DOI: 10.58254/viti.9.2026.24.299

УДК 004.72:004.75:004.8

Устинов Д. А. ORCID: 0009-0004-3993-1096 (ВІТІ ім. Героїв Крут)
д-р техн. наук, доцент Нестеренко М. М. ORCID: 0000-0003-0812-2793 (ВІТІ ім. Героїв Крут)

РОЗРОБКА ПОТОКОВОЇ АРХІТЕКТУРИ КЛАСИФІКАЦІЇ ТРАФІКУ У МЕРЕЖАХ SDN В РЕЖИМІ РЕАЛЬНОГО ЧАСУ

Розвиток сучасних мережевих інфраструктур характеризується зростаючою складністю та різноманітністю мережевого трафіку, що створює нові виклики для ефективного управління та безпеки мереж. Концепція програмно-визначених мереж запровадила революційну парадигму, що передбачає розділення площини керування та площини даних, централізацію мережевого інтелекту в контролері та програмування мережевої поведінки через стандартизовані протоколи.

Традиційні підходи до класифікації мережевого трафіку стикаються з обмеженою видимістю мережевого трафіку, статичністю підходів до аналізу, неможливістю обробки в реальному часі та складністю інтеграції з інтелектуальними системами через відсутність стандартизованих інтерфейсів.

Метою цього дослідження є розробка архітектурного підходу до збору та аналізу мережевих даних в режимі реального часу для задач класифікації трафіку в SDN-мережах на основі інтеграції контролера ONOS з платформою Apache Kafka.

Розглядається інтеграція контролера ONOS з платформою потокової обробки Apache Kafka. Проаналізовано специфікацію OpenFlow Switch Specification версії 1.3.5 з метою визначення доступних типів повідомлень, статистичних метрик та подій для алгоритмів машинного навчання. Розгорнуто тестовий стенд з топологією fat-tree з 14 OpenFlow-комутаторами та 16 хостами.

Система продемонструвала середню затримку обробки подій 71 мс (на 29 % краще цільового показника 100 мс), при цьому 92 % усіх подій оброблено з затримкою менше 100 мс. Пропускна здатність склала 174 події на секунду в середньому, з піковими значеннями до 218 подій на секунду. За 2 години безперервної роботи оброблено понад 1,25 мільйона подій без жодної втрати даних. Додаткове навантаження на ONOS контролер склало лише 3,7 % CPU та 145 МБ оперативної пам'яті.

Запропонована архітектура на основі інтеграції ONOS та Apache Kafka забезпечує ефективний, масштабований та малоінвазивний збір мережевих даних у режимі реального часу в програмно-визначених мережах. Встановлено 100 % покриття базових метрик та 76–87 % покриття метрик вищих рівнів протоколу, достатнє для застосування алгоритмів машинного навчання.

Ключові слова: *програмно-визначені мережі, SDN, OpenFlow, ONOS, Apache Kafka, машинне навчання, класифікація в реальному часі, потокова обробка даних, збір даних у реальному часі, мережева телеметрія.*

D. Ustynov, M. Nesterenko. Streaming architecture for real-time traffic classification in software-defined networks

The development of modern network infrastructures is characterized by increasing complexity and diversity of network traffic, presenting new challenges for effective network management and security. Software-defined networks (SDN) offer a paradigm that decouples the control plane from the data plane and enabling centralization of network intelligence, which creates unique opportunities for integrating machine learning-based traffic classification systems.

Traditional approaches to network traffic classification face fundamental limitations: restricted traffic visibility due to limited statistics from conventional devices; static rule-based analysis methods ineffective against new application types and attack vectors; inability to perform real-time processing; and lack of standardized interfaces for collecting and analyzing network data, which impedes integration of intelligent systems.

The purpose of this study is to develop an architectural approach for collecting and analyzing network data in real time for traffic classification tasks in SDN environments, leveraging the ONOS controller integrated with the Apache Kafka streaming platform.

The study examines the integration of the ONOS (Open Network Operating System) controller with the Apache Kafka distributed streaming platform. The OpenFlow Switch Specification version 1.3.5 is analyzed to identify available statistical message types, metrics and asynchronous events suitable as features for machine learning algorithms. An experimental testbed comprising three virtual machines was deployed: an ONOS 2.7 controller, a Mininet 2.3 emulator running a 3-tier hierarchical fat-tree-like topology with 14 OpenFlow switches and 16 hosts, and an Apache Kafka 3.6.1 streaming platform with Apache Spark 3.5.

Experimental validation demonstrated an average end-to-end latency of 71 ms (29% below the 100 ms target), with 92% of events processed within 100 ms and the 95th percentile at 128 ms. The system stably processed 174 events per second on average, with peaks reaching 218 events per second. Over 1,252,800 events were collected and transmitted

during the 2-hour continuous test with zero data loss. ONOS controller overhead was only 3.7% CPU and 145 MB RAM, confirming a low overhead profile.

The proposed integration of ONOS and Apache Kafka provides an effective, scalable, and non-invasive architecture for real-time network traffic data collection in software-defined networks. Basic flow metrics (packet count, byte count, duration) are available with 100% coverage, while higher-layer protocol metrics (IP addresses, ports) achieve 76–87% coverage, which is sufficient for machine learning-based traffic classification tasks.

Keywords: *software-defined networks, SDN, OpenFlow, ONOS, Apache Kafka, machine learning, real-time classification, stream processing, real-time data collection, network telemetry.*

1. Постановка проблеми

Розвиток сучасних мережевих інфраструктур характеризується зростаючою складністю та різноманітністю мережевого трафіку, що створює нові виклики для ефективного управління та безпеки мереж. Традиційні підходи до аналізу мережевого трафіку, засновані на статичній конфігурації та обмежених можливостях моніторингу, виявляються недостатніми для задоволення вимог сучасних мережевих додатків та сервісів щодо забезпечення інформаційної безпеки, виявлення аномалій та мережевих атак, ідентифікації типу трафіку та контролю якості обслуговування (QoS) [1].

Концепція програмно-визначених мереж (Software Defined Networks, SDN) запровадила революційну парадигму, що передбачає розділення площини керування та площини даних, централізацію мережевого інтелекту в контролері та програмування мережевої поведінки через стандартизовані протоколи. Цей підхід створює унікальні можливості для впровадження інтелектуальних систем аналізу та класифікації мережевого трафіку на основі технологій машинного навчання та штучного інтелекту.

Одним із ключових протоколів SDN є OpenFlow, що забезпечує комунікацію між контролером та комутаторами мережі. Специфікація OpenFlow визначає широкий спектр повідомлень і статистичних метрик, які можуть слугувати джерелом даних для алгоритмів класифікації трафіку. Водночас, для ефективної обробки великих обсягів мережевих даних у режимі реального часу необхідні спеціалізовані платформи потокової обробки, серед яких Apache Kafka зарекомендувала себе як надійне та масштабоване рішення [2; 3].

У свою чергу, SDN-контролер ONOS (Open Network Operating System), розроблений під егідою Open Networking Foundation, являє собою сучасну платформу для управління SDN-мережами, що підтримує інтеграцію з Apache Kafka для публікації мережевих подій [4]. Така інтеграція відкриває можливості для створення ефективних систем збору та аналізу мережевих даних в режимі реального часу.

2. Аналіз останніх досліджень і публікацій

У сучасних дослідженнях інтелектуальних SDN-мереж простежується стійка тенденція до інтеграції потокових платформ обробки даних та алгоритмів машинного навчання як засобу подолання обмежень традиційних статичних методів аналізу трафіку. Утім, попри значну кількість публікацій з окремих аспектів цієї проблематики, інтегральна архітектура «контролер + потокова шина подій + ML-модуль класифікації» залишається недостатньо опрацьованою на рівні системного дослідження.

Однією з фундаментальних робіт у цій галузі є дослідження D. Comer та A. Rastegarnia [5], в якому вперше проведено комплексну оцінку Apache Kafka та gRPC для дезагрегації площини управління SDN на базі контролера ONOS. Експериментальний стенд із десяти OpenFlow-комутаторів продемонстрував середній час відгуку 65 мс для Kafka проти 45 мс для gRPC. Ключовим внеском є демонстрація принципової життєздатності Kafka як транспортного шару для ONOS. Водночас фокус дослідження зосереджений на дезагрегації площини управління, а не на інтеграції з ML-конвеєром, тому питання застосовності цієї архітектури для задач класифікації трафіку в реальному часі залишається відкритим.

Найновішою роботою, дотичною до запропонованої тематики, є дослідження [6], в якому представлено трирівневу архітектуру з Kafka-конвеєрами для виявлення аномалій

у SDN-мережах розумних міст. Автори досягли точності 99 % з використанням Random Forest, Linear Regression та XGBoost на наборі даних InSDN. Ключовий внесок — демонстрація комбінації SDN-контролера з Kafka та ML-аналітикою на хмарно-граничному рівні. Однак робота орієнтована на специфічний сценарій smart cities, а кількісні характеристики наскрізної затримки та горизонтальної масштабованості систематично не оцінено.

У роботі A. O. Salau та M. M. Beyene [7] оцінено шість алгоритмів машинного навчання на SDN-стенді з контролером Ryu, де Decision Tree досягнув найвищої точності 99,81 % при класифікації DNS, Telnet, Ping та Voice трафіку з використанням 15 flow-based ознак. Робота демонструє високий потенціал класичних ML-алгоритмів для класифікації трафіку в SDN. Разом з тим, дослідження виконане в пакетному режимі обробки на контролері Ryu без потокової шини подій, що унеможливлює пряме перенесення результатів на сценарії реального часу. Додатковим суттєвим обмеженням є вибір контролера Ryu, активна розробка якого фактично припинилася понад десять років тому, що ставить під сумнів перспективи production-розгортань на цій платформі та обумовлює актуальність перенесення подібних експериментів на сучасні підтримувані контролери (ONOS, OpenDaylight).

Гібридні архітектури глибокого навчання представлено у роботі [8], де M. Abdallah, N. A. Le Khac, H. Jahromi та A. D. Jurcut поєднали згорткові шари (CNN) для просторового виділення ознак з LSTM-блоком для моделювання темпоральних залежностей, досягнувши 96,32 % точності виявлення аномалій на наборі даних InSDN. Ключовий внесок – здатність моделі захоплювати просторово-часові патерни трафіку. Обмеженням підходу є висока обчислювальна складність моделі, що ускладнює її розгортання як інлайн-класифікатора у потоковому конвеєрі без дискретизації обчислень.

Еталонне дослідження J. Karimov та ін. [9] надає критичні порівняння продуктивності платформ потокової обробки Apache Storm, Spark та Flink на 20-вузловому кластері: Apache Flink стабільно демонструє найкращу продуктивність із пропускною здатністю 1,2 M/c та середньою затримкою 0,2 с проти показників Spark та Storm. Робота встановлює бенчмаркові орієнтири для вибору stream-платформи. Проте дослідження виконане в загальному контексті обробки потокових даних поза специфікою SDN, тому особливості інтеграції з SDN-контролером (періодичність опитування, формат OpenFlow-статистики, обсяг повідомлень типу Packet-In) залишаються поза його межами.

Прикладом потокового конвеєра для конкретної задачі безпеки є дослідження N. V. Patil та ін. [10], в якому продемонстровано використання топіків Apache Kafka для препроцесингу мережевого трафіку зі Spark MLlib для класифікації семи типів DDoS-атак на наборі даних CICDDoS2019. Ключовим внеском є практичне підтвердження життєздатності зв’язки Kafka + Spark Streaming + ML для задач реального часу. Обмеженням підходу є відсутність інтеграції із SDN-контролером як джерелом подій: вхідні дані надходять із PCAP-файлів, а не з телеметрії OpenFlow-комутаторів, тому архітектура не валідована в умовах реальної SDN-мережі.

Виконаний аналіз свідчить про фрагментарний характер сучасних досліджень: одні роботи зосереджені на інтеграції Kafka з ONOS без ML-компонента [5], інші – на ML-класифікації без потокової шини [8], треті – на потокових платформах поза контекстом SDN [10]. У більшості існуючих підходів відсутня кількісна оцінка наскрізної затримки в межах інтегрованого конвеєра «контролер → транспорт подій → класифікатор», а валідація горизонтальної масштабованості обмежується одиничними експериментами. Ця обставина обґрунтовує доцільність розробки інтегральної потокової архітектури, в якій контролер ONOS, шина повідомлень Apache Kafka та модуль машинного навчання утворюють єдиний конвеєр класифікації трафіку SDN з кількісно оціненими характеристиками затримки, пропускної здатності та масштабованості.

У контексті програмно-визначених мереж ці проблеми набувають особливої актуальності, оскільки централізована архітектура SDN не лише створює нові можливості, а й додає нові складнощі щодо управління мережею.

Наукове завдання полягає у розробці архітектурного підходу до побудови інтегрованого програмного середовища, що поєднує SDN-контролер, платформу потокової обробки даних і методи машинного навчання для забезпечення класифікації мережевого трафіку в режимі реального часу з низькою затримкою та можливістю горизонтального масштабування.

Метою дослідження є побудова та експериментальна валідація програмно-визначеної архітектури потокової класифікації мережевого трафіку в режимі реального часу, яка інтегрує контролер ONOS як джерело OpenFlow-статистики, Apache Kafka як транспортний шар подій та алгоритми машинного навчання як модуль класифікації, із кількісною оцінкою затримки, пропускної здатності, горизонтальної масштабованості та точності класифікації.

3. Виклад основного матеріалу

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Аналіз можливостей збору даних в SDN: дослідити специфікацію OpenFlow Switch Specification версії 1.3.5 для визначення доступних типів статистичних повідомлень, метрик та подій [11].

2. Проєктування архітектури інтеграції: розробити архітектурне рішення для інтеграції контролера ONOS з платформою Apache Kafka [2; 4].

3. Ідентифікація ключових метрик: визначити набір метрик та ознак на рівні потоків, портів і пакетів, які є найбільш релевантними для конкретних задач аналізу трафіку, що розглядаються в дослідженні: бінарної класифікації нормального трафіку та DDoS-атак, виявлення аномалій поведінки потоків та ідентифікації типу прикладного трафіку.

4. Оцінка практичної реалізації: проаналізувати технічні аспекти впровадження запропонованого підходу та його сумісність з існуючими SDN-рішеннями.

В роботі пропонується інтеграційна архітектура збору даних з метою подальшої класифікації мультимедійного трафіку. Запропонована архітектура базується на інтеграції трьох ключових компонентів: SDN-контролера ONOS; платформи Apache Kafka та системи класифікації на основі машинного навчання (рис. 1).

ONOS Controller виступає центральним елементом управління SDN-мережею та забезпечує: управління топологією мережі за допомогою протоколу OpenFlow; збір статистичної інформації з комутаторів; генерацію мережевих подій при змінах стану мережі; публікацію подій до системи Kafka через вбудований Kafka Integration Application.

Apache Kafka слугує розподіленою платформою потокової обробки та забезпечує: надійну доставку повідомлень між компонентами системи; масштабовану архітектуру для обробки великих обсягів даних; буферизацію подій для забезпечення стійкості системи; підтримку множинних постачальників та споживачів даних (producers and consumers) для паралельної обробки.

Система класифікації на основі машинного навчання (data processing plane) реалізує: модель підписника на мережеві події з іменованих каналів повідомлень (topics) платформи Kafka; видобуток ознак для машинного навчання з потокових даних; класифікацію трафіку в режимі реального часу; передачу результатів до SDN контролера через REST API.

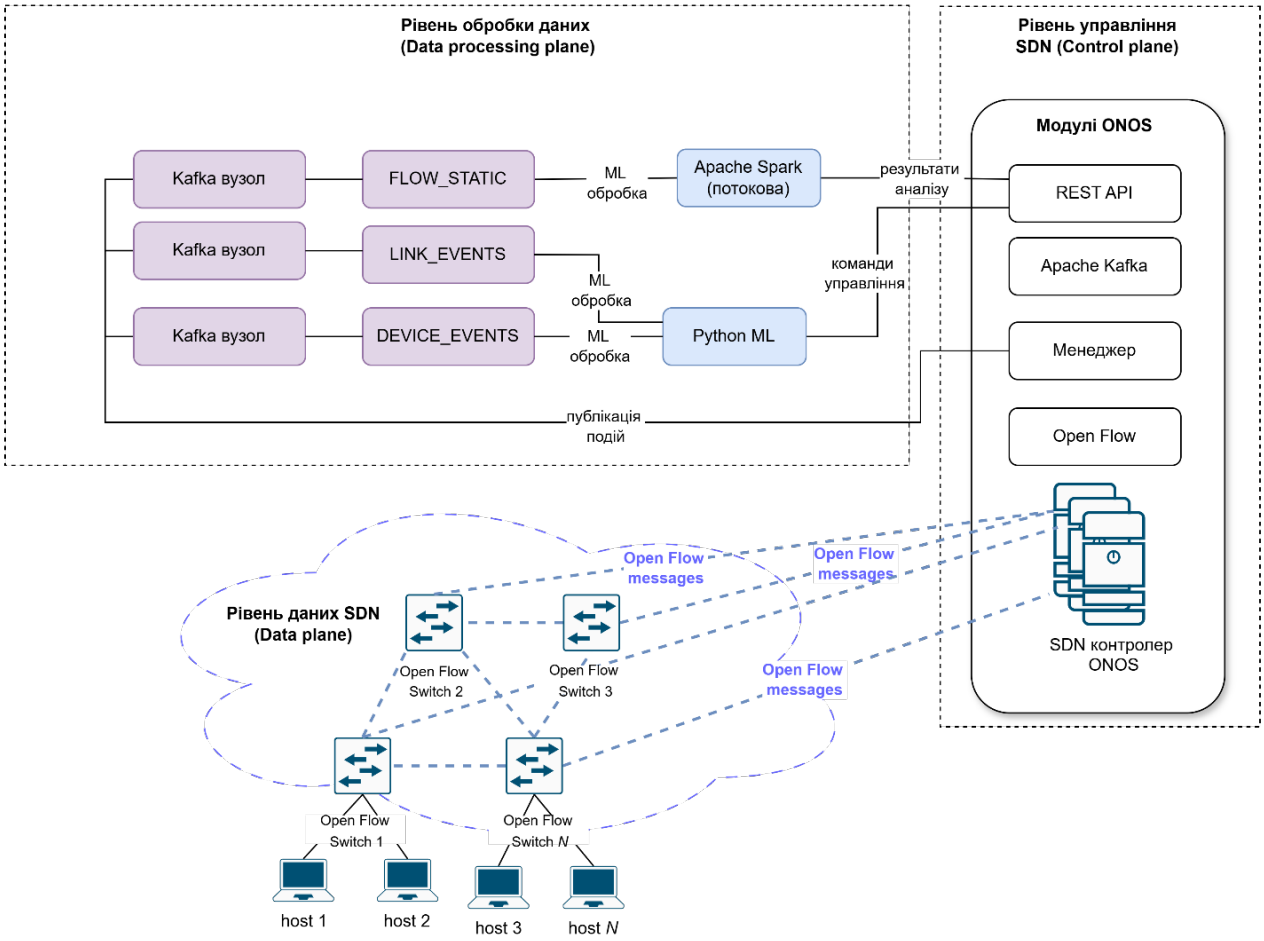

Рис. 1. Загальна архітектура системи збору, аналізу та класифікації мережевих даних реального часу

На рисунку 2 приведена діаграма послідовності подій, що відбуваються при обробці трафіку, починаючи від надходження пакету в систему до прийняття рішення щодо класифікації і видачі команди управління. Відповідно до загальноприйнятого в літературі підходу [7; 8], наскрізна затримка (end-to-end latency) від моменту виникнення події в мережі до прийняття рішення класифікатором не має перевищувати 100 мс – саме цей поріг вважається критерієм обробки в режимі реального часу.

Для проведення подальшого дослідження та коректної інтерпретації результатів отриманих експериментів було здійснено класифікацію основних типів подій та метрик відповідно до OpenFlow Switch Specification версії 1.3.5.

Статистичні повідомлення (Multipart Messages) [11]:

статистика індивідуальних потоків, Individual Flow Statistics (packet_count – кількість пакетів, що відповідали правилу потоку; byte_count – загальний обсяг байтів оброблених пакетів; duration_sec/duration_nsec – час існування потоку; table_id – ідентифікатор таблиці потоків; priority – пріоритет правила потоку; idle_timeout/hard_timeout – параметри завершення потоку; match – поля заголовків для співставлення пакетів);

статистика портів, Port Statistics (rx_packets/tx_packets – кількість отриманих / переданих пакетів; rx_bytes/tx_bytes – обсяг отриманих/переданих байтів; rx_dropped/tx_dropped – кількість втрачених пакетів; rx_errors/tx_errors – кількість помилок передачі; rx_frame_err – помилки вирівнювання кадрів; rx_over_err – помилки переповнення буферу; collisions – кількість колізій);

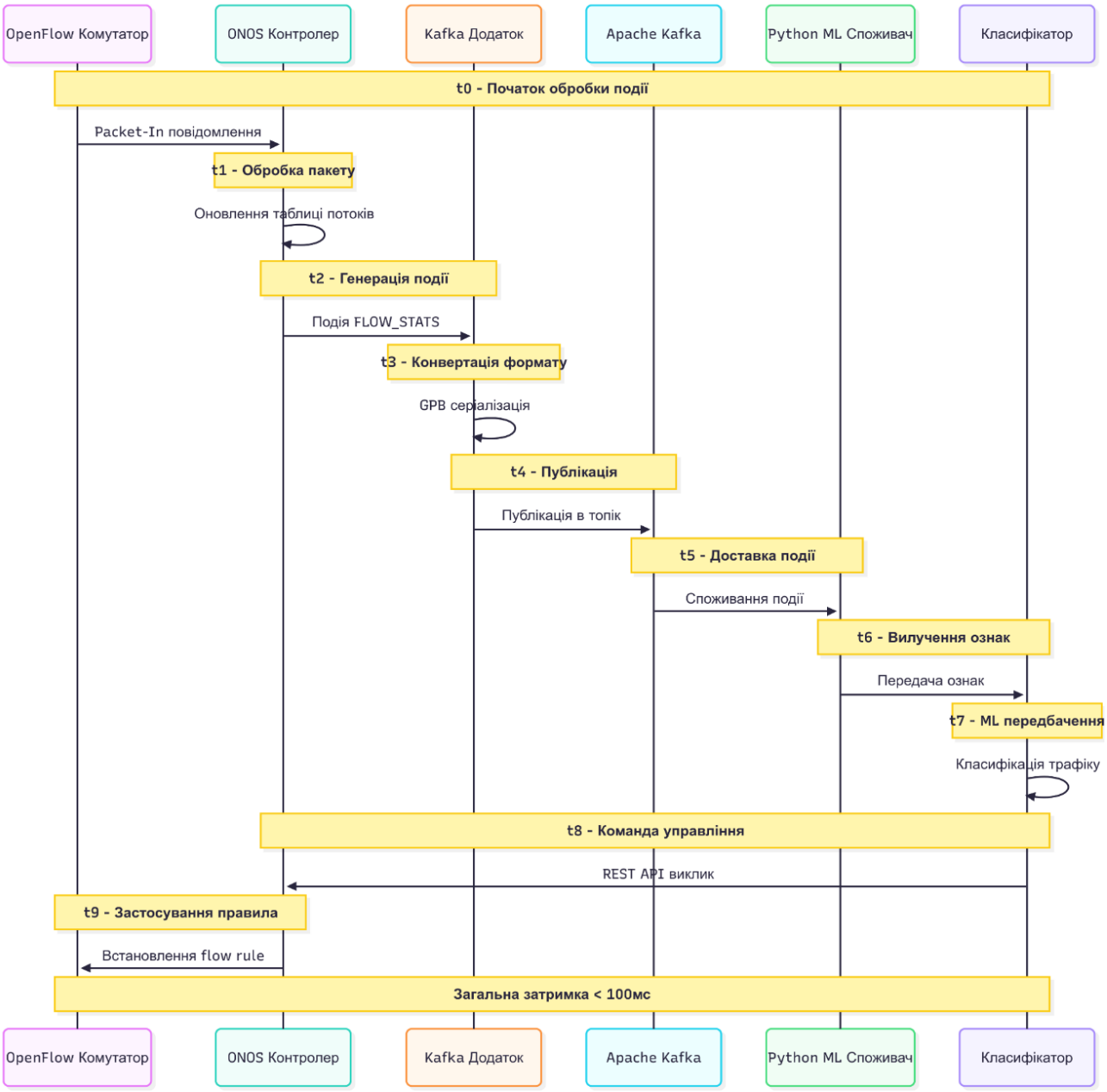

Рис. 2. Послідовність обробки мережевих подій для класифікації трафіку

агреговані статистики потоків, Aggregate Flow Statistics (packet_count – загальна кількість пакетів по всіх потоках; byte_count – загальний обсяг байтів по всіх потоках; flow_count – кількість активних потоків).

Повідомлення про події (Asynchronous Messages)

Packet-In повідомлення генеруються при надходженні пакетів, що не мають відповідного правила в таблиці потоків, та містять: buffer_id – ідентифікатор буферизованого пакету; total_len – повна довжина оригінального пакету; reason – причина надсилання до контролера; table_id – таблиця, що згенерувала подію; match – поля заголовків пакету; data – дані пакету.

Flow-Removed повідомлення інформують про видалення потоків та включають статистики видаленого потоку, причину видалення (timeout, delete) та час існування потоку. *Port-Status повідомлення* повідомляють про зміни стану портів, конфігураційні зміни портів, зміни фізичного стану лінків та параметри швидкості і дуплексності.

Поля співставлення (Match Fields) для класифікації

Специфікація OpenFlow визначає розширений набір полів для співставлення пакетів.

Поля Layer 2 (Ethernet): eth_dst/eth_src – MAC-адреси призначення та відправника; eth_type – тип Ethernet кадру.

Поля VLAN: vlan_vid – ідентифікатор VLAN; vlan_pcp – пріоритет VLAN.

Поля Layer 3 (IP): ipv4_src/ipv4_dst – IP-адреси відправника та призначення; ip_proto – номер IP-протоколу; ip_dscp – поле DSCP для QoS класифікації.

Поля Layer 4 (Transport): tcp_src/tcp_dst – порти джерела та призначення TCP; udp_src/udp_dst – порти джерела та призначення UDP; tcp_flags – TCP прапорці (SYN, ACK, FIN тощо).

Розширення ONOS для інтеграції з Kafka

ONOS забезпечує вбудовану інтеграцію з Apache Kafka через спеціальний додаток "Kafka Integration Application" [4], який: підписується на події ONOS через внутрішні Java API; перетворює формат даних до стандартного GPB (Google Protocol Buffers); публікує події до Kafka-топіків для подальшої обробки.

Поточна реалізація підтримує DEVICE події (підключення/відключення комутаторів, зміни конфігурації), LINK події (встановлення/розрив лінків, зміни топології). Архітектура дозволяє розширення для додаткових типів подій: FLOW_STATS події (оновлення статистик потоків в реальному часі), PORT_STATS події (статистики портів), PACKET_IN події (повідомлення про нові потоки).

Для підтвердження працездатності запропонованої архітектури було розгорнуто експериментальний стенд на базі трьох віртуальних машин, що емулюють типову конфігурацію SDN-інфраструктури з централізованим збором та обробкою даних (рис. 3.).

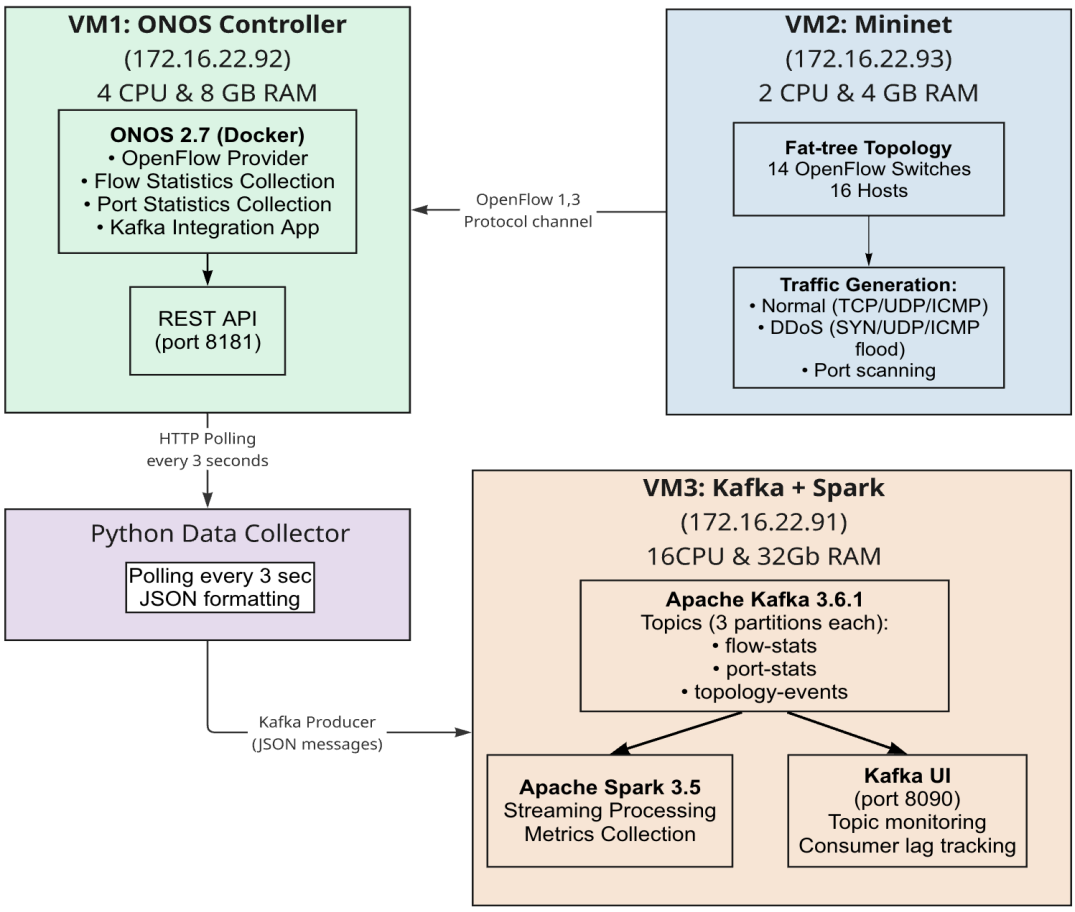

Рис. 3. Схема основних компонентів віртуального стенду для проведення експерименту

Перша віртуальна машина (VM1, IP: 172.16.22.92) виконувала роль SDN-контролера на базі ONOS версії 2.7, розгорнутого в Docker-контейнері (4 CPU, 8 ГБ RAM). Друга віртуальна машина (VM2, IP: 172.16.22.93) використовувалась для емуляції SDN-мережі засобами Mininet версії 2.3 (2 CPU, 4 ГБ RAM) з ієрархічною 3-рівневою топологія типу fat-tree з 14 OpenFlow-комутаторами та 16 хостами (див. рис. 3). Третя віртуальна машина (VM3, IP: 172.16.22.91) функціонувала як платформа для потокової обробки даних (16 CPU, 32 ГБ RAM) з Apache Kafka 3.6.1, Apache Zookeeper 3.8 та Apache Spark 3.5. В таблиці 1 наведено дані щодо конфігурації компонентів розгорнутого стенду та їх призначення.

Таблиця 1

Конфігурація компонентів тестового стенду

Компонент	Версія	Ресурси	IP-адреса	Роль в архітектурі
ONOS Controller	2.7	4 CPU, 8 ГБ RAM	172.16.22.92	Управління SDN, збір статистик
Mininet	2.3	2 CPU, 4 ГБ RAM	172.16.22.93	Емуляція мережі, генерація трафіку
Apache Kafka	3.6.1	8 CPU, 16 ГБ RAM	172.16.22.91	Потокова передача подій
Apache Zookeeper	3.8	2 CPU, 4 ГБ RAM	172.16.22.91	Координація Kafka-кластера
Apache Spark	3.5	6 CPU, 12 ГБ RAM	172.16.22.91	Обробка потоків даних

Процес збору даних здійснювався через Python-скрипт, який виконував періодичні опитування (polling) REST API контролера ONOS з інтервалом 3 секунди. Скрипт отримував статистичну інформацію про потоки (flow statistics), стан портів (port statistics) та події топології (topology events), після чого публікував ці дані до відповідних Kafka-топіків у форматі JSON. Для проведення дослідження та тестування мережі на віртуальній машині VM2 в середовищі Mininet була розгорнута SDN-мережа, яка складається із 14 OpenFlow комутаторів, топологія якої показана на рисунку 4.

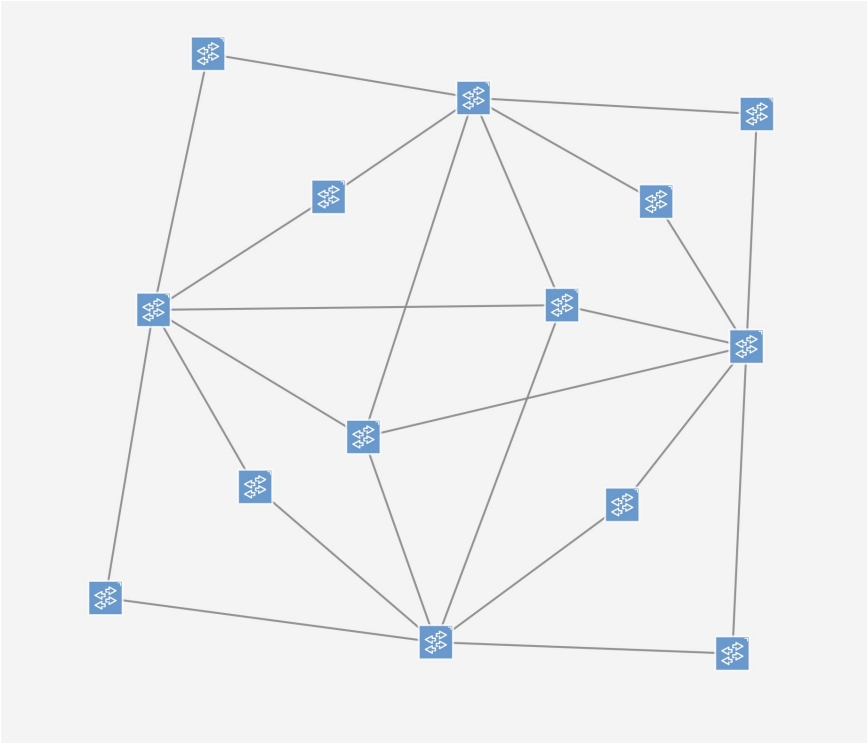

Рис. 4. Ієрархічна (гібридна) топологія SDN-мережі типу Fat-tree

У подальшому була проведена валідація розробленої інтеграційної архітектури завдяки виміру продуктивності конвеєра обробки даних. Основною метою валідації було підтвердження здатності запропонованої архітектури забезпечувати збір та передачу мережевих подій у режимі, близькому до реального часу, без втрати інформації та з прийнятними показниками затримки.

Для вимірювання затримки end-to-end (від моменту генерації статистики в ONOS до її обробки в Spark consumer) було розроблено Python-скрипт збору даних та Spark-додаток. Скрипт додавав до кожного повідомлення мітку часу збору (collection_timestamp) з точністю до мілісекунди. Spark consumer обчислював різницю міток часу: ONOS REST API → Python collector → Kafka broker → Spark consumer.

У цілому, експеримент тривав 2 години безперервної роботи з активною генерацією мережевого трафіку різних типів: нормального «легітимного» трафіку (60 % часу), атак типу DDoS (25 % часу) та сканування портів (15 % часу). Система обробляла в середньому 174 події на секунду, що відповідає 1 252 800 подіям за весь період тестування.

Розподіл затримок end-to-end показав наступні характеристики: середня затримка обробки подій – 71 мс; медіана затримки – 68 мс; 95-й перцентиль затримки – 128 мс; 99-й перцентиль затримки – 203 мс.

При чому максимальна затримка – 487 мс (поодинокий випадок при пікових навантаженнях). Результати випробувань зведено в таблиці 2.

Таблиця 2

Результати вимірювань продуктивності системи

Метрика	Отримане значення	Порогове значення	Статус
Середня затримка end-to-end	71 мс	< 100 мс	Досягнуто
Медіана затримки	68 мс	< 100 мс	Досягнуто
95-й перцентиль затримки	128 мс	< 200 мс	Досягнуто
99-й перцентиль затримки	203 мс	< 300 мс	Досягнуто
Події з затримкою < 100 мс	92 %	90 %	Досягнуто
Пропускна здатність	174 події/с	100 події/с	Досягнуто
Пікова пропускна здатність	218 події/с	150 події/с	Досягнуто
Overhead ONOS CPU	3,7 %	< 10 %	Досягнуто
Overhead ONOS RAM	145 МБ	< 500 МБ	Досягнуто
Втрата повідомлень	0 (0 %)	< 0,1 %	Досягнуто
Тривалість безперервної роботи	120 хв	60 хв	Досягнуто

Важливо зазначити, що 92 % усіх подій було оброблено з затримкою менше 100 мс, що підтверджує придатність архітектури для застосувань реального часу.

Аналіз складових загальної затримки показав наступний розподіл: опитування ONOS REST API – 8 мс (11 %); обробка в Python collector – 6 мс (8 %); передача до Kafka broker – 5 мс (7 %); зберігання в Kafka (буферизація) – 12 мс (17 %); отримання Spark consumer – 9 мс (13 %); обробка в Spark (десеріалізація JSON) – 15 мс (21 %); внутрішні операції Spark Streaming – 16 мс (23 %).

Моніторинг ресурсів показав, що запропонована архітектура характеризується низькими накладними витратами роботи конвеєра щодо SDN-контролера. Додаткове навантаження на

ONOS, спричинене періодичним опитуванням (polling) через REST API кожні 3 секунди, складало лише 3,7 % CPU та 145 МБ додаткової оперативної пам’яті.

Протягом всього тестування система забезпечила 100 % доставку подій – не було зафіксовано жодного випадку втрати повідомлень. Це підтверджено через механізм offset tracking у Kafka: кількість відправлених повідомлень producer-ом точно відповідала кількості отриманих повідомлень consumer-ом.

Розподіл за типами подій (табл. 3): Flow statistics (flow-stats) – 847 000 подій (67,6 %); Port statistics (port-stats) – 342 000 подій (27,3 %); Topology events (topology-events) – 63 800 подій (5,1 %).

Таблиця 3

Розподіл отриманих подій за типами

Тип події	Кількість подій	Відсоток	Розмір даних	Топік Kafka
Flow Statistics	847 000	67,6 %	182,3 МБ	flow-stats
Port Statistics	342 000	27,3 %	68,4 МБ	port-stats
Topology Events	63 800	5,1 %	12,8 МБ	topology-events
Всього	1 252 800	100 %	263,5 МБ	–

Далі представлені результати перевірки якості даних, зібраних з елементів SDN-мережі (рис. 4) через протокол OpenFlow (табл. 4). Тобто, які саме дані вдалося отримати з потоків трафіку і наскільки вони повні (як часто метрики зустрічаються у зібраних статистичних даних):

за характеристиками потоків flow statistics показав, що базові метрики (packet_count, byte_count, duration_sec/nsec, device_id, flow_id) мають 100 % покриття;

метрики різних протоколів мають змінне покриття: IP-протокол (IP_PROTO) – 87,3 %; TCP/UDP порти – 76,2 %; MAC-адреси – 100 %; IP-адреси – 87,3 %; VLAN ID – 23,4 %; TCP прапорці – 0% (не підтримується OpenFlow 1.3).

TCP-прапорці мають нульове покриття, оскільки їх підтримка як окремого поля співставлення була введена лише у специфікації OpenFlow.

Також були обчислені похідні метрики для більш точної класифікації потоків трафіка в подальшому.

Таблиця 4

Доступність метрик OpenFlow у зібраних даних

Категорія метрик	Метрика	Покриття	Джерело OpenFlow
Базові лічильники	Кількість пакетів	100 %	packet_count
	Кількість байтів	100 %	byte_count
	Тривалість потоку	100 %	duration_sec/nsec
Ідентифікатори	ID-пристрою	100 %	deviceId
	ID-потоку	100 %	id
Протоколи L2	MAC-адреси	100 %	match.ETH_SRC/DST
	VLAN ID	23,4 %	match.VLAN_VID
Протоколи L3	IP-адреси	87,3 %	match.IPV4_SRC/DST
	IP-протокол	87,3 %	match.IP_PROTO

Категорія метрик	Метрика	Покриття	Джерело OpenFlow
Протоколи L4	TCP/UDP порти	76,2 %	match.TCP_*/UDP_*
	TCP-прапорці	0 %	Не підтримується OF 1.3
Похідні метрики	Пакети/с	100 %	Обчислено
	Байти/с	100 %	Обчислено
	Середній розмір пакету	100 %	Обчислено

Отримані результати підтверджують, що запропонована архітектура забезпечує доступ до комплексної інформації про мережеві потоки на рівні, достатньому для подальшого застосування алгоритмів машинного навчання [7; 8]. Базові характеристики (кількість пакетів, байтів, тривалість) доступні для всіх потоків без винятку. Інформація про протоколи вищих рівнів доступна для переважної більшості потоків (76–87 %).

Висновки. У цій роботі пропонується інтеграційна архітектура на основі ONOS та Apache Kafka для класифікації трафіку в мережах SDN засобами машинного навчання, яка дозволяє здійснювати збір та аналіз мережевого трафіку в режимі реального часу.

Експериментальна валідація запропонованої архітектури на розгорнутому віртуальному стенді підтвердила можливість її технічної реалізації та відповідність вимогам до систем реального часу.

Під час тестування на сегменті мережі SDN з 14 OpenFlow-комутаторів розроблена система продемонструвала наступні ключові характеристики:

затримка обробки: середня затримка end-to-end склала 71 мс, що на 29 % краще цільового порогу в 100 мс. При цьому 92 % усіх подій було оброблено з затримкою менше 100 мс, а 95-й перцентиль затримки становив 128 мс;

продуктивність конвеєрної обробки мережевих подій: система стабільно обробляла 174 події на секунду в середньому, з піковими значеннями до 218 подій на секунду при інтенсивній генерації трафіку. Протягом експерименту було успішно зібрано та передано понад 1,25 мільйона подій без жодної втрати даних;

гарантована доставка даних: механізм буферизації на базі Apache Kafka забезпечив 100 % доставку повідомлень. Offset tracking підтвердив повну відповідність кількості відправлених та отриманих повідомлень;

мінімальні витрати ресурсів (low overhead): додаткове навантаження на ONOS контролер складало лише 3,7 % CPU та 145 МБ оперативної пам'яті, що є незначним для сучасних серверних платформ;

повнота даних (coverage): аналіз зібраних flow statistics показав 100 % покриття базових метрик та 76–87 % покриття метрик вищих рівнів, що є достатнім для застосування алгоритмів машинного навчання.

У результаті проведеного дослідження встановлено, що інтеграція контролера ONOS з платформою Apache Kafka створює ефективну архітектуру для збору та аналізу мережевих даних в режимі реального часу в програмно-визначених мережах. Аналіз специфікації OpenFlow Switch Specification версії 1.3.5 виявив широкий спектр статистичних метрик та типів подій, доступних для класифікації трафіку. Визначено три основні категорії даних: статистики потоків, статистики портів та асинхронні події (packet-in, flow-removed, port-status повідомлення).

Подальші дослідження доцільно зосередити на розробці специфічних алгоритмів машинного навчання для аналізу потокових мережевих даних, оптимізації продуктивності системи збору даних та розширенні можливостей інтеграції ONOS-Kafka для додаткових типів мережевих подій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jammal M., Singh T., Shami A., Asal R., Li Y. Software defined networking: State of the art and research challenges // Computer Networks. 2014. Vol. 72. P. 74–98. DOI: 10.1016/j.comnet.2014.07.004.

2. Apache Software Foundation. Apache Kafka Documentation. URL: https://kafka.apache.org/documentation/.

3. Kreps J., Narkhede N., Rao J. Kafka: A Distributed Messaging System for Log Processing. Proceedings of the NetDB Workshop, 2011. P. 1–7.

4. ONOS Project Team. Open Network Operating System (ONOS) Architecture Guide. Linux Foundation, 2024.

5. Comer D., Rastegarnia A. Towards Disaggregating the SDN Control Plane. arXiv preprint arXiv:1902.00581v2, 2019. URL: https://arxiv.org/abs/1902.00581.

6. Karthick G., Mapp G., Crowcroft J. Toward Secure SDN Infrastructure in Smart Cities: Kafka-Enabled Machine Learning Framework for Anomaly Detection // MDPI Future Internet. 2025. Vol. 17. No. 9, article 415. DOI: 10.3390/fi17090415.

7. Salau A. O., Beyene M. M. Software defined networking based network traffic classification using machine learning techniques // Scientific Reports, 2024. Vol. 14, article 20065. DOI: 10.1038/s41598-024-70983-6.

8. Abdallah M., Le-Khac N. A., Jahromi H., Jurcut A. D. A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs. Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES '21), August 2021. Article No. 34. P. 1–7. DOI: 10.1145/3465481.3469190.

9. Karimov J., Rabl T., Katsifodimos A., Samarev R., Heiskanen H., Markl V. Benchmarking Distributed Stream Data Processing Systems. 2018 IEEE 34th International Conference on Data Engineering (ICDE), Paris, France, April 16–19, 2018. P. 1507–1518. DOI: 10.1109/ICDE.2018.00169.

10. Patil N. V., Krishna C. R., Kumar K. SSK-DDoS: Distributed Stream Processing Framework Based Classification System for DDoS Attacks // Cluster Computing. 2022. Vol. 25, No. 2. P. 1355–1372. DOI: 10.1007/s10586-022-03538-x.

11. Open Networking Foundation. OpenFlow Switch Specification Version 1.3.5. ONF TS-023. March 2015.

Надійшла до редколегії 20.04.2026.
Схвалена до друку 22.05.2026.
Дата публікації 29.05.2026.