ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ ТА ЕЛЕКТРОННА ІНЖЕНЕРІЯ
INFORMATION AND COMMUNICATION TECHNOLOGIES, ELECTRONIC ENGINEERING

№ 5 (2), 2025

https://doi.org/10.23939/ictee2025.02.

ЗАСТОСУВАННЯ ЗАСОБІВ МЕРЕЖЕВОГО ПРОГРАМУВАННЯ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ УПРАВЛІННЯ ІНФОКОМУНІКАЦІЙНИМИ МЕРЕЖАМИ

О. Єременко [ORCID: 0000-0003-3721-8188], **Р. Мамон** [ORCID: 0009-0004-0538-0900], **Д. Андрушко** [ORCID: 0009-0004-0749-9729], **Л. Мельнікова** [ORCID: 0000-0003-0439-7108]

Харківський національний університет радіоелектроніки, пр. Науки, 14, Харків, 61166, Україна

Відповідальна за рукопис: О. Єременко (e-mail: oleksandra.yeremenko@nure.ua).

(Подано 22 серпня 2025)

Досліджено процеси автоматизації управління інфокомунікаційними мережами з використанням засобів мережевого програмування. У роботі обґрунтовано актуальність автоматизації як ключового напряму розвитку сучасних інфокомунікаційних систем, що зумовлено зростанням масштабів трафіку та складністю архітектури мереж. Показано, що автоматизація дає змогу підвищити ефективність експлуатації, зменшити тривалість виконання завдань і мінімізувати вплив людського чинника. Аналізування відомих підходів засвідчило, що автоматизація охоплює увесь життєвий цикл мережі: від початкового налаштування та введення в експлуатацію до подальшої підтримки й оптимізації. Залежно від сфери застосування вона реалізується як у корпоративних інфраструктурах, так і в середовищі постачальників послуг. Водночас важливого значення набуває вибір оптимальних інструментів, зокрема інтерфейсу командного рядка, який дає змогу створювати сценарії для повторюваних завдань, та графічних вебінтерфейсів, що забезпечують зручність і наочність. Окрему увагу приділено мові Perl, що демонструє широкі можливості для інтеграції з мережевими пристроями та створення сценаріїв управління. Для клієнтської частини вебзастосунків доцільно застосовувати AJAX та JavaScript/DOM, що забезпечують інтерактивність і динамічне оновлення даних. Серверна частина може реалізовуватися за допомогою Node.js, яке характеризується масштабованістю та подієво-орієнтованою архітектурою. У статті наведено прикладне рішення у вигляді розробленого програмного забезпечення для управління та моніторингу IP/MPLS-маршрутизаторів. Його архітектура містить ядро, клієнтську й серверну частини, що забезпечує інтегровану платформу для автоматизації експлуатаційних завдань. Практичні випробування підтвердили зручність використання застосунку, зменшення тривалості виконання операцій і кількості помилок. Перспективи подальших досліджень пов’язані з інтеграцією новітніх протоколів, уніфікацією інтерфейсів і створенням масштабованих систем управління мережами.

Ключові слова: *інфокомунікаційні мережі, автоматизація, мережеве програмування, управління, моніторинг.*

УДК: 621.391

Вступ

Інфокомунікаційні системи та мережі становлять основу сучасної цифрової інфраструктури, забезпечуючи безперервність комунікаційних процесів та обмін даними у різних сферах діяльності [1, 2]. З огляду на зростання обсягів інформаційного трафіку та ускладнення архітектури мереж,

зростає потреба у впровадженні нових підходів до управління, які забезпечують високу швидкість реакції на змінні умови функціонування, а також гарантують надійність і гнучкість сервісів. У цьому контексті ключового значення набувають засоби автоматизації, що дають змогу мінімізувати вплив людського чинника, підвищити ефективність експлуатації та зменшити тривалість виконання різноманітних завдань [3–6].

Автоматизація управління інфокомунікаційними мережами (ІКМ) полягає у використанні спеціалізованого програмного забезпечення для виконання широкого спектра завдань від початкового налаштування до моніторингу та оптимізації процесів функціонування [4–8]. Водночас забезпечення оперативного, гнучкого та узгодженого управління нині є визначальною вимогою як для класичних мереж, так і для архітектур програмно-конфігурованих мереж [1, 2]. Отже, сучасні підходи передбачають використання інтерфейсів програмного керування (API), які замінюють ручне введення CLI-команд і дають змогу автоматизувати, наприклад, конфігурацію мережевих пристроїв за допомогою програмних скриптів мовами Python, Java, Perl тощо.

Процес автоматизації охоплює увесь життєвий цикл роботи мережі: від початкового розгортання обладнання (день 0), через конфігурацію та введення в експлуатацію (день 1), до подальшої підтримки, модернізації та оптимізації (день 2) [3–5]. На кожному з етапів автоматизація спрямована на зниження витрат часу і ресурсів, а також на підвищення надійності функціонування мережевих елементів.

Залежно від сфери застосування, автоматизація може реалізовуватися як у корпоративних інфраструктурах (мережі центрів обробки даних, хмарні сервіси, кампусні та локальні мережі), так і у середовищі постачальників послуг (широкосмуговий доступ, VPN-рішення, віртуалізовані мережеві функції) [5]. У будь-якому випадку метою є забезпечення керованості складних гетерогенних середовищ і підвищення ефективності взаємодії між фізичними й віртуальними компонентами мереж. Тому автоматизація управління ІКМ розглядається як ключовий напрям розвитку сучасних систем зв’язку, в якому поєднуються можливості мережевого програмування та технологій віртуалізації.

Отже, метою статті є дослідження процесів автоматизації управління інфокомунікаційними мережами із застосуванням засобів мережевого програмування, а також демонстрація прикладного рішення у вигляді розробленого застосунку для управління обладнанням і моніторингу IP/MPLS-маршрутизаторів із використанням графічного вебінтерфейсу.

2. Переваги та основні типи автоматизації інфокомунікаційних мереж

Відомо, що зміни у мережі, які виконуються вручну, нерідко призводять до неузгодженості конфігурацій. Це спричинено різними стилями налаштувань, які застосовують інженери, недостатньою документацією або людськими помилками [5, 7]. Така неузгодженість спричиняє зниження продуктивності мережі, ускладнення її обслуговування та зростання операційних витрат. За таких умов автоматизація дає змогу мінімізувати зазначені недоліки ручного адміністративного втручання, забезпечуючи низку ключових переваг [3–5, 7] (рис. 1).

Автоматизація підвищує оперативність управління ІКМ, даючи змогу значно швидше реагувати на нові вимоги. Крім того, завдяки уникненню ручного втручання підвищується надійність: конфігурації виконуються послідовно, що робить систему менш уразливою до людських помилок і забезпечує вищий рівень доступності. Водночас автоматизація сприяє зниженню експлуатаційних витрат, адже усуває потребу у виконанні рутинних завдань, зменшує тривалість відновлення після збоїв і час простою обладнання.

Важливою перевагою є також стандартизація конфігурацій. Використання шаблонів і чітко визначеної логіки налаштувань уніфікує роботу із мережею, полегшує завдання адміністраторів та зменшує кількість можливих похибок. Внесення змін стає швидшим і контрольованим, оскільки він реалізується за допомогою автоматизованих інструментів, а не ручного введення команд. Це, зі свого боку, дає змогу зменшитти час на розгортання нових сервісів і підвищує загальну гнучкість мережі.

Автоматизація спрощує роботу мережевих адміністраторів, які можуть здійснювати дистанційне керування через централізовані контролери. Вони надають доступ до єдиної панелі управління з упорядкованим інтерфейсом, що дає змогує ефективно контролювати мережу навіть у разі її географічної розподіленості.

Рис. 1. Ключові переваги автоматизації мереж

Вищий рівень автоматизації забезпечує додаткові можливості, зокрема постійний моніторинг та аналітику, що дає змогу своєчасно виявляти відхилення у функціонуванні мережі. Так, наприклад, динамічна оптимізація на основі алгоритмів машинного навчання забезпечує підтримку бажаного стану мережі та прийняття автоматизованих рішень щодо коригування параметрів. Завдяки цьому істотно прискорюється усунення несправностей, зменшується тривалість їх локалізації та відновлення працездатності.

Окремої уваги потребує підвищення рівня безпеки. Адже автоматизовані системи зменшують ризик людських помилок під час конфігурацій, а також дають змогу оперативно реагувати на загрози. Використання спеціалізованих алгоритмів забезпечує захист від потенційних атак та підвищує стійкість інфраструктури.

У підсумку автоматизація ІКМ забезпечує комплексний позитивний ефект, який охоплює підвищення надійності, гнучкості та продуктивності, зниження витрат, а також створює підґрунтя для упровадження інтелектуальних технологій управління життєвим циклом мереж.

Зазвичай виокремлюють різні основні типи мережевої автоматизації, які доцільно враховувати під час практичної реалізації [3–5, 8–11]. Перший тип пов’язаний із використанням спеціалізованого програмного забезпечення для автоматизації мереж, відомого також як інтелектуальна мережева автоматизація. Такі програмні рішення пропонують готові шаблони конфігурацій і процедур, що зменшує потребу в ручному написанні сценаріїв. На відміну від скриптової автоматизації, програмні засоби здатні охоплювати ширший спектр завдань, зокрема виявлення і усунення несправностей, оптимізацію роботи обладнання та централізоване управління.

Другим типом є автоматизація, що ґрунтується на сценаріях. Вона передбачає використання набору інструкцій для виконання повторюваних дій у мережі. Цей підхід вважається одним із найпоширеніших завдяки відносній простоті та доступності. Історично для таких завдань застосовували мови Tcl і Perl, проте сьогодні все більшого поширення набувають універсальні мови програмування із відкритим вихідним кодом, зокрема Python, Ruby, Ansible, Bash та Go. Їхня популярність пояснюється не лише гнучкістю та широкими можливостями, а й наявністю численних бібліотек та підтримки спільноти, що спрощує автоматизацію мережевих операцій.

У практичній реалізації автоматизації управління мережами важливу роль відіграють інтерфейси взаємодії із обладнанням. Найпоширеніші підходи – використання командного рядка (CLI) та графічного інтерфейсу користувача (GUI). Кожен із цих підходів має і переваги, й обмеження, які наведено в табл. 1 [8].

Таблиця 2

Переваги та недоліки використання GUI та CLI для мережевої автоматизації [8]

Критерій	GUI	CLI
Швидкість	Виконання операцій порівняно повільне, оскільки залежить від графічного відображення та взаємодії	Команди виконуються швидко, що забезпечує високу оперативність під час роботи з мережею
Крива навчання	Інтерфейс інтуїтивно зрозумілий, користувачі можуть опанувати його навіть самостійно	Засвоєння потребує знання великої кількості команд, що ускладнює навчання нових користувачів
Зручність для користувача	Високий рівень зручності завдяки використанню звичних іконок та візуальних елементів	Менш зручний для недосвідчених користувачів, оскільки робота здійснюється тільки через команди
Компоненти	Реалізується через набір графічних елементів, що забезпечують наочність та інтуїтивність користування	Інтерфейс текстовий і передбачає роботу з командним рядком
Основні користувачі	Призначений для широкого кола користувачів: від кінцевих до системних адміністраторів	Переважно використовують досвідчені користувачі, зокрема програмісти та адміністратори систем

Найсучаснішим напрямом є автоматизація на основі намірів (Intent-Based Networking, IBN). Її сутність полягає у використанні моделей, які відображають бізнес-цілі та очікування користувачів, а також у застосуванні штучного інтелекту й методів машинного навчання для динамічного управління політиками. У цьому випадку адміністратору достатньо визначити бажаний рівень обслуговування програм або користувачів, після чого система самостійно адаптує конфігурацію мережі з урахуванням заданих цілей. Такий підхід дає змогу гнучко реагувати на відхилення, автоматично перерозподіляючи ресурси й відновлюючи необхідний рівень сервісу для критично важливих додатків.

Усі типи автоматизації можна застосовувати до різних мереж, якими керують оператори або через API. Це стосується глобальних (WAN), локальних (LAN), хмарних, безпроводових мереж, мереж центрів обробки даних [3–5] тощо. Отже, мережева автоматизація – універсальний інструмент, здатний охоплювати сучасні інфокомунікаційні середовища та забезпечувати їх ефективне функціонування.

3. Архітектура програмного забезпечення та використані технологічні рішення

У межах роботи розроблено та протестовано програмне забезпечення (ПЗ), яке надає оператору мережі можливість виконання виробничих завдань на обладнанні Nokia 7250 Interconnect Router. Це обладнання належить до класу IP/MPLS-маршрутизаторів і підтримує управління як за допомогою командного рядка (CLI), так і через графічний вебінтерфейс (GUI). На рис. 2 наведено архітектурну схему функціонування програмного забезпечення та застосованих технологічних рішень.

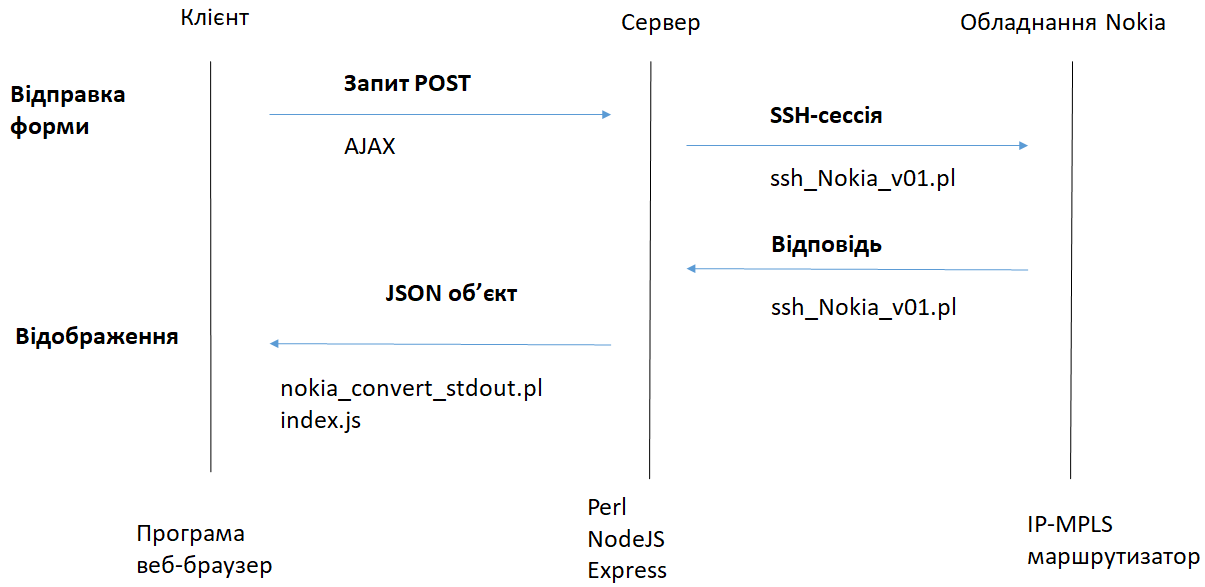


Рис. 2. Архітектурна схема функціонування розробленого програмного забезпечення

Архітектура розробленого програмного забезпечення побудована за клієнт-серверною моделлю та передбачає взаємодію між веббраузером користувача, серверною частиною і мережним обладнанням. Користувач формує запит через графічний вебінтерфейс, після чого дані надсилаються на сервер методом HTTP POST із використанням технології AJAX. Серверна частина обробляє отриману інформацію за допомогою скриптів Node.js та Perl, які встановлюють SSH-сесію із маршрутизатором Nokia та виконують відповідні CLI-команди. Відповідь пристрою передається у форматі JSON, що забезпечує зручне відображення результатів у браузері користувача.

3.1. Характеристика Perl у застосуванні до мережевої автоматизації

Мова програмування Perl є універсальним інструментом із широкими можливостями для реалізації завдань системного адміністрування, опрацювання текстів, управління даними та мережевої автоматизації. Вона підтримує процедурне програмування із використанням змінних, виразів, блоків коду та підпрограм, що робить її зручною для написання як невеликих скриптів, так і масштабних застосунків [12, 13].

Істотною перевагою Perl є наявність великої кількості вбудованих функцій, що дають змогу ефективно працювати з текстовими даними та взаємодіяти з операційною системою на низькому рівні. Крім того, мова забезпечує простий механізм роботи з асоціативними масивами, що робить її придатною для вирішення завдань управління даними та організації опрацювання великих інформаційних потоків. Завдяки виразності синтаксису навіть складні програми можна реалізувати компактним кодом, що підвищує продуктивність розроблення та полегшує подальший супровід.

Нині основною та найпоширенішою є версія Perl 5, яку активно використовують у створенні CGI-скриптів, вебпрограмуванні та автоматизації адміністрування в середовищі Linux/Unix. У контексті мережевої автоматизації Perl застосовують для розроблення скриптів, що забезпечують взаємодію із інтерфейсами командного рядка мережевого обладнання, виконання рутинних конфігураційних завдань, генерування звітів, а також для інтеграції з іншими системами управління. Серед типових сфер застосування можна виділити автоматизацію завдань адміністрування серверів і мереж, вебпрограмування, керування базами даних, розроблення GUI, а також використання у біоінформатиці (рис. 3).

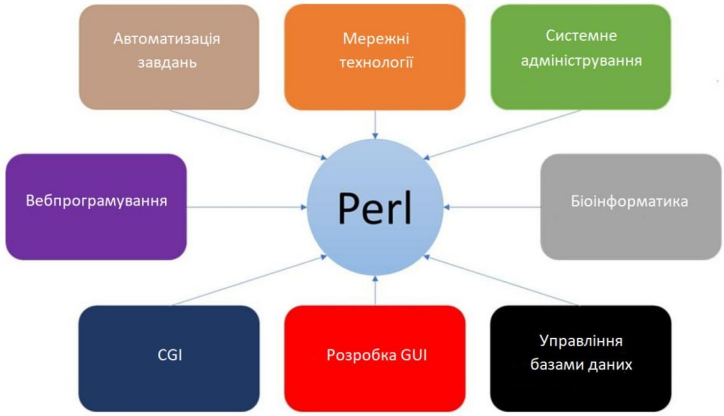


Рис. 3. Сфери застосування мови Perl

Виділимо також мову Raku (раніше відому як Perl 6), яка офіційно відокремлена від Perl 5 і розвивається як самостійна. Попри відмінності, між цими мовами зберігається певний рівень інтеграції: код, написаний на Raku, може взаємодіяти із програмами на Perl 5 за допомогою спеціальних модулів. Проте саме Perl 5 залишається основною версією, яку використовують для мережевої автоматизації та системного адміністрування.

Отже, основними перевагами Perl є універсальність, широкий набір вбудованих функцій, підтримка різних парадигм програмування, а також застосовність у багатьох сферах – від вебпрограмування до управління інфокомунікаційними мережами. Завдяки цим властивостям Perl залишається поширеним інструментом для розробників та адміністраторів, які реалізують завдання автоматизації в сучасних інфраструктурах [12, 13].

3.2. Технології клієнтської частини застосунку

Особливості використання AJAX

Під час розроблення вебзастосунку була реалізована клієнт-серверна архітектура взаємодії. Для відображення результатів використано технологію AJAX та платформу Node.js, що ґрунтується на мові програмування JavaScript.

AJAX (Asynchronous JavaScript and XML) застосовують у вебпрограмуванні для створення динамічних і швидкодіючих вебсторінок [14]. Її головна особливість – можливість асинхронного обміну невеликими обсягами даних між клієнтом і сервером у фоновому режимі, без необхідності повного перезавантаження сторінки. Такий підхід забезпечує плавну інтерактивну взаємодію користувача із вебзастосунком і зменшує обсяг трафіку, що передається між сторонами.

Функціонування AJAX ґрунтується на використанні об’єкта XMLHttpRequest у JavaScript, який дає змогу асинхронно надсилати та отримувати дані від сервера. Принцип роботи полягає в тому, що після ініціації певної події у веббраузері створюється відповідний запит, що надсилається на сервер. Отримана відповідь обробляється клієнтським кодом і використовується для оновлення лише тих елементів сторінки, які потребують змін, без втручання у відображення всієї вебсторінки.

Завдяки такому механізму AJAX підвищує швидкість відгуку системи та забезпечує більшу зручність для користувачів. Це робить технологію одним із базових інструментів для розроблення сучасних інтерактивних вебзастосунків, орієнтованих на ефективне використання мережевих ресурсів і високий рівень інтерфейсної динаміки.

Формати надсилання та отримання структурованих даних

Технологія AJAX підтримує використання кількох форматів даних для організації обміну інформацією між клієнтом і вебсервером [14]. Серед них можна виділити:

- прості текстові повідомлення (Plain Text), що можуть містити окремі слова або речення;

· мову розмітки XML, яка визначає правила для кодування документів у зручному для людини та машини вигляді;

· полегшений формат обміну даними JSON, орієнтований на простоту читання і запису як для людини, так і для комп'ютера;

· стандартну мову розмітки HTML, використовувану для відображення інформації у браузері.

Використання різних форматів забезпечує гнучкість у структуризації та передаванні даних, даючи розробникам змогу вибирати найвідповідніший підхід залежно від конкретних завдань.

Серед перелічених варіантів особливе значення має формат JSON, оскільки саме його заплановано практично застосовувати у межах цього проєкту (рис. 2). JSON – формат обміну даними, який зберігає простоту для людини та зручність для машинної обробки [15]. Формат JSON походить із синтаксису мови JavaScript, проте повністю незалежний від неї, оскільки засоби його оброблення реалізовані для більшості сучасних мов програмування.

Застосування JSON у сфері мережевої автоматизації пояснюється низкою переваг. Передусім цей формат має зрозумілу структуру, що забезпечує легке читання і сприйняття даних. Крім того, він є менш багатослівним порівняно з XML, що дає змогу зменшити тривалість аналізу та генерації даних, знижуючи навантаження на обчислювальні ресурси. Ще однією перевагою є універсальність: JSON підтримується більшістю мов програмування, що робить його придатним для багатьох інструментів і скриптів у сфері автоматизації мереж. Також він дає змогу відображати складні структури даних, зокрема вкладені масиви та об'єкти. JSON – один із найпоширеніших форматів обміну даними в мережевій індустрії, оскільки його підтримують більшість API та сучасні платформи для управління конфігурацією.

Завдяки цим властивостям JSON набув поширення для конфігурації мережевих пристроїв, обміну даними між різними системами та автоматизації завдань адміністрування. У сценаріях автоматизації за допомогою JSON-об'єкта можна проаналізувати параметри конфігурації, після чого використати їх для налаштування інтерфейсу пристрою через API чи інструменти управління, зокрема Ansible. Простота структури JSON робить його зручним для розширення та модифікації, оскільки до нього можна легко додати нові параметри чи описи інтерфейсів відповідно до потреб.

Використання JavaScript / DOM для відображення інформації та взаємодії з нею

JavaScript – мова програмування, яка забезпечує створення інтерактивних вебсторінок і сьогодні є невід'ємною складовою вебзастосунків. Її використання дає змогу реалізовувати динамічну взаємодію із користувачем у режимі реального часу без необхідності повного перезавантаження сторінки. Завдяки цьому забезпечується вищий рівень інтерактивності та зручності роботи з вебресурсами.

Document Object Model (DOM) – програмний інтерфейс для вебдокументів, який подає сторінку у вигляді ієрархічного дерева об'єктів [16]. Така структура дає змогу програмам змінювати вміст, стиль і структуру документа. У цьому контексті JavaScript використовується для доступу до об'єктів DOM і керування ними, що уможливлює динамічне оновлення HTML- та CSS-елементів. Це дає змогу модифікувати візуальне відображення та інформаційне наповнення сторінки без втрати цілісності документа.

Отже, поєднання JavaScript і DOM забезпечує механізми для реалізації динамічного відображення та інтерактивної взаємодії із вебсторінками. Це охоплює оновлення вмісту, створення візуальних ефектів, перевірку форм, а також опрацювання подій користувача, зокрема натискання клавіш або кліків.

3.3. Технології серверної частини застосунку

Node.js – ефективне середовище виконання JavaScript, побудоване на високопродуктивному рушії V8, розробленому для браузера Chrome. Завдяки неблокувальній подієво-орієнтованій моделі Node.js ефективно обробляє велику кількість одночасних з'єднань та забезпечує масштабованість

застосунків, особливо у сценаріях, що передбачають інтенсивний обмін даними. Такі характеристики роблять Node.js перспективним інструментом у веброзробленні, зокрема у сфері мережевої автоматизації.

У контексті автоматизації мереж Node.js використовують для побудови вебінтерфейсів і прикладних інструментів, що забезпечують виконання типових завдань адміністрування. Його модель асинхронного введення – виведення дає змогу ефективно працювати з базами даних, API-за-питами та файловими системами, що є важливими складовими сучасних систем управління мережею.

Вагома перевага – розвинена екосистема модулів npm, у якій наявні бібліотеки, орієнтовані на автоматизацію мережевих процесів. Зокрема, Node-RED є відомою платформою з графічним середовищем розроблення, яка підтримує інтеграцію з API, пристроями IoT та мережевим обладнанням [17]. Існують також бібліотеки на зразок node-netconf, що реалізують клієнтські можливості протоколу NETCONF, хоча деякі з них перебувають у статусі архівованих і не мають активної підтримки.

Отже, використання Node.js у сфері мережевої автоматизації сприяє створенню надійних і продуктивних вебзастосунків, які спрощують управління мережею, зменшують кількість ручних операцій та підвищують ефективність експлуатації інфокомунікаційної інфраструктури.

4. Програмна реалізація завдань управління та моніторингу в інфокомунікаційній мережі

4.1. Ядро програмного забезпечення

У цьому розділі продемонстровано роботу програмного забезпечення, розробленого для виконання завдань управління та моніторингу в інфокомунікаційній мережі. Подані скріншоти та приклади відображають основний функціонал системи, що забезпечує взаємодію із мережевим обладнанням та виконання основних операцій. Опис обмежено ключовими можливостями, необхідними для розуміння принципів функціонування ПЗ.

Отже, скрипт *ssh_Nokia_v01.pl* (рис. 4) приймає як параметри командного рядка IP-адресу маршрутизатора, ім’я користувача, номер опитуваного порту та здійснює їх перевірку в циклі `for` для масиву ARGV.

```
#!/bin/perl -w

for (my $i=$[; $i<=$#ARGV; $i++) {
        if ( $ARGV[$i] eq '-u' ) {
                if ( $ARGV[$i+1] =~ /^\S+$/ ) {
                        $user = $ARGV[$i+1];
                }
                else {
                        print "Error: User can contain non-whitespace character only, got - $ARGV[$i+1]\nExit\n";
                        exit 1;
                }
        }
        elsif ( $ARGV[$i] eq '-ip' ) {
                if ( $ARGV[$i+1] =~ /^(?:[0-9]{1,3}\.){3}[0-9]{1,3}$/ ) {
                        $ip = $ARGV[$i+1];
                }
                else {
                        print "Error: $ARGV[$i+1] doesn't look like IPv4\nExit\n";
                        exit 1;
                }
        }
        elsif ( $ARGV[$i] eq '-P' ) {
                if ( $ARGV[$i+1] =~ /^\d+\/\d+\/\d+$/ ) {
                        $port = $ARGV[$i+1];
                }
                else {
                        print "Error: $ARGV[$i+1] doesn't look like Port\nExit\n";
                        exit 1;
                }
        }
}

if ( $user && $ip ) {
        main();
}
else {
        printUsage();
}
```

Рис. 4. Скрипт, що описує ключову функціональність застосунку

Далі скрипт здійснює перевірку доступності порту 22 на обладнанні та виконує команду `show port <port> detail` на маршрутизаторі, підключившись до нього за допомогою протоколу SSH (рис. 5).

```
sub main() {
        `nc -w 1 -z $ip 22`;
        my $check_ssh = $? >> 8;
        if ( $check_ssh eq 0 ) {
                $ssh_cmd = "ssh $user\@$ip < ./sli.txt";
                $res = `$ssh_cmd` or warn $!;
                print $res."\n";
        }
        else {
                print "ERROR: It seems Router $ip is unreachable\nExit...";
                exit 1;
        }
}

exit 0;
```

Рис. 5. Головна функція застосунку щодо перевірки доступності порту 22 на обладнанні

Наведемо приклад виконання цього скрипту за допомогою такої команди в CLI:

```
./ssh_Nokia_v01.pl -ip 192.168.1.1 -u admin -P 1/1/1
```

Результат виконання цієї команди наведено на рис. 6.

```
Description          : IXRe-23-1/1/7
Interface            : 1/1/1                    Oper Speed       : 10 Gbps
Link-level           : Ethernet                 Config Speed     : 10 Gbps
Admin State          : up                       Oper Duplex      : full
Oper State           : up - Active in LAG 2
Config Duplex        : N/A
Physical Link        : Yes                      MTU              : 9212
Single Fiber Mode    : No                       Min Frame Length : 64 Bytes
Loopback Mode        : none                     Loopback SwapMac : No
IfIndex              : 1292402689               Hold time up     : 0 seconds
Last State Change    : 02/07/2024 16:28:55      Hold time down   : 0 seconds
Hold Time Down Rmng: 0 cs                       Hold Time Up Rmng: 0 cs
Last Cleared Time    : 02/07/2024 16:28:49      DDM Events       : Enabled
Phys State Chng Cnt: 7
RS-FEC Config Mode : None
RS-FEC Oper Mode     : None
```

Рис. 6. Частковий результат виконання програми ssh_Nokia_v01.pl

Для подальшого опрацювання та відображення результату роботи цього скрипта у вікні браузера дані записують у файл *nokia_raw_data.txt*. Наступний крок – скрипт *nokia_convert_stdout.pl* виконує аналіз файлу *nokia_raw_data.txt* і запис результату в файл *render_obj.txt* (рис. 7).

```
        if ( $_ =~ /Tx\sOutput\sPower\s\(dBm\)\s+(\W\d+\.\d+)\s+(\W\d+\.\d+)\s+(\W\d+\.\d+)\s+(\W\d+\.\d+)\s+(\W\d+\.\d+)/ ){
                        $txop_val = '"txop_val":"'.$1.'"';
                        $txop_ha = '"txop_ha":"'.$2.'"';
                        $txop_hw = '"txop_hw":"'.$3.'"';
                        $txop_lw = '"txop_lw":"'.$4.'"';
                        $txop_la = '"txop_la":"'.$5.'"';
        }

        if ( $_ =~ /Rx\sOptical\sPower\s\(avg\sdBm\)\s+(\W\d+\.\d+)\s+(\W\d+\.\d+)\s+(\W\d+\.\d+)\s+(\W\d+\.\d+)\s+(\W\d+\.\d+
)/ ){
                        $rxop_val = '"rxop_val":"'.$1.'"';
                        $rxop_ha = '"rxop_ha":"'.$2.'"';
                        $rxop_hw = '"rxop_hw":"'.$3.'"';
                        $rxop_lw = '"rxop_lw":"'.$4.'"';
                        $rxop_la = '"rxop_la":"'.$5.'"';
        }
}

$render_obj = "{$descr,$eth,$speed,$ll,$cs,$as,$od,$pl,$mtu,$od,
                                $pl,$mtu,$ts,$tt,$mn,
                                $txw,$cc,$sn,$pn,$oc,$lls,
                                $temp_val,$temp_ha,$temp_hw,$temp_lw,$temp_la,
                                $sv_val,$sv_ha,$sv_hw,$sv_lw,$sv_la,
                                $txbias_val,$txbias_ha,$txbias_hw,$txbias_lw,$txbias_la,
                                $txop_val,$txop_ha,$txop_hw,$txop_lw,$txop_la,
                                $rxop_val,$rxop_ha,$rxop_hw,$rxop_lw,$rxop_la}";

open RO, "> render_obj.txt" or die $!;
print RO $render_obj;
close RO;

exit 0;
"nokia_convert_stdout.pl" [dos] 147L, 4389B written                    122,1-8       Bot
```

Рис. 7. Фрагмент програми оброблення отриманих даних

Для написання вищезазначених скриптів вибрано інтерпретатор Perl, оскільки він дає змогу ефективно виконувати команди CLI та, водночас, використовувати механізм розбору текстових даних за допомогою регулярних виразів – `perlre`.

4.2. Клієнтська частина вебзастосунку

Для забезпечення відображення результатів у цій роботі реалізовано схему взаємодії клієнт-сервер (рис. 2) на основі технології AJAX та платформи Node.js мови програмування Javascript.

У каталозі *public* ПЗ було створено файли *index.html* та *main.html* з html-кодом сторінок, що містять js-файли клієнтської частини застосунку *login.js* та *script.js* відповідно (рис. 8–11).

```
document.body.appendChild(document.createElement('br'));

var login_form = document.createElement('form');
login_form.setAttribute('id', 'loginForm');

login_form.appendChild(login_label);
login_form.appendChild(input_login);
document.body.appendChild(login_form);

var submitLogin = document.createElement('button');
submitLogin.innerText = 'Login';
submitLogin.setAttribute('id', 'submitLogin');

document.body.appendChild(pswd_label);
document.body.appendChild(input_pswd);
document.body.appendChild(document.createElement('br'));

login_form.appendChild(submitLogin);
submitLogin.addEventListener('click', function(event) {
    event.preventDefault();
    var inl = document.getElementById('input_login').value;
    var inp = document.getElementById('input_pswd').value;
//      var login_body = 'Login='+encodeURIComponent(inl)+'&Password='+encodeURIComponent(inp);

    var encodedCredentials = btoa(`${inl}:${inp}`);

    if (encodedCredentials === 'YWRtaW46MTIzNDU2Nw==') {
        window.location.href = 'main.html';
    } else {
          alert("Incorrect login");
    }
});
```

Рис. 8. Фрагмент вихідного коду файлу login.js

```
form.appendChild(slabel);
form.appendChild(select);
document.body.appendChild(form);

var submitButton = document.createElement('button');
submitButton.innerText = 'Submit';
submitButton.setAttribute('id', 'submitButton');

form.appendChild(submitButton);

submitButton.addEventListener('click', function(event) {
        event.preventDefault();
        var inputValue = document.getElementById('myInput').value;
        var inputIP = document.getElementById('InputIP').value;
        var selectValue = document.getElementById('mySelect').value;

        if (!checkIpAddress(inputIP)) {
                alert("ERROR: Wrong IP format!")
                return;
        };

        var body = 'inputIP=' + encodeURIComponent(inputIP) +
                '&inputValue=' + encodeURIComponent(inputValue) +
                '&selectValue=' + encodeURIComponent(selectValue);

        xhr.open('POST', '/resources');
        xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
        xhr.send(body);
});
```

Рис. 9. Фрагмент файлу клієнтської частини script.js, *що відправляє запити (request) на сервер*

```
let xhr = new XMLHttpRequest();
let render_obj;

xhr.onreadystatechange = () => {
   if (xhr.readyState === 4) {
      if (xhr.status === 200) {
                //pasrse and render xhr.responseText as table
                render_obj = JSON.parse(xhr.responseText);
                console.log(render_obj);
```

Рис. 10. Фрагмент файлу клієнтської частини script.js, *що обробляє відповідь сервера (response)*

```
                var table1 = document.createElement('table');
                Object.keys(render_obj).forEach(function(key) {
                        var row = document.createElement('tr');
                        var cell1 = document.createElement('td');
                        cell1.textContent = key;
                        var cell2 = document.createElement('td');
                        cell2.textContent = render_obj[key];
                        row.appendChild(cell1);
                        row.appendChild(cell2);
                        table1.appendChild(row);
                });

                document.getElementById('root').appendChild(table1);
```

Рис. 11. Фрагмент файлу клієнтської частини script.js, *що створює елементи DOM*

Скрипт *login.js*, показаний на рис. 8, відповідає за перевірку облікових даних (рис. 12), які вводить користувач.

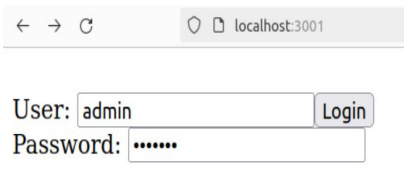


Рис. 12. Сторінка автентифікації

У разі введення некоректних даних автентифікації буде згенеровано відповідне повідомлення (рис. 13).

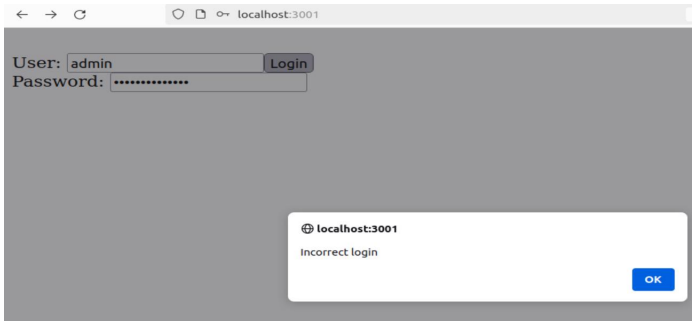


Рис. 13. Повідомлення під час перевірки автентифікації

В разі успішної автентифікації користувачеві буде доступна головна сторінка (рис. 14).

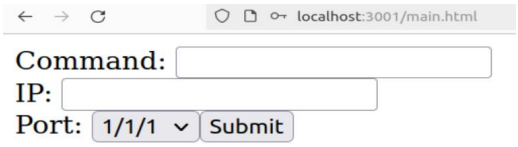


Рис. 14. Сторінка доступу до меню виконання команд

Після введення необхідних даних (рис. 15) користувач відправляє команду на виконання та отримує результат (рис. 16).

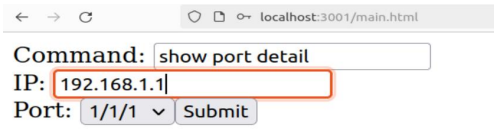
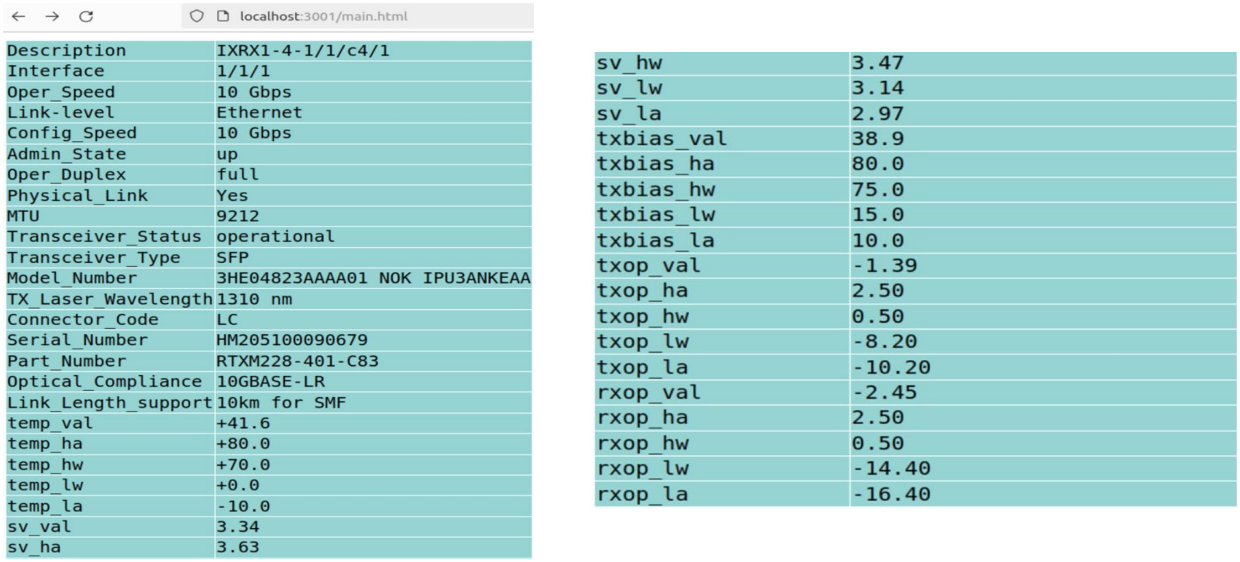


Рис. 15. Форма введення даних команди



Description	IXRX1-4-1/1/c4/1
Interface	1/1/1
Oper_Speed	10 Gbps
Link-level	Ethernet
Config_Speed	10 Gbps
Admin_State	up
Oper_Duplex	full
Physical_Link	Yes
MTU	9212
Transceiver_Status	operational
Transceiver_Type	SFP
Model_Number	3HE04823AAAA01 NOK IPU3ANKEAA
TX_Laser_Wavelength	1310 nm
Connector_Code	LC
Serial_Number	HM205100090679
Part_Number	RTXM228-401-C83
Optical_Compliance	10GBASE-LR
Link_Length_support	10km for SMF
temp_val	+41.6
temp_ha	+80.0
temp_hw	+70.0
temp_lw	+0.0
temp_la	-10.0
sv_val	3.34
sv_ha	3.63

sv_hw	3.47
sv_lw	3.14
sv_la	2.97
txbias_val	38.9
txbias_ha	80.0
txbias_hw	75.0
txbias_lw	15.0
txbias_la	10.0
txop_val	-1.39
txop_ha	2.50
txop_hw	0.50
txop_lw	-8.20
txop_la	-10.20
rxop_val	-2.45
rxop_ha	2.50
rxop_hw	0.50
rxop_lw	-14.40
rxop_la	-16.40

Рис. 16. Результат роботи клієнтської частини ПЗ

4.3. Серверна частина вебзастосунку

Серверну частину реалізовано на платформі Node.js із використанням фреймфорку ***express*** та модулів ***fs*** і ***body-parser*** (рис. 17).

```
const app = express();
const fs = require("fs");
const bodyParser = require("body-parser");

let router_ip;
let cmd_res;

app.use(express.static(__dirname + "/public"));
app.use(bodyParser.urlencoded({extended: false}));
```

Рис. 17. Фрагмент файлу серверної частини index.js, *що підключає необхідні фреймворки та модулі*

Маршутизація HTTP-запитів здійснюється за допомогою методу post. Запис даних у локальний файл на сервері забезпечує функція `writeFileSync` (рис. 18).

```
app.post("/resources", (req, res) => {

        //get data from client
        let rq = req;
        router_ip = rq.body["inputIP"];
        let cmd = rq.body["inputValue"];
        let port = rq.body["selectValue"];

        let wrcmd = 'environment no more\n'+`${cmd} `+`${port} `+'detail';
        let wrfile = __dirname + "/../sli.txt"
        exec_ssh_Nokia(router_ip);      //execute ssh_Nokia pl and write raw_data file
        fs.writeFileSync(wrfile, wrcmd, "utf8");        //write sli.txt

        cmd_res = get_render_obj();
        res.json(cmd_res);      //send json to client
});
```

Рис. 18. Фрагмент файлу серверної частини index.js, *що забезпечує маршрутизацію клієнтських викликів та запуск скрипту* ssh_Nokia_v01.pl

Після старту сервер прослуховує порт 3001 (рис. 19).

```
[nodemon] starting `node index.js`
Server has been started at localhost:3001
```

Рис. 19. Повідомлення про старт Node.JS сервера

Виконання локальних програм на сервері можливе завдяки модулю `node:child_process` (рис. 20).

```
let rd_file = __dirname + "/../nokia_raw_data.txt";
const { spawn } = require('child_process');
const rd = spawn( __dirname + '/../ssh_Nokia_v01.pl',['-u', 'admin', '-ip', router_ip]);
rd.stdout.on('data', (data) => {
  console.log(`stdout: ${data}`);
  fs.writeFileSync(rd_file, data);
});
```

Рис. 20. Фрагмент файлу серверної частини index.js, *що застосовує модуль* node:child_process

Необхідно зазначити, що розроблене ПЗ протестовано на лабораторній мережі компанії Nokia Bell NV, розташованій в м. Антверпен, Бельгія (реальні IP-адреси та облікові дані змінено з міркувань безпеки).

4.4. Якісна характеристика розробленого програмного забезпечення

Розроблене програмне забезпечення має низку характеристик, що визначають його практичну цінність. Зокрема, ПЗ:

- забезпечує можливість виконання команд оболонки за умови мінімальних навичок роботи із CLI;
- підтримує роботу через вебінтерфейс (GUI);
- його вартість низька порівняно із пропрієтарними програмними рішеннями.

Програмне забезпечення орієнтоване на використання персоналом ланок технічного обслуговування та диспетчерських центрів. Під час експлуатації інфокомунікаційної мережі інженерному персоналу часто доводиться звертатися за допомогою до фахівців центрів управління. Водночас завдання, що не потребують поглиблених знань управління мережею, як-от визначення статусу портів, зміна їх адміністративного стану, вимірювання параметрів ліній та інші подібні операції, можуть виконуватися безпосередньо на місцях (у приміщеннях вузлів агрегації чи виносних концентраторах обладнання).

Наприклад, щоб отримати інформацію про статус порта маршрутизатора, у традиційній практиці необхідно звернутися до інженера центру управління, сформулювати завдання, дочекатися відповіді та проаналізувати надані дані. Залежно від завантаженості персоналу цей процес може тривати від 3 до 15 хв.

Натомість, використовуючи розроблене програмне забезпечення, достатньо мати мережевий доступ до обладнання, відкрити у браузері сторінку localhost:3001 (або запустити консольний скрипт), пройти автентифікацію, ввести IP-адресу маршрутизатора та виконати необхідну команду. Отриманий результат відображається безпосередньо на екрані користувацького пристрою, що зменшує тривалість виконання завдання до приблизно двох хвилин. Отже, запропонований підхід до розподілу завдань усуває необхідність обміну даними між різними секторами експлуатаційних служб і дає змогу істотно пришвидшити виконання рутинних операцій.

Висновки

У статті досліджено та описано практичну реалізацію засобів мережевого програмування для автоматизації процесів управління інфокомунікаційними мережами. Проаналізовано відомі підходи до автоматизації, визначено їхні переваги та недоліки, а також розроблено ПЗ, що забезпечує виконання типових завдань управління мережею та її моніторингу.

Результати виконаних досліджень підтвердили, що автоматизація є ключовим інструментом для сучасних інфокомунікаційних систем, оскільки дає змогу істотно знизити операційні витрати, підвищити ефективність роботи мережі та забезпечити її адаптивність до змін у середовищі функціонування. Використання автоматизованих засобів продемонструвало можливість зменшити тривалість виконання завдань конфігурації та моніторингу з декількох хвилин до кількох секунд, а також кількість помилок, пов’язаних із людським фактором.

Дослідження виявило, що оптимальний вибір інструментів для автоматизації залежить від конкретних завдань і умов їх виконання. Використання інтерфейсу командного рядка (CLI) залишається актуальним методом налаштування мережевого обладнання, оскільки дає змогу створювати сценарії для виконання однотипних завдань. Водночас графічні вебінтерфейси (GUI) мають перевагу в частині зручності та наочності.

Огляд інструментів програмування показав, що мову Perl широко використовують для розроблення систем мережевої автоматизації завдяки її гнучкості, підтримці бібліотек і здатності інтегруватися із різними пристроями. Для клієнтської частини вебзастосунку доцільно застосовувати технології AJAX і JavaScript/DOM, які забезпечують інтерактивність та динамічне оновлення даних. Серверну частину можна ефективно реалізовувати засобами Node.js, що відзначається високою масштабованістю та подієво-орієнтованою архітектурою.

Програмна реалізація завдань управління та моніторингу продемонструвала, що створене програмне забезпечення може виступати інтегрованою платформою для виконання операцій над ІКМ. Його архітектура, у яку входять ядро, клієнтська та серверна частини, забезпечує ефективне функціонування та можливість подальшого розширення. Практичні тести засвідчили зручність використання як для персоналу обслуговування, так і для диспетчерських центрів.

Перспективи подальших досліджень полягають у розвитку гнучких і масштабованих систем автоматизації управління мережею, інтеграції новітніх протоколів та створенні універсальних рішень для роботи із обладнанням різних виробників. Також важливим напрямом є уніфікація інтерфейсів і розроблення додаткових модулів, які дають змогу знизити витрати на експлуатацію мереж та навчання персоналу. Отже, результати роботи підтвердили, що автоматизація процесів управління інфокомунікаційними мережами є критично важливим фактором для забезпечення їх ефективності, надійності та конкурентоспроможності.

Список використаних літературних джерел

[1] *Лемешко, О. В., Єременко, О. С., Невзорова, О. С. (2020). Потокові моделі та методи маршрутизації в інфокомунікаційних мережах: відмовостійкість, безпека, масштабованість. Харків: ХНУРЕ. 308 с. DOI: https://doi.org/10.30837/978-966-659-282-1*

[2] *Лемешко, О. В., Єременко, О. С., Євдокименко, М. О., Шаповалова, А. С., Слейман, Б. (2022). Моделювання та оптимізація процесів безпечної та відмовостійкої маршрутизації в телекомунікаційних мережах: монографія. М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. Харків: ХНУРЕ, 198 с. DOI: https://doi.org/10.30837/978-966-659-378-1*

[3] *Abuelenain, K., Doyle, J., Karneliuk, A., Jain, V. (2021). Network Programmability and Automation Fundamentals. Cisco Press, 1232 p.*

[4] *Oswalt, M., Adell, C., Lowe, S. S., Edelman, J. (2023). Network Programmability and Automation: Skills for the Next-Generation Network Engineer, 2nd edn. O'Reilly Media Inc., 825 p.*

[5] *Pinto, I., Lacunza, A. A. (2021). Network Automation Trends and Strategy. Cisco Systems Inc., 17 p., available at: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/network-automation-strategy-wp.pdf*

[6] *Yeremenko, O., Savchenko, R., Yakovenko, K., Shestopalov, S. (2025). "Study of Reliability and Fault Tolerance Management in Information and Communication Networks: Modeling and Testing of Default Gateway Redundancy Protocols". Infocommunication technologies and electronic engineering, Vol. 5, No. 1, pp. 15–33. DOI: https://doi.org/10.23939/ictee2025.01.015*

[7] *Ellis-Barker, S. (2023), "7 benefits of network automation technology to power your brand". Available at: https://www.hamilton-barnes.com/resources/blog/7-benefits-of-network-automation-technology-to-power-your-brand/*

[8] *Know Computing, "Features, Advantages, and Disadvantages Command line interface (CLI)". Available at: https://www.knowcomputing.com/features-advantages-and-disadvantages-command-line/*

[9] *Petryschuk, S. (2022), "Networking Automation Software: Pros & Cons.". Available at: https://www.auvik.com/franklyit/blog/best-network-automation-software/*

[10] *Telecom Network Automation, Cloud-native automation for Telecom networks. Available at: https://cloud.google.com/telecom-network-automation*

[11] *GeeksforGeeks (2025), What is Network Automation? Available at: https://www.geeksforgeeks.org/what-is-network-automation/*

[12] *Schwartz, R.L., Foy, B. D., Phoenix, T. (2021) Learning Perl: Making Easy Things Easy and Hard Things Possible, 8th edn. O'Reilly Media Inc., 395 p.*

[13] *Lornfeld, J. (2025), Perl Programming: A Complete Learning Journey Covering Basic and Advanced Topics. 200 p.*

[14] *W3Schools, AJAX Introduction. Available at: https://www.w3schools.com/asp/asp_ajax_intro.asp*

[15] *Introducing JSON. Available at: https://www.json.org/json-en.html*

[16] *W3Schools, JavaScript HTML DOM. Available at: https://www.w3schools.com/js/js_htmldom.asp*

[17] *Node-RED, User Guide. Node-RED Documentation. Available at: https://nodered.org/docs/user-guide/*

APPLICATION OF NETWORK PROGRAMMING TOOLS FOR AUTOMATING INFORMATION AND COMMUNICATION NETWORK MANAGEMENT PROCESSES

Oleksandra Yeremenko, Roman Mamon, Dmytro Andrushko, Liubov Melnikova

Kharkiv National University of Radio Electronics, 14, Nauky Ave., Kharkiv, 61166, Ukraine

The article is devoted to the study of the automation of management processes of information and communication networks using network programming tools. The work substantiates the relevance of automation as a key direction in the development of modern information and communication systems, driven by the growth in traffic volumes and the complexity of network architecture. It is shown that automation allows for increasing operational efficiency, reducing task execution time, and minimizing the human factor. An analysis of existing approaches has shown that automation covers the entire network lifecycle: from initial configuration and commissioning to ongoing support and optimization. Depending on the application area, it is implemented both in corporate infrastructures and in service providers' environments. At the same time, it is essential to choose the optimal tools, in particular the command line interface, which allows you to create scripts for repetitive tasks, and graphical web interfaces that provide convenience and clarity. Special attention is paid to the Perl language, which demonstrates broad opportunities for integration with network devices and the creation of management scripts. For the client side of web applications, it is advisable to use AJAX and JavaScript / DOM, which provide interactivity and dynamic data updates. The server side can be implemented using Node.js, which is characterized by scalability and event-driven architecture. The article presents an applied solution in the form of developed software for managing and monitoring IP/MPLS routers. Its architecture includes a core, client, and server parts, providing an integrated platform for automating operational tasks. Practical tests have confirmed the ease of use of the application, reduced operation time, and fewer errors. Prospects for further research are related to the integration of the latest protocols, the unification of interfaces, and the creation of scalable network management systems.

Keywords: *information and communication networks, automation, network programming, management, monitoring.*