далі можуть використовуватися Echo, Packet-In/Packet-Out та інші повідомлення для підтримки функцій керування та передачі даних. У даному сценарії важливим є сам факт стабільного встановлення каналу керування, оскільки саме через нього реалізується застосування політик та реакція на події/навантаження.



а)



б)

Рис. 4.24. Конфігурування площини керування: а) запуск віддаленого SDN-контролера у приватній хмарі, б) під'єднання SDN-вузла під час старту мережі

Фреймворк допускає варіативне розміщення компонентів (обмежений ресурс, MQTT-брокер, IoT-хаб, клієнти), перед запуском сценаріїв виконано актуалізацію конфігурації відповідно до фактичної топології. На рис. 4.25 показано приклад оновлень - параметри приведено у відповідність до розміщення вузла обмеженого ресурсу (Flask API) та MQTT-брокера на окремих хостах. Така конфігураційна політика необхідна для забезпечення відтворюваності експериментів, перенесення стенда в інше середовище не вимагає переписування коду, а зводиться до зміни конфігурації.

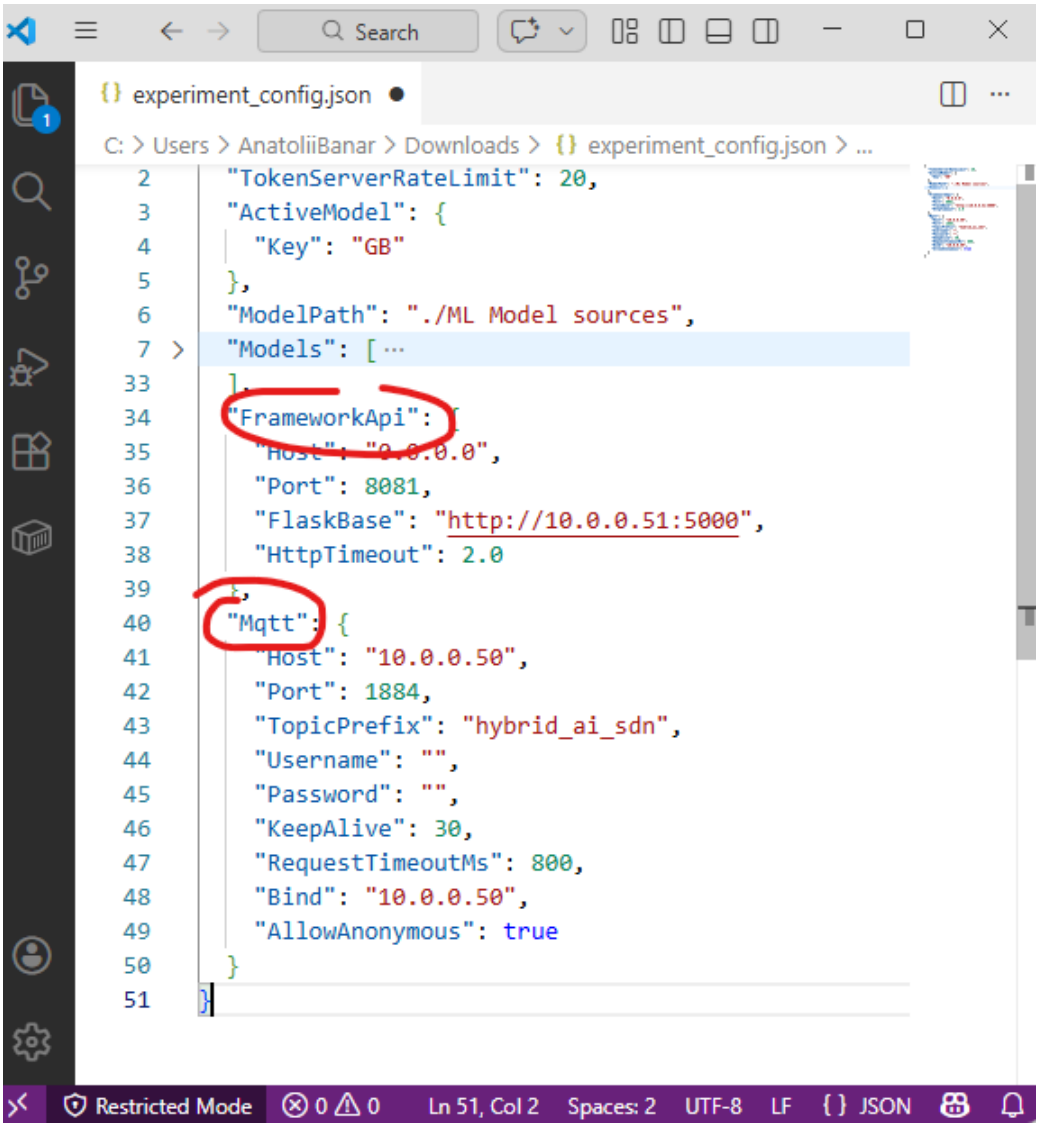


Рис. 4.25. Оновлення конфігурації відповідно до топології

Після встановлення з’єднання з контролером та узгодження конфігурації розгортається повний стенд дослідження (рис. 4.26), що включає компоненти

інфраструктурного рівня (мережеві вузли/трафік), рівня керування (SDN-контролер) та прикладного рівня (web-сервіс, візуалізація, операції керування). З огляду на вимоги до роботи в умовах обмежених ресурсів, важливою є наявність механізмів оперативного спостереження за активністю клієнтів, успішністю доступу до обмеженого ресурсу та ключовими показниками роботи AI-агента (час прийняття рішення, метрики справедливості), що реалізовано на прикладному рівні фреймворку та використовується у поточному сценарії як інструментальна підтримка апробації.

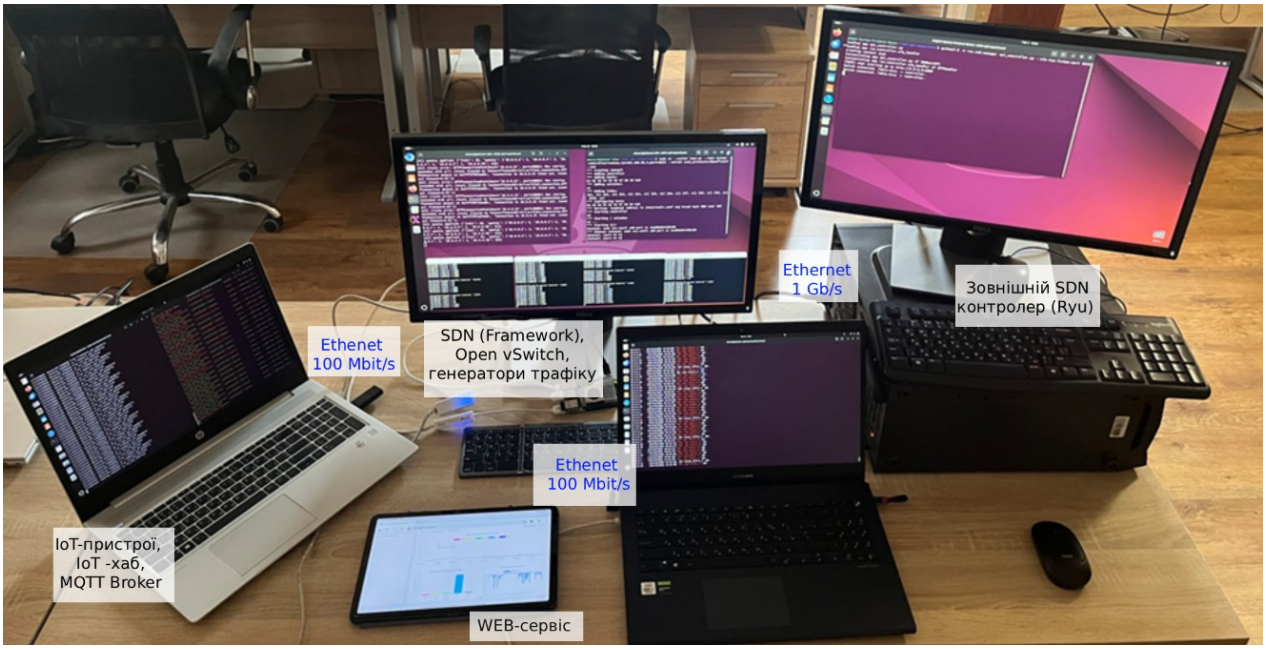


Рис. 4.26. Реалізований апаратний стенд розробленого фреймворку

Після статичного підтвердження конфігурації, наступним етапом є перехід до динамічних сценаріїв, які імітують типові ситуації керування в SDN-IoT середовищі, такі як зміна стану IoT-пристрою, зміна інтенсивності трафіку, застосування політики блокування/ізоляції активного джерела, а також спостереження роботи AI-агента в реальному часі.

Перший динамічний сценарій демонструє керування IoT-пристроями через MQTT із прикладного веб-сервісу. На рис. 4.27 показано інтерфейс зі списком IoT-пристроїв та операцію переведення CO2-сенсора в режим енергозбереження.

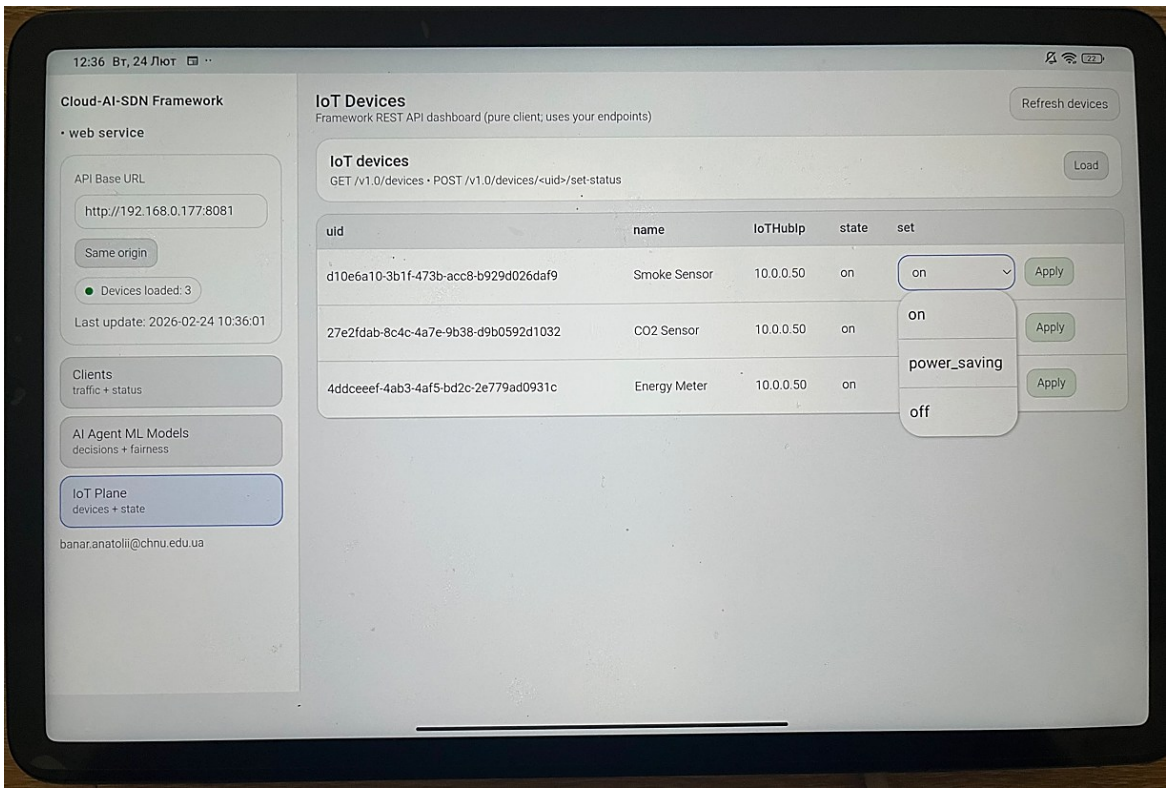


Рис. 4.27. Зміна стану IoT-пристрою використовуючи MQTT із веб-сервісу

Це дозволяє підтвердити технічну реалізацію каналу керування через MQTT-брокер та забезпечити умови для апробації інтелектуальних методів управління та можливість оперативного маніпулювання параметрами трафіку, і відповідно мати змогу дослідити поведінку AI-модуля та коректність спрацювання політик доступу в умовах обмежених ресурсів. Результат операції підтверджено на рис. 4.28, у консолі IoT-хабу відображається оновлений стан CO2-сенсора, а також очікуване зменшення активності, яке стає помітним у загальній кількості трафіку від вузла IoT-хабу.

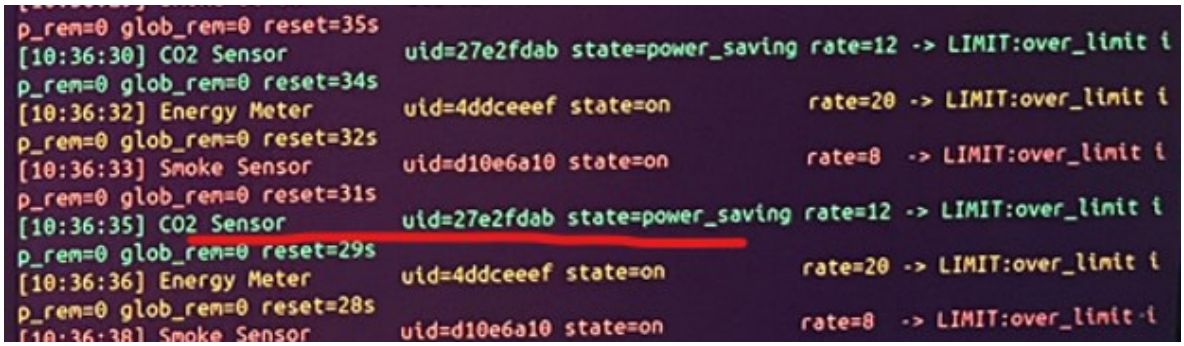


Рис. 4.28. Зміна енергоспоживання CO2-сенсора в консолі IoT-хабу

Наступний сценарій спрямований на перевірку політик доступу та реакції системи на «жорсткі» керувальні дії - блокування джерела трафіку. На рис. 4.29 наведено операцію блокування IP хоста IoT-хабу, одразу після застосування політики візуально спостерігається зменшення активності відповідного клієнта на графіку веб-інтерфейсу. Далі, на рис. 4.30 зафіксовано наслідок блокування - IP-адреса заблокована і запити не проходять.



Рис. 4.29. Блокування IP хоста IoT-хабу

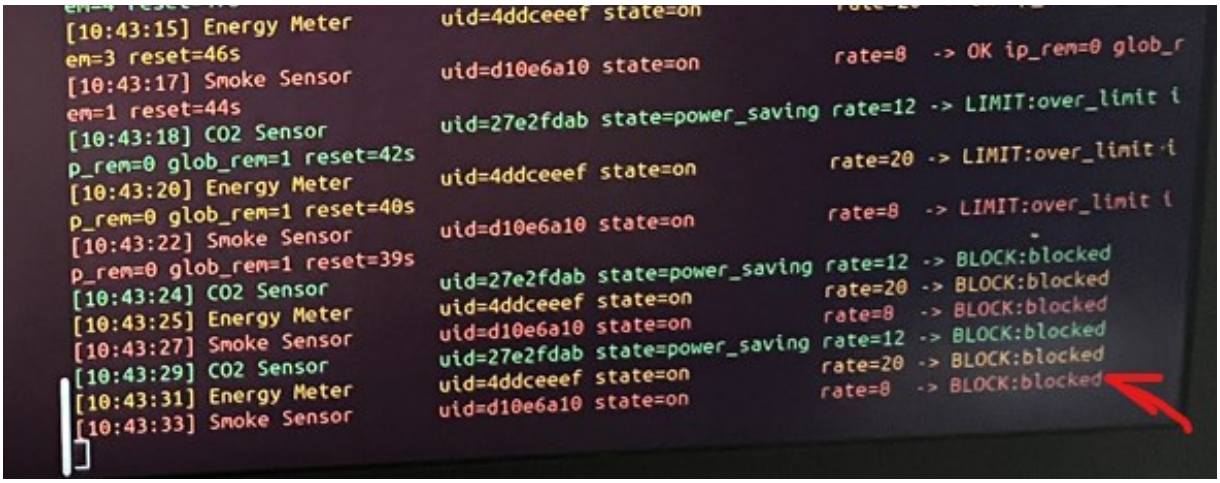


Рис. 4.30. Результат блокування IP хоста IoT-хабу

Завершальний сценарій апробації пов’язаний з аналізом роботи AI-агента в інтегрованій системі. На рис. 4.31 наведено робочі графіки, які узагальнюють як часові характеристики прийняття рішень ML-моделями, так і метрики справедливості розподілу ресурсу.



Рис. 4.31. Робочі графіки AI-агента: час прийняття рішення для різних моделей та динаміка індексу Джейна

Зафіксовано, що модель Random Forest на експериментальному стенді має найбільший час прийняття рішень, що узгоджується з результатами порівняльного аналізу в п. 4.1. Також індекс Джейна перебуває на стабільно високому рівні, поодинокі провали спостерігаються у моменти перемикання моделі, що є очікуваним ефектом при зміні конфігурації керування та перебудові політик доступу. Варто зазначити, що на даному етапі оцінюється не абстрактна якість вибраної ML -моделі , а її практична придатність в умовах інтеграції. Час прийняття рішення впливає на швидкість реакції керування, а справедливість на

здатність системи забезпечити прогнозований доступ клієнтів до обмеженого ресурсу під час сплесків навантаження та гарантувати заявлені показники QoS.

Таким чином, проведена апаратна апробація підтвердила працездатність запропонованої архітектури фреймворку умовах гібридного розміщення компонентів (edge-вузол і приватна хмара), а також продемонструвала експлуатаційні властивості:

- можливість реконфігурації топології та розміщення компонентів через параметри конфігурації;

- наявність керування IoT-площиною через MQTT із відображенням результатів у системі;

- підтримку політик ізоляції/блокування активних джерел трафіку;

- узгоджену роботу AI-агента з прогнозованою поведінкою метрик часу прийняття рішення та справедливості розподілу ресурсу. Окремо слід відзначити гнучкість практичної реалізації: фреймворк підтримує зміну компонентів на інші реалізації (наприклад, інший виробник обладнання, новий SDN контролер або масштабування, альтернативні реалізації Ml-моделей) без впливу на інтерфейс і роботу мережі, що забезпечує широкі можливості для подальших досліджень та масштабування запропонованих методів у реальні середовища.

Висновки до розділу 4

У розділі наведено результати експериментальної та апаратної апробації запропонованих підходів, які охоплюють порівняльну оцінку обраних ML-моделей для ШІ-модуля SDN, перевірку їх працездатності й накладних витрат у ресурсно-обмеженому середовищі, а також розробку й інтеграцію прикладного веб-сервісу керування роботою Cloud-AI-SDN фреймворку та SDN-мережею, забезпечення моніторингу стану компонентів і системних метрик. Отримані результати підтвердили ефективність запропонованих методів і технологій для покращення адаптивного керування доступом до обмежених ресурсів із збереженням стабільної роботи мережевої інфраструктури.

Показано, що вибір ML-моделі визначається компромісом між якістю керування та ресурсними витратами. Легковагові моделі характеризуються нижчими накладними витратами та кращою придатністю до роботи в режимі реального часу на edge-платформах, тоді як більш складні ансамблеві підходи здатні забезпечувати вищу якість прийняття рішень за умови збереження прийнятного рівня затримки циклу керування та стабільності роботи системи. Апаратні експерименти підтвердили, що навіть за обмежених CPU/RAM-ресурсів вибрані моделі можуть бути використані у складі SDN-контурів керування, а за результатами інтегральної оцінки доцільно застосовувати більш точну модель (GB) як основну для подальших досліджень, залишаючи простішу (LR) як резервну для сценаріїв із жорсткішими обмеженнями. При цьому встановлено, що використання моделі GB забезпечує вищу якість прийняття керуючих рішень і створює передумови для більш рівномірного розподілу доступу до обмеженого ресурсу без критичного зростання накладних витрат.

У межах практичної реалізації фреймворку розроблено та інтегровано прикладний веб-сервіс та інтерфейс взаємодії, що забезпечує керування компонентами експерименту та сценаріями IoT-контуру, а також підтримує відтворюваність проведення серій експериментів. Реалізація моніторингу системних метрик підвищила спостережуваність стенда й дозволила пов’язувати зміни в поведінці системи з фактичним станом апаратних ресурсів, що було необхідним для коректної інтерпретації результатів у ресурсно-обмежених умовах.

ВИСНОВКИ

У дисертаційній роботі розв’язано науково-практичне завдання покращення адаптивного керування доступом до обмежених ресурсів із забезпеченням рівномірного розподілу доступу та збереженням стабільної роботи мережевої інфраструктури телекомунікаційної мережі за рахунок розроблення та експериментального обґрунтування методів і технологій застосування алгоритмів штучного інтелекту в програмно-конфігурованих мережах. Запропонований підхід ґрунтується на поєднанні централізованого SDN-керування, телеметрії та прогнозної аналітики ШІ в межах єдиного циклу кервання, що дозволяє приймати адаптивні рішення з урахуванням ресурсного бюджету середовища виконання.

Показано, що інтеграція алгоритмів ШІ в SDN є доцільною насамперед як засіб підвищення адаптивності та автоматизації керування у динамічних мережевих умовах, однак вимагає ретельного балансування між точністю алгоритмів та витратами ресурсів на їх роботу. Обґрунтовано, що коректна оцінка ефективності AI-керування повинна спиратися не лише на традиційні метрики якості ML-моделей, а й на мережево-орієнтовані показники ефективності та справедливості розподілу обмежених ресурсів, оскільки саме вони визначають практичну результативність керувальних впливів для різних груп клієнтів/вузлів.

Запропоновано та реалізовано відтворювану гібридну архітектуру Cloud-AI-SDN фреймворку, яка забезпечує - керованість і програмованість мережі на рівні SDN-контролера, інтеграцію інтелектуальної аналітики в цикл прийняття рішень без жорсткої прив’язки до домену кінцевих вузлів, масштабованість і спостережуваність через сервісні компоненти моніторингу та керування. Показано, що архітектурно доцільним є винесення логіки роботи з ШІ в окремий сервіс, що зменшує ризик блокування контуру подій контролера та дозволяє керовано масштабувати обчислення.

Практична значущість запропонованого підходу полягає в тому, що він дозволяє SDN-контролеру через агента штучного інтелекту адаптивно обмежувати запити лише до критичного ресурсу, зберігаючи стабільний доступ

усіх клієнтів до решти мережевої інфраструктури та сервісів, які визначають загальну QoE. Тобто керування спрямовується на локалізацію дефіциту ресурсу у «вузькому місці», не перетворюючи обмеження на глобальну деградацію мережевих послуг.

Експериментально встановлено, що застосування AI-керування забезпечує покращення якості розподілу доступу до обмеженого ресурсу на 11,78% порівняно зі статичними правилами, що підтверджує досягнення мети дисертаційної роботи. Отриманий ефект досягнуто без втрати керованості системи та за умови контрольованих накладних витрат, що є принциповим для апаратних середовищ із обмеженими CPU/RAM-ресурсами. Виконано відбір і валідацію моделей машинного навчання для інтелектуального керування у запропонованому сценарії. Показано, що придатність моделей визначається компромісом «якість-витрати», а кінцевий вибір має враховувати не лише якість прийняття рішень, але й ресурсний бюджет та вимоги до прогнозованої реактивності керувального циклу. Проведена апаратна апробація на edge-платформі підтвердила практичну здійсненність реалізації AI-керування доступом до обмеженого ресурсу в SDN середовищі та придатність обраних моделей/архітектурних рішень для подальшого розвитку фреймворку та перенесення на прикладні сценарії.

Розроблено та інтегровано веб-сервіс керування Cloud-AI-SDN фреймворком і SDN-мережею, який забезпечує конфігурацію режимів роботи, керування доступом до обмежених ресурсів та спостережуваність стану компонентів і ключових метрик системи, що підвищує відтворюваність експериментів, інтерпретованість результатів і практичну придатність запропонованого рішення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bhuiyan Z. A., Islam S., Islam M. M., Ullah A. B. M. A., Naz F., Rahman M. S. On the (in)Security of the Control Plane of SDN Architecture: A Survey. *IEEE Access*. 2023. Vol. 11. P. 91550-91582. DOI: 10.1109/ACCESS.2023.3307467

2. Caccioli Capra F. Definition and Applications of SDN, NFV, Edge Computing and AI/ML Techniques. *Journal of Machine and Computing*. 2022. Vol. 2, no. 3. P. 103-113. DOI: 10.53759/7669/jmc202202015

3. Банар А., Воробець Г. Перспективні напрями розвитку, удосконалення і застосувань мережі SDN на основі методів штучного інтелекту. *Вісник Хмельницького національного університету. Серія: Технічні науки*. 2025. Т. 355, № 4. С. 15-21. DOI: 10.31891/2307-5732-2025-355-1

4. Open Flow Standard [Електронний ресурс]. URL: https://opennetworking.org/softwaredefined-standards/specifications (дата звернення: 12.12.2024)

5. Akyildiz I. F., Lee A., Wang P., Luo M., Chou W. A roadmap for traffic engineering in SDN-OpenFlow networks. *Computer Networks*. 2014. Vol. 71. P. 1-30. DOI: 10.1016/j.comnet.2014.06.002

6. Al-Najjar A., Khan F. H., Portmann M. Network traffic control for multi-homed end-hosts via SDN. *IET Communications*. 2020. Vol. 14. P. 3312-3323. DOI: 10.1049/iet-com.2020.0335

7. DeLany R., Smith A., Li Y., Du L. SDN Dynamic Controller Configuration to Mitigate Compromised Controllers. *2023 IEEE Transportation Electrification Conference & Expo (ITEC)*, Detroit, MI, USA, 2023. P. 1-5. DOI: 10.1109/ITEC55900.2023.10186974

8. Priyadarsini M., Bera P. Software defined networking architecture, traffic management, security, and placement: A survey. *Computer Networks*. 2021. Vol. 192. Art. № 108047. DOI: 10.1016/j.comnet.2021.108047

9. Diouf M., Ouya S., Klein J., Bissyandé T. Software Security in Software-Defined Networking: A Systematic Literature Review. *arXiv*. 2025. arXiv:2502.13828. DOI: 10.48550/arXiv.2502.13828

10. Yujie R., Muqing W., Yiming C. An Effective Controller Placement Algorithm Based on Clustering in SDN. *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, Chengdu, China, 2020. P. 2294-2299. DOI: 10.1109/ICCC51575.2020.9345045

11. Manso C., Vilalta R., Casellas R., Martínez R., Muñoz R. Cloud-native SDN Controller Based on Micro-Services for Transport Networks. *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, Ghent, Belgium, 2020. P. 365-367. DOI: 10.1109/NetSoft48620.2020.9165377.

12. Pérez R., Rivera M., Salgueiro Y., Baier C. R., Wheeler P. Moving Microgrid Hierarchical Control to an SDN-Based Kubernetes Cluster: A Framework for Reliable and Flexible Energy Distribution. *Sensors*. 2023. Vol. 23. Art. № 3395. DOI: 10.3390/s23073395

13. Lin F. P.-C., Tsai Z. Hierarchical Edge-Cloud SDN Controller System With Optimal Adaptive Resource Allocation for Load-Balancing. *IEEE Systems Journal*. 2020. Vol. 14, no. 1. P. 265-276. DOI: 10.1109/JSYST.2019.2894689

14. Банар А. Ю., Воробець Г. І. Особливості і переваги застосування штучного інтелекту у хмарній реалізації SDN контролера. *Physical and technological problems of transmission, processing and storage of information in infocommunication systems (PREDT - 2025): Proceedings of Xth International Scientific-Practical Conference* (Chernivtsi, 15-17 May 2025). С. 189-190. URL: https://drive.google.com/drive/folders/1WtGHyXGLooQgBSCqNW32qOVOmj0-bQr8

15. Банар А., Воробець Г. Хмарні SDN-контролери з підтримкою ШІ: архітектура, масштабованість та безпека (порівняльне дослідження). *Безпека інфокомунікаційних систем та Інтернету речей*. 2025. Т. 3, № 1. С. 01011:1-6. DOI: 10.31861/sisiot2025.1.01011

16. Rostami M., Goli-Bidgoli S. An overview of QoS-aware load balancing techniques in SDN-based IoT networks. *Journal of Cloud Computing*. 2024. Vol. 13. Art. № 89. DOI: 10.1186/s13677-024-00651-7

17. Ospina Cifuentes B. J., Suárez Á., García Pineda V., Alvarado Jaimes R., Montoya Benitez A. O., Grajales Bustamante J. D. Analysis of the use of artificial intelligence in software-defined intelligent networks: A survey. *Technologies*. 2024. Vol. 12. Art. № 99. DOI: 10.3390/technologies12070099

18. Rahouti M., Xiong K., Xin Y. Secure Software-Defined Networking Communication Systems for Smart Cities: Current Status, Challenges, and Trends. *IEEE Access*. 2021. Vol. 9. P. 12083-12113. DOI: 10.1109/ACCESS.2020.3047996.

19. Krishnan P., Jain K., Aldweesh A., Prabu P., Buyya R. OpenStackDP: a scalable network security framework for SDN-based OpenStack cloud infrastructure. *Journal of Cloud Computing*. 2023. Vol. 12. Art. № 26. DOI: 10.1186/s13677-023-00406-w

20. Taniguchi Y., Tsutsumi H., Iguchi N., Watanabe K. Design and Evaluation of a Proxy-Based Monitoring System for OpenFlow Networks. *The Scientific World Journal*. 2016. Art. № 6513649. DOI: 10.1155/2016/6513649

21. Nunes B. A., Mendonca M., Nguyen X.-N., Obraczka K., Turletti T. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Communications Surveys & Tutorials*. 2014. Vol. 16, no. 3. P. 1617-1634. DOI: 10.1109/SURV.2014.012214.00180

22. Zhou W., Li L., Luo M., Chou W. REST API Design Patterns for SDN Northbound API. *Proceedings of the 2014 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, Victoria, BC, Canada, 2014. P. 358-365. DOI: 10.1109/WAINA.2014.153

23. Krishna S. M., Sharma R. Survey on Application Programming Interfaces in Software Defined Networks and Network Function Virtualization. *Global Transitions Proceedings*. 2021. Vol. 2, no. 2. P. 468-474. DOI: 10.1016/j.gltp.2021.08.018

24. Doriguzzi-Corin R., Salvadori E., Aranda Gutiérrez P. A., Stritzke C., Leckey A., Phemius K., Rojas E., Guerrero C. NetIDE: Removing vendor lock-in in SDN. *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. 2015. P. 1-2. DOI: 10.1109/NETSOFT.2015.7116170

25. Du S., Lee J., Kim K. Proposal of gRPC as a New Northbound API for Application Layer Communication Efficiency in SDN. *Proceedings of the 2018 ACM International Conference on Information Networking (ICOIN)*. 2018. P. 1-6. DOI: 10.1145/3164541.3164563

26. Zhao Y., Li Y., Zhang X., Geng G., Zhang W., Sun Y. A Survey of Networking Applications Applying the Software Defined Networking Concept Based on Machine Learning. *IEEE Access*. 2019. Vol. 7. P. 95397-95417. DOI: 10.1109/ACCESS.2019.2928564

27. 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/s44230-023-00025-3

28. Lantz B., Heller B. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-IX)*. 2010. P. 19:1-19:6. DOI: 10.1145/1868447.1868466

29. Gil B., Trezentos P. Impacts of Data Interchange Formats on Energy Consumption and Performance in Smartphones. *Proceedings of the 2011 Workshop on Open Source and Design of Communication (OSDOC '11)*, Lisboa, Portugal, 2011. P. 1-6. DOI: 10.1145/2016716.2016718

30. Braun W., Menth M. Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices. *Future Internet*. 2014. Vol. 6, no. 2. P. 302-336. DOI: 10.3390/fi6020302

31. Dong L., Chen L., He B., Wang W. The Research on Designs of Multiple Flow Tables in the OpenFlow Protocol. *Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN)*, Hangzhou, China, 2018. P. 1-2. DOI: 10.1109/ICCCN.2018.8487378

32. Qiao S., Hu C., Guan X., Zou J. Taming the Flow Table Overflow in OpenFlow Switch. *Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM '16)*, Florianopolis, Brazil, 2016. P. 591-592. DOI: 10.1145/2934872.2959063

33. Nallusamy P., Saravanen S., Krishnan M. Decision Tree-Based Entries Reduction Scheme Using Multi-Match Attributes to Prevent Flow Table Overflow in

SDN Environment. *International Journal of Network Management*. 2021. Vol. 31, no. 4. Art. № e2141. DOI: 10.1002/nem.2141

34. Huang L., Shen Q., Zhou F., Shao W., Cui X. Nonuniform bandwidth reservation for tunnels in MPLS network using meter tables of OpenFlow. *Transactions on Emerging Telecommunications Technologies*. 2018. Vol. 29, iss. 5. Art. № e3286. DOI: 10.1002/ett.3286

35. Schiff L., Borokhovich M., Schmid S. Reclaiming the Brain: Useful OpenFlow Functions in the Data Plane. In: *Proceedings of the 13th ACM Workshop on Hot Topics in Networks (HotNets-XIII)*. Los Angeles, CA, USA, Association for Computing Machinery, 2014. P. 1-7. DOI: 10.1145/2670518.2673874

36. Banerjee S., Kannan K. Tag-In-Tag: Efficient flow table management in SDN switches. In: *Proceedings of the 10th International Conference on Network and Service Management (CNSM 2014)*. 2014. P. 109-117. DOI: 10.1109/CNSM.2014.7014147

37. Ma H., Yang Y., Mi Z. A distributed storage framework of FlowTable in software defined network. *Computers & Electrical Engineering*. 2015. Vol. 43. P. 155-168. DOI: 10.1016/j.compeleceng.2014.10.012

38. Garrich Alabarce M., Bravalheri A. Overview of South-Bound Interfaces for Software-Defined Optical Networks. In: *Proceedings of the 20th International Conference on Transparent Optical Networks (ICTON 2018)*. Bucharest, Romania, 2018. P. 1-5. DOI: 10.1109/ICTON.2018.8473494

39. Щур В., Кулаков Ю. Аналіз сучасних методів балансування навантаження в SDN мережах. *Herald of Khmelnytskyi National University. Technical Sciences*. 2023. Т. 323, № 4. С. 352-357. DOI: 10.31891/2307-5732-2023-323-4-352-357

40. Ряба Н. М., Правило В. В. Забезпечення виконання заданих показників якості обслуговування в мережах SDN. *Збірник матеріалів Міжнародної науково-технічної конференції «Перспективи телекомунікацій»*. 2018. URL: https://conferenc-journal.its.kpi.ua/article/view/131516/127598 (дата звернення: 14.11.2024).

41. Єфанова К. А., Пономаренко З. М., Масесов М. О. Актуальність впровадження та використання мережі SDN. *Сучасні інформаційні технології у сфері безпеки та оборони*. 2018. Т. 32, № 2. С. 147-150. DOI: 10.33099/2311-7249/2018-32-2-147-150

42. Prabakaran P., Isravel D. P., Silas S. A Review of SDN-Based Next Generation Smart Networks. *2019 3rd International Conference on Computing and Communications Technologies (ICCCT)*, Chennai, India, 2019. P. 80-85. DOI: 10.1109/ICCCT2.2019.8824871

43. Даков С. Ю. Метод підвищення ефективності SDN мережі за допомогою керуючого програмного забезпечення. *Вісник Хмельницького національного університету. Технічні науки*. 2018. № 6(2). С. 216-223. DOI: 10.31891/2307-5732-2018-267-6(2)-216-223

44. Бовда Е. М. Метод управління перерозподілом навантаження в SDN мережах. *Збірник наукових праць Військового інституту телекомунікацій та інформатизації*. 2017. Вип. 2. С. 6-15. URL: http://nbuv.gov.ua/UJRN/Znpviti_2017_2_3 (дата звернення: 10.12.2024)

45. Пелех П. Оцінка QoE на основі вимірювання QoS для відео реального часу у безпровідних мережах. *Herald of Khmelnytskyi National University. Technical Sciences*. 2024. Т. 337, № 3(2). С. 237-245. DOI: 10.31891/2307-5732-2024-337-3-35

46. Marshoodulla S. Z., Saha G. An Approach towards Removal of Data Heterogeneity in SDN-Based IoT Framework. *Internet of Things*. 2023. Vol. 22. Art. № 100763. DOI: 10.1016/j.iot.2023.100763

47. Rahouti M., Xiong K., Xin Y., Jagatheesaperumal S. K., Ayyash M., Shaheed M. SDN Security Review: Threat Taxonomy, Implications, and Open Challenges. *IEEE Access*. 2022. Vol. 10. P. 45820-45854. DOI: 10.1109/ACCESS.2022.3168972

48. Abdulqadder I. H., Zou D., Aziz I. T., Yuan B., Li W. SecSDN-Cloud: Defeating Vulnerable Attacks Through Secure Software-Defined Networks. *IEEE Access*. 2018. Vol. 6. P. 8292-8301. DOI: 10.1109/ACCESS.2018.2797214

49. Jiang J., Lin C., Han G., Abu-Mahfouz A. M., Shah S. B. H., Martínez-García M. How AI-enabled SDN technologies improve the security and functionality of industrial IoT network: Architectures, enabling technologies, and opportunities. *Digital Communications and Networks*. 2023. Vol. 9, no. 6. P. 1351-1362. DOI: 10.1016/j.dcan.2022.07.001

50. Guo A., Yuan C. Network Intelligent Control and Traffic Optimization Based on SDN and Artificial Intelligence. *Electronics*. 2021. Vol. 10. P. 700. DOI: 10.3390/electronics10060700

51. Wassie G., Ding J., Wondie Y. Traffic prediction in SDN for explainable QoS using deep learning approach. *Scientific Reports*. 2023. Vol. 13. Art. № 20607. DOI: 10.1038/s41598-023-46471-8

52. Mohammadi R., Akleylek S., Ghaffari A., Shirmarz A. Automatic delay-sensitive applications quality of service improvement with deep flows discrimination in software defined networks. *Cluster Computing*. 2023. Vol. 26. P. 437-459. DOI: 10.1007/s10586-022-03729-6

53. Alhilali A. H., Montazerolghaem A. Artificial intelligence based load balancing in SDN: A comprehensive survey. *Internet of Things*. 2023. Vol. 22. Art. № 100814. DOI: 10.1016/j.iot.2023.100814

54. Xiang M., Chen M., Wang D., Luo Z. Deep Reinforcement Learning-based load balancing strategy for multiple controllers in SDN. *e-Prime - Advances in Electrical Engineering, Electronics and Energy*. 2022. Vol. 2. Art. № 100038. DOI: 10.1016/j.prime.2022.100038

55. Siddiqui S., Hameed S., Shah S. A., Ahmad I., Aneiba A., Draheim D., Dustdar S. Toward Software-Defined Networking-Based IoT Frameworks: A Systematic Literature Review, Taxonomy, Open Challenges and Prospects. *IEEE Access*. 2022. Vol. 10. P. 70850-70901. DOI: 10.1109/ACCESS.2022.3188311

56. Prasad V. K., Dansana D., Bhavsar M. D., Acharya B., Gerogiannis V. C., Kanavos A. Efficient Resource Utilization in IoT and Cloud Computing. *Information*. 2023. Vol. 14, no. 11. Art. № 619. DOI: 10.3390/info14110619

57. Lilhore U. K., Simaiya S., Sharma Y. K., Rai A. K., Padmaja S. M., Vajid K. N., Kumar V., Alroobaea R., Alsufyani H. Cloud-Edge Hybrid Deep Learning Framework for Scalable IoT Resource Optimization. *Journal of Cloud Computing*. 2025. Vol. 14, no. 1. Art. № 5. DOI: 10.1186/s13677-025-00729-w

58. Eghbali Z., Zolfy Lighvan M. A hierarchical approach for accelerating IoT data management process based on SDN principles. *Journal of Network and Computer Applications*. 2021. Vol. 181. Art. № 103027. DOI: 10.1016/j.jnca.2021.103027

59. Firouzi R., Rahmani R. A Distributed SDN Controller for Distributed IoT. *IEEE Access*. 2022. Vol. 10. P. 42873-42882. DOI: 10.1109/ACCESS.2022.3168299

60. Jiménez-Lázaro M., Herrera J. L., Berrocal J., Galán-Jiménez J. Improving the Energy Efficiency of Software-Defined Networks through the Prediction of Network Configurations. *Electronics*. 2022. Vol. 11. Art. № 2739. DOI: 10.3390/electronics11172739

61. Udayaprasad P. K., Shreyas J., Srinidhi N. N., Kumar S. M. D., Dayananda P., Askar S. S., Abouhawwash M. Energy Efficient Optimized Routing Technique With Distributed SDN-AI to Large Scale I-IoT Networks. *IEEE Access*. 2024. Vol. 12. P. 2742-2759. DOI: 10.1109/ACCESS.2023.3346679

62. Pérez-Díaz J. A., Valdovinos I. A., Choo K.-K. R., Zhu D. A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning. *IEEE Access*. 2020. Vol. 8. P. 155859-155872. DOI: 10.1109/ACCESS.2020.3019330.

63. Банар А. Ю., Воробець Г. І. Open-source фреймворк для моделювання та оптимізації SDN-мереж із використанням алгоритмів штучного інтелекту. *Проблеми інформатики та комп'ютерної техніки (ПІКТ - 2025): праці XIV Міжнар. наук.-практ. конф.* (м. Чернівці, 13-15 лист. 2025 р.). С. 22-24. URL: https://drive.google.com/file/d/12FGgnfM6NA8HPo66h0TgVrfbPHAYDMFC/view

64. Mozo A., Karamchandani A., de la Cal L., Gómez-Canaval S., Pastor A., Gifre L. A Machine-Learning-Based Cyberattack Detector for a Cloud-Based SDN Controller. *Applied Sciences*. 2023. Vol. 13. Art. № 4914. DOI: 10.3390/app13084914

65. Wani A., Revathi S., Khaliq R. SDN-Based Intrusion Detection System for IoT Using Deep Learning Classifier (IDSIoT-SDL). *CAAI Transactions on Intelligent Technology*. 2021. Vol. 6, no. 3. P. 281-290. DOI: 10.1049/cit2.12003

66. Pathak Y., Prashanth P. V. N., Tiwari A. AI Meets SDN: A Survey of Artificial Intelligent Techniques Applied to Software-Defined Networks. In: Kumar M., Gill S. S., Samriya J. K., Uhlig S. *6G Enabled Fog Computing in IoT*. Springer. DOI: 10.1007/978-3-031-30101-8_16

67. Банар А. Ю. Перспективні напрями оптимізації і захисту даних в SDN мережі з використанням штучного інтелекту. *Проблеми інформатики та комп'ютерної техніки (ПІКТ - 2023): праці XII Міжнар. наук.-практ. конф.* (м. Чернівці, 10-12 лист. 2023 р.). С. 91-92. URL: https://drive.google.com/file/d/10m9pfYczudmIanv0na1MgZqMSoGyFPA-/edit

68. Xu T., Chen C., Hu K., Zhuang Y. A Game Model and Fault Recovery Algorithm for SDN Multi-Domain. *Sensors*. 2025. Vol. 25, no. 1. Art. № 164. DOI: 10.3390/s25010164

69. Tong V., Souihi S., Tran H. A., Mellouk A. Machine Learning based Root Cause Analysis for SDN Network. *2021 IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, 2021. P. 1-6. DOI: 10.1109/GLOBECOM46510.2021.9685185

70. Панченко О.М. Адаптивне управління ресурсами та якістю обслуговування у програмно-конфігурованих сервісноорієнтованих телекомунікаційних мережах: дис. ... д-ра філософії: 172 – Телекомунікації та радіотехніка / Панченко Олексій Михайлович. – Львів, 2021. – 200 с

71. Банар А. Ю., Воробець Г. І. Побудова та управління SDN мереж з використанням методів штучного інтелекту. *Інтелектуальні інформаційні системи: Матеріали всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів* (м. Миколаїв, 4-5 груд. 2025 р.). С. 10-12. URL: https://dspace.chmnu.edu.ua/jspui/handle/123456789/3054

72. Varoquaux G., Buitinck L., Louppe G., Grisel O., Pedregosa F., Mueller A. Scikit-learn: Machine Learning Without Learning the Machinery. *GetMobile: Mobile*

Computing and Communications. 2015. Vol. 19, no. 1. P. 29-33. DOI: 10.1145/2786984.2786995

73. Menghani G. Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. *ACM Computing Surveys*. 2023. Vol. 55, no. 12. Art. № 259. DOI: 10.1145/3578938

74. Silva D., Carvalho L. I., Soares J., Sofia R. C. A Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA. *Applied Sciences*. 2021. Vol. 11, no. 11. Art. № 4879. DOI: 10.3390/app11114879

75. Hmissi F., Ouni S. SDN-DMQTT: SDN-Based Platform for Re-configurable MQTT Distributed Brokers Architecture. In: Zaslavsky A., Ning Z., Kalogeraki V., Georgakopoulos D., Chrysanthis P. K. (eds.). *Mobile and Ubiquitous Systems: Computing, Networking and Services*. Cham: Springer Nature Switzerland, 2024. P. 393-411. DOI: 10.1007/978-3-031-63992-0_26

76. Nguyen L., Ha S., Le T., Hoang Luong H., Vo K., Nguyen K., Nguyen A., Dao T., Nguyen H. BMDD: A novel approach for IoT platform (broker-less and microservice architecture, decentralized identity, and dynamic transmission messages). *PeerJ Computer Science*. 2022. Vol. 8. Art. № e950. DOI: 10.7717/peerj-cs.950

77. Park J.-H., Kim H.-S., Kim W.-T. DM-MQTT: An Efficient MQTT Based on SDN Multicast for Massive IoT Communications. *Sensors*. 2018. Vol. 18, no. 9. Art. № 3071. DOI: 10.3390/s18093071

78. Husnain M., Hayat K., Cambiaso E., Fayyaz U. U., Mongelli M., Akram H., Abbas S. G., Shah G. A. Preventing MQTT Vulnerabilities Using IoT-Enabled Intrusion Detection System. *Sensors*. 2022. Vol. 22, no. 2. Art. № 567. DOI: 10.3390/s22020567

79. Fawwaz D. Z., Chung S. H., Ahn C. W., Kim W. S. Optimal Distributed MQTT Broker and Services Placement for SDN-Edge Based Smart City Architecture. *Sensors*. 2022. Vol. 22, no. 9. Art. № 3431. DOI: 10.3390/s22093431

80. Tseng Y. H., Wang C., Wei Y. T., Chiang Y. T. Cloud-edge MQTT messaging for latency mitigation and broker memory footprint reduction. *PeerJ Computer Science*. 2025. Vol. 11. Art. № e2741. DOI: 10.7717/peerj-cs.2741

81. Sokolova M., Japkowicz N., Szpakowicz S. Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation. In: Sattar A., Kang Bh. (eds). *AI 2006: Advances in Artificial Intelligence*. Lecture Notes in Computer Science. 2006. Vol. 4304. Springer, Berlin, Heidelberg. DOI: 10.1007/11941439_114

82. Cabot J. H., Ross E. G. Evaluating prediction model performance. *Surgery*. 2023. Vol. 174, no. 3. P. 723-726. DOI: 10.1016/j.surg.2023.05.023

83. Chapanduka G., Nleya B., Chidzonga R. Towards Fair and QoS-Aware Bandwidth Allocation in Next-Generation Multi-Gigabit WANs. *Electronics*. 2025. Vol. 14, no. 23. Art. № 4658. DOI: 10.3390/electronics14234658

84. Max-Min Fairness. *ScienceDirect*. URL: https://www.sciencedirect.com/topics/computer-science/max-min-fairness (дата звернення: 06.11.2025).

85. Gupta N., Maashi M. S., Tanwar S., Badotra S., Aljebreen M., Bharany S. A Comparative Study of Software Defined Networking Controllers Using Mininet. *Electronics*. 2022. Vol. 11. Art. № 2715. DOI: 10.3390/electronics11172715

86. Sultana N., Chilamkurti N., Peng W., Alhadad R. Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications*. 2019. Vol. 12, no. 2. P. 493-501. DOI: 10.1007/s12083-017-0630-0

87. Garba U. H., Toosi A. N., Pasha M. F., Khan S. SDN-Based Detection and Mitigation of DDoS Attacks on Smart Homes. *Computer Communications*. 2024. Vol. 221. P. 29-41. DOI: 10.1016/j.comcom.2024.04.001

88. Kreutz D., Ramos F. M. V., Veríssimo P. E., Rothenberg C. E., Azodolmolky S., Uhlig S. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*. 2015. Vol. 103, no. 1. P. 14-76. DOI: 10.1109/JPROC.2014.2371999

89. Nadeem M. W., Goh H. G., Ponnusamy V., Aun Y. DDoS Detection in SDN Using Machine Learning Techniques. *Computers, Materials & Continua*. 2021. Vol. 71, no. 1. P. 771-789. DOI: 10.32604/cmc.2022.021669

90. Salau A. O., Beyene M. M. Software defined networking based network traffic classification using machine learning techniques. *Scientific Reports*. 2024. Vol. 14. Art. № 20060. DOI: 10.1038/s41598-024-70983-6

91. Saito T., Rehmsmeier M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLOS ONE*. 2015. Vol. 10, no. 3. Art. № e0118432. DOI: 10.1371/journal.pone.0118432

92. Банар А. Ю., Воробець Г. І. Алгоритми штучного інтелекту для оптимізації функціонування SDN: сучасні підходи та перспективи. *Зв'язок*. 2025. № 4. С. 11-18. DOI: 10.31673/2412-9070.2025.041241

93. Cox D. R. The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*. 1958. Vol. 20, no. 2. P. 215-232. DOI: 10.1111/j.2517-6161.1958.tb00292.x

94. Breiman L. Random Forests. *Machine Learning*. 2001. Vol. 45, no. 1. P. 5-32. DOI: 10.1023/A:1010933404324

95. Friedman J. H. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*. 2001. Vol. 29, no. 5. P. 1189-1232. DOI: 10.1214/aos/1013203451

96. Nyaramneni S., Potluri T., Somavarapu J. Advanced Ensemble Machine Learning Models to Predict SDN Traffic. *Procedia Computer Science*. 2023. Vol. 230. P. 417-426. DOI: 10.1016/j.procs.2023.12.097

97. Cortes C., Vapnik V. Support-Vector Networks. *Machine Learning*. 1995. Vol. 20. P. 273-297. DOI: 10.1007/BF00994018

98. Li X., Yuan D., Hu H., Ran J., Li S. DDoS Detection in SDN Switches using Support Vector Machine Classifier. *Proceedings of the 2015 Joint International Mechanical, Electronic and Information Technology Conference*. Atlantis Press, 2015. P. 344-348. DOI: 10.2991/jimet-15.2015.63

99. Rumelhart D. E., Hinton G. E., Williams R. J. Learning Representations by Back-Propagating Errors. *Nature*. 1986. Vol. 323. P. 533-536. DOI: 10.1038/323533a0

100. McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Shenker S., Turner J. OpenFlow: Enabling Innovation in Campus

Networks. *ACM SIGCOMM Computer Communication Review*. 2008. Vol. 38, no. 2. P. 69-74. DOI: 10.1145/1355734.1355746

101. OpenFlow Switch Specification. URL: https://yuba.stanford.edu/~derickso/openflow-spec-v1.0.0-cookieenhancements.pdf (дата звернення: 06.11.2025)

102. Cluster Configuration in Owl. URL: https://wiki.onosproject.org/pages/viewpage.action?pageId=28836788 (дата звернення: 06.11.2025).

103. Yeganeh S. H., Ganjali Y. Kandoo: A Framework for Efficient and Scalable Offloading of Control Applications. *Proceedings of the 1st ACM International Workshop on Hot Topics in Software Defined Networks (HotSDN '12)*. 2012. P. 19-24. DOI: 10.1145/2342441.2342446

104. Bhardwaj S., Panda S. N. Performance Evaluation Using RYU SDN Controller in Software-Defined Networking Environment. *Wireless Personal Communications*. 2022. Vol. 122. P. 701-723. DOI: 10.1007/s11277-021-08920-3

105. Prete L. R., Shinoda A. A., Schweitzer C. M., de Oliveira R. L. S. Simulation in an SDN Network Scenario Using the POX Controller. *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, Bogota, Colombia, 2014. P. 1-6. DOI: 10.1109/ColComCon.2014.6860403

106. Murillo-Calderon F. M., Hidalgo-Oñate A. G. Real-Time Network Information Management Using Floodlight in Convergence with SDN. In: Garcia M. V., Reyes J. P., Nuñez C., Gordón-Gallegos C. (eds). *Proceedings of the International Conference on Computer Science, Electronics and Industrial Engineering (CSEI 2024)*. Lecture Notes in Networks and Systems. 2026. Vol. 1516. Springer, Cham. DOI: 10.1007/978-3-031-98768-7_46

107. Shah S. A., Faiz J., Farooq M., Shafi A., Mehdi S. A. An Architectural Evaluation of SDN Controllers. *2013 IEEE International Conference on Communications (ICC)*, Budapest, Hungary, 2013. P. 3504-3508. DOI: 10.1109/ICC.2013.6655093

108. Berde P., Gerola M., Hart J., Higuchi Y., Kobayashi M., Koide T., Lantz B., O'Connor B., Radoslavov P., Snow W., Parulkar G. ONOS: Towards an Open, Distributed SDN OS. *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking (HotSDN '14)*, Chicago, Illinois, USA, 2014. P. 1-6. DOI: 10.1145/2620728.2620744

109. Sakic E., Kellerer W. Response Time and Availability Study of RAFT Consensus in Distributed SDN Control Plane. *IEEE Transactions on Network and Service Management*. 2018. Vol. 15, no. 1. P. 1-14. DOI: 10.1109/TNSM.2017.2775061

110. Medved J., Varga R., Tkacik A., Gray K. OpenDaylight: Towards a Model-Driven SDN Controller Architecture. *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Sydney, NSW, Australia, 2014. P. 1-6. DOI: 10.1109/WoWMoM.2014.6918985

111. Singh A., Kaur N., Kaur H. Extensive Performance Analysis of OpenDayLight (ODL) and Open Network Operating System (ONOS) SDN Controllers. *Microprocessors and Microsystems*. 2022. Vol. 95. Art. № 104715. DOI: 10.1016/j.micpro.2022.104715

112. Warraich S. H., Aziz Z., Khurshid H., Hameed R., Saboor A., Awais M. SDN Enabled and OpenFlow Compatible Network Performance Monitoring System. arXiv e-prints. 2020. arXiv:2005.07765. DOI: 10.48550/arXiv.2005.07765

113. Gordon H., Batula C., Tushir B., Dezfouli B., Liu Y. Securing Smart Homes via Software-Defined Networking and Low-Cost Traffic Classification. *Proceedings of the 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, Madrid, Spain, 2021. P. 1049-1057. DOI: 10.1109/COMPSAC51774.2021.00143

114. Arzo S. T., Scotece D., Bassoli R., Barattini D., Granelli F., Foschini L., Fitzek F. H. P. MSN: A Playground Framework for Design and Evaluation of MicroServices-Based SDN Controller. *Journal of Network and Systems Management*. 2022. Vol. 30, no. 1. Art. № 19. DOI: 10.1007/s10922-021-09631-7

115. Salazar-Chacón G. D., Marrone L. OpenSDN Southbound Traffic Characterization: Proof-of-Concept Virtualized SDN-Infrastructure. *Proceedings of the 2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, BC, Canada, 2020. P. 0282-0287. DOI: 10.1109/IEMCON51383.2020.9284938

116. Banar A., Vorobets H. AI-based adaptive management of limited resources in SDN-IoT ecosystems. *Radioelectronic and Computer Systems*. 2025. Vol. 2025, no. 4. P. 154-170. DOI: 10.32620/reks.2025.4.11

117. Díaz-Montiel A. A., Lantz B., Yu J., Kilper D., Ruffini M. Real-Time QoT Estimation Through SDN Control Plane Monitoring Evaluated in Mininet-Optical. *IEEE Photonics Technology Letters*. 2021. Vol. 33, no. 18. P. 1050-1053. DOI: 10.1109/LPT.2021.3075277

118. Ryu application programming model. URL: https://ryu.readthedocs.io/en/latest/ryu_app_api.html (дата звернення: 09.11.2025)

ДОДАТКИ

Додаток А

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації:

***Наукові праці у виданнях, включених до переліку наукових фахових видань України та проіндексованих у наукометричній базі даних Scopus*:**

1. **Banar A.,** Vorobets H. AI-based adaptive management of limited resources in SDN-IoT ecosystems. *Radioelectronic and Computer Systems*. 2025. Vol. 2025. № 4. P. 154-170. DOI: 10.32620/reks.2025.4.11, (Scopus, Q3, https://www.scimagojr.com/journalsearch.php?q=21101038702&tip=sid&clean=0). *(Внески авторів: **Банар А.Ю**. - огляд літератури, розробка методу управління SDN-мережею на базі інтелектуальних агентів, побудова топології мережі та архітектури фреймворку, налаштування інтелектуального управління, програмна реалізація алгоритмів керування, підготовка тексту роботи. Воробець Г.І. - формулювання концепції інтелектуального управління мережею, визначення вимог до системи управління, наукове редагування тексту.)*

Наукові праці у виданнях, що входять до переліку наукових фахових видань України:

2. **Банар А. Ю.,** Воробець Г. І. Перспективні напрями розвитку, удосконалення і застосувань мережі SDN на основі методів штучного інтелекту. *Вісник Хмельницького національного університету. Серія: Технічні науки*. 2025. Т. 355. № 4. С. 15-21. DOI: 10.31891/2307-5732-2025-355-1 *(Внески авторів: **Банар А.Ю.** - аналіз архітектурних особливостей SDN та викликів безпеки, систематизація методів штучного інтелекту для завдань балансування трафіку та забезпечення QoS/QoE, розробка класифікації загроз та методів їх протидії у програмно-конфігурованих мережах, підготовка тексту статті та візуалізацій. Воробець Г.І. - постановка завдання дослідження, визначення*

концепції інтеграції ШІ в SDN, аналіз огляду літератури та наукове редагування рукопису.)

3. **Банар А. Ю.,** Воробець Г. І. Алгоритми штучного інтелекту для оптимізації функціонування SDN: сучасні підходи та перспективи. *Зв'язок*. 2025. № 4. С. 11-18. DOI: 10.31673/2412-9070.2025.041241 *(Внески авторів: **Банар А. Ю.** - дослідження та порівняльний аналіз ефективності алгоритмів машинного навчання для завдань SDN, розробка класифікації напрямів інтеграції ШІ (безпека, маршрутизація, енергоефективність), формування порівняльної таблиці ефективності методів, написання основного тексту. Воробець Г.І. - формування структури дослідження, перевірка результатів порівняльного аналізу, формулювання висновків щодо обмежень використання ШІ, науковий огляд рукопису перед друком.)*

4. **Банар А. Ю.,** Воробець Г. І. Хмарні SDN-контролери з підтримкою ШІ: архітектура, масштабованість та безпека (порівняльне дослідження). *Безпека інфокомунікаційних систем та Інтернету речей*. 2025. Т. 3. № 1. С. 01011:1-6. DOI: 10.31861/sisiot2025.1.01011 *(Внески авторів: **Банар А.Ю**. - дослідження архітектур локальних та хмарних SDN-контролерів (монолітна, мікросервісна, ієрархічна), аналіз проблем масштабованості та затримок у хмарному середовищі, аналіз сценаріїв інтеграції AI-модулів у хмарну інфраструктуру, підготовка тексту та графічних матеріалів. Воробець Г.І. - концептуалізація порівняльного дослідження, визначення критеріїв оцінки архітектур, наукове редагування статті.)*

Наукові праці, які засвідчують апробацію матеріалів дисертації:

5. **Банар А. Ю**. Перспективні напрями оптимізації і захисту даних в SDN мережі з використанням штучного інтелекту. *Проблеми інформатики та комп'ютерної техніки (ПІКТ - 2023):* праці XII Міжнар. наук.-практ. конф. (м. Чернівці, 10 - 12 лист. 2023 р.). С. 91-92.

6. **Банар А. Ю.**, Воробець Г. І. Особливості і переваги застосування штучного інтелекту у хмарній реалізації SDN контролера. *Physical and*

technological problems of transmission, processing and storage of information in infocommunication systems (PREDT - 2025): Proceedings of Xth International Scientific-Practical Conference. (Чернівці, 15-17 травня 2025). С. 189-190. *(Внески авторів:* ***Банар А.Ю.*** *- проведення порівняльного аналізу архітектур хмарних SDN-контролерів та оцінка їх масштабованості, дослідження аспектів інтеграції ШІ-модулів для керування трафіком, редагування тексту тез, розробка презентаційних матеріалів та очний виступ із доповіддю на конференції. Воробець Г.І. - постановка завдання дослідження, науковий огляд питань безпеки хмарних рішень, підготовка тексту тез, перевірка матеріалів доповіді.)*

7. **Банар А. Ю**., Воробець Г. І. Open-source фреймворк для моделювання та оптимізації SDN-мереж із використанням алгоритмів штучного інтелекту. *Проблеми інформатики та комп'ютерної техніки (ПІКТ - 2025):* праці XIV Міжнар. наук.-практ. конф. (м. Чернівці, 13 - 15 лист. 2025 р.). С. 22-24. *(Внески авторів:* ***Банар А.Ю.*** *- обґрунтування вибору open-source інструментів для реалізації SDN-фреймворку, програмна реалізація компонентів архітектури та проведення експериментального моделювання, написання тез, підготовка презентаційних матеріалів та очний виступ із пленарною доповіддю. Воробець Г.І. - огляд використаного відкритого програмного забезпечення для досліджень в SDN, формулювання завдань роботи, перевірка матеріалів доповіді.)*

8. **Банар А. Ю**., Воробець Г. І. Побудова та управління SDN мереж з використанням методів штучного інтелекту. *Інтелектуальні інформаційні системи*: Матеріали всеукраїнської науково-практичної конференції молодих вчених, аспірантів і студентів. (м. Миколаїв, 4-5 груд. 2025 р.). С. 10-12. *(Внески авторів:* ***Банар А.Ю.*** *- розробка архітектури та побудова експериментальної інтелектуальної SDN-платформи, реалізація алгоритмів для адаптивного розподілу ресурсів, підготовка тексту тез та виступ із доповіддю. Воробець Г.І. - формулювання концепції об'єднання SDN та ШІ для гетерогенних систем, визначення вимог до відтворюваності експериментів, перевірка матеріалів доповіді та підготовка до виступу.)*

Додаток Б

Відомості про апробацію результатів дисертації

- Науковий семінар кафедри радіотехніки та інформаційної безпеки Чернівецького національного університету імені Юрія Федьковича (м. Чернівці, 13 березня 2026), форма участі - очна.

- Міжнародний науково-технічний семінар «Критичні комп'ютерні технології та системи (КриКТехС)» (м. Харків, 29 січня 2026), форма участі - дистанційна.

- П'ятнадцята міжнародна конференція «Dependable Systems, Services and Technologies (DESSERT'2025)» (Греція, Афіни, 19-21 грудня 2025), форма участі - дистанційна.

- Всеукраїнська науково-практична конференція молодих вчених, аспірантів і студентів «Інтелектуальні інформаційні системи» (м. Миколаїв, 4-5 грудня 2025), форма участі - дистанційна з публікацією тез.

- Чотирнадцята міжнародна конференція «Проблеми інформатики та комп'ютерної техніки (ПІКТ-2025)» (м. Чернівці, 13-15 листопада 2025), форма участі - очна з публікацією тез.

- Десята міжнародна науково-практична конференція «Physical and technological problems of transmission, processing and storage of information in infocommunication systems (PREDT - 2025)» (м. Чернівці, 15 травня 2025), форма участі - очна з публікацією тез.

- Дванадцята міжнародна конференція «Проблеми інформатики та комп'ютерної техніки (ПІКТ-2023)» (м. Чернівці, 10 - 12 листопада 2023), форма участі - очна з публікацією тез.

Додаток В

Акти впровадження результатів дисертаційної роботи

Акт
впровадження результатів дисертаційної роботи
Банара Анатолія Юрійовича
на тему: «Методи і технології використання алгоритмів штучного інтелекту в програмно-конфігурованих мережах в умовах обмежених ресурсів», представленої до захисту на здобуття наукового ступеня доктора філософії з галузі знань 17 – «Електроніка та телекомунікації» за спеціальністю 172 – «Телекомунікації та радіотехніка»

Цей акт підтверджує, що «Yukon Software Ltd» використала та впровадила результати, представлені в дисертаційному дослідженні здобувача ступеня доктора філософії Чернівецького національного університету імені Юрія Федьковича Банара Анатолія Юрійовича, а саме:

- У межах науково-дослідної діяльності компанії впроваджено метод адаптивного керування доступом до обмеженого ресурсу прикладного рівня в SDN, що забезпечує локалізацію керуючого впливу в точці дефіциту ресурсу та підвищує рівномірність розподілу доступу між користувачами;
- Використано підхід до вибору моделі машинного навчання для AI-керування в умовах обмежених ресурсів за розширеними критеріями (якість керування, час прийняття рішення, завантаження обчислювальних ресурсів, енергоспоживання), що дозволило підвищити практичну придатність впровадження інтелектуальних механізмів керування в ресурсно-обмежених середовищах.

Використання запропонованих методів і технологій дозволило підвищити рівень автоматизації та адаптивності керування доступом до дефіцитних сервісів, зменшити вплив пікових навантажень на критичні компоненти інфраструктури та покращити стабільність надання послуг в умовах обмежених ресурсів.

10.11.2025 Директор «Yukon Software Ltd» _____ Шкурей М. Р.

Акт
впровадження результатів дисертаційної роботи
Банара Анатолія Юрійовича
на тему: «Методи і технології використання алгоритмів штучного інтелекту в програмно-конфігурованих мережах в умовах обмежених ресурсів», представленої до захисту на здобуття наукового ступеня доктора філософії в галузі знань 17 – «Електроніка та телекомунікації» за спеціальністю 172 – «Телекомунікації та радіотехніка»

Цей акт підтверджує, що ТОВ «ШАРПМАЙНДЗ ЮЕЙ» використало та впровадило результати, представлені в дисертаційному дослідженні здобувача ступеня доктора філософії Чернівецького національного університету імені Юрія Федьковича Банара Анатолія Юрійовича, а саме:

- Впроваджено метод адаптивного керування доступом до обмеженого ресурсу прикладного рівня в програмно-конфігурованій мережі, що передбачає локалізацію керуючого впливу безпосередньо в точці дефіциту ресурсу та забезпечує рівномірний розподіл доступу без коригування загальних характеристик мережі;
- Впроваджено удосконалену архітектуру інтелектуальної SDN-мережі, яка об'єднує мережеві, прикладні та IoT-компоненти в єдиному контурі керування. Застосування засобів аудиту, автентифікації та подієвої взаємодії в межах цієї архітектури дозволило підвищити стійкість інфраструктури до збоїв та забезпечити відтворюваність сценаріїв управління;
- Застосовано підхід вибору моделі машинного навчання для AI-керування в умовах обмежених ресурсів на основі розширених критеріїв (класифікаційні показники, час прийняття рішення, енергоспоживання), що дозволяє обирати модель керування відповідно до ресурсних обмежень середовища та вимог конкретного сценарію.

Результати роботи дозволяють організації підвищити рівень наукоємності власних розробок у сфері телекомунікацій, та дозволяють пропонувати інноваційні рішення у сфері проєктування інтелектуальних SDN-мереж, адаптованих до роботи в умовах обмежених ресурсів.

12.02.2026

Головний
операційний директор
Гордієнко А. Ю.

«Затверджую»
Директор Навчально-наукового
інституту фізико-технічних та
комп'ютерних наук Чернівецького
національного університету
імені Юрія Федьковича
кандидат технічних наук, доцент


Петро ШПАТАР

Акт
впровадження в навчальному процесі результатів дисертаційної роботи
Банара Анатолія Юрійовича
«Методи і технології використання алгоритмів штучного інтелекту в програмно-конфігурованих мережах в умовах обмежених ресурсів»
представленої до захисту на здобуття наукового ступеня доктора філософії з галузі знань 17 – «Електроніка та телекомунікації», за спеціальністю 172 – «Телекомунікації та радіотехніка»

Комісія Чернівецького Національного університету імені Юрія Федьковича в складі:

заступника директора ННІФТКН з наукової роботи, доктора фізико-математичних наук, професора Дуболазова О. В.;

завідувача кафедри радіотехніки та інформаційної безпеки, доктора технічних наук, професора Саміли А.П.;

завідувача кафедри комп'ютерних систем та мереж, кандидата фізико-математичних наук, доцента Воробця Г. І.,

склали цей акт про те, що результати дисертаційної роботи здобувача наукового ступеня доктора філософії за спеціальністю «Телекомунікації та радіотехніка» Банара А.Ю. впроваджені в навчальний процес кафедр РТІБ та КСМ, зокрема в дисциплінах: «Захист та моніторинг комп'ютерних мереж», «Мережні інформаційні технології Cisco», та «Комп'ютерні мережі».

При викладанні цих дисциплін використовуються наступні матеріали досліджень, які отримані в дисертаційній роботі:

- Методика побудови та конфігурування програмно-конфігурованої мережі (SDN), що включає етапи формування топології, налаштування SDN-контролера та правил керування трафіком, організацію збору телеметрії, відтворюваний запуск експериментальних сценаріїв і аналіз результатів;
- Метод адаптивного керування доступом до обмеженого ресурсу прикладного рівня в SDN з локалізацією керуючого впливу в точці дефіциту ресурсу, що використовується для пояснення принципів адресного керування доступом і забезпечення рівномірності розподілу ресурсів;
- Програмний модуль та експериментальний стенд фреймворку Cloud-AI-SDN для відлагодження, навчання та перевірки роботи ML-моделей в SDN-середовищі, що забезпечує відтворюване формування топології й політик керування, збір телеметрії для навчання, запуск сценаріїв навантаження та аналіз результатів під час виконання практичних і лабораторних робіт;
- Методика вибору моделі машинного навчання для AI-керування за розширеними критеріями (класифікаційні показники, стійкість до хибнопозитивних спрацювань, час прийняття рішення, завантаженість обчислювальних ресурсів, енергоспоживання), що використовується для демонстрації компромісів між якістю і обчислювальними витратами та обґрунтування вибору моделей для різних сценаріїв;

Використання зазначених результатів дозволило підвищити якість навчального процесу із вказаних дисциплін.

Заступник директора ННІФТКН
з наукової роботи, д. ф.-м.н., професор _____ Олександр ДУБОЛАЗОВ

Завідувач кафедри РТІБ
д. т. н., професор _____ Андрій САМІЛА

Завідувач кафедри КСМ
кандидат ф.-м. н., доцент _____ Георгій ВОРОБЕЦЬ