Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Система діагностики та графічного аналізу стану вузлів розподіленої мережі на базі SDN-технологій
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 022072.22.01.109 ПЗ

Виконав здобувач IV курсу, група КІ2-22-1 _____ Артем ПЛЮСНІН
Підпис Ініціали, прізвище

Керівник _____ Андрій ГАРМАТЮК
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер канд.фіз.-мат.наук, доц. _____ Тетяна КИСІЛЬ
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС _____ Ольга ПАВЛОВА
«01» червня 2026 р. Підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
Рівень вищої освіти ПЕРШИЙ (БАКАЛАВРСЬКИЙ)
Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ
Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ
Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КПС
_____ Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Плюсніну Артему Андрійовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Система діагностики та графічного аналізу стану вузлів розподіленої мережі на базі SDN-технологій

Керівник проекту (роботи) Гарматюк Андрій Вікторович, асистент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7
2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.
3. Вихідні дані до роботи Завдання на кваліфікаційну роботу
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз об'єкта дослідження та постановка задачі
Проєктування системи діагностики та графічного аналізу стану вузлів розподіленої мережі
Програмна реалізація системи графічного моніторингу SDN-інфраструктури
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Архітектура системи графічного моніторінгу
Алгоритм обробки телеметричних даних та синтезу топології
Інтерфейс системи графічного моніторінгу SDN-мережі

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «_10_»_____01_____2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконано
3	Робота над розділом 1 – аналіз об'єкта досліджень та постановка задачі	01.03.2026	виконано
4	Робота над розділом 2 – проєктування системи діагностики та графічного аналізу стану вузлів розподіленої мережі	01.04.2026	виконано
5	Робота над розділом 3 – програмна реалізація системи графічного моніторингу SDN-інфраструктури	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконано
7	Попередній захист ВКР	26.05.2026	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач_____ ___Артем ПЛЮСНІН___
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи_____ ____Андрій ГАРМАТЮК___
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 022072.22.01.109 ПЗ	Пояснювальна записка	66		
			Графічні матеріали			
2		КвРКІ 022072.22.01.109 Е8	Архітектура системи графічного моніторингу	1		
3		КвРКІ 022072.22.01.109 Е8	Алгоритм обробки телеметричних даних та синтезу топології	1		
4		КвРКІ 022072.22.01.109 Е8	Інтерфейс системи графічного моніторингу SDN-мережі			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 022072.22.01.109 ВП		
Розробив		Плюснін			Відомість проекту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Гарматюк				У	
Н. контр.		Кисіль				ХНУ, КІ2-22-1	
Затв.		Павлова		01.06			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно-апаратної система моніторингу параметрів комп'ютерного обладнання».

Автор роботи: Артем ПЛЮСНІН.

Керівник роботи: Андрій ГАРМАТЮК.

Пояснювальна записка: 66 с., 31 рис., 12 табл., 3 дод., 50 джерел.

Графічна частина: 3 креслення.

SDN, МОНІТОРИНГ МЕРЕЖІ, TELEMETRY, REACT, NODE.JS, REST API, CISCO CATALYST CENTER, ВІЗУАЛІЗАЦІЯ ТОПОЛОГІЇ.

Кваліфікаційна робота бакалавра присвячена розробці програмної системи графічного моніторингу та аналізу стану вузлів розподіленої SDN-мережі промислового підприємства. Актуальність теми зумовлена зростанням складності сучасних мережевих інфраструктур та необхідністю забезпечення безперервного контролю стану мережевого обладнання у режимі реального часу. Своєчасне виявлення перевантажень, втрати зв'язку та аварійних ситуацій дозволяє підвищити стабільність функціонування промислової мережі, скоротити час локалізації несправностей та підвищити ефективність адміністрування SDN-інфраструктури.

Метою роботи є проєктування, реалізація та тестування програмної системи для збору, обробки та графічної візуалізації телеметричних даних SDN-мережі у режимі реального часу. Для досягнення поставленої мети було виконано аналіз сучасних підходів до побудови систем мережевого моніторингу та технологій Software Defined Networking, розроблено архітектуру програмного комплексу, реалізовано інтеграцію з Cisco Catalyst Center через REST API, створено алгоритми обробки телеметрії та синтезу топології, а також розроблено інтерактивний вебінтерфейс для візуального аналізу стану мережевої інфраструктури.

Підпис здобувача

30.05.2026

Дата

ЗМІСТ

ВСТУП

Розвиток цифрових технологій та концепції «Індустрія 4.0» підвищує залежність промислових підприємств від стабільної роботи мережевої інфраструктури. Автоматизовані виробничі системи, промислові контролери та мережеве обладнання працюють у спільному інформаційному середовищі, тому навіть короткочасні мережеві збої можуть призводити до втрати даних або порушення технологічних процесів. У зв’язку з цим важливим є використання систем централізованого моніторингу та діагностики мереж.

Традиційні засоби моніторингу, що базуються на журналах подій, SNMP-запитах і статичних схемах топології, не завжди забезпечують достатню швидкодію та зручність аналізу в умовах сучасних SDN-мереж. Це створює потребу у використанні інтерактивної візуалізації, централізованого збору телеметрії та сучасних вебінтерфейсів для контролю стану мережі.

У роботі розглянуто задачу проєктування та реалізації системи графічного моніторингу SDN-інфраструктури. Основну увагу приділено інтеграції із SDN-контролером, побудові інтерактивної топології мережі, обробці телеметричних даних і візуальному аналізу стану вузлів у режимі реального часу.

Метою дипломної роботи є розробка системи графічного моніторингу та аналізу стану вузлів SDN-мережі для підвищення ефективності адміністрування мережевої інфраструктури.

Об’єкт дослідження – процеси моніторингу та діагностики активного мережевого обладнання у програмно-конфігурованих мережах.

Предмет дослідження – методи та програмні засоби інтерактивної візуалізації мережевих топологій і механізми збору та обробки телеметричних даних у SDN-інфраструктурі.

1 АНАЛІЗ ОБ'ЄКТА ДОСЛІДЖЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз сучасного стану та проблем моніторингу промислових мережевих інфраструктур

У сучасних умовах розвитку цифрових технологій та концепції «Індустрія 4.0» промислові підприємства значною мірою залежать від стабільної роботи мережевої інфраструктури. Автоматизовані виробничі системи, промислові контролери, SCADA-системи та мережеве обладнання формують єдине інформаційне середовище, у якому навіть короткочасні мережеві збої можуть призводити до порушення технологічних процесів або втрати даних. Для підприємств машинобудівної галузі мережа є важливою частиною виробничої інфраструктури, від якої залежить стабільність роботи обладнання та ефективність виробництва [27, 30, 42].

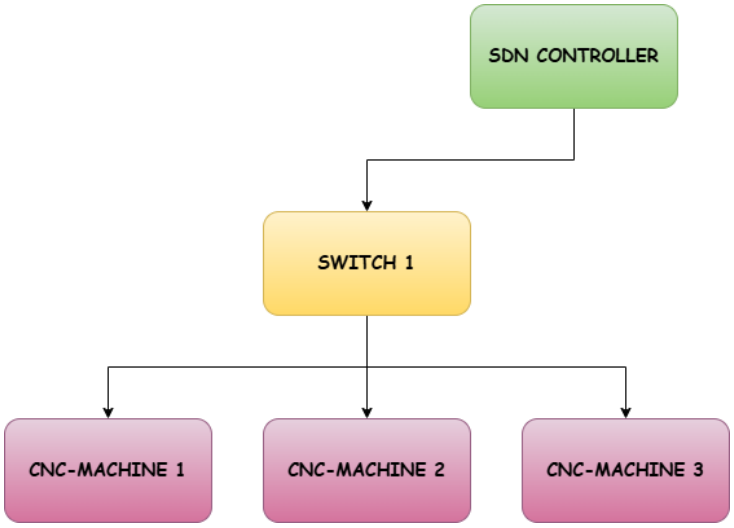


Рисунок 1.1 – Узагальнена структура промислової мережевої інфраструктури підприємства

Сучасні промислові мережі характеризуються високою складністю та постійними змінами структури. У межах одного підприємства часто використовуються як сучасні інтелектуальні пристрої, так і застаріле обладнання

з різними протоколами передачі даних. Це ускладнює централізоване адміністрування та формування цілісної картини стану мережі.

Більшість традиційних мереж побудовані за класичною ієрархічною моделлю, у якій логіка управління розподілена між окремими мережевими пристроями. Кожен маршрутизатор або комутатор працює автономно, використовуючи власні алгоритми та локальні таблиці маршрутизації. Такий підхід ускладнює централізований контроль і знижує гнучкість керування мережею.

Додатковою проблемою є використання застарілих методів моніторингу, зокрема SNMP та текстових журналів подій. У разі виникнення несправностей адміністратору доводиться вручну аналізувати інформацію з різних джерел, що потребує значних часових витрат і підвищує ймовірність помилок. Ситуація ускладнюється через постійну зміну структури мережі, внаслідок чого документація та схеми топології швидко втрачають актуальність [5, 10].

Окремою проблемою є недостатній рівень візуалізації. Текстові журнали та таблиці параметрів не дозволяють швидко оцінити загальний стан мережі, особливо в умовах великої кількості вузлів і складних взаємозв’язків між ними. Це створює потребу у використанні систем, здатних автоматично формувати графічне представлення топології та відображати стан мережі у режимі реального часу [32 – 33].

Важливим фактором також є конвергенція IT та OT-сегментів. На відміну від офісних мереж, промислові системи мають забезпечувати мінімальні затримки передачі даних і стабільність роботи каналів зв’язку. Навіть незначні затримки можуть негативно впливати на роботу виробничого обладнання. Більшість традиційних систем моніторингу не враховують специфіку таких вимог.

Крім того, із розвитком віддаленого доступу та хмарних сервісів зростає актуальність питань інформаційної безпеки. Відсутність централізованого

контролю підвищує ризик несанкціонованого доступу та порушення роботи обладнання.

Таблиця 1.1 – Основні проблеми традиційних систем моніторингу промислових мереж

Проблема	Наслідок
Гетерогенність обладнання	Ускладнення адміністрування
Відсутність централізованого контролю	Повільна локалізація збоїв
Ручна діагностика	Високі часові витрати
Відсутність актуальної топології	Помилки при аналізі
Недостатня візуалізація	Складність сприйняття

Таким чином, сучасні промислові мережі характеризуються високою складністю, гетерогенністю обладнання та недостатньою ефективністю традиційних методів моніторингу. Це створює потребу у використанні нових підходів до централізованого управління та візуалізації мережевої інфраструктури.

Одним із перспективних рішень є використання програмно-конфігурованих мереж (SDN), які забезпечують централізоване управління, спрощують збір телеметричних даних та дозволяють формувати актуальну картину стану мережі. Розробка системи графічного моніторингу дозволить підвищити ефективність адміністрування мережі та скоротити час локалізації несправностей [1 – 6].

1.2 Дослідження архітектурної парадигми SDN як фундаментальної основи для систем діагностики

Ускладнення сучасних промислових мереж та зростання вимог до швидкодії, надійності й централізованого контролю зумовлюють необхідність

використання нових підходів до побудови мережевої інфраструктури. Традиційні архітектури дедалі частіше демонструють обмеження у питаннях масштабованості, гнучкості та оперативності адміністрування. Одним із перспективних напрямів розвитку мережевих технологій є концепція програмно-конфігурованих мереж – Software Defined Networking (SDN) [1].

У традиційних мережах маршрутизатори та комутатори одночасно виконують функції передачі трафіку й управління ним. У сучасних розподілених середовищах це ускладнює масштабування, моніторинг і централізоване адміністрування мережі.

Архітектура SDN базується на розділенні площини передачі даних (Data Plane) та площини управління (Control Plane). Мережеві пристрої виконують функції пересилання пакетів, а логіка керування переноситься до централізованого програмного компонента – SDN-контролера. Такий підхід спрощує адміністрування, дозволяє швидко змінювати політики маршрутизації та забезпечує глобальне бачення стану мережі [1, 5, 9].

Використання SDN є доцільним через складну структуру мережі та необхідність стабільної роботи автоматизованих систем управління. Централізована модель дозволяє підвищити ефективність моніторингу та скоротити час локалізації несправностей.

Архітектура SDN складається з трьох основних рівнів:

– Infrastructure Layer – фізичне мережеве обладнання;

– Control Plane – рівень керування, на якому функціонує SDN-контролер;

– Application Layer – прикладні системи моніторингу, аналітики та візуалізації.

У межах даного дослідження система графічного моніторингу належить до рівня Application Layer та взаємодіє з контролером через програмні інтерфейси API.

Важливою перевагою SDN є централізований збір телеметричної інформації. У роботі як базовий елемент управління розглядається контролер

Cisco Catalyst Center, який забезпечує отримання актуальних даних про стан мережевої інфраструктури та формує глобальне уявлення про мережу.

Для взаємодії між компонентами SDN використовуються програмні інтерфейси API. Southbound API забезпечує зв'язок між контролером і мережевими пристроями, а Northbound API використовується для взаємодії контролера з прикладними системами. У сучасних реалізаціях Northbound API зазвичай базується на REST-архітектурі та дозволяє отримувати дані у форматі JSON [19, 21, 43].

На відміну від традиційного моніторингу через SNMP-запити, REST API забезпечує централізований доступ до телеметричних даних за меншої кількості мережевих звернень. Через API система може отримувати інформацію про топологію мережі, стан вузлів, завантаження каналів, використання ресурсів, а також показники затримок і втрат пакетів [7 – 8].

SDN також створює умови для побудови динамічної графічної моделі мережі, у якій вузли відображаються як вершини графа, а канали зв'язку – як ребра. Це дозволяє швидше виявляти перевантаження, аномалії та потенційні точки відмови [32 – 33].

Додатковою перевагою є можливість реалізації предиктивного аналізу на основі телеметричних даних. Система може прогнозувати можливі проблеми та дозволяє своєчасно виконувати профілактичне обслуговування мережі [34].

Централізована архітектура SDN також підвищує рівень інформаційної безпеки, оскільки контролер здатний оперативно виявляти аномальну активність і змінювати політики маршрутизації у разі виникнення загроз.

Таким чином, технологія SDN є ефективною основою для створення сучасних систем діагностики та графічного моніторингу промислових мереж. Централізоване управління, підтримка API та можливість інтеграції аналітичних засобів дозволяють підвищити ефективність адміністрування мережевої інфраструктури та забезпечити стабільність роботи підприємства.

1.3 Порівняльний аналіз програмних засобів та фреймворків для побудови високореактивних інтерфейсів

Під час розробки сучасних систем моніторингу важливе значення має вибір технологічної платформи для реалізації користувацького інтерфейсу. Саме інтерфейс забезпечує відображення телеметричних даних, взаємодію оператора з системою та оперативне реагування на зміни стану мережі. Для промислових SDN-мереж особливо важливими є висока швидкість оновлення даних, стабільність роботи та зручність візуального представлення інформації.

Традиційні вебінтерфейси з повним перезавантаженням сторінок є недостатньо ефективними для систем реального часу. Тому сучасні диспетчерські системи переважно будуються за архітектурою Single Page Application (SPA) [21 – 22].

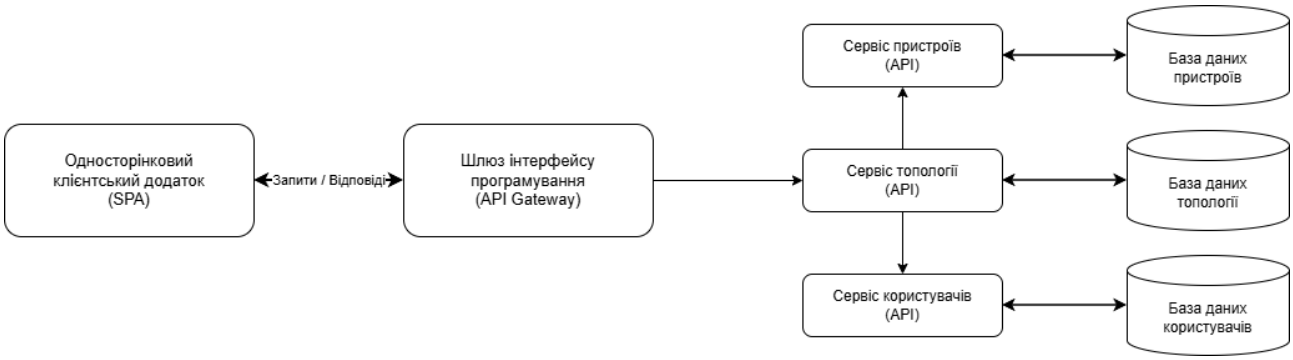


Рисунок 1.2 – Архітектура Single Page Application (SPA) з асинхронним оновленням даних

SPA-підхід передбачає одноразове завантаження основної структури застосунку з подальшим асинхронним оновленням даних без перезавантаження сторінки. Це забезпечує плавну роботу інтерфейсу та безперервне відображення телеметричної інформації.

У межах дослідження було проведено порівняльний аналіз популярних JavaScript-рішень для створення високореактивних інтерфейсів: Angular, Vue.js та React.

Angular є повноцінним фронтенд-фреймворком із великою кількістю вбудованих можливостей. Він підтримує TypeScript, маршрутизацію, механізми роботи з формами та керування станом застосунку. Angular добре підходить для масштабних корпоративних систем, однак має складну архітектуру та потребує значних ресурсів браузера. Для систем реального часу це може негативно впливати на продуктивність [12].

Vue.js є легшим та простішим у використанні фреймворком із реактивною моделлю даних. Він забезпечує швидку розробку та зручну компонентну структуру. Водночас при створенні великих систем із великою кількістю інтерактивних елементів можуть виникати труднощі з масштабуванням і підтримкою складної логіки взаємодії між компонентами.

Найбільш доцільним рішенням для даного проєкту є бібліотека React, розроблена компанією Meta. React використовує декларативний підхід і компонентну архітектуру, що спрощує масштабування застосунку та повторне використання компонентів [11, 22, 33].

Однією з основних переваг React є використання Virtual DOM, який дозволяє оновлювати лише ті елементи інтерфейсу, що фактично змінилися. Це зменшує навантаження на браузер та підвищує продуктивність системи [23].

Для задач моніторингу важливою є також технологія Reconciliation, яка забезпечує ефективне оновлення компонентів при частій зміні телеметричних даних. React оновлює лише змінені компоненти, що дозволяє підтримувати стабільну роботу інтерфейсу навіть при великій кількості одночасних оновлень.

Додатковою перевагою React є односпрямований потік даних (One-way Data Flow), який забезпечує передбачуваність роботи застосунку та спрощує налагодження.

Важливу роль у виборі React відіграє також його розвинена екосистема. Існує велика кількість бібліотек для побудови графіків, інтерактивних діаграм та візуалізації мережевих топологій, що спрощує створення диспетчерських систем моніторингу.

У межах проєкту React використовується разом із Vite – сучасним інструментом збірки, який забезпечує швидку компіляцію та підтримує механізм Hot Module Replacement (HMR). Це дозволяє миттєво оновлювати модулі застосунку без повного перезавантаження сторінки та прискорює процес розробки [13].

Таким чином, проведений аналіз показав, що React у поєднанні з Vite найбільш повно відповідає вимогам системи моніторингу SDN-мережі. Висока продуктивність, ефективна робота з телеметричними даними, масштабованість і розвинена екосистема роблять цю платформу оптимальним рішенням для побудови графічної системи моніторингу мережевої інфраструктури.

1.4. Огляд методів графічного аналізу та бібліотек візуалізації мережевих топологій

У сучасних системах моніторингу важливу роль відіграє графічне представлення мережевої інфраструктури. Зі збільшенням кількості мережевих вузлів і ускладненням топології використання лише текстових журналів та таблиць стає малоефективним. Для оператора важливо швидко визначати взаємозв’язки між вузлами, виявляти перевантажені сегменти мережі та локалізувати несправності.

У сучасних системах моніторингу мережева інфраструктура зазвичай подається у вигляді графа, де вузли мережі відображаються як вершини, а канали зв’язку – як ребра. Такий підхід дозволяє наочно демонструвати структуру мережі та виконувати аналіз маршрутів передачі даних.

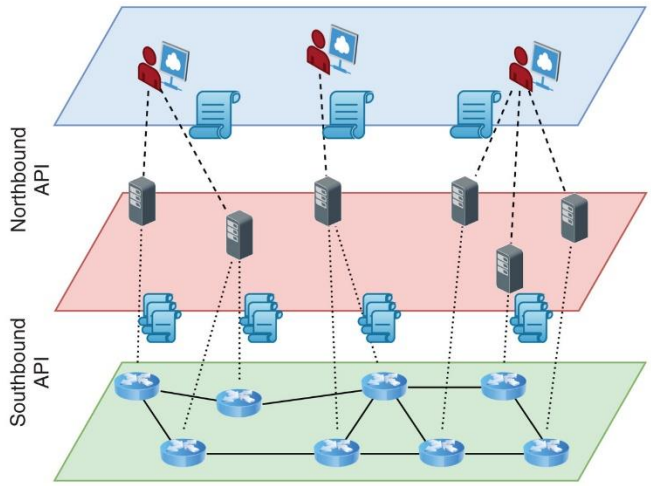


Рисунок 1.3 – Приклад графового представлення SDN-мережі

Для SDN-мереж особливо важливою є підтримка динамічного оновлення топології, оскільки стан вузлів і параметри трафіку можуть змінюватися у режимі реального часу. Тому система візуалізації повинна забезпечувати інтерактивність, масштабування та швидке оновлення елементів інтерфейсу.

У процесі дослідження було проаналізовано декілька популярних бібліотек графічної візуалізації: D3.js, Cytoscape.js та React Flow [14, 45].

1.4.1. Низькорівневий підхід на базі бібліотеки D3.js

D3.js є однією з найбільш відомих бібліотек для візуалізації даних. Вона забезпечує високий рівень гнучкості та дозволяє створювати складні інтерактивні графіки й графові структури.

Основною перевагою D3.js є повний контроль над графічними елементами та їх поведінкою. Проте бібліотека є низькорівневим інструментом і потребує значного обсягу ручного програмування. Розробнику необхідно самостійно реалізовувати механізми масштабування, переміщення вузлів та оновлення графа.

Додатковою проблемою є складна інтеграція D3.js з React через відмінності між імперативною та декларативною моделями рендерингу. Для

систем моніторингу реального часу це ускладнює підтримку коду та збільшує складність розробки.

1.4.2. Аналітичний підхід із використанням Cytoscape.js

Cytoscape.js спеціалізується на роботі з графовими структурами та мережевими топологіями. Бібліотека підтримує автоматичне розташування вузлів і має вбудовані алгоритми побудови графів.

Перевагою Cytoscape.js є висока продуктивність при роботі з великими графами та підтримка складних топологій. Це дозволяє використовувати бібліотеку для відображення великих мережевих структур.

Водночас Cytoscape.js використовує власний механізм рендерингу на базі Canvas, що ускладнює інтеграцію HTML-компонентів усередині вузлів. Крім того, інтеграція з React зазвичай потребує використання додаткових адаптерів.

1.4.3. Декларативний вузловий підхід на базі React Flow (@xyflow)

Найбільш доцільним рішенням для даного проєкту визначено бібліотеку React Flow, яка входить до екосистеми @xyflow. Вона орієнтована на створення node-based інтерфейсів та інтерактивних графових структур у середовищі React.

React Flow побудована на концепції вузлів і зв’язків, що добре відповідає структурі SDN-мережі. Кожен вузол реалізується як окремий React-компонент і може містити телеметричні дані, індикатори стану та елементи керування.

Основною перевагою React Flow є нативна інтеграція з React, що спрощує керування станом застосунку та забезпечує стабільну роботу інтерфейсу при великій кількості оновлень у реальному часі.

Бібліотека підтримує масштабування, переміщення вузлів, MiniMap, кастомні з’єднання та багаторівневу структуру графів. Це дозволяє ефективно відображати складні сегменти мережевої інфраструктури.

Для системи моніторингу React Flow є оптимальним рішенням, оскільки дозволяє поєднати візуалізацію топології мережі з інтеграцією телеметричних даних та елементів аналітики.

Таблиця 1.2 – Аналіз функціональних можливостей графічних бібліотек

Критерій порівняння	D3.js	Cytoscape.js	React Flow
Рівень абстракції	Низький (Primitive-based)	Середній (Graph-based)	Високий (Node-based)
Гнучкість дизайну вузлів	Абсолютна	Обмежена (Canvas)	Повна (React/HTML)
Швидкість розробки	Низька	Середня	Висока
Підтримка React	Через сторонні обгортки	Обмежена	Нативна (Native)
Складність ієрархії	Потребує ручного розрахунку	Вбудовані алгоритми	Підтримка Subflows

Таким чином, React Flow найбільш повно відповідає вимогам системи графічного моніторингу SDN-мережі. Бібліотека забезпечує оптимальне поєднання продуктивності, інтеграції з React та зручності реалізації інтерактивних мережевих топологій.

1.5. Обґрунтування мети та постановка завдань дослідження

Проведений аналіз промислових мережевих інфраструктур, архітектури SDN та сучасних засобів візуалізації показав необхідність використання нових підходів до моніторингу та діагностики мережевого середовища. У сучасних умовах мережева інфраструктура є критично важливою частиною виробничого

процесу, тому питання стабільності, керованості та швидкого виявлення несправностей набувають особливого значення.

Ця проблема є актуальною через складну структуру SDN-мережі, велику кількість взаємопов’язаних пристроїв та необхідність постійного контролю телеметричних даних. Традиційні методи моніторингу на основі SNMP, журналів подій і статичних схем топології вже не забезпечують достатню швидкість аналізу та зручність роботи оператора.

Особливо складною є проблема обробки великих обсягів телеметричної інформації, яку генерують сучасні SDN-контролери. Аналіз даних у текстовому вигляді ускладнює локалізацію несправностей та збільшує час реагування на аварійні ситуації.

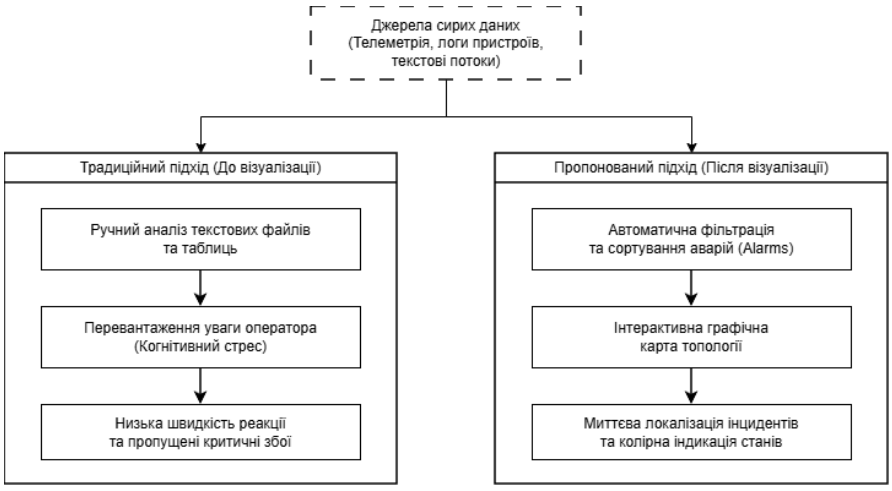


Рисунок 1.4 – Проблема перевантаження оператора телеметричними даними (до/після візуалізації)

У зв’язку з цим виникає необхідність створення системи, здатної перетворювати телеметричні дані у зрозуміле графічне представлення в режимі реального часу. Така система повинна забезпечувати інтерактивний аналіз топології мережі, візуальне відображення стану вузлів та швидкий доступ до параметрів обладнання.

Метою кваліфікаційної роботи є проєктування та розробка програмної системи візуального моніторингу та графічного аналізу стану вузлів SDN-

мережі. Реалізація системи повинна підвищити ефективність контролю мережевої інфраструктури та скоротити час локалізації несправностей.

Об’єктом дослідження є процеси моніторингу та діагностики активного мережевого обладнання у програмно-конфігурованих мережах.

Предметом дослідження є методи та програмні засоби інтерактивної візуалізації мережевих топологій, а також механізми збору та обробки телеметричних даних через API SDN-контролерів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– дослідити архітектуру SDN та методи отримання телеметричних даних;

– обґрунтувати вибір технологічного стека програмної системи;

– реалізувати серверний прошарок для взаємодії із SDN-контролером;

– розробити алгоритм побудови графічної топології мережі;

– створити інтерактивний користувацький інтерфейс системи моніторингу;

– забезпечити стабільність і продуктивність системи при роботі з потоковими телеметричними даними.

Результатом виконання роботи повинна стати програмна система моніторингу SDN-мережі, яка поєднуватиме централізований збір телеметрії, графічний аналіз топології та сучасний вебінтерфейс. Використання такої системи дозволить підвищити ефективність адміністрування мережі та забезпечити стабільнішу роботу інфраструктури.

1.6 Висновки до першого розділу

У першому розділі проаналізовано сучасний стан промислових мережевих інфраструктур та основні проблеми моніторингу SDN-мереж. Встановлено, що традиційні методи адміністрування й контролю вже не забезпечують необхідної швидкості, гнучкості та зручності аналізу в умовах складних мережевих середовищ.

У ході дослідження визначено, що технологія Software Defined Networking є ефективною основою для побудови сучасних систем моніторингу завдяки централізованому управлінню, підтримці API та можливості збору телеметричних даних у режимі реального часу.

Також проведено аналіз сучасних JavaScript-фреймворків і бібліотек візуалізації мережевих топологій. Для реалізації системи моніторингу обрано React, Vite та React Flow, які забезпечують високу продуктивність, масштабованість і підтримку інтерактивної графічної візуалізації.

На основі проведеного аналізу сформульовано мету та основні завдання кваліфікаційної роботи, пов’язані з розробкою системи візуального моніторингу та графічного аналізу стану вузлів SDN-мережі.

2. ПРОЄКТУВАННЯ СИСТЕМИ ДІАГНОСТИКИ ТА ГРАФІЧНОГО АНАЛІЗУ СТАНУ ВУЗЛІВ РОЗПОДІЛЕНОЇ МЕРЕЖІ

2.1. Концептуальна модель та багаторівнева архітектура системи

Проєктування системи діагностики та графічного аналізу стану вузлів мережі передбачає створення середовища, здатного забезпечувати збір, обробку та візуалізацію телеметричних даних у режимі реального часу.

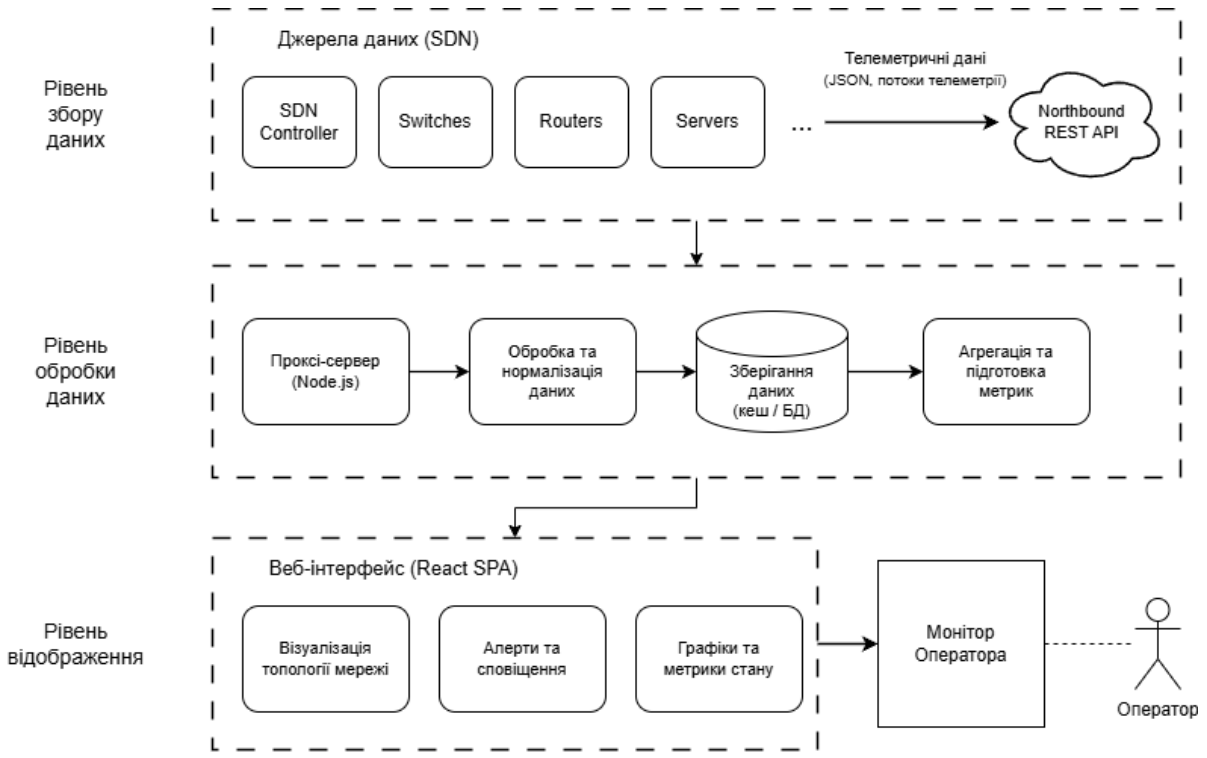


Рисунок 2.1 – Загальна концептуальна модель системи

Основною задачею системи є перетворення великих обсягів технічної інформації у зрозуміле графічне представлення, яке дозволяє оператору швидко оцінювати стан мережі та реагувати на аварійні ситуації.

Архітектура системи базується на принципі розділення відповідальностей, що дозволяє відокремити процеси збору даних, їх обробки та відображення. Такий підхід підвищує гнучкість, масштабованість і спрощує модернізацію системи [36, 39].

З урахуванням особливостей промислового середовища система реалізована у вигляді багаторівневої архітектури, яка складається з трьох основних рівнів [35].

Перший рівень відповідає за збір телеметричних даних із SDN-інфраструктури. Джерелом інформації виступають мережеві пристрої та SDN-контролер, які передають дані про стан вузлів, канали зв'язку та параметри трафіку. Взаємодія з інфраструктурою здійснюється через стандартизовані API.

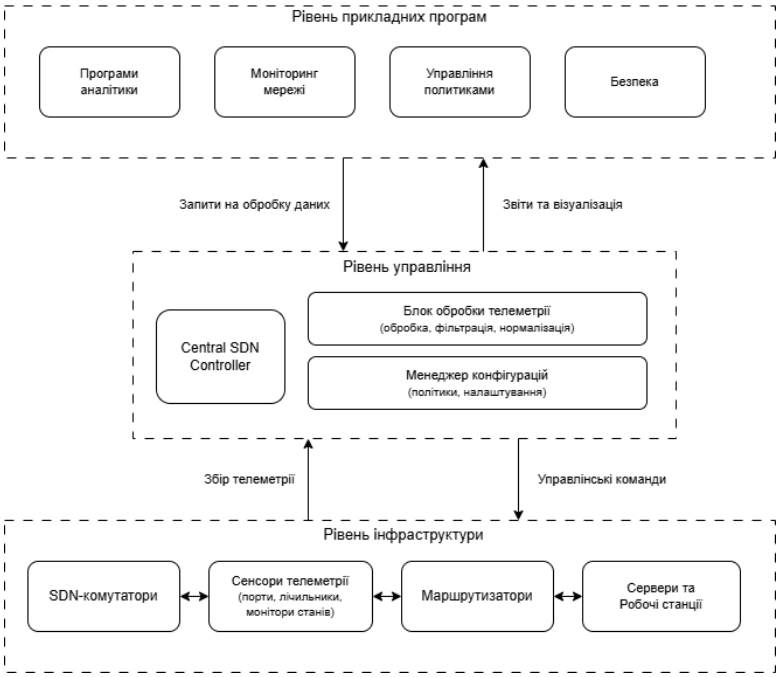


Рисунок 2.2 – Схема SDN-рівня (джерела телеметрії + мережеві пристрої + потік даних)

Другий рівень виконує функції проміжного програмного прошарку. На цьому етапі здійснюється обробка, фільтрація та нормалізація телеметричних даних. Крім того, рівень забезпечує контроль доступу, ізоляцію службової інформації та взаємодію між SDN-контролером і клієнтською частиною системи.

Важливою функцією цього рівня є зменшення інформаційного шуму шляхом відсіювання другорядних даних і підготовки лише необхідної інформації для подальшої візуалізації.

Третій рівень відповідає за графічне представлення даних та взаємодію з користувачем. На цьому рівні формується інтерактивна топологія мережі, яка дозволяє оператору аналізувати стан вузлів, переглядати параметри пристроїв і швидко виявляти проблемні сегменти мережі.

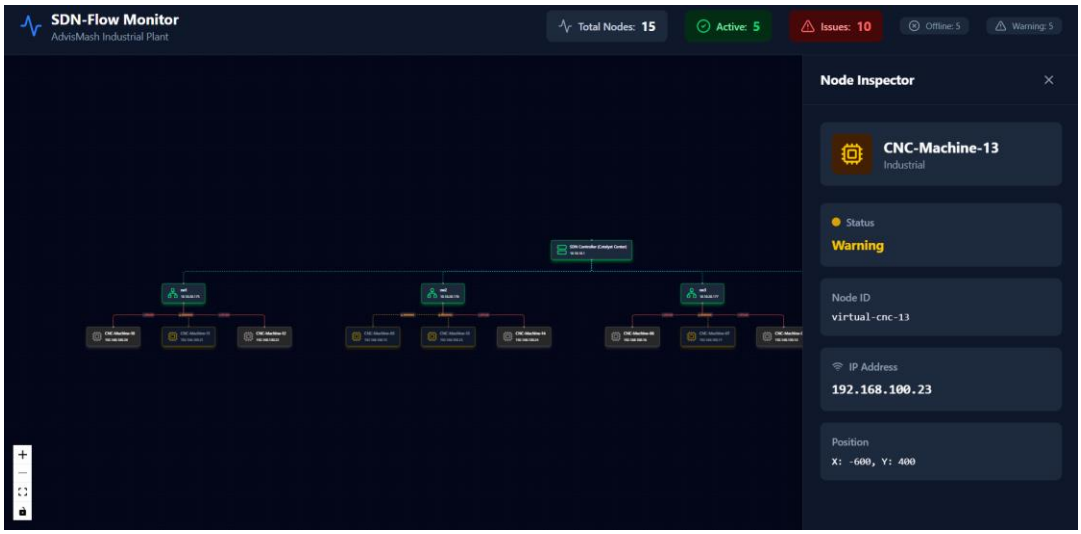


Рисунок 2.3 – Інтерфейс системи моніторингу мережі

Інтерфейс системи підтримує масштабування, навігацію між сегментами мережі та оновлення даних у режимі реального часу. Це забезпечує актуальність інформації та спрощує процес моніторингу мережевої інфраструктури.

Таким чином, запропонована багаторівнева архітектура дозволяє створити гнучку та масштабовану систему моніторингу, здатну ефективно працювати в умовах промислової SDN-інфраструктури.

Таблиця 2.1 – Рівні архітектури та їх функції

Рівень	Функція
Data Source	Збір телеметрії з SDN
Middleware	Обробка, фільтрація, безпека

Кінець таблиці 2.1

UI Layer	Візуалізація та взаємодія

2.2. Інформаційне забезпечення та моделювання структур даних топології

Інформаційне забезпечення системи моніторингу та графічного аналізу мережі забезпечує збір, обробку та структуризацію телеметричних даних, отриманих від SDN-інфраструктури. Основною задачею цього рівня є перетворення сирих даних у структуровану модель, придатну для подальшого аналізу та візуалізації.

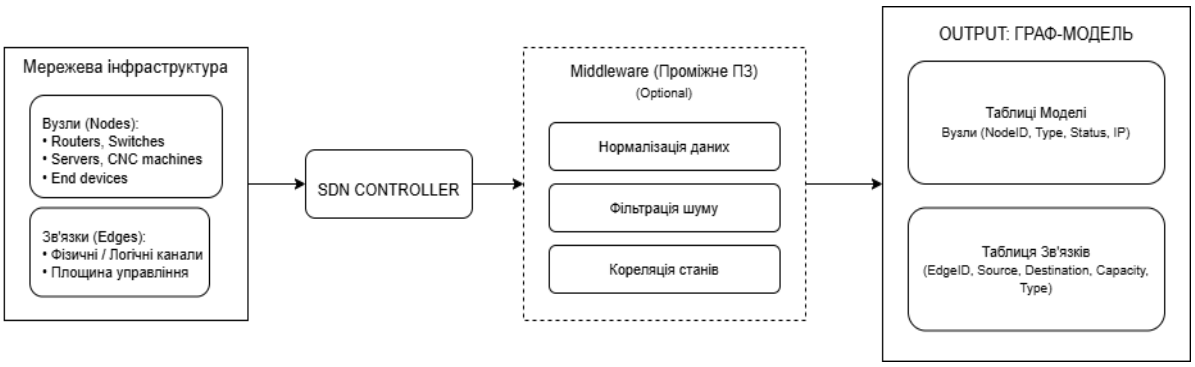


Рисунок 2.4 – Концептуальна модель інформаційного забезпечення системи

У межах системи мережева інфраструктура розглядається як сукупність взаємопов’язаних інформаційних об’єктів. Такий підхід дозволяє створити логічну модель мережі, яка відображає фізичні та логічні взаємозв’язки між пристроями.

Основними інформаційними сутностями системи є вузли мережі та топологічні зв’язки. Основними інформаційними сутностями системи є вузли мережі та топологічні зв’язки, що забезпечують формування єдиної моделі мережевої інфраструктури для її моніторингу та візуалізації.

Вузол мережі виступає цифровою моделлю фізичного пристрою та містить:

– ідентифікаційні параметри пристрою;
– мережеві реквізити;
– телеметричні показники;
– інформацію про поточний стан;
– параметри позиціонування у графічній топології.

Телеметричні дані вузлів включають показники доступності, завантаження процесора, використання пам’яті та інші параметри, необхідні для моніторингу стану мережі.

Другим типом інформаційних сутностей є топологічні зв’язки, які описують взаємодію між вузлами мережі.

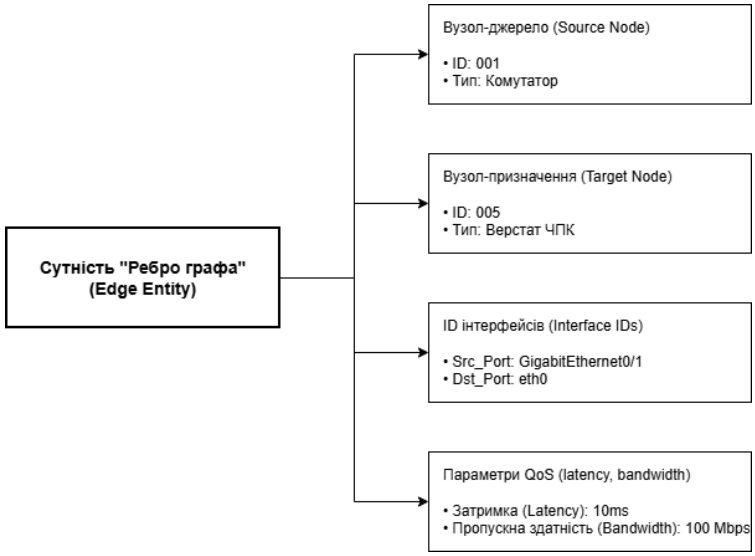


Рисунок 2.5 – Структура топологічного зв’язку

Топологічний зв’язок містить інформацію про напрям передачі даних, інтерфейси підключення та параметри якості каналу зв’язку. Це дозволяє не лише відображати структуру мережі, а й аналізувати стабільність і навантаження окремих сегментів.

Обробка інформації в системі реалізована у вигляді послідовного конвеєра. На першому етапі виконується фільтрація та нормалізація телеметричних даних, отриманих від SDN-контролера. Це дозволяє зменшити кількість надлишкової інформації та привести дані до єдиного формату.

Таблиця 2.2 – Етапи обробки даних у системі

Етап	Функція
Фільтрація	Видалення зайвих даних
Нормалізація	Приведення до єдиного формату
Семантична кореляція	Побудова графа мережі

Після нормалізації виконується семантична кореляція даних, у межах якої формується графова модель мережевої інфраструктури та встановлюються логічні зв’язки між пристроями.

Важливою частиною інформаційного забезпечення є підтримка актуальності даних у режимі, наближеному до реального часу. Для цього використовуються механізми синхронізації та кешування, які дозволяють зменшити навантаження на SDN-контролер і підвищити продуктивність системи.

Таким чином, інформаційне забезпечення створює основу для ефективної роботи системи моніторингу, забезпечуючи перетворення телеметричних потоків у структуровану модель мережі, придатну для аналізу та графічної візуалізації.

2.3. Функціональна декомпозиція підсистем моніторингу

Функціональна декомпозиція системи моніторингу дозволяє розподілити процеси збору, обробки та візуалізації даних між окремими програмними

компонентами. Такий підхід спрощує розробку, підвищує стабільність роботи системи та забезпечує можливість подальшого масштабування.

Основним принципом побудови системи є модульність. Кожна підсистема виконує власний набір функцій та взаємодіє з іншими компонентами через стандартизовані інтерфейси. Це дозволяє зменшити залежність між модулями та локалізувати можливі помилки без впливу на роботу всієї системи [26, 39].

У межах архітектури виділено чотири основні функціональні підсистеми.

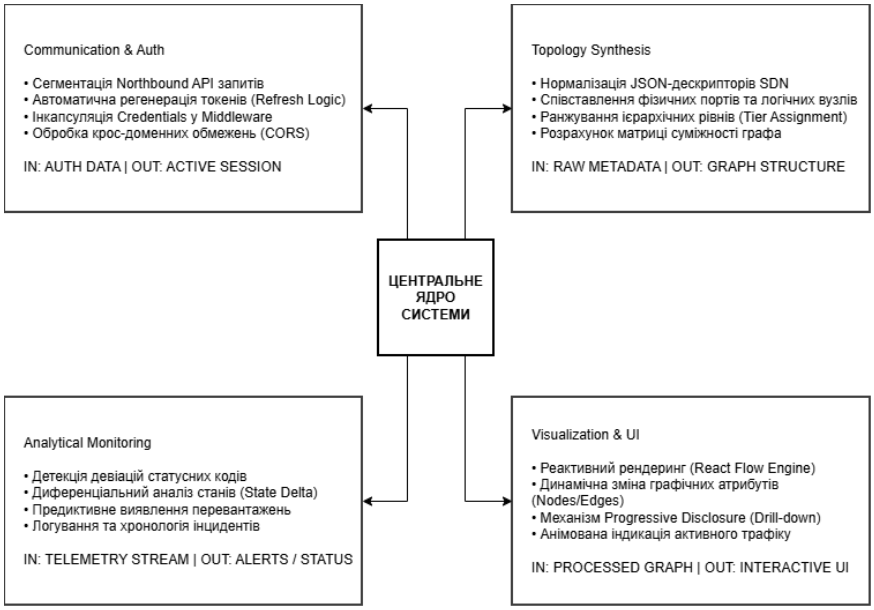


Рисунок 2.6 – Функціональна декомпозиція системи.

Першою є підсистема мережевої комунікації та управління доступом. Вона забезпечує взаємодію із SDN-контролером, автентифікацію користувачів, керування сесіями та безпечний обмін даними. Також підсистема виконує контроль інтенсивності запитів для запобігання перевантаженню мережевої інфраструктури.

Другою є підсистема топологічного синтезу та логічної обробки даних. Її основною задачею є перетворення сирих телеметричних даних у структуровану модель мережі. На цьому рівні формується граф топології, виконується аналіз зв’язків між пристроями та уніфікація даних, отриманих від різних типів обладнання.

Третьою є підсистема аналітичного моніторингу та предиктивного аналізу. Вона здійснює контроль стану мережі, аналіз телеметричних показників і виявлення відхилень у роботі вузлів. Крім того, підсистема виконує журналювання подій та аналіз зв’язності мережі.

Четвертою є підсистема візуалізації та користувацького інтерфейсу. Вона відповідає за побудову інтерактивної топологічної карти мережі та відображення стану вузлів у режимі реального часу. Інтерфейс підтримує масштабування, навігацію між сегментами мережі та перегляд детальної інформації про окремі пристрої.

Для відображення стану мережі використовуються візуальні індикатори, які дозволяють швидко визначати проблемні сегменти та аварійні вузли.

Таблиця 2.3 – Порівняльна характеристика підсистем

Підсистема	Вхід	Функція	Результат
Communication	credentials	доступ/сесії	захищений API
Topology	raw data	побудова графа	структура мережі
Analytics	telemetry	аналіз станів	алерти
Visualization	graph	UI	інтерфейс

Таким чином, функціональна декомпозиція дозволяє представити систему моніторингу у вигляді незалежних логічних модулів, кожен з яких виконує окрему функцію. Такий підхід підвищує гнучкість архітектури та спрощує подальший розвиток системи моніторингу SDN-мережі.

2.4. Обґрунтування та математичне моделювання алгоритмів обробки даних

Ефективність системи моніторингу SDN-мережі залежить від алгоритмів обробки телеметричних даних, які забезпечують аналіз стану вузлів, побудову топології та візуалізацію змін у режимі реального часу.

У промислових мережах кількість подій та обсяг телеметричних даних можуть бути значними, тому алгоритмічна частина системи виконує функцію фільтрації та обробки інформації, залишаючи лише дані, необхідні для діагностики.

Алгоритмічне забезпечення системи включає:

- управління сесійними з’єднаннями;
- побудову та обробку топології мережі;
- моніторинг і візуалізацію станів вузлів.

Рисунок 2.7 – Узагальнена схема алгоритмічної обробки даних у системі моніторингу SDN-мереж

Система побудована на асинхронній подієвій моделі, у межах якої обробка даних виконується під час виникнення подій. Це дозволяє зменшити затримки та забезпечити роботу в режимі, наближеному до реального часу [25].

2.4.1. Алгоритм управління сесійними дескрипторами та безпеки з'єднання

Алгоритм управління сесіями забезпечує безперервний доступ до SDN-контролера та підтримує актуальність авторизаційних токенів.

Перевірка актуальності токена (pre-check)

Перед виконанням кожного запиту здійснюється перевірка умови:

$$T_{current} < T_{expire} - \Delta t \quad (2.1)$$

де:

$T_{current}$ – поточний системний час;

T_{expire} – час завершення дії токена;

Δt – часовий буфер безпеки.

У разі завершення строку дії токена система виконує повторну авторизацію. Для уникнення конфліктів використовується механізм блокування оновлення сесії та черга запитів.

2.4.2. Алгоритм ієрархічного синтезу та геометричного розташування графа

Алгоритм використовується для побудови топології мережі у вигляді ієрархічного графа. Центральним елементом структури є SDN-контролер.

1. Призначення рівнів вузлам

Кожному вузлу присвоюється рівень:

R = 0 – контролер

R = 1 – core-комутатори

R = 2 – access-комутатори

R = 3 – кінцеві пристрої

2. Вертикальне позиціонування

$$Y = R \cdot H_{step} \tag{2.2}$$

де:

H_{step} – фіксована відстань між рівнями.

3. Горизонтальне розташування

Позиція по осі X визначається як балансування піддерев:

$$X = \frac{(\sum X_i)}{n} \tag{2.3}$$

де:

X_i – координати дочірніх вузлів;

n – кількість дочірніх елементів.

Це забезпечує симетричне розташування елементів графа.

4. Усунення накладань

Якщо відстань між вузлами менша за допустимий мінімум, виконується їх розведення:

$$distance(node_i, node_j) < D_{\min} \rightarrow shift(node_j) \tag{2.4}$$

5. Побудова зв’язків

Зв’язки між вузлами формуються як криві траєкторії з плавною інтерполяцією для уникнення перетинів.

2.4.3. Алгоритм динамічного моніторингу станів та візуального алертингу

Алгоритм забезпечує безперервний контроль стану вузлів у режимі реального часу та формує візуальні сигнали про зміни стану мережі.

Порівняння станів

$$\Delta S = S\_current - S\_previous \quad (2.5)$$

Якщо $\Delta S \neq 0$, виконується оновлення відповідного вузла.

У системі використовуються такі стани:

– ONLINE – нормальна робота;
– WARNING – перевищення порогових значень;
– ALARM – критичний стан або недоступність вузла.

Для візуалізації використовуються кольорові індикатори:

– ONLINE – зелений;
– WARNING – жовтий;
– ALARM – сірий або червоний.

Логування подій

Кожна зміна стану фіксується у журналі подій:

```
EventLog = {time, node_id, old_state, new_state}
```
(2.6)

Це забезпечує можливість подальшого аналізу інцидентів та діагностики причин відмов.

2.5. Проєктування інтерактивного інтерфейсу та логіки взаємодії (UX/UI)

Інтерфейс системи моніторингу призначений для швидкого аналізу стану мережевої інфраструктури та оперативного реагування на зміни у роботі SDN-мережі.

Основною задачею UX/UI-проєктування є зменшення когнітивного навантаження на оператора та забезпечення швидкого доступу до критично важливої інформації [27, 35].

Інтерфейс реалізовано у вигляді інтерактивного графічного полотна, на якому мережева інфраструктура відображається як динамічний граф.

Система підтримує:

- переміщення робочої області (Pan);
- масштабування (Zoom);
- переміщення вузлів (Drag-and-drop).

Для різних типів обладнання використовуються окремі графічні позначення:

- сервери;
- комутатори;
- маршрутизатори;
- промислові пристрої;
- SDN-контролери.

Для швидкого визначення стану вузлів використовується кольорова індикація.

Таблиця 2.4 – Візуальна семантика

Стан вузла	Колір	Характеристика
ONLINE	Зелений	Нормальна робота
WARNING	Помаранчевий	Попередження або перевищення порогів
CRITICAL	Червоний	Критичний збій або втрата доступності

Інтерфейс використовує темну тему оформлення, що зменшує навантаження на зір та покращує сприйняття графічних елементів.

Для уникнення перевантаження інтерфейсу використовується принцип поступового розкриття інформації (Progressive Disclosure). Детальні параметри вузла відображаються лише після взаємодії користувача з елементом мережі.

Приклад логіки взаємодії:

```
onNodeClick(node):
show(node_details)
```

Для покращення сприйняття інформації використовуються динамічні ефекти:

- анімація активного трафіку;
- пульсація аварійних вузлів;
- плавні переходи між станами;
- динамічна зміна кольору елементів.

Це дозволяє оператору швидко визначати проблемні сегменти мережі та оперативно реагувати на аварійні ситуації.

2.6. Проєктні рішення щодо забезпечення кібербезпеки та відмовостійкості

Під час проєктування системи моніторингу SDN-мережі питання кібербезпеки та відмовостійкості розглядаються як базові складові архітектури. Це пов’язано з тим, що система взаємодіє з критично важливою мережевою інфраструктурою підприємства та промисловим обладнанням [29, 47].

У системі використовується принцип багаторівневого захисту (Defense in Depth), відповідно до якого кожен рівень архітектури виконує окрему функцію безпеки [29, 47].

Стратегії забезпечення інформаційної безпеки

Одним із ключових рішень є використання проміжного серверного рівня (proxy-рівня), який виступає єдиною точкою взаємодії між клієнтською частиною та SDN-контролером. Це дозволяє ізолювати службові дані від клієнтського інтерфейсу.

Усі конфіденційні параметри доступу зберігаються лише на серверному рівні у змінних середовища.

Приклад логіки зберігання:

```
SERVER_ENV={
API_KEY,
ADMIN_PASSWORD,
TOKEN_SECRET
}
```

Такий підхід зменшує ризик компрометації облікових даних та підвищує рівень захисту системи.

Для контролю доступу використовується механізм перевірки дозволених джерел запитів.

Спрощена логіка перевірки:

```
If request.origin in ALLOWED_DOMAINS:
allow()
else:
reject()
```

Це дозволяє обмежити несанкціонований доступ до серверного API.

Усі телеметричні дані проходять етап перевірки та очищення перед подальшою обробкою:

```
Clean-data = sanitize(raw-data)
```
(2.7)

Такий механізм дозволяє виключити обробку некоректних або потенційно небезпечних даних.

Проєктні рішення щодо відмовостійкості

Відмовостійкість системи забезпечується механізмами автоматичного відновлення з’єднань та підтримки роботи при часткових збоях.

Система підтримує автоматичне оновлення сесійних токенів без переривання моніторингу.

Приклад логіки:

```
if session.expired:
refresh_session()
```

У випадку недоступності SDN-контролера система використовує останній збережений стан мережі:

```
if api_unavailable:
load_cached_state()
```

Це дозволяє уникнути повного припинення роботи інтерфейсу при короткочасних збоях зовнішнього API.

Для підвищення стабільності використовується механізм кешування станів мережі, який зменшує кількість повторних звернень до контролера та прискорює відновлення системи після помилок з’єднання.

Також реалізовано принцип граціозної деградації. У разі перевантаження або часткової відмови система тимчасово відключає другорядні функції, зберігаючи базові механізми моніторингу.

Окремим компонентом є підсистема журналювання подій, яка забезпечує аудит дій та фіксацію змін стану системи [28].

Приклад структури журналу подій:

```
log_event({
timestamp,
event_type,
node_id,
state
})
```

Журналювання дозволяє виконувати аналіз інцидентів, виявляти причини збоїв та контролювати роботу системи.

Таким чином, використані проєктні рішення забезпечують захист інформаційної інфраструктури, стабільність роботи системи та підтримку безперервного моніторингу SDN-мережі в умовах промислового середовища.

2.7 Висновки до другого розділу

У другому розділі виконано проєктування системи діагностики та графічного аналізу стану вузлів SDN-мережі для кваліфікаційної роботи. Сформовано концептуальну модель системи та визначено її багаторівневу архітектуру.

Було розроблено інформаційну модель мережевої топології, яка забезпечує перетворення телеметричних даних у графову структуру, придатну для аналізу та візуалізації. Проведена функціональна декомпозиція дозволила визначити основні підсистеми та їх роль у процесі моніторингу мережі.

Також розглянуто алгоритми управління сесіями, побудови топології та динамічного моніторингу станів вузлів, що забезпечують стабільну роботу системи в режимі реального часу.

Окрему увагу приділено проєктуванню інтерактивного користувацького інтерфейсу, орієнтованого на швидке сприйняття інформації та зручність роботи оператора. Додатково визначено основні рішення щодо забезпечення кібербезпеки, контролю доступу та відмовостійкості системи.

У результаті сформовано архітектурну та алгоритмічну основу програмного комплексу для подальшої реалізації системи моніторингу SDN-мережі промислового підприємства.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ГРАФІЧНОГО МОНІТОРИНГУ SDN-ІНФРАСТРУКТУРИ

3.1 Інтеграція з SDN-контролером та базове налаштування середовища

3.1.1 Організація взаємодії з хмарною інфраструктурою Cisco Catalyst Center Sandbox

Для розробки та тестування системи графічного моніторингу було використано хмарне середовище Cisco Catalyst Center Sandbox у конфігурації Always-On. Це дозволило працювати з реальним SDN-контролером без необхідності розгортання локальної мережевої інфраструктури [46].

Використання Cisco Sandbox забезпечило доступ до телеметричних даних, мережевих пристроїв і фізичної топології корпоративної SDN-мережі. На відміну від емуляторів або статичних наборів даних, середовище дозволяє тестувати систему в умовах, наближених до реальної експлуатації.

Взаємодія із контролером здійснюється через Northbound REST API. Доступ до API забезпечує отримання телеметричних даних, інформації про мережеві пристрої та топологію SDN-мережі. Обмін даними виконується за допомогою HTTP-запитів із поверненням результатів у форматі JSON. Базова адреса API:

https://sandboxdnac.cisco.com

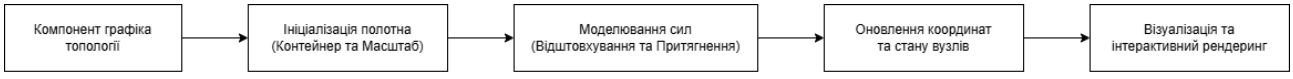


Рисунок 3.1 – Схема взаємодії клієнтського застосунку з Cisco Catalyst Center через серверний прошарок

Процес взаємодії починається з автентифікації через endpoint:

```
/dna/system/api/v1/auth/token
```

Після авторизації система отримує токен доступу для подальших API-запитів.

У backend-рівні використовується кешування токена з часом життя 55 хвилин [7].

Таблиця 3.1 – Основні параметри механізму автентифікації Cisco Catalyst Center

Параметр	Значення	Призначення
Тип автентифікації	Basic Authentication	Первинне підтвердження доступу
Endpoint авторизації	/dna/system/api/v1/auth/token	Отримання токена доступу
Тривалість життя токена	60 хвилин	Час дії сесії Cisco
Час кешування у системі	55 хвилин	Попереджувальне оновлення токена
Механізм повторної авторизації	Automatic Retry	Відновлення сесії при 401
Формат передачі даних	JSON	Формат API-відповідей
Протокол взаємодії	HTTPS	Захищений обмін даними

При отриманні HTTP 401 система автоматично оновлює токен доступу.

У такому випадку система повторно виконує авторизацію та повторює запит.

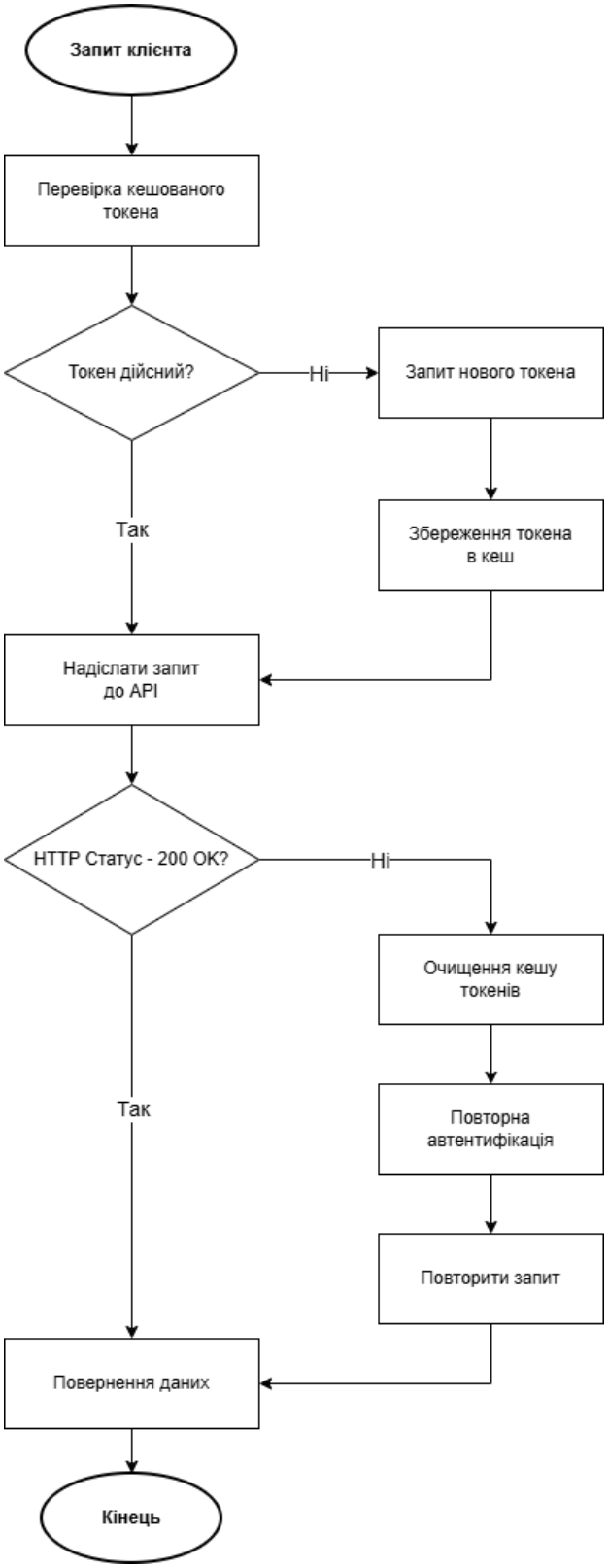


Рисунок 3.2 – Алгоритм повторної автентифікації при втраті токена доступу

Особливістю Cisco Sandbox є використання самопідписаних SSL-сертифікатів. Для підтримки HTTPS-з’єднання у середовищі Node.js було

використано власний HTTPS-агент із вимкненою перевіркою сертифіката. Таке рішення застосовується лише у тестовому середовищі [20].

Після проходження авторизації система виконує отримання телеметричних даних із декількох API-ресурсів Cisco Catalyst Center.

Основні endpoint:

- `/dna/intent/api/v1/network-device`
- `/dna/intent/api/v1/topology/physical-topology`

Endpoint /network-device використовується для отримання інформації про пристрої, а /physical-topology – для побудови структури мережі.

Отримані дані проходять етап нормалізації та перетворюються у внутрішній формат системи. Це дозволяє відокремити клієнтський застосунок від специфіки Cisco API та використовувати єдину модель даних для подальшої візуалізації у React Flow.

Інтеграція з Cisco Catalyst Center Sandbox дозволила тестувати систему з використанням реальної телеметрії та топології SDN-мережі.

Рисунок 3.3 – Процес нормалізації телеметричних даних

Завдяки попередній обробці та нормалізації даних вдалося забезпечити стабільну роботу клієнтського інтерфейсу навіть при циклічному оновленні телеметрії. Використання єдиного внутрішнього формату також спростило подальшу обробку інформації, побудову мережевого графа та інтеграцію механізмів моніторингу у frontend-рівні системи.

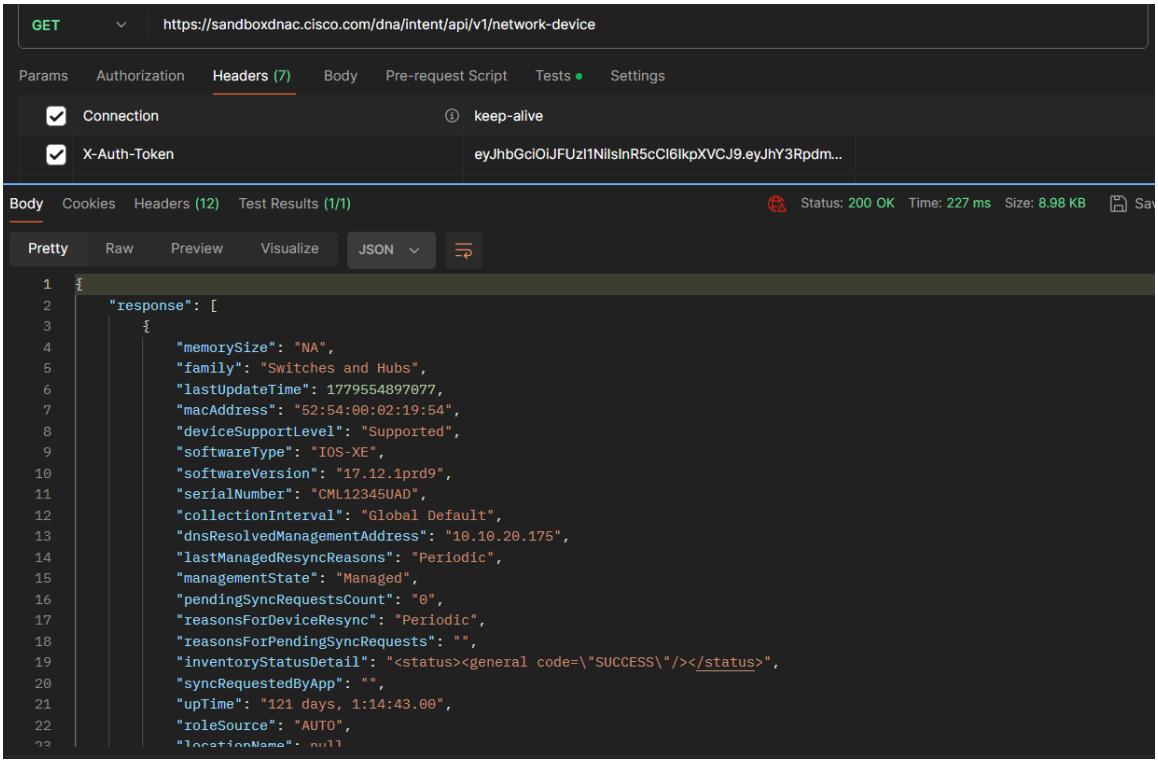


Рисунок 3.4 – Отримання мережевих пристроїв через Cisco Catalyst Center API

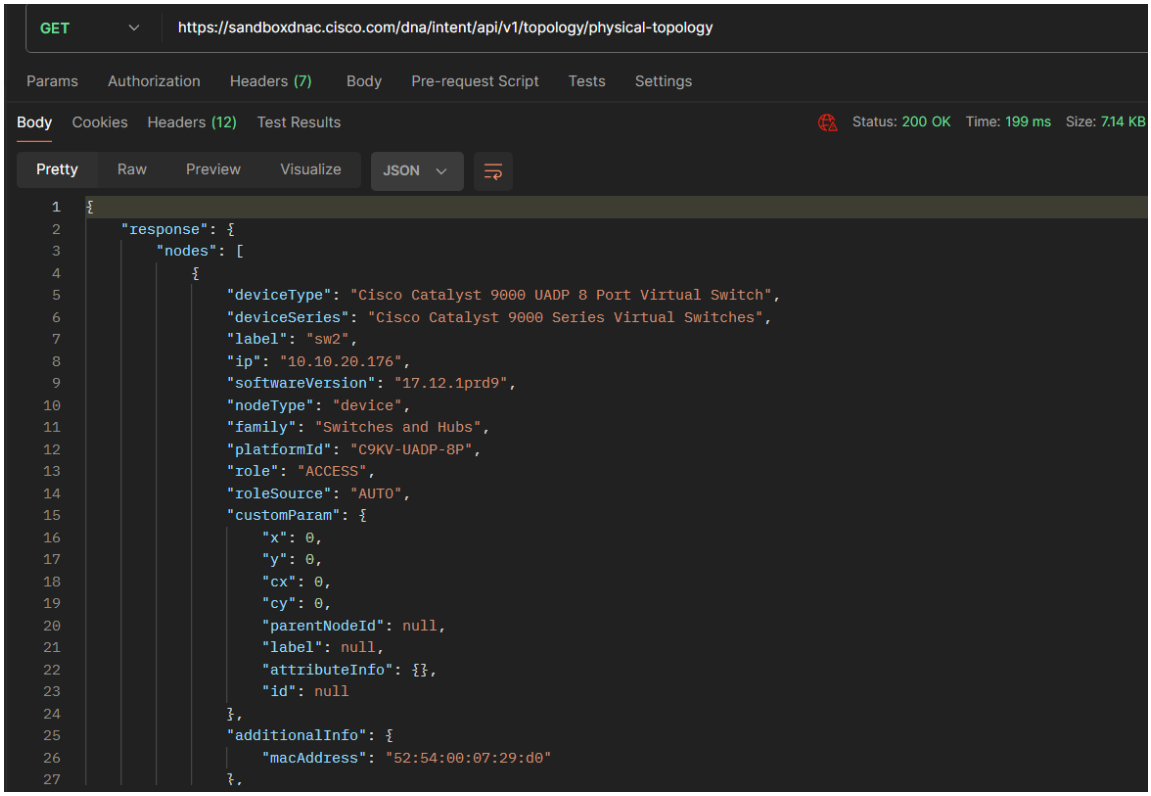


Рисунок 3.5 – Отримання фізичної топології через Cisco Catalyst Center API

3.1.2 Архітектура проєкту та розподіл відповідальності компонентів

Система графічного моніторингу SDN-інфраструктури побудована за клієнт-серверною архітектурою з використанням принципу Separation of Concerns – розділення відповідальності між окремими компонентами. Такий підхід дозволив створити масштабовану та гнучку структуру, у якій кожен модуль виконує власний набір функцій [35, 39].

У межах системи виділено три основні рівні:

- рівень доступу до даних (Data Access Layer);
- рівень бізнес-логіки (Business Logic Layer);
- рівень інтерфейсу користувача (Presentation/UI Layer).

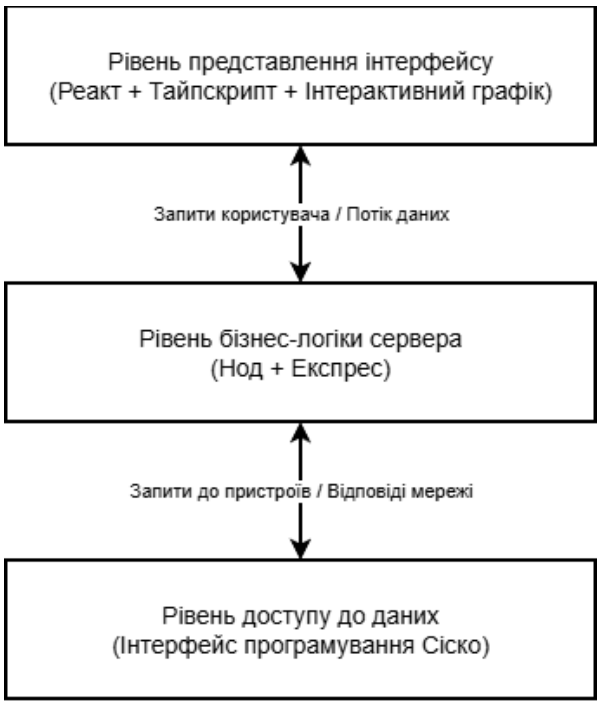


Рисунок 3.6 – Багаторівнева архітектура системи графічного моніторингу SDN-інфраструктури

Серверна частина реалізована на базі Node.js із використанням Express.js. Backend-рівень відповідає за:

– автентифікацію;
– виконання API-запитів;
– агрегацію телеметричних даних;
– нормалізацію інформації;
– формування структури топології.

Використання окремого backend-рівня дозволило ізолювати механізми авторизації та приховати службові дані від клієнтської частини. У результаті frontend-застосунок взаємодіє лише з внутрішнім API системи.

Додатково backend виконує обробку топології мережі:
– визначення ієрархічних рівнів вузлів;
– фільтрацію надлишкових зв’язків;
– валідацію структури графа;
– усунення дублікатів;
– генерацію віртуальних вузлів.

Таблиця 3.2 – Розподіл відповідальності між компонентами системи

Компонент	Основні функції
Frontend (React)	Візуалізація топології, взаємодія з користувачем, відображення телеметрії
Backend (Node.js/Express.js)	Автентифікація, обробка API-запитів, нормалізація даних, синтез топології
Cisco Catalyst Center	Надання телеметричних даних та інформації про мережеву топологію
Zustand Store	Глобальне управління станом клієнтського застосунку
React Flow	Рендеринг графових структур та інтерактивної топології

Для виконання HTTP-запитів використовується бібліотека Axios. Асинхронна модель дозволяє паралельно отримувати дані з декількох API-ресурсів.

Клієнтська частина реалізована як SPA-застосунок на базі React. React використовується для побудови SPA-інтерфейсу з підтримкою динамічного оновлення компонентів.

TypeScript використовується для суворої типізації компонентів і телеметричних структур.

Збірка frontend-застосунку виконується за допомогою Vite, який забезпечує:

– швидкий запуск середовища розробки;
– підтримку Hot Module Replacement (HMR);
– оптимізацію production-збірки.

Архітектура frontend-рівня побудована за модульним принципом. Компоненти інтерфейсу, сервіси, логіка стану та допоміжні утиліти розміщені у окремих модулях.

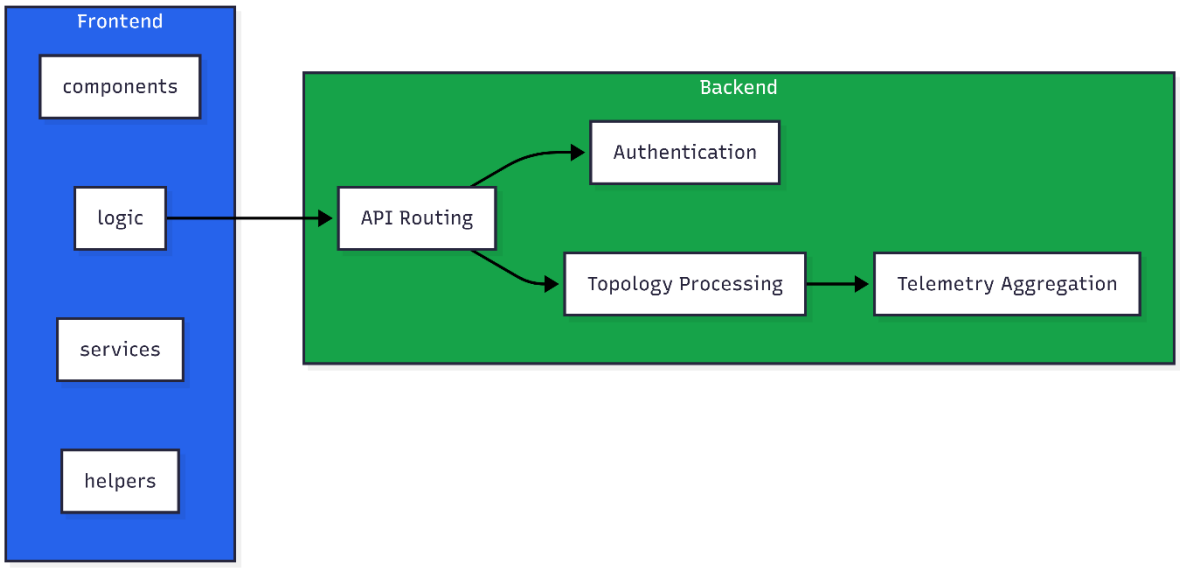


Рисунок 3.7 – Структурна організація модулів програмного комплексу

Для глобального управління станом використовується бібліотека Zustand. На відміну від складніших рішень, таких як Redux, Zustand забезпечує простішу структуру керування станом та зменшує кількість зайвих повторних рендерингів компонентів.

3.2 Проєктування та реалізація серверного прошарку (Backend)

3.2.1 Алгоритм автентифікації, кешування токенів та маршрутизація запитів

Серверний прошарок системи моніторингу виконує роль проміжного рівня між клієнтським застосунком та SDN-контролером Cisco Catalyst Center. Backend-рівень відповідає за автентифікацію, обробку API-запитів та підготовку телеметрії.

Використання окремого серверного прошарку дозволяє:

- ізолювати облікові дані та токени доступу;
- зменшити навантаження на Cisco API;
- централізувати обробку помилок;
- реалізувати механізми кешування та повторної авторизації.

Серверна частина реалізована на базі Node.js та Express.js. Для виконання HTTP-запитів використовується бібліотека Axios, яка забезпечує підтримку асинхронної взаємодії та роботу із HTTPS-з'єднаннями [16, 25, 44].

Основою механізму авторизації є функція getValidToken, яка відповідає за отримання та перевірку токена доступу.

Після успішної авторизації токен зберігається у пам'яті сервера та повторно використовується для наступних API-запитів. Такий підхід дозволяє зменшити кількість звернень до endpoint авторизації та підвищити швидкодію системи. У випадку завершення строку дії токена backend автоматично виконує повторну автентифікацію без втручання користувача.

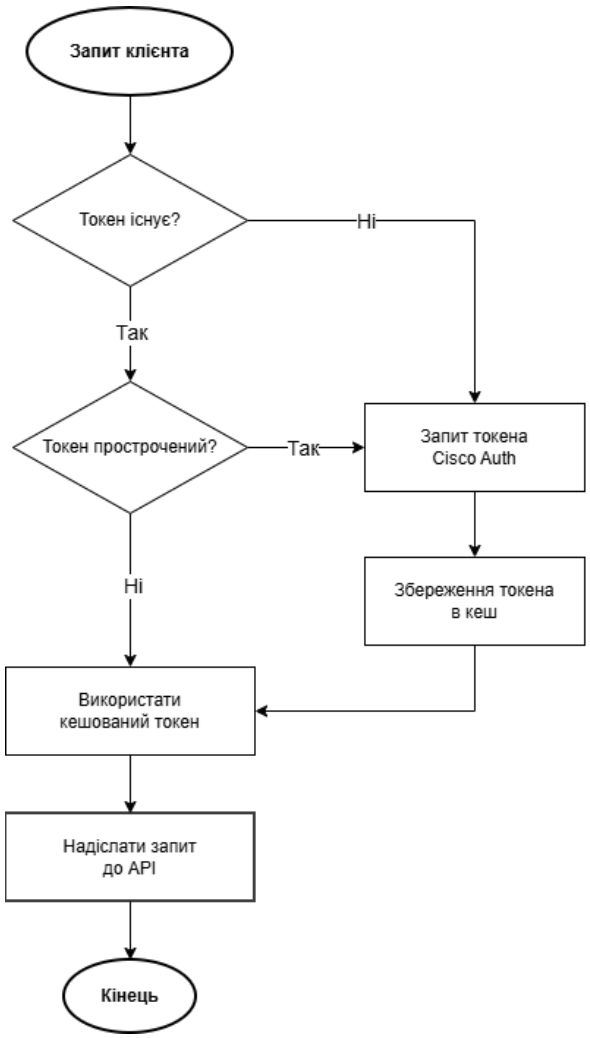


Рисунок 3.8 – Алгоритм отримання та кешування токена доступу

Для оптимізації роботи системи використовується механізм кешування токенів у пам’яті сервера. Час життя кешу визначається як:

$$T = 55 \cdot 60 \cdot 1000 \tag{3.1}$$

де:

T – час життя кешованого токена у мілісекундах.

Використання 55-хвилинного інтервалу дозволяє оновлювати токен до завершення строку його дії та уникати розриву сесії під час моніторингу мережі.

Для уніфікації мережевої взаємодії реалізовано функцію makeCiscoRequest, яка:

– додає заголовки авторизації;
– конфігурує HTTPS-з’єднання;
– виконує обробку помилок;
– контролює тайм-аути запитів.

У випадку отримання відповіді HTTP 401 Unauthorized система автоматично:

– очищує кеш токенів;
– виконує повторну авторизацію;
– повторно відправляє запит до API.

Це дозволяє автоматично відновлювати сесію при втраті токена.

Для контролю стабільності роботи використовуються тайм-аути:

– 10 секунд – для авторизації;
– 15 секунд – для отримання топології.

Таблиця 3.3 – Основні параметри серверного прошарку

Параметр	Значення	Призначення
Backend Framework	Express.js	Обробка HTTP-запитів
Runtime Environment	Node.js	Виконання серверної логіки
HTTP Client	Axios	Взаємодія з Cisco API
Token Cache Lifetime	55 хвилин	Оптимізація авторизації
Auth Timeout	10 секунд	Контроль часу авторизації
API Timeout	15 секунд	Контроль отримання топології
Authentication Type	Basic Authentication	Доступ до Cisco Sandbox
Data Format	JSON	Формат телеметричних даних

Архітектура backend-рівня побудована на розділенні повної синхронізації топології та окремого оновлення телеметрії.

У системі реалізовано два основні REST-маршрути:

- `/api/topology`
- `/api/topology/status`

Маршрут /api/topology використовується для побудови повної структури мережевого графа. У межах цього процесу сервер паралельно отримує:

- список мережевих пристроїв;
- інформацію про фізичну топологію.

Для паралельного виконання API-запитів використовується Promise.all.

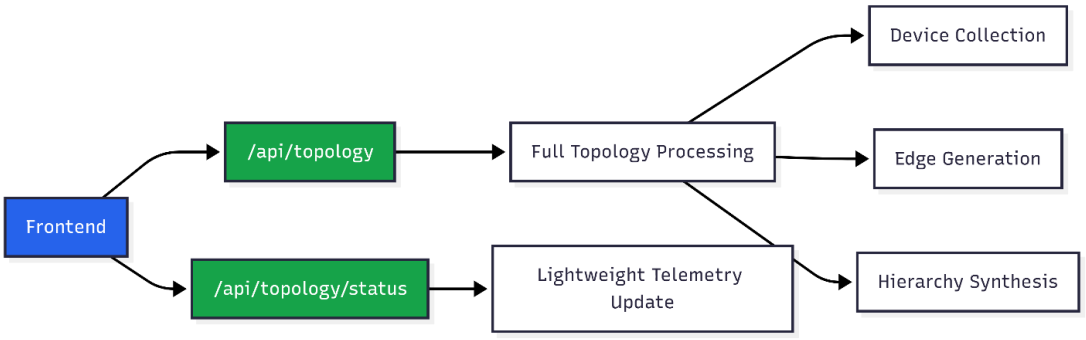


Рисунок 3.9 – Маршрутизація запитів у серверному прошарку

Після отримання телеметричних даних backend виконує:

- нормалізацію JSON-структур;
- синтез ієрархії вузлів;
- генерацію віртуальних вузлів;
- фільтрацію зв’язків;
- перевірку цілісності графа.

Рисунок 3.10 – Конвеєр обробки телеметричних даних у backend-рівні

У результаті frontend отримує готову структуру топології, придатну для безпосереднього рендерингу у React Flow.

Маршрут /api/topology/status використовується для легковагового оновлення телеметрії без повторного формування графа. У цьому режимі оновлюються:

– статуси вузлів;
– CPU-навантаження;
– час безвідмовної роботи;
– телеметричні параметри пристроїв.

Це дозволяє оновлювати телеметрію без повторного формування топології.

Додатково backend-рівень підтримує централізовану систему журналювання подій, яка фіксує:

– помилки авторизації;
– результати фільтрації зв’язків;
– проблеми побудови топології;
– перевірки цілісності графа.

Серверний прошарок забезпечує взаємодію із Cisco API та підготовку телеметрії для візуалізації SDN-мережі.

3.2.2 Синтез ієрархічної топології та гібридна генерація віртуальних вузлів

Однією з ключових задач системи графічного моніторингу є автоматичне формування структурованої топології мережі на основі телеметричних даних Cisco Catalyst Center. Оскільки SDN-контролер повертає інформацію у вигляді сирих JSON-структур без координатного позиціонування та логіки графічного компонування, виникає необхідність попереднього синтезу топології на серверному рівні.

Для вирішення даної задачі у backend-прошарку реалізовано алгоритм generateHierarchicalPositions, який виконує автоматичне просторове компонування вузлів та забезпечує побудову впорядкованої мережевої структури без накладання графічних елементів.

Алгоритм базується на ієрархічній моделі SDN-мережі та поділяє всі вузли на три основні рівні:

– ядро мережі;
– комутатори доступу;
– кінцеві промислові пристрої.

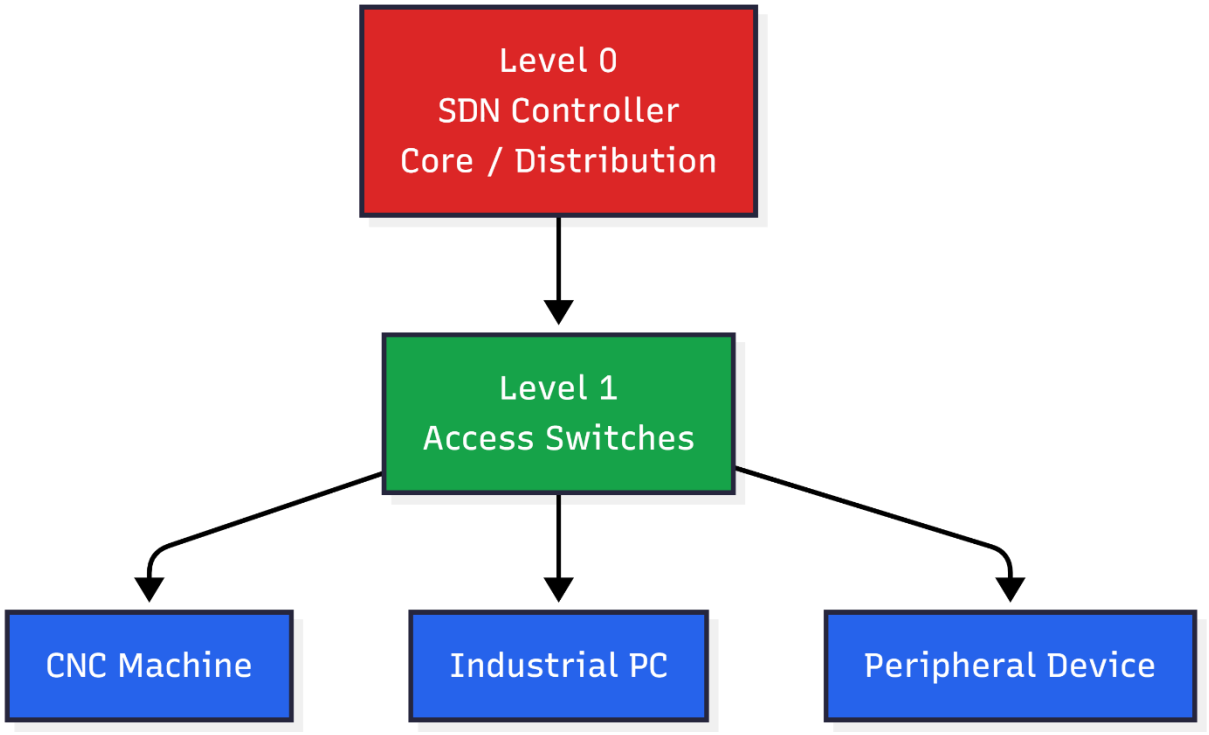


Рисунок 3.11 – Ієрархічна структура мережевої топології

Кожному рівню відповідає фіксована координата по вертикальній осі. Рівень ядра мережі розташовується у верхній частині графа:

$$y = 0 \tag{3.2}$$

На цьому рівні розміщуються:

– SDN-контролер;
– комутатори Core;
– вузли Distribution-рівня.

Подібне компонування дозволяє сформувати візуальний центр управління мережею та швидко ідентифікувати критичні елементи інфраструктури.

Другий рівень використовується для комутаторів доступу:

$$y = 200 \quad (3.3)$$

Для запобігання накладанню зв’язків між комутаторами використовується збільшений горизонтальний інтервал:

$$d = 1200 \quad (3.4)$$

де:

d – відстань між комутаторами у пікселях.

Збільшене горизонтальне рознесення суттєво покращує читабельність топології та мінімізує візуальні конфлікти між ребрами графа.

Третій рівень використовується для кінцевих промислових пристроїв:

$$y = 400 \quad (3.5)$$

На даному рівні розміщуються:

– CNC-верстати;

– промислові ПК;

– периферійне обладнання;

– виробничі контролери.

Кінцеві вузли групуються виключно під відповідними комутаторами доступу, що забезпечує логічне відображення структури виробничої мережі. Для уникнення накладання елементів використовується горизонтальний крок 350 пікселів між вузлами.

Окрему роль у системі відіграє алгоритм фільтрації мережевих зв’язків, реалізований у модулі mapLinksToEdges.

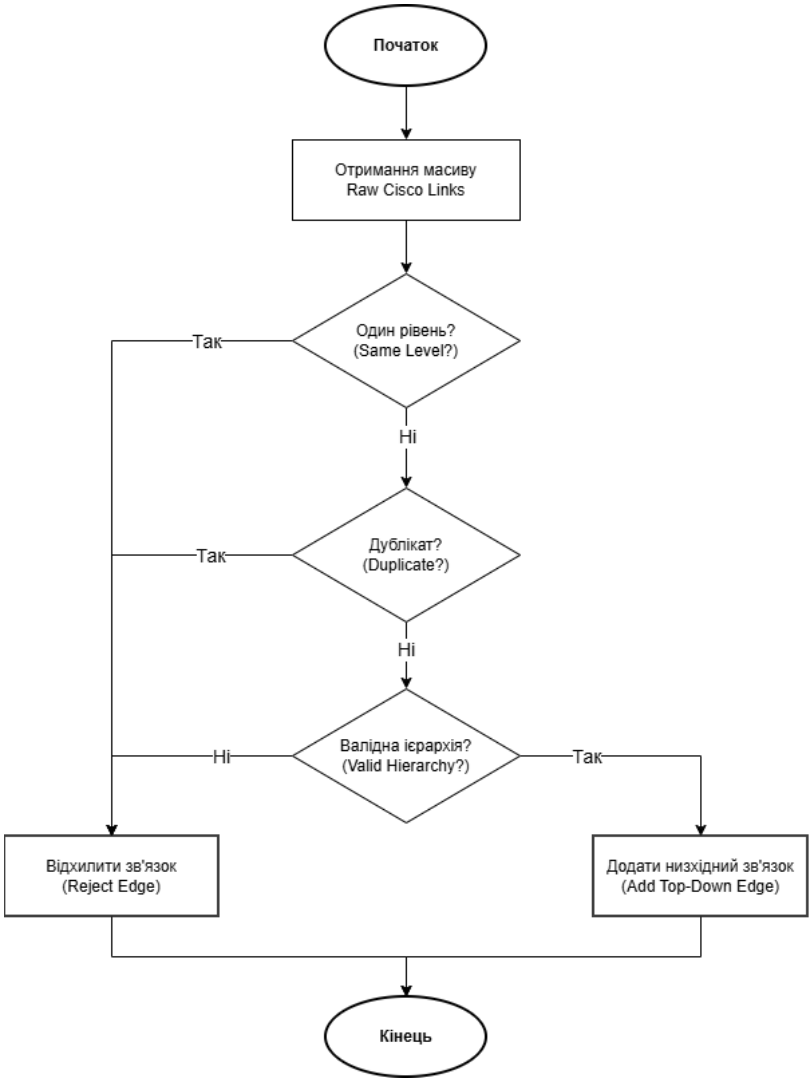


Рисунок 3.12 – Алгоритм фільтрації мережевих зв’язків

У процесі побудови графа система автоматично:

– відкидає peer-to-peer-з’єднання між вузлами одного рівня;

– усуває дублікати ребер;

– перевіряє коректність ієрархії;

– контролює існування вузлів перед створенням зв’язків.

У результаті візуалізуються лише top-down-з’єднання, які формують деревоподібну структуру графа та значно покращують когнітивне сприйняття топології оператором [33].

Важливою особливістю системи є підтримка гібридної генерації віртуальних вузлів. Тестове середовище Cisco Catalyst Center Sandbox містить

обмежену кількість кінцевих пристроїв, що не дозволяє повноцінно моделювати складну промислову інфраструктуру.

Для усунення цього обмеження реалізовано алгоритм generateVirtualIndustrialNodes, який автоматично доповнює топологію віртуальними CNC-вузлами.

У системі використовується мінімальний поріг:

$$N_{min} = 15$$

Якщо кількість реальних вузлів менша за дане значення, backend-рівень автоматично генерує додаткові CNC-вузли із унікальними ідентифікаторами, IP-адресами та телеметричними параметрами.

Таблиця 3.4 – Параметри генерації та компонування топології

Параметр	Значення	Призначення
Level 0 Coordinate	y = 0	Розміщення ядра мережі
Level 1 Coordinate	y = 200	Рівень комутаторів доступу
Level 2 Coordinate	y = 400	Рівень промислових вузлів
Switch Spacing	1200 px	Запобігання перетину ребер
CNC Spacing	350 px	Групування промислових вузлів
Minimum Device Count	15	Поріг генерації virtual nodes
Edge Filter Type	Top-Down	Побудова деревоподібної структури

Генерація віртуальних вузлів використовується як механізм моделювання промислового середовища та дозволяє:

- перевіряти масштабованість системи;
- тестувати стабільність рендерингу;
- моделювати великі виробничі сегменти;
- оцінювати поведінку системи при збільшенні кількості вузлів.

Для забезпечення реалістичності частині віртуальних вузлів автоматично призначаються аварійні або деградаційні стани:

– Partial;
– Warning;
– Unreachable;
– Offline.

Подібний підхід дозволяє моделювати:

– втрату зв’язку;
– деградацію обладнання;
– перевантаження мережі;
– аварійні сегменти SDN-інфраструктури.

Функція generateVirtualEdges виконує рівномірний розподіл віртуальних вузлів між фізичними комутаторами доступу, що дозволяє формувати симетричну та збалансовану структуру графа.

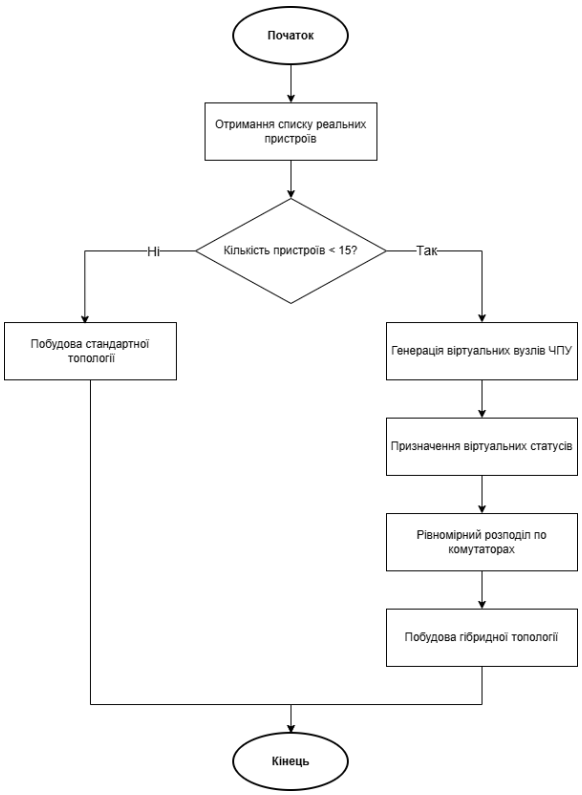


Рисунок 3.13 – Гібридна генерація та розподіл віртуальних промислових вузлів

Для покращення сприйняття система використовує кольорове кодування зв’язків:

– зелений – нормальна робота;
– жовтий – попередження;
– червоний – втрата зв’язку;
– анімований блакитний – магістральні SDN-з’єднання.

Додатково backend-рівень виконує перевірку цілісності графа шляхом контролю відповідності ідентифікаторів вузлів та ребер. Це дозволяє виявляти некоректні зв’язки ще до передачі даних у frontend-застосунок та запобігати помилкам рендерингу.

Таким чином, реалізований алгоритм забезпечує автоматичне формування структурованої, масштабованої та візуально оптимізованої топології SDN-мережі, придатної для інтерактивного моніторингу у режимі реального часу.

3.3 Реалізація клієнтського вебінтерфейсу та візуалізації (Frontend)

3.3.1 Динамічний рендеринг мережевого графа за допомогою React Flow

Frontend-рівень системи відповідає за візуалізацію SDN-інфраструктури, навігацію по топології та взаємодію оператора з мережею. Основою інтерфейсу стала бібліотека React Flow (@xyflow/react), яка дозволяє будувати інтерактивні графи з підтримкою масштабування, оновлення вузлів у реальному часі та кастомних компонентів [14, 45].

На відміну від статичних схем або таблиць, React Flow дозволяє представляти кожен вузол як окремий React-компонент із власною логікою та станом. Це особливо важливо для систем моніторингу, де телеметрія постійно змінюється, а оновлювати потрібно лише окремі елементи, а не всю топологію.

Центральним елементом підсистеми є компонент NetworkCanvas, який відповідає за:

– ініціалізацію React Flow;

– рендеринг вузлів та зв’язків;
– навігацію по графу;
– обробку подій користувача;
– інтеграцію допоміжних UI-компонентів.

На вхід компонента <ReactFlow/> передаються:

– масив вузлів (nodes);
– масив зв’язків (edges);
– конфігурація nodeTypes;
– параметри навігації;
– обробники подій.

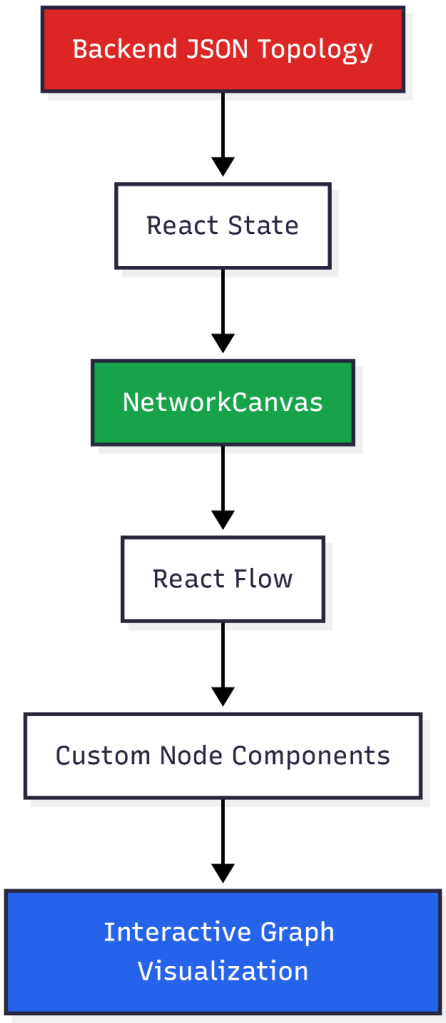


Рисунок 3.14 – Архітектура рендерингу мережевого графа у React Flow

Важливу роль у системі відіграє механізм nodeTypes, який дозволяє прив’язувати типи вузлів до власних React-компонентів. Завдяки цьому кожен тип обладнання має окреме візуальне представлення.

У системі реалізовано компоненти для:

– SDN-контролера;
– серверів;
– комутаторів;
– CNC-пристроїв.

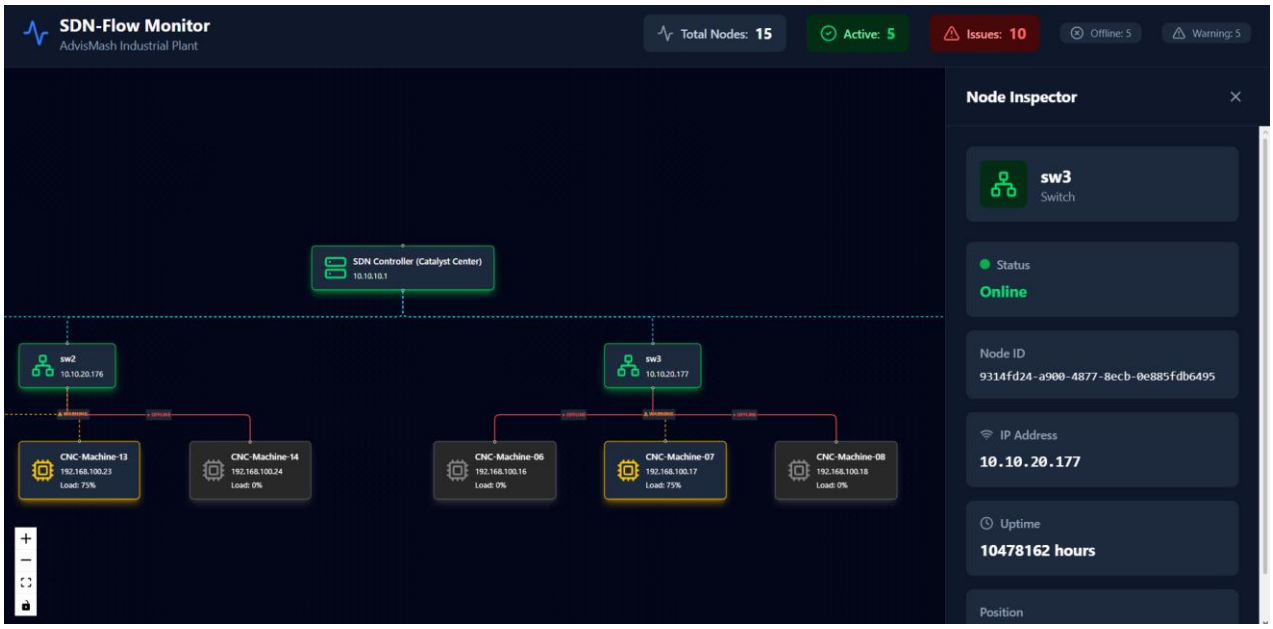


Рисунок 3.15 – Візуалізація мережевої топології у React Flow

Такий підхід спрощує масштабування інтерфейсу та додавання нових типів обладнання.

Для оптимізації продуктивності використовуються React-хуки useMemo та useCallback. Вони дозволяють уникати зайвих ререндерів під час циклічного оновлення телеметрії.

У NetworkCanvas також реалізовано базові механізми взаємодії з топологією:

– вибір вузлів;
– масштабування;
– переміщення полотна;
– автоматичне центрування;
– скидання виділення.

Подія onNodeClick відкриває панель NodeInspector, яка показує детальну інформацію про вибраний вузол:

– IP-адресу;
– статус доступності;
– CPU-навантаження;
– uptime;
– телеметричні параметри;
– службову інформацію.

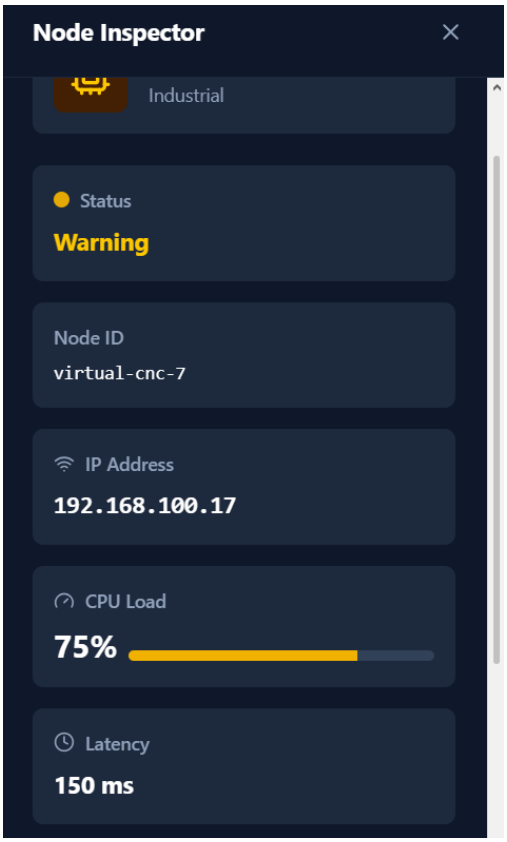


Рисунок 3.16 – Контекстна панель телеметричного аналізу NodeInspector

NodeInspector реалізований як окремий бічний компонент, який динамічно оновлюється залежно від вибраного вузла. Це дозволяє переглядати телеметрію без переходів між сторінками.

Подія onPaneClick використовується для скидання поточного вибору при натисканні на вільну область полотна.

Для автоматичної адаптації графа під розміри екрана використовується параметр fitView. Під час старту система автоматично масштабує топологію так, щоб усі вузли залишались у видимій області.

Також встановлено обмеження мінімального та максимального масштабування, щоб уникнути втрати орієнтації на полотні.

Для покращення навігації використовуються допоміжні компоненти:

- `<Background/>`;
- `<Controls/>`.

Таблиця 3.5 – Основні компоненти підсистеми візуалізації

Компонент	Призначення
React Flow	Рендеринг графової структури
NetworkCanvas	Центральне полотно топології
NodeInspector	Відображення телеметрії вузла
Background	Формування координатної сітки
Controls	Керування масштабом та навігацією
Custom Nodes	Візуалізація типів обладнання

Компонент <Background/> формує координатну сітку для кращої орієнтації у великих топологіях, а <Controls/> надає кнопки масштабування та навігації.

Окрему увагу приділено відображенню станів вузлів. Для цього використовуються:

- кольорові індикатори;
- CSS-анімації;

– hover-ефекти;

– плавні переходи;

– пульсація аварійних вузлів.

Такі механізми дозволяють оператору швидше помічати проблемні сегменти мережі та реагувати на зміни стану інфраструктури.

У frontend-рівні також реалізовано обробку помилок мережевої взаємодії. Якщо backend або SDN-контролер стають недоступними, система припиняє рендеринг графа та відображає компонент NetworkError.

Компонент повідомляє про втрату з’єднання та показує статус автоматичних спроб повторного підключення.

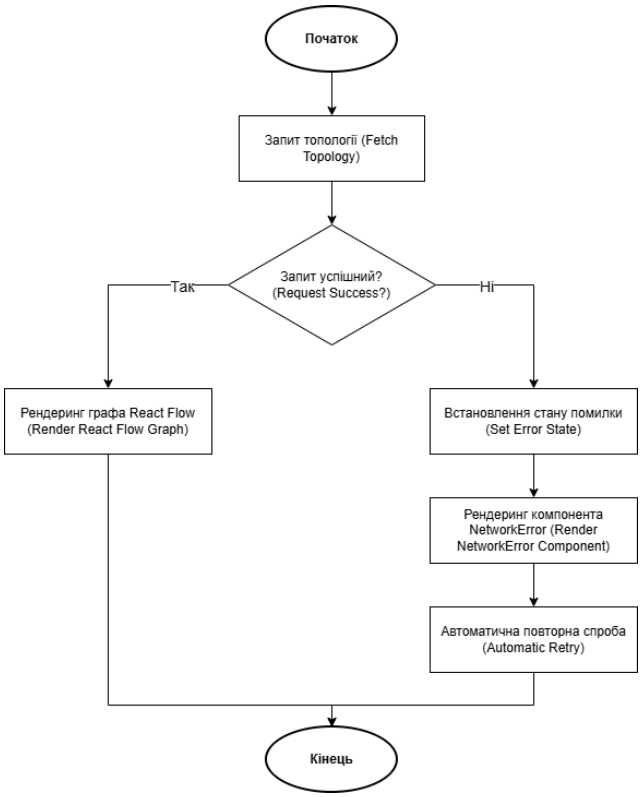


Рисунок 3.17 – Обробка помилок мережевої взаємодії у frontend-рівні

У результаті реалізований frontend забезпечує інтерактивну та масштабовану візуалізацію SDN-інфраструктури з підтримкою оновлення телеметрії у реальному часі та швидкого аналізу стану мережі.

3.3.2 Глобальне управління станом та обробка телеметрії

Для систем моніторингу в реальному часі важливо забезпечити постійне оновлення телеметрії без просадок продуктивності інтерфейсу. У випадку SDN-моніторингу надмірні ререндери можуть погіршувати плавність роботи графа та сповільнювати реакцію інтерфейсу.

У класичному React передача стану часто виконується через props, але при великій кількості компонентів це призводить до проблеми Prop Drilling – передачі даних через багато рівнів вкладеності. У результаті збільшується кількість повторних рендерів та знижується продуктивність застосунку [23].

Для вирішення цієї проблеми у системі використовується менеджер глобального стану Zustand. На відміну від складніших рішень, він має простий API, легко інтегрується з React та створює менше службового навантаження.

У frontend-рівні реалізовано централізоване сховище useNetworkStore, яке містить:

– масив вузлів (nodes);
– масив зв’язків (edges);
– інформацію про вибраний вузол (selectedNode);
– методи оновлення стану.

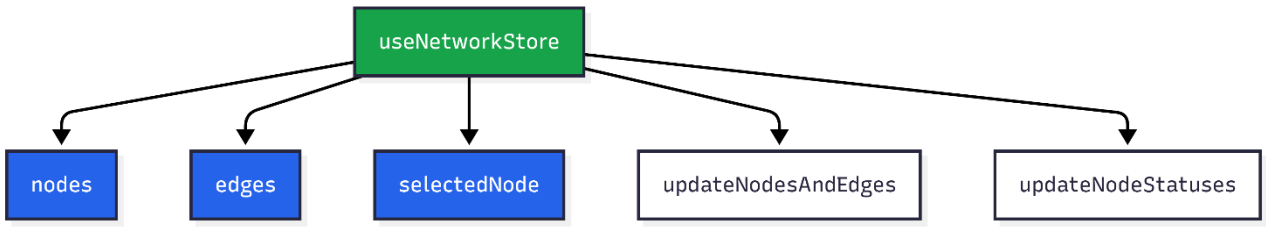


Рисунок 3.18 - Архітектура глобального сховища стану Zustand

Структура сховища побудована так, щоб оновлення торкались лише потрібних компонентів і не викликали зайвих змін інтерфейсу.

У системі реалізовано дві окремі стратегії оновлення стану.

Таблиця 3.6 – Стратегії оновлення стану у frontend-рівні

Метод оновлення	Призначення	Частота використання
updateNodesAndEdges	Повне завантаження структури топології	Початкове завантаження
updateNodeStatuses	Точкове оновлення телеметричних параметрів	Кожні 10 секунд
Global State Sync	Синхронізація frontend/backend	При отриманні topology
Polling Update	Оновлення статусів вузлів	Циклічний моніторинг

Перша стратегія – updateNodesAndEdges. Вона виконує повне оновлення масивів вузлів та зв’язків і використовується лише під час початкового завантаження топології.

Такий підхід дозволяє один раз синхронізувати frontend із backend та сформувати базову структуру графа без постійного перевантаження інтерфейсу.

Друга стратегія – updateNodeStatuses. У цьому режимі frontend отримує лише масив телеметричних статусів (statusNodes), після чого оновлює відповідні вузли за ID.

Оновлюються лише окремі поля:

– status;
– uptime;
– load;
– latency.

При цьому структура графа та координати вузлів не змінюються.

Такий підхід дозволяє:

– зменшити кількість рендерів;

– уникнути перебудови топології;

– забезпечити плавне оновлення індикаторів;

– знизити навантаження на React Flow.

Для оновлення стану використовується immutable-підхід через map() та spread-оператор. Це забезпечує правильну роботу Virtual DOM та коректне визначення змін React [23].

Для додаткової оптимізації застосовуються shallow selectors (useShallow), які запобігають рендерам компонентів, якщо відповідна частина глобального стану не змінювалась.

Логіку отримання телеметрії винесено в окремий React-хук useNetworkData.

Рисунок 3.19 – Життєвий цикл отримання телеметричних даних

Під час ініціалізації компонента через useEffect виконується функція loadTopology, яка звертається до endpoint /api/topology та завантажує повну структуру мережі.

Після первинного завантаження система переходить у режим циклічного моніторингу через setInterval.

Інтервал polling-оновлення становить

$$t = 10 \text{ с} \tag{3.6}$$

де:

t – період циклічного опитування backend-рівня.

Кожні 10 секунд frontend виконує запит до оптимізованого endpoint'у /api/topology/status, після чого отримані телеметричні дані передаються у метод updateNodeStatuses.

Розділення повного завантаження топології та легковагового polling-моніторингу дозволяє:

- зменшити обсяг мережевого трафіку;
- мінімізувати навантаження на backend;
- уникнути повторної побудови графа;
- знизити кількість re-render операцій;
- забезпечити більш плавну роботу React Flow.

Такий підхід дозволяє оновлювати лише телеметричні параметри без повної перебудови структури топології. У результаті інтерфейс працює стабільніше, а зміни стану вузлів відображаються швидше та без помітних затримок для користувача.

Рисунок 3.20 – Оптимізація оновлення телеметрії без повного ререндерингу графа

Для запобігання витокам пам’яті у useNetworkData реалізовано очищення таймера через clearInterval. При демонтуванні компонента polling коректно завершується та не залишає фонових HTTP-запитів.

У результаті реалізована frontend-архітектура забезпечує стабільне управління глобальним станом та ефективне оновлення телеметрії навіть при великій кількості вузлів і високій частоті змін мережевих даних.

3.4 Висновки по третьому розділу

У третьому розділі було реалізовано програмну систему графічного моніторингу SDN-інфраструктури з використанням сучасних вебтехнологій та засобів програмно-керованих мереж. Система забезпечує інтеграцію з Cisco Catalyst Center, автоматичне формування топології мережі, обробку телеметрії та її візуалізацію у режимі реального часу.

У ході реалізації організовано взаємодію з Cisco Catalyst Center Sandbox через REST API, реалізовано механізми автентифікації, кешування токенів та автоматичного відновлення сесії. Використання Sandbox-середовища дозволило працювати з реальною телеметрією без розгортання власної SDN-інфраструктури.

Backend-рівень на базі Node.js та Express.js забезпечує обробку API-запитів, фільтрацію зв'язків, генерацію віртуальних вузлів та підготовку топології до подальшої візуалізації. Також реалізовано алгоритми ієрархічного компонування графа та перевірки цілісності мережевої структури.

Frontend-рівень, створений на основі React, TypeScript та React Flow, забезпечує інтерактивне відображення мережі, підтримку навігації, масштабування та перегляду телеметрії вузлів. Для оптимізації роботи застосовано Zustand, polling-моніторинг та точкове оновлення телеметрії без повного ререндерингу графа.

У результаті було створено модульну та масштабовану систему моніторингу SDN-мереж, придатну для подальшого розвитку та інтеграції з інтелектуальними засобами аналізу мережевої інфраструктури.

ВИСНОВКИ

У роботі було розроблено та реалізовано систему графічного моніторингу і діагностики вузлів SDN-мережі. Система забезпечує централізований збір телеметрії, аналіз стану мережевої інфраструктури, візуалізацію топології та виявлення проблемних сегментів у режимі реального часу.

У процесі роботи досліджено сучасні підходи до побудови SDN-мереж, методи обробки телеметричних даних та засоби створення високореактивних вебзастосунків. Розроблене рішення дозволяє підвищити ефективність адміністрування мережі, скоротити час локалізації несправностей та покращити стабільність роботи інфраструктури підприємства.

У першому розділі проведено аналіз сучасних промислових мереж та проблем традиційних систем моніторингу. Розглянуто особливості SDN-архітектури, централізованого керування та методів візуалізації мережевих топологій. Також виконано порівняння JavaScript-фреймворків і бібліотек візуалізації. За результатами аналізу обрано стек React, TypeScript, Vite та React Flow як найбільш придатний для створення високореактивної системи моніторингу [13, 15, 24].

У другому розділі виконано проєктування системи та сформовано її багаторівневу архітектуру. Розроблено модель обробки телеметрії, структуру топологічних даних та механізми взаємодії між підсистемами. Реалізовано функціональну декомпозицію модулів моніторингу, алгоритми побудови топології, аналізу станів вузлів та візуального алертингу. Окрему увагу приділено UX/UI-рішенням, кібербезпеці, контролю доступу та відмовостійкості системи.

У третьому розділі реалізовано програмну частину системи. Виконано інтеграцію з Cisco Catalyst Center Sandbox через REST API. Розроблено backend на базі Node.js та Express.js, який забезпечує автентифікацію, кешування токенів, обробку телеметрії та підготовку даних для frontend-рівня. Реалізовано алгоритми

синтезу ієрархічної топології, фільтрації зв’язків та генерації віртуальних промислових вузлів.

Також створено клієнтський вебінтерфейс на базі React Flow з підтримкою інтерактивної навігації, масштабування та динамічного оновлення телеметрії. Для оптимізації роботи використано Zustand та механізм polling-моніторингу без повного ререндерингу графа.

У результаті створено масштабовану систему моніторингу SDN-мережі, яка забезпечує стабільне відображення топології, підтримує механізми візуального алертингу та дозволяє оперативно виявляти проблемні сегменти мережевої інфраструктури.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Kreutz D., Ramos F. M. V., Verissimo P. E. et al. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*. 2015. Vol. 103(1). P. 14-76.

2. Nunes B. A. A., Mendonca M., Nguyen X.-N. et al. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Communications Surveys & Tutorials*. 2014. Vol. 16(3). P. 1617-1634.

3. Hu F., Hao Q., Bao K. A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation. *IEEE Communications Surveys & Tutorials*. 2014. Vol. 16(4). P. 2181-2206.

4. Benzekki K., El Fergougui A., Elbelrhiti Elalaoui A. Software-defined networking (SDN): a survey. *Security and Communication Networks*. 2016. Vol. 9(18). P. 5803-5833.

5. Kim H., Feamster N. Improving Network Management with Software Defined Networking. *IEEE Communications Magazine*. 2013. Vol. 51(2). P. 114-119.

6. Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. *ONF White Paper*. 2012. 12 p.

7. Cisco Systems. Cisco Catalyst Center Platform Overview. URL: https://developer.cisco.com/docs/dna-center/ (дата звернення: 22.03.2026).

8. Cisco Systems. Cisco DNA Center API Documentation. URL: https://developer.cisco.com/docs/dna-center/#!cisco-dna-center-platform-overview (дата звернення: 22.03.2026).

9. Medved J., Varga R., Tkacik A. et al. OpenDaylight: Towards a Model-Driven SDN Controller Architecture. *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. Sydney, 2014. P. 1-6.

10. Stallings W. Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud. Boston : Addison-Wesley, 2016. 366 p.

11. *React Documentation*. URL: https://react.dev (дата звернення: 22.03.2026).

12. TypeScript Documentation. URL: https://www.typescriptlang.org/docs (дата звернення: 22.03.2026).

13. Vite Documentation. URL: https://vitejs.dev/guide (дата звернення: 22.03.2026).

14. React Flow Documentation. URL: https://reactflow.dev (дата звернення: 22.03.2026).

15. Zustand Documentation. URL: https://zustand.docs.pmnd.rs (дата звернення: 22.03.2026).

16. Node.js Documentation. URL: https://nodejs.org/en/docs (дата звернення: 22.03.2026).

17. Express.js Documentation. URL: https://expressjs.com (дата звернення: 22.03.2026).

18. Axios Documentation. URL: https://axios-http.com/docs/intro (дата звернення: 22.03.2026).

19. Mozilla Developer Network. REST API Introduction. URL: https://developer.mozilla.org/en-US/docs/Glossary/REST (дата звернення: 22.03.2026).

20. Mozilla Developer Network. HTTPS. URL: https://developer.mozilla.org/en-US/docs/Glossary/HTTPS (дата звернення: 22.03.2026).

21. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : dissertation. University of California, Irvine, 2000. 162 p.

22. Freeman A. Pro React 18. New York : Apress, 2023. 1074 p.

23. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. Sebastopol : O'Reilly Media, 2023. 310 p.

24. Cherny B. Programming TypeScript: Making Your JavaScript Applications Scale. Sebastopol : O'Reilly Media, 2019. 324 p.

25. Tilkov S., Vinoski S. Node.js: Using JavaScript to Build High-Performance Network Programs. *IEEE Internet Computing*. 2010. Vol. 14(6). P. 80-83.

26. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

27. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2015. 816 p.

28. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York : McGraw-Hill Education, 2019. 880 p.

29. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems – Requirements. Geneva : ISO, 2022.

30. Tanenbaum A. S., Wetherall D. J. Computer Networks. 5th ed. Boston : Pearson, 2011. 960 p.

31. Bhattacharyya D., Kim T.-H., Pal S. A Comparative Study of Wireless Sensor Networks and Their Routing Protocols. *Sensors*. 2010. Vol. 10(12). P. 10506-10523.

32. Barabasi A.-L. Network Science. Cambridge : Cambridge University Press, 2016. 474 p.

33. Newman M. Networks: An Introduction. Oxford : Oxford University Press, 2010. 784 p.

34. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 775 p.

35. Richards M., Ford N. Fundamentals of Software Architecture. Sebastopol : O'Reilly Media, 2020. 419 p.

36. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 533 p.

37. Spinellis D. Code Quality: The Open Source Perspective. Boston : Addison-Wesley, 2006. 680 p.

38. Hunt A., Thomas D. The Pragmatic Programmer: Your Journey to Mastery. 20th Anniversary Edition. Boston : Addison-Wesley, 2019. 352 p.

39. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

40. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Boston : Prentice Hall, 2008. 464 p.

41. Grinberg M. Flask Web Development. Sebastopol : O’Reilly Media, 2018. 316 p.

42. Kurose J. F., Ross K. W. Computer Networking: A Top-Down Approach. 8th ed. Boston : Pearson, 2021. 864 p.

43. Richardson L., Ruby S. RESTful Web Services. Sebastopol : O’Reilly Media, 2007. 448 p.

44. OpenJS Foundation. Express.js Guide. URL: https://expressjs.com/en/guide/routing.html (дата звернення: 22.03.2026).

45. React Flow Examples. URL: https://reactflow.dev/examples (дата звернення: 22.03.2026).

46. Cisco Systems. Cisco DevNet Sandbox Overview. URL: https://developer.cisco.com/site/sandbox/ (дата звернення: 22.03.2026).

47. OWASP Foundation. OWASP Top Ten Web Application Security Risks. URL: https://owasp.org/www-project-top-ten/ (дата звернення: 22.03.2026).

48. Google Developers. Web Performance Optimization. URL: https://web.dev/performance/ (дата звернення: 22.03.2026).

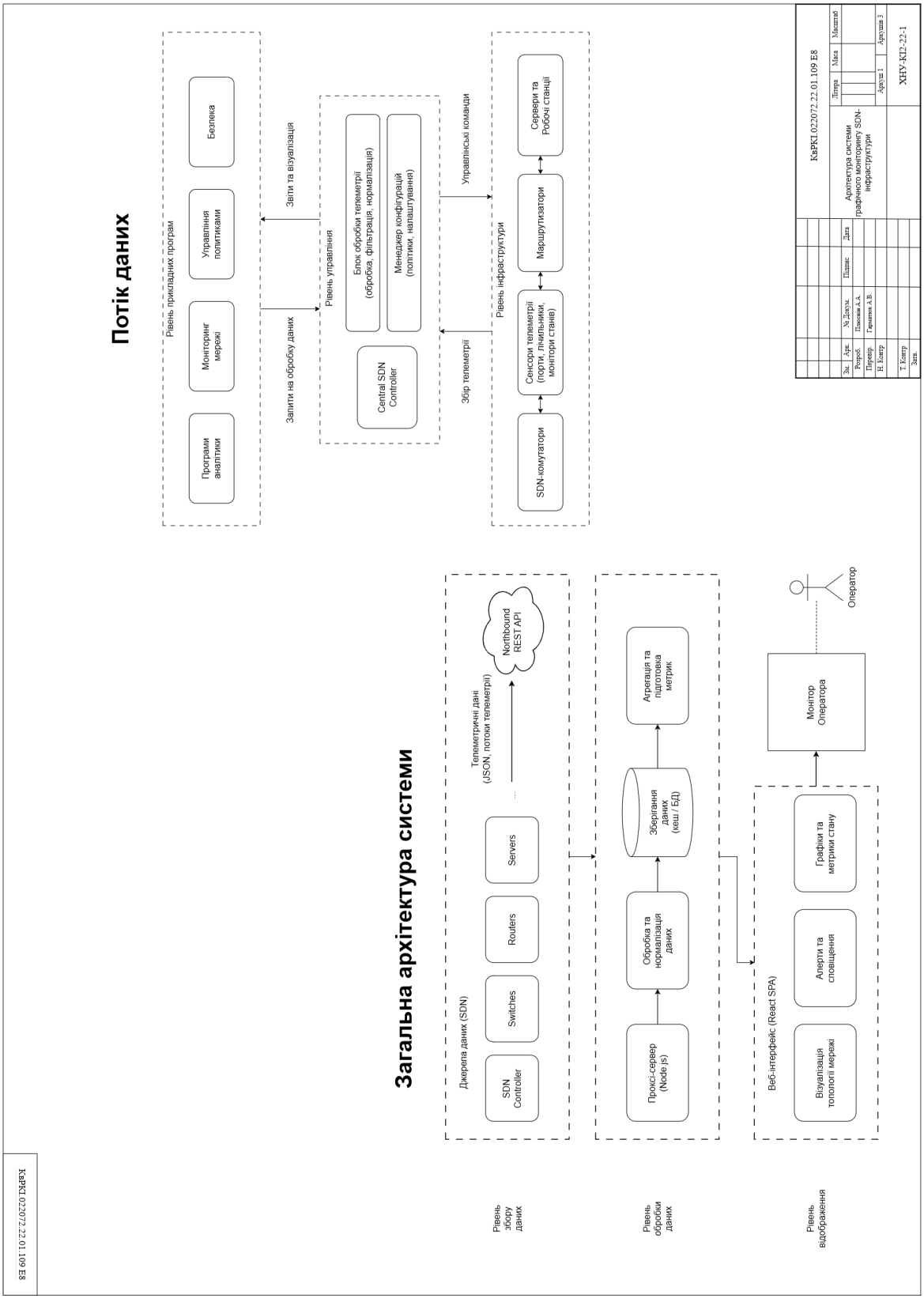
49. W3C. Scalable Vector Graphics (SVG) 2 Specification. URL: https://www.w3.org/TR/SVG2/ (дата звернення: 22.03.2026).

50. Mozilla Developer Network. Fetch API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (дата звернення: 22.03.2026).

ДОДАТОК А

(обов’язковий)

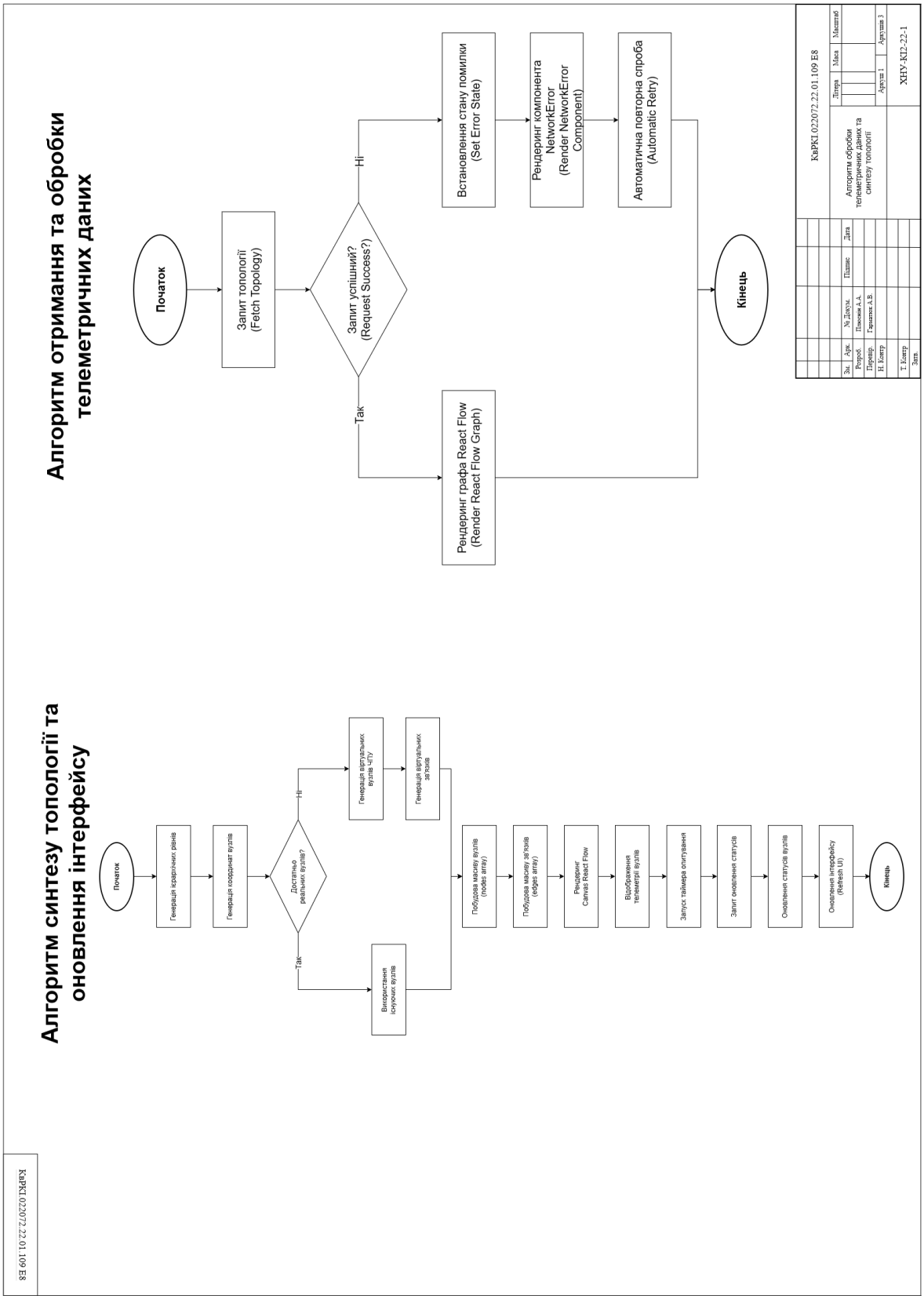
Копія креслення «Архітектура системи графічного моніторингу»



ДОДАТОК Б

(обов’язковий)

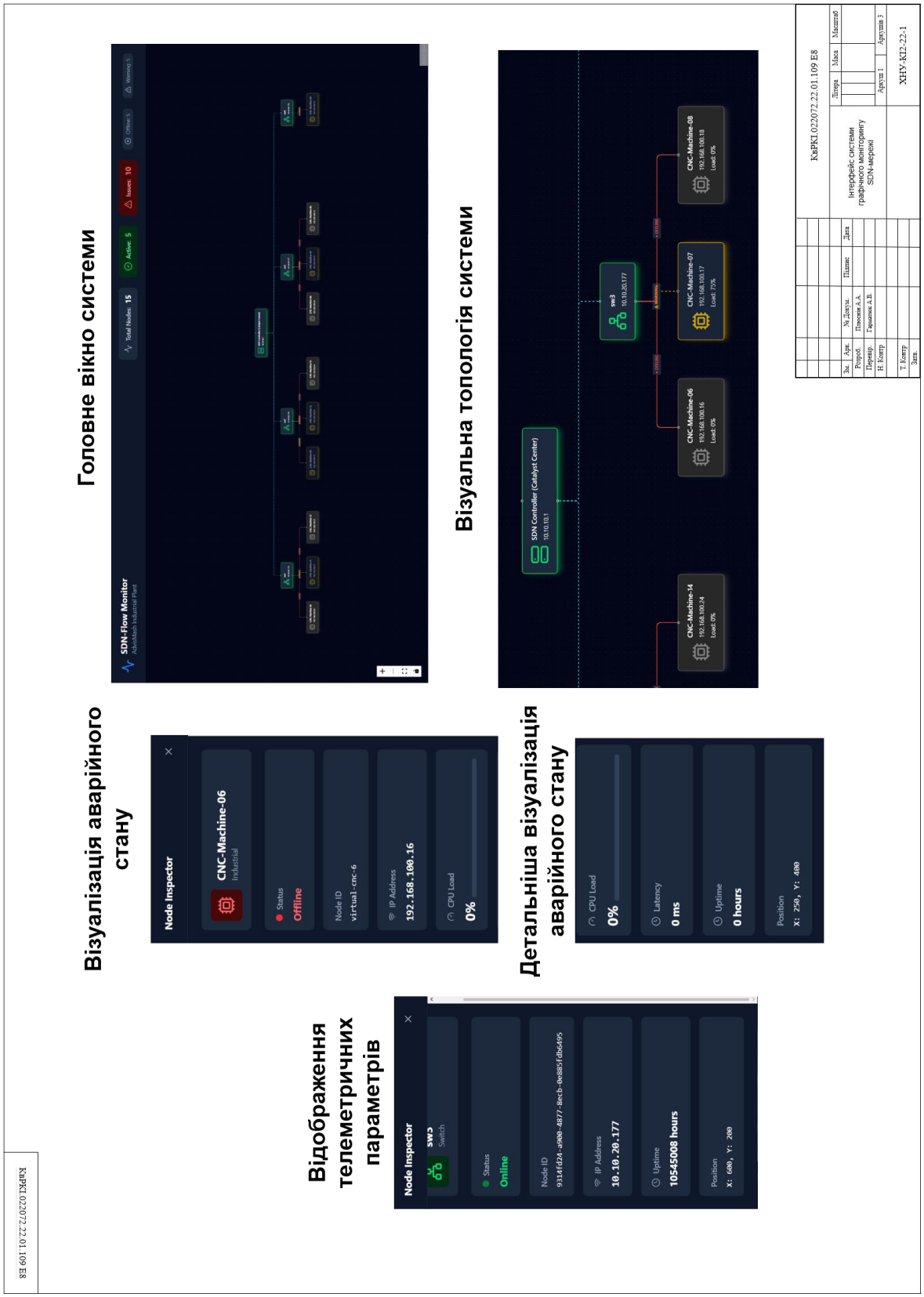
Копія креслення «Алгоритм обробки телеметричних даних та синтезу топології»



ДОДАТОК В

(обов’язковий)

Копія креслення «Інтерфейс системи графічного моніторингу SDN-мережі»



Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Артем ПЛЮСНІН

Співавтор:

Назва: Система діагностики та графічного аналізу стану вузлів розподіленої мережі на базі SDN-технологій

Експерт: Андрій ГАРМАТЮК

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:5.26%

Коефіцієнт подібності 2:0.68%

Мікропробіли: 3

Заміна букв: 3

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-06-03 01:36:20.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-06-03 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 14%**

	Документ		Сумарний збіг по Базі Даних	
ID: 273157 Назва: БКР Система діагностики та графічного аналізу стану вузлів розподіленої мережі на базі SDN-технологій Додано в БД: 2026-06-02 Автора: Артем ПЛЮСНІН Керівники: Андрій ГАРМАТЮК Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	67662	646	2635 (4%)	41 (6%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Плюснін Артем Андрійович

Тема: Система діагностики та графічного аналізу стану вузлів розподіленої мережі на базі SDN-технологій

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3___ Кількість сторінок записки ___66___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є проєктування, програмна реалізація та тестування системи для збору, обробки та графічної візуалізації телеметричних даних розподіленої SDN-мережі підприємства у режимі реального часу.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області: проаналізовано сучасні проблеми моніторингу промислових мереж (концепція «Індустрія 4.0»), досліджено архітектуру програмно-конфігурованих мереж (SDN), методи отримання телеметрії та обґрунтовано вибір сучасного стека вебтехнологій (Node.js, React, React Flow) для вирішення поставленої задачі.В другому розділі розроблено багаторівневу клієнт-серверну архітектуру системи; спроєктовано інформаційні моделі мережевих вузлів та топологічних зв'язків; розроблено алгоритм синтезу ієрархічної топології графа та фільтрації з'єднань; описано логіку гібридної генерації віртуальних промислових об'єктів для моделювання навантаження.В третьому розділі виконано практичну програмну реалізацію системи: створено серверний прошарок (Middleware) для безпечної інтеграції з хмарним SDN-контролером Cisco Catalyst Center; реалізовано алгоритми кешування токенів; розроблено високореактивний клієнтський

інтерфейс із застосуванням менеджера стану Zustand для оптимізації рендерингу телеметрії; проведено валідацію роботи алгоритмів та оцінку продуктивності системи.

4. Позитивні сторони роботи: Використання передового стеку технологій розробки (React, TypeScript, Express.js), успішна інтеграція з реальною хмарною інфраструктурою корпоративного рівня (Cisco Sandbox), глибоке розуміння архітектурних патернів оптимізації та висока практична цінність розробленої системи моніторингу.

5. Негативні сторони роботи: Недостатньо уваги приділено порівняльному аналізу продуктивності розробленого рішення з існуючими комерційними системами моніторингу, а також наявні незначні стилістичні неточності в оформленні тексту.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні, здобувач продемонстрував ґрунтовні інженерні знання, а сама робота заслуговує на позитивну оцінку.

8. Інші зауваження: ______________________________

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) ____________
Бедзамон Леонід Петрович, зав-каф ІПЗ, ХНУ

“01” червня 2026 р.

(підпис)

Зав. кафедри КІС
д-р. філософії Ользі ПАВЛОВІЙ

Артем ПЛЮСНІН
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-22-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Система діагностики та графічного аналізу стану вузлів розподіленої мережі на базі SDN-технологій
Автор Артем ПЛЮСНІН
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: Андрій ГАРМАТЮК

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 5,26%; та системою Anti-Plagiarism складає 1,0%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри _____ Підпис Ольга ПАВЛОВА Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми _____ Підпис Андрій НІЧЕПОРУК Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Підпис Андрій ГАРМАТЮК Ім'я, ПРІЗВИЩЕ