ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ІНЖЕНЕРІЇ ТА ТРАНСПОРТУ

КАФЕДРА АВТОМАТИЗАЦІЇ, РОБОТОТЕХНІКИ І МЕХАТРОНІКИ

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Автоматизована система моніторингу та прогнозування збоїв у телекомунікаційних мережах»

«Automated System for Monitoring and Forecasting Failures in Telecommunication Networks»

Виконав: студент 6 курсу, групи 6А спеціальності 174 – «Автоматизація, комп'ютерно-інтегровані технології та робототехніка освітньо-професійної програми – «Автоматизація та комп'ютерно-інтегровані технології»
Корнійчук В.О.

Керівник: к.т.н., доцент Лебеденко Ю.О.

Рецензент_____
(прізвище та ініціали)

Херсон – 2025 рік

Херсонський національний технічний університет

Факультет Інженерії та транспорту
Кафедра Автоматизації, робототехніки і мехатроніки
Освітньо-кваліфікаційний рівень магістр
Спеціальність 174 – «Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітньо-професійна програма «Автоматизація та комп'ютерно-інтегровані технології»

ЗАТВЕРДЖУЮ:
Завідувач кафедри автоматизації,
робототехніки і мехатроніки
Селіверстов І.А.
«___» _____ 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу студенту

Корнійчуку Віталію Олеговичу

1. Тема проєкту: Автоматизована система моніторингу та прогнозування збоїв у телекомунікаційних мережах

Automated System for Monitoring and Forecasting Failures in Telecommunication Networks
керівник проєкту: к.т.н., доцент Лебеденко Ю.О.
затверджена наказом вищого навчального закладу від 28.08.2025 р. № 364-с
2. Строк подання студентом проєкту «10» грудня 2025 р.
3. Вихідні дані до проєкту: Дослідити методи та засоби підвищення ефективності процесів керування мережевим трафіком з метою прогнозування збоїв у телекомунікаційних мережах
4. Зміст розрахунково-пояснювальної записки 1) Аналіз технологічних рішень для забезпечення якості обслуговування у телекомунікаційних мережах; 2) Аналіз самоподібності мережевого трафіку та побудова фрактальної математичної моделі; 3) Розробка автоматизованої системи моніторингу та прогнозування збоїв у телекомунікаційних мережах
5. Перелік графічного матеріалу 1) Постановка задачі дослідження; 2) Етапи проектування нейронних мереж; 3) Критичні ситуації. Навчання нейронної мережі; 4) Алгоритм роботи нейромережі; 5) Основні типи нейронів; 6) Схема моделювання нейромережі

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	Лебеденко Ю.О., к.т.н., доцент		

7. Дата видачі завдання « 5 « жовтня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Літературний огляд	15.10.2025	
2	Аналіз технологічних рішень для забезпечення якості обслуговування у телекомунікаційних мережах	25.10.2025	
3	Аналіз самоподібності мережевого трафіку та побудова фрактальної математичної моделі	01.11.2025	
4	Розробка автоматизованої системи моніторингу та прогнозування збоїв у телекомунікаційних мережах	15.11.2025	
5	Обробка результатів	20.11.2025	
6	Висновки по роботі	25.11.2025	
7	Оформлення ПЗ і креслень	05.12.2025	

Студент Корнійчук В.О.

(підпис)

Керівник проєкту Лебеденко Ю.О.

(підпис)

.

Номер рядка	Формат	Позначення	Найменуванння	Кіль-кість	Шифр до-кумента	Приміт-ка
1	А4		Завдання на проєктування	2		
2	А4	ХНТУ174.КРМ.25.10.РФ	Реферат	3	РФ	
3	А4	ХНТУ174.КРМ.25.10.ПЗ	Пояснювальна записка	110	ПЗ	
4	А1	Демонстраційний плакат	Постановка задачі дослідження	1		
5	А1	Демонстраційний плакат	Етапи проектування	1		
			нейронних мереж			
6	А1	Демонстраційний плакат	Критичні ситуації	1		
			Навчання нейронної			
			мережі			
7	А1	Демонстраційний плакат	Алгоритм роботи	1		
			нейромережі			
8	А1	Демонстраційний плакат	Основні типи нейронів	1		
9	А1	Демонстраційний плакат	Схема моделювання	1		
			нейромережі			

Зм.	Лист	№ докум	Підпис	Дата	ХНТУ174.КРМ.25.10 ВП		
Розроб.		Корнійчук В.О.			Відомість об'єму роботи	Літ.	Лист 1 / Аркушів 1
Перевір.		Лебеденко Ю.О.					
Реценз.						ХНТУ, гр.6А	
Н. Контр.		Сарафаннікова					
Затверд.		Селіверстов І.А.					

РЕФЕРАТ

Кваліфікаційна робота магістра: 110 сторінок, 20 рисунків, 3 таблиці, 1 додаток. Графічна частина – 6 аркушів формату А1.

У кваліфікаційній роботі магістра проведено системний аналіз технічного функціонування мереж. Проведений аналіз базових мережних топологій та мережних технічних і програмних засобів, приводяться основні мережні технології. Обґрунтований вибір математичної моделі для дослідження основних характеристик трафіку. Розроблений алгоритм перерозподілу смуги пропускання на базі нейромережевих технологій. Побудована модель короткострокового прогнозування мережевого трафіку на базі нейронних мереж. Розроблений метод інтелектуального керування трафіком на основі системного аналізу мереж, алгоритмів перерозподілу смуги пропускання і буферної пам'яті маршрутизаторів, нейромережевої моделі короткострокового прогнозування мережевого трафіку. На основі даного методу керування розроблена система керування трафіком. Проведено імітаційне моделювання розробленої системи, яке показало підвищення ефективності керування трафіком.

МЕРЕЖІ ПЕРЕДАЧІ ДАННИХ, НЕЙРОННІ МЕРЕЖІ, КЕРУВАННЯ ТРАФІКОМ, МАТЕМАТИЧНА МОДЕЛЬ НЕЙРОМЕРЕЖІ, КОМУТАТОР, ПРОГРАМНИЙ ПРОДУКТ, АЛГОРИТМ РОБОТИ ПРОГРАМИ

THE ABSTRACT

Qualification master's work: 110 pages, 20 figures, 3 table, 1 appendix. A graphic part – 6 sheets of a format A1.

In the master's qualification work the systematic analysis of technical functioning was carried out. The analysis of basic network topologies and network technical and software tools, the main network technologies are given. The choice of the mathematical model for the study of the main characteristics of traffic is substantiated. The algorithm of redistribution of bandwidth is developed on the basis of neural network technologies. A model of short-term forecasting of network traffic based on neural networks is constructed. The method of intellectual traffic management on the basis of system analysis of networks, algorithms of redistribution of bandwidth and buffer memory of routers, neural network model of short-term forecasting of network traffic is developed. On the basis of this method of management a traffic management system is developed. Simulated simulation of the developed system was carried out, which showed an increase in the efficiency of traffic management.

DATA TRANSMISSION NETWORK, NEURAL NETWORKS, TRAFFIC MANAGEMENT, MATHEMATICAL MODEL OF NEURAL NETWORK, SWITCH, PROGRAM PRODUCT, PROGRAM OPERATION ALGORITHM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК І СКОРОЧЕНЬ 5
ВСТУП 6
1 АНАЛІЗ ТЕХНОЛОГІЧНИХ РІШЕНЬ ДЛЯ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ОБСЛУГОВУВАННЯ У ТЕЛЕКОМУНІКАЦІЙНИХ МЕРЕЖАХ 9
1.1 Технологічні основи функціонування телекомунікаційних мереж 9
1.1.1 Технологія Ethernet 11
1.1.2 Стек протоколів TCP / IP 13
1.2 Функціональний рівень 15
1.3 Механізми керування 16
1.3.1 Механізми керування інтенсивністю трафіку 16
1.3.2 Механізми забезпечення гарантованої якості обслуговування 19
1.4 Аналіз підходів для автоматизованого моніторингу та прогнозування збоїв у телекомунікаційних мережах 31
1.4.1 Самоподібність мережевого трафіку як ключова концепція для прогнозування збоїв 31
1.4.2 Обгрунтування підходу для керування трафіком 39
1.5 Висновок до розділу 1 41
2 АНАЛІЗ САМОПОДІБНОСТІ МЕРЕЖЕВОГО ТРАФІКУ ТА ПОБУДОВА ФРАКТАЛЬНОЇ МАТЕМАТИЧНОЇ МОДЕЛІ 43
2.1 Виявлення характеристик та властивостей самоподібних процесів у телекомунікаційних системах 43
2.2 Експериментальне дослідження та визначення параметрів мережевого трафіку 47
2.3 Побудова математичної моделі трафіку на основі фрактального броунівського руху 58

2.4 Верифікація та оцінювання можливості прогнозування трафіку на основі фрактальної моделі 65
2.5 Висновки до розділу 2 66
3 РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ МОНІТОРИНГУ ТА ПРОГНОЗУВАННЯ ЗБОЇВ У ТЕЛЕКОМУНІКАЦІЙНИХ МЕРЕЖАХ 67
3.1 Розробка алгоритмічного забезпечення інтелектуальної системи моніторингу та керування на основі нейромережевих технологій 67
3.1.1 Алгоритм перерозподілу смуги пропускання 67
3.1.2 Реалізація прогнозування трафіку на основі нейромережевих технологій 70
3.1.3 Алгоритм перерозподілу буферної пам'яті маршрутизаторів 77
3.1.4 Алгоритм інтелектуального керування трафіком 80
3.2 Результати імітаційного моделювання 84
3.3 Архітектура та структурна схема інтелектуальної системи керування трафіком 88
3.4 Програмне забезпечення та реалізація системи керування трафіком 89
3.5 Впровадження автоматизованої системи прогнозування трафіку 90
3.6 Висновок до розділу 3 94
ВИСНОВКИ 95
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 96
ДОДАТОК А ЛІСТИНГ ПРОГРАМИ 99

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК І СКОРОЧЕНЬ

ARP – Address Resolution Protocol (протокол розділення адрес);

CPU – Central Processing Unit (центральний процесор);

ФБР – Фрактальний Броунівський Рух;

I/O – Input/Output (введення/виведення);

IOS-XE – Internetwork Operating System - eXtensible Edition (операційна система Cisco);

IP – Internet Protocol (Міжмережевий протокол);

ISO – International Organization for Standardization (Міжнародна організація зі стандартизації);

MAE – Mean Absolute Error (середня абсолютна похибка);

MRTG – Multi Router Traffic Grapher (програма для графічного відображення мережевої статистики);

OSI – Open Systems Interconnection (взаємодія відкритих систем);

QoS – Quality of Service (якість обслуговування);

SNMP – Simple Network Management Protocol (простий протокол керування мережею);

SSH – Secure Shell (безпечна оболонка);

tcng – Traffic Control Next Generation (утиліта керування трафіком Linux);

TCP – Transmission Control Protocol (протокол керування передачею);

ДСЗ – довгострокова залежність;

ЕОМ – Електронно-обчислювальна машина;

НМ – Нейронна мережа;

ОЗП – Оперативний запам'ятовувальний пристрій.

ВСТУП

В умовах стрімкої цифрової трансформації суспільства та глобалізації економічних процесів, телекомунікаційні мережі перетворилися на фундаментальну інфраструктуру. Вони є основою для функціонування сучасних організаційно-технічних комплексів, забезпечуючи миттєвий обмін даними, зв'язок між розподіленими системами та доступ до глобальних інформаційних ресурсів. Від промислової автоматизації та «Інтернету речей» (IoT) до фінансових сервісів та хмарних обчислень – стабільна робота мережевих технологій є вирішальним фактором конкурентоспроможності та ефективності.

Водночас, зі зростанням масштабу та складності телекомунікаційних систем, кардинально посилюються вимоги до їх надійності, відмовостійкості та якості обслуговування (QoS). Будь-які збої, затримки чи втрати даних у мережевій інфраструктурі, особливо в промислових або бізнес-критичних середовищах, можуть призвести до значних фінансових та репутаційних втрат. Ефективне керування такими гетерогенними комплексами вимагає не лише потужного технічного забезпечення, але й передових інформаційних та програмних інструментів для аналізу їхнього стану, діагностики проблем та, що найважливіше, прогнозування потенційних збоїв.

Актуальність теми. Сучасні телекомунікаційні мережі, зокрема ті, що функціонують у промислових середовищах, є складними, критично важливими інфраструктурами. Їхня стабільність та ефективність, що регулюються стандартами OSI та стеком TCP/IP, є ключовими для забезпечення безперервності технологічних процесів, Інтернету речей (IoT) та систем керування.

Проте традиційні підходи до моніторингу та керування мережами, що базуються на статичних конфігураціях та усереднених показниках,

виявляються недостатньо ефективними. Головною проблемою є фрактальна (самоподібна) природа сучасного мережевого трафіку. Ця властивість призводить до непередбачуваних пікових навантажень, які не фіксуються стандартними методами моніторингу, спричиняючи переповнення буферів комутаційного обладнання та, як наслідок, значні втрати пакетів.

Такі збої критично погіршують якість обслуговування (QoS) для сервісів, чутливих до затримок, таких як системи SCADA, відеоконтроль у реальному часі чи IP-телефонія. Виникає нагальна потреба у переході від реактивних методів (усунення збоїв постфактум) до проактивних систем, здатних прогнозувати погіршення стану мережі до його настання. Розробка інтелектуальних систем моніторингу, здатних прогнозувати збої на основі аналізу самоподібного трафіку, є надзвичайно актуальною науково-практичною задачею.

Метою даної кваліфікаційної роботи є підвищення надійності функціонування промислових телекомунікаційних мереж шляхом розробки автоматизованої системи, здатної здійснювати моніторинг, діагностику та прогнозування збоїв на основі нейромережевого аналізу фрактального трафіку.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1. Провести системний аналіз функціонування та архітектурних особливостей сучасних промислових телекомунікаційних мереж.

2. Дослідити та змоделювати трафік в промислових мережах, виконати аналіз його фрактальних (самоподібних) властивостей та їх впливу на якість обслуговування (QoS) і виникнення збоїв.

3. Розробити алгоритмічне забезпечення інтелектуальної системи моніторингу, включаючи розробку нейромережевої моделі для прогнозування інтенсивності трафіку та виявлення аномалій, що передують збоям.

4. Розробити та програмно реалізувати прототип інтелектуальної системи керування трафіком, що включає діагностику збоїв, та перевірити її ефективність шляхом імітаційного моделювання.

Об’єкт дослідження – процеси функціонування та передачі даних у промислових телекомунікаційних мережах.

Предмет дослідження – математичні моделі мережевого трафіку, а також методи та алгоритми моніторингу, діагностики та нейромережевого прогнозування збоїв у промислових мережах.

Методи дослідження. Для вирішення поставлених задач використовувався комплексний підхід, що включає: методи системного аналізу та теорії телекомунікаційних мереж (для аналізу архітектур), теорія фракталів та статистичний аналіз (для дослідження самоподібності трафіку та розрахунку показника Херста), теорія штучних нейронних мереж (для розробки моделі прогнозування), а також методи імітаційного моделювання та об'єктно-орієнтованого програмування (для практичної реалізації системи).

Практична значущість одержаних результатів. Практична значущість роботи полягає у розробці програмного забезпечення та алгоритмів, здатних не просто реагувати на збої в мережі, а прогнозувати їх виникнення на основі аналізу прихованих характеристик трафіку. Це дозволяє проактивно адаптувати параметри комутаційного обладнання (наприклад, політики QoS, розміри черг), що призводить до підвищення надійності та відмовостійкості промислових мереж, мінімізації втрат пакетів та забезпечення стабільної якості обслуговування для критичних технологічних процесів.

1 АНАЛІЗ ТЕХНОЛОГІЧНИХ РІШЕНЬ ДЛЯ ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ОБСЛУГОВУВАННЯ У ТЕЛЕКОМУНІКАЦІЙНИХ МЕРЕЖАХ

1.1 Технологічні основи функціонування телекомунікаційних мереж

Сучасні телекомунікаційні мережі проєктуються та функціонують відповідно до загальновизнаних міжнародних норм та стандартів. Фундаментальною концептуальною рамкою, що визначає структуровані засади комунікації між мережевими пристроями, є Еталонна модель взаємодії відкритих систем (OSI). Ця модель була представлена Міжнародною організацією зі стандартизації (ISO) у 1984 році.

Модель OSI визначає загальну архітектуру взаємодії в мережах, пропонуючи семирівневу ієрархічну структуру, проте не регламентує конкретних операційних методів чи протоколів. Уся розробка нових мережевих стандартів здійснюється в руслі цієї моделі.

Кожен із семи рівнів моделі має чітко визначену зону відповідальності в загальному процесі комунікації:

– Фізичний (Physical): Передача бітів по фізичному середовищу.

– Канальний (Data Link): Керування доступом до середовища, формування кадрів та контроль помилок.

– Мережевий (Network): Маршрутизація та логічна адресація даних (пакетів).

– Транспортний (Transport): Наскрізна доставка даних, сегментація та відновлення.

– Сеансовий (Session): Керування сеансами зв'язку.

– Представницький (Presentation): Форматування даних, шифрування/дешифрування.

– Прикладний (Application): Забезпечення мережевого доступу

для програм користувача.

Зв'язок між суміжними рівнями забезпечується через стандартизовані інтерфейси, а комунікація з аналогічним рівнем у віддаленій системі відбувається на основі протокольних домовленостей. Реалізація рівнів може бути досягнута як апаратними, так і програмними засобами.

Незважаючи на свою концептуальну значимість, модель OSI ніколи не була впроваджена у повному обсязі з відповідним набором протоколів. На сьогоднішній день у світових мережах домінує стек протоколів TCP/IP, який є спрощеним, чотирирівневим варіантом моделі (рис. 1.1).

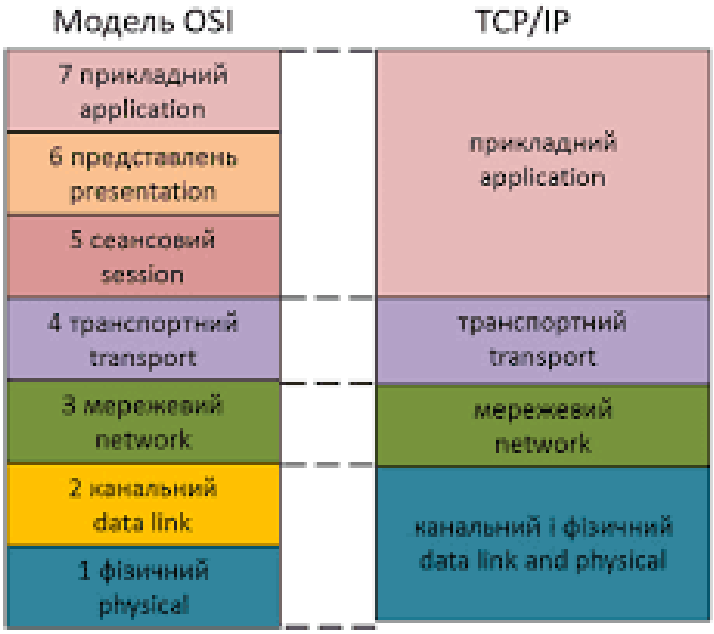


Рисунок 1.1 – Порівняння моделей OSI та TCP/IP

Хоча прив’язка окремих інтернет-протоколів до рівнів моделі OSI не є абсолютно строгою, вона все ж виконує важливу роль у структуризації мережевих технологій. Така умовність зумовлена історичними чинниками, зокрема конкуренцією між різними підходами до побудови мереж та незавершеністю самого проєкту OSI. Незважаючи на це, модель OSI

зберігає своє значення як базова теоретична конструкція, що слугує орієнтиром для виробників мережевого обладнання, формування міжнародних стандартів у сфері телекомунікацій, а також для освітніх програм і професійної підготовки фахівців у галузі комунікацій.

У контексті мережевої інфраструктури важливо розглядати її склад як сукупність елементів, що забезпечують ефективну взаємодію між користувачами та системами. До таких елементів належать кінцеві обчислювальні пристрої, які можуть функціонувати як окремо, так і в складі кластерів, а також активне мережеве обладнання, що виконує функції комутації, маршрутизації та керування трафіком. Зокрема, до пристроїв комутації та підключення належать ретранслятори, мости та комутатори, які забезпечують передачу даних у межах локальних мереж. Для зв'язку між різнорідними мережами використовуються маршрутизатори та шлюзи, що дозволяє реалізувати міжмережеву взаємодію.

Функціонування сучасних телекомунікаційних систем неможливе без застосування ключових протоколів стека TCP/IP, серед яких варто виокремити TCP, UDP та IP, що відповідають за встановлення з'єднань, передачу пакетів та адресацію. Крім того, важливу роль відіграють механізми керування трафіком, які включають інструменти для регулювання пропускної здатності та забезпечення якості обслуговування, зокрема технології шейпінгу та полісінгу, що дозволяють оптимізувати розподіл мережевих ресурсів відповідно до заданих параметрів.

1.1.1 Технологія Ethernet

Технологія Ethernet відіграє провідну роль серед методів пакетної передачі даних, що застосовуються в локальних та корпоративних мережах. У цьому контексті під мережею слід розуміти інтегровану систему узгоджених протоколів і апаратно-програмних засобів, яка забезпечує

повноцінне функціонування обчислювального середовища.

Основи роботи Ethernet визначаються в межах двох нижніх рівнів еталонної моделі OSI. На фізичному рівні регламентуються параметри електричних сигналів, типи кабельних з'єднань та характеристики середовища передачі. Канальний рівень, своєю чергою, відповідає за структуру кадрів, а також за механізми доступу до спільного середовища, включаючи виявлення та обробку колізій – хоча останнє стало менш актуальним із поширенням комутованих мереж.

Історично апаратна підтримка Ethernet базувалася на повторювачах, які працювали на фізичному рівні. Двопортові повторювачі забезпечували регенерацію сигналу, дозволяючи з'єднувати віддалені пристрої. Згодом з'явилися багатопортові повторювачі – хаби, що здійснювали широкомовну передачу, тобто надсилали отримані дані на всі порти одночасно. Такий підхід, хоча й простий, створював значні труднощі в масштабних мережах: зростала кількість колізій, а трафік ставав надмірно навантаженим. У сучасних умовах ці пристрої практично вийшли з ужитку через обмежену продуктивність і низький рівень безпеки.

Суттєвий прорив у розвитку Ethernet забезпечили комутатори – багатопортові мости, що працюють на канальному рівні. На відміну від хабів, комутатори здійснюють селективну передачу даних, спрямовуючи кадри лише до відповідного адресата. Це стало можливим завдяки використанню MAC-адресної таблиці, яка динамічно формується в процесі роботи. Коли комутатор отримує кадр, він фіксує MAC-адресу відправника та асоціює її з відповідним портом. Якщо адреса одержувача вже відома, кадр надсилається безпосередньо на потрібний порт. У випадку, коли адреса ще не внесена до таблиці, комутатор тимчасово вдається до широкомовної передачі, що дозволяє виявити новий вузол і оновити таблицю.

Попри ефективність, Ethernet має фундаментальні обмеження, пов'язані з його прив'язкою до фізичного та канального рівнів. MAC-адреса,

яка є унікальною і немаршрутизованою, функціонує лише в межах одного широкомовного домену. Це означає, що Ethernet не здатен самостійно забезпечити логічне розділення мереж або керування трафіком поза межами локального сегмента.

Для побудови масштабних, географічно розподілених мереж, що охоплюють кілька сегментів або підмереж, необхідно залучати засоби мережевого рівня – зокрема маршрутизатори та протоколи, такі як IP. Саме вони дозволяють подолати обмеження Ethernet і забезпечити гнучке, кероване з'єднання між різними частинами мережевої інфраструктури.

1.1.2 Стек протоколів TCP / IP

Протокол IP (Internet Protocol), що функціонує на мережевому рівні еталонної моделі OSI, становить фундаментальний компонент стеку TCP/IP. Його основне призначення полягає в забезпеченні доставки датаграм між вузлами мережі, причому ця доставка є ненадійною – протокол не гарантує збереження послідовності або повторну передачу втрачених пакетів.

Кожен пристрій у мережі ідентифікується за допомогою IP-адреси – унікального числового ідентифікатора, який у версії IPv4 представлений як 32-бітне значення. Призначення IP-адрес може здійснюватися вручну адміністратором або автоматично, з використанням спеціалізованих протоколів, таких як DHCP. Саме ці адреси дозволяють протоколу IP здійснювати маршрутизацію інформації між різними мережевими сегментами.

Передача датаграм між мережами реалізується за допомогою маршрутизаторів – пристроїв, що також працюють на мережевому рівні. Після отримання пакета маршрутизатор аналізує його цільову IP-адресу та звертається до таблиці маршрутизації, яка містить інформацію про можливі шляхи доставки, включаючи адресу наступного вузла (хопа) та відповідні

метрики. Якщо відповідний маршрут знайдено, датаграма передається далі; у протилежному випадку вона відкидається.

На транспортному рівні моделі OSI функціонують два ключові протоколи – TCP та UDP, які забезпечують зв’язок між процесами на кінцевих вузлах.

Протокол TCP (Transmission Control Protocol) орієнтований на надійність: він встановлює з’єднання між джерелом і призначенням, контролює цілісність переданих даних, забезпечує їхню послідовність і, за потреби, ініціює повторну передачу. Завдяки вбудованим механізмам контролю помилок і регулювання потоку, TCP є оптимальним для застосунків, де критично важлива достовірність переданої інформації – зокрема, для веб-сервісів, електронної пошти та інших транзакційних систем.

Натомість UDP (User Datagram Protocol) реалізує принцип мінімального втручання: він не гарантує доставку, не перевіряє послідовність пакетів і не передбачає механізмів повторної передачі. Такий підхід дозволяє зменшити затримки та підвищити ефективність використання пропускної здатності. UDP застосовується в ситуаціях, де швидкість передачі є пріоритетною, а втрати даних не критичні – наприклад, у потоковому відео, онлайн-іграх або DNS-запитах.

Окрему роль у функціонуванні мережі відіграє протокол ARP (Address Resolution Protocol), який забезпечує перетворення IP-адрес у фізичні MAC-адреси. Для встановлення з’єднання з пристроєм у межах локального сегмента необхідно знати його MAC-адресу. Якщо відповідний запис відсутній у локальній ARP-таблиці, ініціюється широкомовний запит, що поширюється мережею з метою виявлення пристрою з потрібною IP-адресою. У відповідь на запит вузол-призначення надсилає ARP-відповідь, яка дозволяє джерелу оновити свою таблицю відповідностей.

1.2 Функціональний рівень

Ефективне керування мережевим трафіком передбачає глибоке розуміння характеристик сервісів, що функціонують у межах інформаційної інфраструктури. Оскільки вимоги до надійності, швидкості та пропускної здатності трафіку безпосередньо залежать від типу сервісу, його класифікація набуває критичного значення для забезпечення якості обслуговування (QoS), оптимального розподілу ресурсів та адаптації мережевих механізмів до прикладних задач.

Одним із базових критеріїв класифікації є чутливість сервісу до втрати пакетів. У цьому аспекті розрізняють сервіси, які толерантно сприймають втрати – вони здатні компенсувати відсутні пакети за допомогою повторної передачі або допускають тимчасове зниження швидкості без істотного впливу на функціональність. Протилежну групу становлять сервіси, для яких навіть незначні втрати призводять до деградації якості або повної втрати працездатності. Це особливо актуально для систем реального часу, таких як голосовий зв’язок або критичні медичні моніторингові платформи.

Не менш важливим параметром є чутливість до затримок. Деякі сервіси, як-от електронна пошта, залишаються функціональними навіть за умов значних часових відхилень. Інші допускають лише незначні затримки, які не впливають на загальну якість. Існують також сервіси, де затримка призводить до часткового зниження функціональності, як у випадку відеострімінгу з буферизацією. Нарешті, надчутливі сервіси – наприклад, VoIP або відеоконференції – потребують мінімальних затримок, оскільки навіть коливання понад 150–200 мс можуть спричинити порушення синхронізації та сприйняття контенту.

З технічного погляду, важливим критерієм класифікації є тип транспортного протоколу, що використовується на відповідному рівні моделі OSI. Протоколи на основі UDP не передбачають встановлення

з’єднання, контролю цілісності чи підтвердження доставки, що робить їх придатними для застосунків, де пріоритетом є швидкість, а не надійність – зокрема для потокового мультимедіа, DNS-запитів або спрощених файлових сервісів. Натомість TCP забезпечує з’єднання, контроль послідовності та гарантовану доставку, що робить його незамінним для критичних сервісів, таких як передача файлів, електронна пошта або веб-доступ.

Останній аспект класифікації стосується інтенсивності трафіку. Сервіси зі стабільною інтенсивністю, як-от аудіо- та відеоконференції, характеризуються рівномірним потоком даних, що дозволяє прогнозувати навантаження на мережу. У протилежність їм – сервіси з пульсуючим або непередбачуваним трафіком, як у випадку завантаження файлів, де навантаження може змінюватися від нуля до повного заповнення каналу.

Узагальнюючи, комплексна класифікація мережевих сервісів за чутливістю до втрат і затримок, типом транспортного протоколу та характером трафіку є необхідною умовою для побудови адаптивної, ефективної та стійкої мережевої інфраструктури, здатної задовольнити вимоги сучасних цифрових застосунків.

1.3 Механізми керування

1.3.1 Механізми керування інтенсивністю трафіку

У сучасних телекомунікаційних системах механізм «кошика маркерів» (Token Bucket) є одним із ключових інструментів для дозування та регулювання інтенсивності мережевого трафіку. Його функціонування базується на простому принципі: кожен пакет даних, що надходить до мережі, може бути переданий лише за умови наявності достатньої кількості

маркерів у кошику. Кількість маркерів, необхідна для передачі, відповідає розміру пакета в байтах. Максимальна кількість маркерів, яку може вміщувати кошик, визначається узгодженим розміром сплеску (Committed Burst Size, B_c).

У стандартній реалізації, коли розмір пакета перевищує кількість доступних маркерів, передача блокується, а пакет відкидається. Це обумовлено тим, що розширений розмір сплеску (*BE*) дорівнює *Bc*, і механізм не передбачає можливості тимчасового перевищення ліміту. Такий підхід, хоча й простий, має суттєві обмеження, зокрема схильність до явища «відкидання хвоста» (tail-drop), що негативно впливає на якість обслуговування.

Для подолання цих обмежень розроблено модифіковану версію кошика маркерів, яка допускає екстрене перевищення розміру сплеску (тобто $B_E > Bc$). У цьому варіанті допускається «займання» маркерів понад наявний ліміт, що дозволяє передавати пакети навіть за умов тимчасового дефіциту ресурсів. Для опису поведінки такого механізму вводяться поняття поточного боргу (D_A) – кількості маркерів, зайнятих у даний момент – та накопиченого боргу (D_c), який є сумою всіх D_A з моменту останнього відкидання пакета.

Після кожної передачі пакета, що потребує займаних маркерів, значення D_A зростає, а D_c накопичується. Якщо відбувається відкидання пакета, D_c обнуляється, тоді як D_A зберігається. У випадку, коли D_A перевищує розширений поріг, подальші пакети блокуються до моменту, коли борг буде погашено за рахунок надходження нових маркерів. Таким чином, механізм забезпечує поступове зниження навантаження, що наближає його до поведінки алгоритму довільного раннього виявлення (Random Early Detection, RED).

У момент надходження нового пакета система порівнює значення D_c з B_E. Якщо борг перевищує допустимий поріг, пакет відкидається, а D_c

обнуляється. Інакше – пакет передається, при цьому D_A збільшується на відповідну кількість маркерів, а D_c оновлюється відповідно до нового значення D_A. Важливо зазначити, що сам факт відкидання пакета не призводить до зменшення кількості маркерів у кошику, тобто не змінює межі сплеску або інтенсивності трафіку.

Узгоджена швидкість передачі даних (Committed Information Rate, CIR) вимірюється в байтах за секунду. Поточний розмір сплеску відображається у лічильнику, який може перевищувати Bc, досягаючи значення $B_c + D_A$. Лічильник оновлюється при кожному надходженні нового пакета.

Реалізація механізму узгодження швидкості доступу (Committed Access Rate, CAR) накладає певні обмеження: мінімальний крок збільшення інтенсивності становить 8 Кбіт/с, а найменше значення B_c і B_E – 8000 байт. Крім того, B_E завжди дорівнює або перевищує B_c.

Механізм «кошика маркерів» є фундаментальним для реалізації двох стратегій керування трафіком – шейпінгу (traffic shaping) та полісінгу (traffic policing). Шейпінг передбачає буферизацію пакетів у комутаційному пристрої з подальшою їх передачею з постійною, заздалегідь визначеною швидкістю. Це дозволяє логічно розділити фізичний канал на кілька потоків і оптимізувати використання смуги пропускання, особливо в умовах наявності вузьких місць.

Полісінг, навпаки, реалізує жорстке обмеження: пакети, що перевищують встановлену інтенсивність, відкидаються. Такий підхід може бути неефективним у поєднанні з протоколом TCP, який у разі втрат пакетів знижує швидкість передачі, а потім знову її підвищує, що призводить до циклічної перевантаженості мережі.

Узагальнюючи, механізм «кошика маркерів» з можливістю екстреного розширення є більш гнучким і адаптивним інструментом для

керування трафіком, здатним забезпечити баланс між продуктивністю, стабільністю та якістю обслуговування в сучасних мережах.

1.3.2 Механізми забезпечення гарантованої якості обслуговування

У сучасних мережах передачі даних забезпечення якості обслуговування (Quality of Service, QoS) реалізується через три концептуально відмінні моделі: Best Effort Service (найкраща можлива доставка), Integrated Service (інтегральна модель) та Differentiated Service (диференційована модель) [3, 10, 11]. Кожна з них відображає різний рівень складності, гнучкості та масштабованості у керуванні мережевими ресурсами.

1. Модель Best Effort є базовою реалізацією маршрутизації трафіку в IP-мережах, яка не передбачає жодного механізму пріоритезації або резервування ресурсів. Передача даних у цьому випадку здійснюється за принципом «зробити все можливе», тобто система намагається доставити пакети, не надаючи жодних гарантій щодо затримки, пропускної здатності чи цілісності. У практичному застосуванні ця модель є прийнятною для сервісів, що не є критичними до часу доставки (наприклад, FTP або HTTP), однак виявляється неефективною для трафіку, чутливого до затримок і джитеру – зокрема, для голосових або відео сервісів.

Особливістю Best Effort є її неспроможність адаптуватися до різких змін навантаження. У разі сплесків трафіку, навіть за наявності резервів пропускної здатності, можуть виникати перевантаження, що призводить до втрат пакетів і деградації якості обслуговування. Ці обмеження стали підґрунтям для розробки більш складних моделей QoS, здатних забезпечити контрольовану поведінку мережі.

2. Інтегральна модель (Integrated Service, IntServ). Модель інтегрованого обслуговування, запропонована в RFC 1633 (1994), передбачає наскрізне резервування ресурсів між відправником і

одержувачем за схемою «точка-точка». Основним інструментом реалізації цієї моделі є протокол RSVP (Resource ReSerVation Protocol), розроблений IETF [RFC 2205]. RSVP використовує сигнальний механізм для конфігурації маршруту, забезпечуючи відповідність вимогам QoS на кожному проміжному вузлі. Ініціатором резервування виступає одержувач, який надсилає запит до відправника, а кожен вузол на шляху перевіряє наявність необхідних ресурсів та авторизацію користувача.

У рамках IntServ визначено два класи сервісу:

- клас контрольованого навантаження (controlled-load) [RFC 2211], який моделює поведінку мережі за умов помірного завантаження;

- клас гарантованого обслуговування (guaranteed QoS) [RFC 2212], що забезпечує жорсткі гарантії щодо затримки та пропускної здатності.

Для реалізації планування обслуговування в IntServ використовуються алгоритми Weighted Fair Queuing (WFQ) та Weighted Random Early Detection (WRED). Зокрема, WFQ є типовим для інтерфейсів з низькою пропускною здатністю. У цьому випадку трафік розподіляється на потоки, кожен з яких обслуговується окремою чергою. Формування потоків здійснюється на основі IP-адрес джерела і призначення, портів, а також поля ToS (Type of Service) IP-заголовка. Загальна кількість черг обмежується 256, а їх обслуговування відбувається за циклічним принципом (round-robin), з урахуванням вагових коефіцієнтів: чим вищий пріоритет потоку, тим менша частка пропускної здатності йому надається (наприклад, для сервісу telnet). Ємність черг також регламентується: максимальна – 4096 пакетів, типова – 64.

У межах моделі інтегрованого обслуговування (IntServ) важливу роль відіграє алгоритм Weighted Random Early Detection (WRED), який реалізує механізм статистичного відкидання пакетів на основі оцінки середньої довжини черги. Його застосування є доцільним у тих випадках, коли обробка трафіку здійснюється незалежно на кожному транзитному вузлі,

без централізованого керування. Такий підхід дозволяє ефективно реагувати на перевантаження мережі та підтримувати стабільність її функціонування.

Принцип дії WRED полягає в тому, що рішення про прийняття або відкидання пакета приймається на основі статистичної оцінки середньої довжини черги у момент його надходження. Якщо це значення менше мінімального порогового значення, пакет безумовно додається до черги. У випадку, коли середня довжина черги перебуває в межах між мінімальним і максимальним порогами, пакет може бути або прийнятий, або відкинутий – залежно від встановленої ймовірності, яка визначається типом трафіку. Якщо ж середня довжина перевищує максимальний поріг, пакет відкидається безумовно.

Особливістю WRED є його здатність диференційовано реагувати на обсяг трафіку, що надходить від різних джерел. Зокрема, пакети з великих потоків мають вищу ймовірність бути відкинутими, ніж пакети з малих потоків. Така поведінка є особливо ефективною для адаптивних протоколів, зокрема TCP, які реагують на втрати пакетів зниженням швидкості передачі. Таким чином, WRED сприяє саморегуляції трафіку, запобігаючи перевантаженню мережі та забезпечуючи більш справедливий розподіл ресурсів.

У практичному застосуванні WRED набув широкого поширення в маршрутизаторах, де він використовується як базовий механізм керування чергами на вихідних інтерфейсах. Його ефективність особливо проявляється в умовах високої динаміки трафіку, коли необхідно забезпечити адаптивну реакцію на зміну навантаження без втрати загальної продуктивності. Важливо зазначити, що трафік, який не належить до IP-класу або має нульовий пріоритет, обробляється з підвищеною ймовірністю втрати, що додатково стимулює оптимізацію поведінки відправника.

Таким чином, алгоритм WRED є ключовим елементом у реалізації QoS в інтегрованих мережах, забезпечуючи баланс між ефективністю

використання ресурсів і гнучкістю адаптації до змінних умов передачі даних.

Інтегральна реалізація концепції QoS (Quality of Service) на мережевому рівні (L3) ґрунтується на архітектурних принципах, що значною мірою наслідують підходи, закладені в технології ATM (Asynchronous Transfer Mode) на канальному рівні (L2). Зокрема, модель IntServ (Integrated Services) запозичує ідею сервісних класів, які дозволяють забезпечити детерміновану якість обслуговування для різних типів трафіку, орієнтуючись на параметри затримки, джитеру, втрат та пропускної здатності.

У контексті практичної реалізації, для каналів із пропускною здатністю до 2048 кбіт/с (наприклад, інтерфейси типу E1) у середовищі CISCO IOS за замовчуванням застосовується стратегія обробки черг Weighted Fair Queuing (WFQ). Цей механізм дозволяє розподіляти ресурси між потоками трафіку пропорційно до їхньої ваги, забезпечуючи справедливий доступ до мережевих ресурсів навіть за умов обмеженої пропускної здатності.

Модель IntServ, у відповідності до логіки ATM, визначає 4 класи (див. табл. 1.1).

Інтегрована модель забезпечення якості обслуговування (IntServ) базується на чітко визначеній архітектурі, що включає низку функціональних компонентів, кожен з яких виконує специфічну роль у процесі резервування та обробки трафіку.

До основних елементів належать:

- модуль резервування ресурсів (Flow Resource Reservation), який координує взаємодію між усіма іншими компонентами;

- модуль контролю доступу (Flow Admission Control), що відповідає за перевірку доступності ресурсів;

- класифікатор пакетів (Packet Classifier), який визначає параметри QoS для кожного окремого пакета;

- диспетчер черг (Packet Scheduler), що реалізує механізм черговості обслуговування відповідно до заданих характеристик потоку.

Таблиця 1.1 – Класи моделі IntServ

Клас QoS	Аналог ATM	Характеристики	Призначення
QoS Class 1	Клас послуг A (CBR)	Характеристики, що збігаються з виділеним цифровим каналом «точка-точка». Забезпечує жорсткі гарантії.	Критичні послуги реального часу.
QoS Class 2	Клас послуг B (VBR)	Прийнятний для перегляду аудіо та відео при відеоконференціях або передачі мультимедіа.	Сервіси реального часу з менш жорсткими вимогами.
QoS Class 3	Клас послуг 3 (ABR)	Призначений для передачі, орієнтованої на з'єднання (наприклад, за допомогою Frame Relay).	Трафік, чутливий до втрат, але менш чутливий до затримок.
QoS Class 4	Клас послуг 4 (UBR)	Аналог IP-передачі в умовах Best Efforts. Не надає жодних гарантій.	Ненадійні, некритичні сервіси.

Ключовим елементом інтегрованої моделі є протокол RSVP (Resource ReSerVation Protocol), який виконує функцію сигналізації та узгодження параметрів QoS між усіма маршрутизаторами на шляху передачі даних.

Ініціація процесу резервування завжди здійснюється одержувачем, який надсилає запит до хоста-відправника. Запити поширюються в одному напрямку, доки не відбувається їх об'єднання з іншими запитами на ті ж ресурси, що особливо ефективно в контексті багатоадресної (multicast) передачі.

Успішне резервування можливе лише за умови підтримки RSVP усіма маршрутизаторами на маршруті. Перед наданням ресурсів запит проходить подвійну перевірку:

- Policy Control – визначає правомочність користувача на резервування;

- Admission Control – оцінює наявність фізичних ресурсів для забезпечення заданого рівня QoS.

У разі позитивного результату обидві перевірки активують програму-демон RSVP, яка вносить відповідні параметри до класифікатора пакетів та диспетчера черг. Це дозволяє забезпечити індивідуалізовану обробку трафіку з урахуванням його пріоритету, затримки та інших характеристик, критичних для сервісів реального часу.

У межах реалізації алгоритму Weighted Fair Queuing (WFQ) важливу роль відіграє поле ToS (Type of Service) в IP-заголовку. Якщо відповідні біти цього поля встановлені, WFQ адаптує обробку трафіку таким чином, щоб черги з вищим пріоритетом обслуговувалися швидше. Це досягається шляхом коригування вагового коефіцієнта, який є обернено пропорційним до рівня пріоритету: чим вищий пріоритет потоку, тим менша вага черги, а отже – більша частка пропускної здатності, що їй надається.

Такий механізм дозволяє не лише забезпечити необхідний рівень QoS для критичних сервісів, але й ефективно протидіяти перевантаженню каналу передачі даних, зберігаючи стабільність мережі навіть за умов високої динаміки трафіку.

Попри високий рівень гарантій, який забезпечує модель IntServ, її застосування в масштабних мережах супроводжується значними накладними витратами, зумовленими необхідністю наскрізного резервування ресурсів на кожному проміжному вузлі. Це обмежує її масштабованість і ускладнює впровадження в умовах великої кількості користувачів та потоків.

3. Модель диференційованого обслуговування. З метою подолання обмежень, притаманних інтегральній моделі QoS, зокрема її низької масштабованості та високих накладних витрат, організація IETF запропонувала альтернативний підхід – модель диференційованого обслуговування (Differentiated Services, DiffServ). На відміну від IntServ, ця модель не потребує централізованого резервування ресурсів і орієнтована на гнучке керування трафіком у великих пакетних мережах.

DiffServ реалізує класифікацію та обробку трафіку на основі покрокової поведінки (Per-Hop Behavior, PHB), яка визначається на кожному окремому вузлі маршруту. Такий підхід дозволяє досягти високої ефективності та масштабованості, оскільки обробка пакетів здійснюється локально, без потреби в глобальній сигналізації. Поведінка PHB задається провайдером послуг і визначає, як саме маршрутизатор має обробляти пакети певного класу.

Класифікація трафіку в DiffServ здійснюється на основі 6-бітового поля DSCP (Differentiated Services Code Point), яке розміщується в полі DS IP-заголовка. Це поле фактично замінює застаріле поле ToS, визначене в RFC 791. Запис значення до DSCP виконується на граничних маршрутизаторах при вході трафіку до мережі, після чого обробка здійснюється відповідно до визначеного PHB. Стандарти RFC 2474 та RFC 2475 регламентують набір допустимих значень DSCP та відповідні механізми обробки черг.

Кожна мережа, що підтримує DiffServ, повинна реалізовувати щонайменше два класи PHB. Найвищий рівень обслуговування забезпечується класом Express Forwarding (EF), який гарантує мінімальні затримки, низький джиттер та фіксовану смугу пропускання. У цьому класі можливе використання RSVP-протоколу для додаткового резервування ресурсів, що дозволяє досягти параметрів, близьких до сервісів реального часу. DiffServ також демонструє високу адаптивність у середовищах із обмеженою пропускною здатністю, що робить її придатною для широкого спектра застосувань.

Історично поле ToS IP-заголовка мало 4 біти, що теоретично дозволяло визначити 16 класів обслуговування. На практиці використовуються лише п’ять основних кодів:

- 1000 – мінімальна затримка;
- 0100 – максимальна пропускна здатність;
- 0010 – максимальна надійність;
- 0001 – мінімальна вартість;
- 0000 – звичайний (базовий) сервіс.

У межах DiffServ можуть застосовуватись різні механізми обробки черг, зокрема WRED, що дозволяє адаптивно керувати трафіком залежно від його класу та поточного стану черг. Для мінімізації джитеру можливе попереднє резервування пропускної здатності під час встановлення каналу, наприклад, у мережах ATM.

Окрему увагу заслуговує технологія Cisco Content Networking, розроблена для забезпечення QoS з урахуванням вмісту переданих даних. На відміну від традиційних підходів, де пріоритет визначався лише за IP-адресою або протоколом, сучасні мережі потребують глибшої класифікації. Наприклад, в межах одного IP-з’єднання можуть одночасно передаватися відео, веб-запити та інші типи трафіку, кожен із яких має різні вимоги до QoS. Це зумовлює необхідність аналізу заголовків на рівнях L3–L7,

включаючи URL, MIME-типи та HTTP-запити. Ускладнення виникає через динамічне призначення портів, що ускладнює ідентифікацію застосунків, однак технології глибокої інспекції (Deep Packet Inspection) дозволяють частково вирішити цю проблему.

Таким чином, модель DiffServ у поєднанні з сучасними технологіями класифікації трафіку забезпечує гнучке, масштабоване та ефективне керування якістю обслуговування в IP-мережах, що відповідає вимогам сучасних мультимедійних і критичних до затримок сервісів.

У сучасних IP-мережах, де традиційні методи класифікації трафіку за адресами та портами виявляються недостатніми, особливо в умовах динамічного використання портів і мультисервісного навантаження, компанією Cisco було розроблено спеціалізований механізм – Network Based Application Recognition (NBAR). Цей інструмент дозволяє здійснювати глибоку класифікацію трафіку на основі аналізу вмісту пакетів, що є критично важливим для реалізації QoS у гетерогенному середовищі.

NBAR функціонує лише для трафіку, який маршрутизується з використанням технології Cisco Express Forwarding (CEF), що забезпечує високопродуктивну маршрутизацію на основі попередньо побудованих таблиць. У межах NBAR здійснюється аналіз до 512 байт заголовків, включаючи рівні L2, L3, L4, а також HTTP-заголовки та URL (до 400 байт). Це дозволяє класифікувати HTTP-трафік не лише за IP-адресами та портами, а й за mime-типами або GET-запитами, що особливо актуально для розмежування мультимедійного, веб-орієнтованого та файлового трафіку в межах одного протоколу.

Кількість одночасно підтримуваних URL або mime-типів обмежується 24, що зумовлено апаратними ресурсами маршрутизатора. Аналіз URL-заголовків також дозволяє виявляти несанкціоновану передачу великих файлів, що сприяє зменшенню ризику зловживання мережевими ресурсами. Окрім TCP та UDP, NBAR підтримує класифікацію широкого спектра

протоколів, зокрема NETBIOS, NTP, POP3/SPOP3, SFTP, IPSec, RSVP, ICMP, IP-in-IP, Kerberos, IMAP/SIMAP, HTTPS, SNMP, SSH, X-Windows, L2TP, LDAP, SOCKS, а також низки маршрутних протоколів [10].

Гнучкість NBAR забезпечується підтримкою Packet Description Language Module (PDLM) – модуля, що дозволяє завантажувати шаблони класифікації безпосередньо до маршрутизатора. Завантаження PDLM може здійснюватися з консолі або іншими засобами, а список підтримуваних протоколів постійно оновлюється. Важливо підкреслити, що NBAR сам по собі не реалізує механізми QoS, а лише виконує класифікацію трафіку, після чого до визначених класів застосовуються відповідні політики обслуговування (наприклад, WFQ, WRED або CBWFQ).

З технічного погляду, класифікація 500 потоків потребує приблизно 1 МБ оперативної пам’яті. На маршрутизаторах нижчого класу (наприклад, серії Cisco 2600) використання NBAR може споживати до 15% обчислювальних ресурсів процесора, що необхідно враховувати при проектуванні системи QoS.

На канальному рівні (L2) повноцінне забезпечення QoS реалізується за допомогою протоколу MPLS (Multiprotocol Label Switching). Хоча його початкове призначення полягало у формуванні VPN та прискоренні комутації, MPLS також дозволяє класифікувати трафік і забезпечувати гарантовану якість обслуговування. Присвоєння міток здійснюється на крайових маршрутизаторах (Label Edge Routers, LER), де виконується повний аналіз заголовків. Подальша маршрутизація пакета базується виключно на мітці, що значно знижує навантаження на маршрутизатори в порівнянні з традиційною IP-адресацією.

Попри активний розвиток технологій забезпечення якості обслуговування (QoS), широке впровадження протоколів MPLS та IEEE 802.1Q у локальних мережах залишається обмеженим. Основною причиною цього є недостатня підтримка відповідних функцій з боку мережевого

обладнання, особливо в сегменті малих і середніх підприємств. Водночас, стандарт IEEE 802.1Q, який визначає структуру маркованого Ethernet-кадру, відіграє ключову роль у реалізації механізмів пріоритезації трафіку в VLAN-сегментованих середовищах.

Застосування IEEE 802.1Q дозволяє інтегрувати механізми QoS на канальному рівні (L2) з політиками, що реалізуються на мережевому рівні (L3), забезпечуючи наскрізне керування якістю обслуговування. Це особливо важливо в умовах мультисервісного навантаження, коли необхідно забезпечити диференційовану обробку трафіку залежно від його типу, пріоритету та вимог до затримки.

Маркування кадру в межах IEEE 802.1Q здійснюється шляхом вставки спеціальної мітки, яка містить такі поля:

- TPID (Tagged Protocol Identifier) – значення 0x8100 (2 октети), що вказує на наявність VLAN-тегу та визначає спосіб обробки кадру;

- User Priority – 3 біти, призначені для встановлення рівня пріоритету кадру;

- CFI (Canonical Format Indicator) – 1 біт, який регулює формат MAC-адресації;

- VLAN Identifier – 12 біт, що ідентифікує віртуальну локальну мережу.

Після додавання VLAN-мітки до Ethernet-кадру відповідно до стандарту IEEE 802.1Q виконується перерахунок контрольної суми (FCS), що забезпечує цілісність переданих даних. Така структура дозволяє комутаторам ідентифікувати пріоритет кадру та здійснювати його обробку відповідно до заданих політик, що є основою для реалізації механізмів QoS на канальному рівні (L2). Зокрема, поле User Priority у VLAN-мітці дозволяє класифікувати трафік за рівнем обслуговування, що критично важливо для підтримки сервісів із різними вимогами до затримки, пропускної здатності та надійності.

Для розширення функціональності QoS у середовищі IEEE 802.3 було запропоновано механізм Subnet Bandwidth Manager (SBM), який інтегрується з протоколом RSVP. SBM забезпечує підтримку множинних черг на основі кодів пріоритету, дозволяючи реалізувати диференційовану обробку трафіку безпосередньо на канальному рівні. Це особливо актуально для сервісів реального часу, таких як VoIP або потокове відео, де необхідна гарантована смуга пропускання, мінімальні затримки та низький рівень джитеру.

Попри наявність технічних засобів, широке впровадження IEEE 802.1Q та MPLS у локальних мережах наразі залишається обмеженим. Основною причиною є недостатня підтримка відповідних функцій у мережевому обладнанні, особливо в сегменті недорогих комутаторів і маршрутизаторів. Водночас, потенціал цих технологій у контексті QoS є значним: вони формують інфраструктурну основу для побудови адаптивних, масштабованих і керованих мережевих середовищ, здатних задовольнити вимоги сучасних інформаційних сервісів.

Відповідно до специфікації IEEE 802.1D (1998), у локальних мережах визначаються два типи пріоритетів:

- Access Priority – використовується комутаторами для визначення порядку передачі кадрів у середовищі змагання за доступ;

- User Priority – формується на рівні MAC-підрівня та вбудовується у заголовок кадру для подальшої обробки іншими мережевими пристроями.

Нульове значення User Priority відповідає найнижчому рівню пріоритету. У деяких випадках Access Priority та User Priority можуть збігатися, що спрощує обробку черг. Комутатори здатні підтримувати одночасну роботу з кількома чергами, використовуючи класи трафіку для визначення відносного пріоритету: кадри з високим класом передаються першочергово, а кадри з низьким – лише за умови відсутності високопріоритетних запитів.

Одним із найбільш перспективних протоколів для забезпечення QoS, який наразі не має широкої реалізації в локальних мережах, є MPLS (Multiprotocol Label Switching). Хоча його первинне призначення полягало у формуванні VPN та прискоренні комутації, сучасне застосування охоплює класифікацію трафіку та забезпечення QoS. На крайових маршрутизаторах LER (Label Edge Router) здійснюється повний аналіз заголовка пакета, після чого йому присвоюється мітка фіксованої довжини. Подальша маршрутизація пакета базується виключно на цій мітці, що дозволяє мінімізувати витрати на комутацію порівняно з традиційною IP-адресацією.

У складі сучасних VPN-мереж часто використовуються ділянки ATM та IP (PPP або Ethernet). Завдяки низькій вартості ATM-каналів, з'єднання елементів MPLS здійснюється через постійні віртуальні канали PVC ATM, де поле VPI виконує функцію мітки. Вибір класу ATM визначається політикою сервіс-провайдерів, хоча для MPLS типовими є класи CBR (Constant Bit Rate) та VBR (Variable Bit Rate), які забезпечують відповідний рівень QoS.

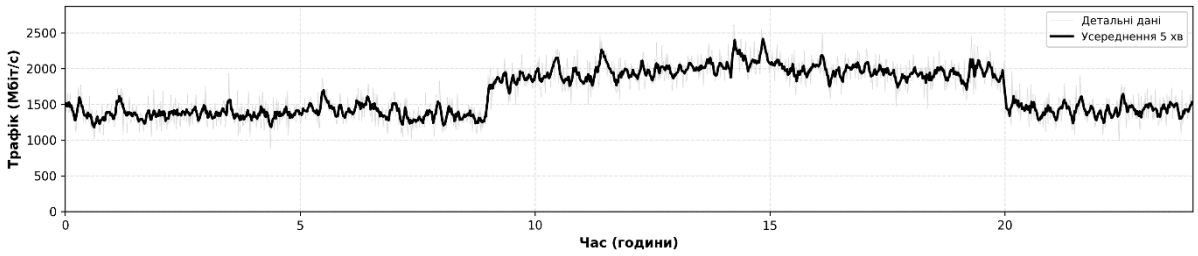
До ключових переваг диференційованої моделі QoS, що реалізується через DiffServ та MPLS, належать: висока масштабованість; відсутність службового (overhead) трафіку; можливість інкрементального розгортання без необхідності глобальної перебудови мережевої інфраструктури.

1.4 Аналіз підходів для автоматизованого моніторингу та прогнозування збоїв у телекомунікаційних мережах

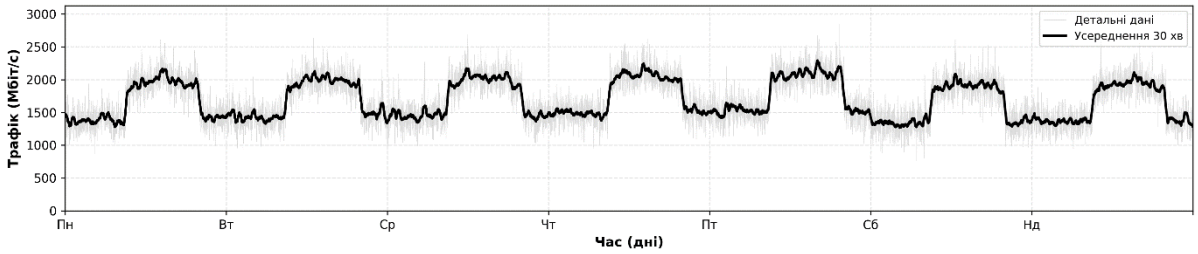
1.4.1 Самоподібність мережевого трафіку як ключова концепція для прогнозування збоїв

Самоподібність мережевого трафіку є фундаментальною властивістю, що характеризує повторюваність навантаження при різних масштабах часової агрегації. Це означає, що графічні залежності щільності потоку

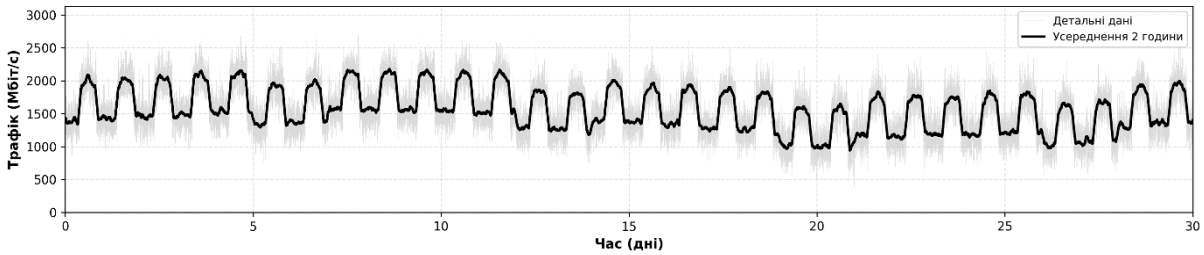
інформації від часу – незалежно від вибраної одиниці виміру (секунда, хвилина, година, доба тощо) – демонструють подібну структуру, зберігаючи характерні особливості незалежно від рівня деталізації (рис. 1.2).



а)



б)



в)

Рисунок 1.2 – Статистика по трафіку:
а) за за 1 день (усереднення 5 хв.); б) за 1 тиждень (усереднення 30 хв.);
в) за 1 місяць (усереднення 2 години)

Математично самоподібність можна описати через властивість самокореляції: якщо набір значень функції, що моделює трафік, поділити на групи, то суми значень усередині кожної групи будуть підпорядковуватись тій самій самоподібній функції, що й вихідний процес. Така поведінка

вказує на наявність довготривалої залежності (long-range dependence), яка не зникає навіть при масштабуванні часових інтервалів.

Феномен самоподібності мережевого трафіку, вперше емпірично зафіксований у середовищі Ethernet [1, 10], виявив фрактальну структуру навантаження, яка зберігається незалежно від масштабу часової агрегації. Згодом аналогічні властивості були виявлені й у глобальних інформаційних системах, зокрема в трафіку WWW [5], що підтвердило універсальність цього явища в різних типах мережевих середовищ. Самоподібність проявляється у вигляді повторюваних патернів навантаження, які залишаються стабільними при зміні часових інтервалів – від секунд до годин і діб.

З наукової точки зору, така поведінка трафіку суперечить класичним моделям, що базуються на припущенні про пуасонівський розподіл, який передбачає незалежність подій у часі. Самоподібні процеси, навпаки, демонструють довготривалу залежність (long-range dependence), що вимагає застосування спеціалізованих математичних апаратів для їх адекватного моделювання. Зокрема, до таких інструментів належать фрактальний аналіз, моделі дробового броунівського руху, а також методи оцінки параметра Херста, який дозволяє кількісно оцінити рівень самоподібності.

Практичне значення самоподібності полягає в її впливі на ефективність керування мережевими ресурсами. Однією з ключових проблем сучасного мережевого планування є невідповідність між реальними характеристиками трафіку та його статистичними оцінками. У більшості випадків інфраструктура проєктується на основі усереднених значень, що репрезентують агрегований трафік множини потоків із різною щільністю. Такий підхід, хоча і зручний для аналітичного моделювання, не враховує локалізовані пікові навантаження, які є характерними для TCP/IP-середовищ.

TCP/IP-протоколи схильні до прояву короткочасних, але інтенсивних викидів навантаження, що можуть спричинити втрати пакетів навіть за умов, коли сумарна потреба не перевищує граничну пропускну здатність. Це явище пояснюється автокореляцією трафіку – властивістю, за якої поточне навантаження залежить від попередніх станів системи. Як наслідок, класичні моделі, що не враховують фрактальну природу трафіку, виявляються недостатніми для точного прогнозування, балансування навантаження та забезпечення QoS.

Таким чином, у контексті побудови QoS-орієнтованих мережевих архітектур, врахування самоподібності трафіку є не лише теоретично обґрунтованим, але й практично необхідним. Це дозволяє перейти від спрощених моделей до більш точних і адаптивних механізмів керування, здатних забезпечити стабільну якість обслуговування в умовах високої динаміки мережевого середовища [8].

Для протоколу TCP було розроблено механізм запобігання перевантаженню, який базується на процедурі підтвердження доставки (ACK). Після прийняття кожного пакета кінцева точка надсилає повідомлення з номером наступного очікуваного сегмента. Щоб уникнути простоїв, спричинених очікуванням підтвердження, TCP використовує механізм «вікон» – відправник надсилає одразу кілька пакетів, кількість яких визначається розміром вікна. Копії цих пакетів зберігаються в черзі повторної передачі. Якщо протягом динамічно обчислюваного інтервалу не надходить підтвердження, пакет передається повторно.

Фрактальна поведінка TCP-протоколу особливо яскраво проявляється в його реалізаціях Reno та Vegas. У TCP Reno керування розміром вікна здійснюється з метою досягнення максимальної пропускної здатності при мінімальних витратах на обробку підтверджень. Алгоритм має дві фази: повільний старт (slow start) та уникнення перевантаження (congestion avoidance). Перехід між фазами визначається параметром

порогового значення ssth(t). У фазі повільного старту, коли поточне значення розміру вікна cwnd(t) менше ssth(t), вікно збільшується лінійно – на одну одиницю за кожне підтвердження. У фазі уникнення перевантаження (cwnd(t) ≥ ssth(t)) приріст вікна відбувається за оберненою залежністю – на величину, зворотну поточному значенню cwnd(t).

Такий підхід дозволяє адаптувати швидкість передачі до реального стану мережі, знижуючи ймовірність перевантаження та втрат пакетів. Водночас, циклічна динаміка зміни розміру вікна створює умови для фракталізації трафіку, що підтверджується емпіричними дослідженнями та математичним моделюванням. У результаті, навіть у стабільних умовах, TCP-трафік демонструє властивості самоподібності, що необхідно враховувати при побудові QoS-орієнтованих мережевих архітектур.

У протоколі TCP ключовим механізмом забезпечення надійності передачі даних та запобігання перевантаженню мережі є динамічне регулювання розміру вікна перевантаження (congestion window, cwnd). Поведінка цього параметра залежить від способу виявлення втрати пакетів – через таймаут або за допомогою алгоритму швидкої повторної передачі (Fast Retransmit).

У разі втрати пакета, виявленої за таймаутом, TCP Reno скидає значення cwnd(t) до одиниці, а порогове значення ssth(t) встановлюється як половина розміру вікна, досягнутого на момент збою. Після цього протокол повертається до фази повільного старту, повторно нарощуючи розмір вікна. Такий підхід дозволяє швидко зменшити навантаження на мережу, однак супроводжується значною деградацією пропускної здатності.

Інша ситуація виникає, коли втрата пакета детектується за допомогою Fast Retransmit – тобто, коли відправник отримує кілька дубльованих підтверджень (ACK) з однаковим номером очікуваного сегмента. У цьому випадку значення cwnd(t) та ssth(t) оновлюються інакше: обидва параметри встановлюються на половину поточного розміру вікна. Після цього TCP

Reno переходить у фазу швидкого відновлення (Fast Recovery), у якій cwnd(t) збільшується на один сегмент за кожне дубльоване підтвердження. Коли надходить нове, не дубльоване підтвердження для повторно надісланого пакета, cwnd(t) встановлюється рівним ssth(t), і протокол повертається до фази уникнення перевантаження.

Альтернативний підхід реалізовано в TCP Vegas – протоколі, запропонованому у 1994 році Лоуренсом Бракмо, Шоном О'Меллі та Ларрі Петерсоном. На відміну від Reno, TCP Vegas не покладається виключно на втрати пакетів як індикатор перевантаження. Натомість він використовує проактивний механізм моніторингу часу обігу пакета (Round-Trip Time, RTT), що дозволяє виявляти ознаки перевантаження до фактичної втрати даних.

У TCP Vegas поведінка системи визначається параметрами α, β, мінімальним RTT та поточним RTT. Відправник обчислює різницю між очікуваною та реальною пропускною здатністю, порівнюючи відношення поточного розміру вікна до мінімального та поточного RTT. Якщо ця різниця менша за α, система вважає, що мережа недовантажена, і збільшує розмір вікна. Якщо різниця потрапляє в інтервал [α, β], вікно залишається незмінним. Якщо ж вона перевищує β, це інтерпретується як ознака наближення до перевантаження, і розмір вікна зменшується.

Фаза повільного старту в TCP Vegas подібна до Reno, однак має важливу відмінність: збільшення cwnd(t) на одиницю відбувається лише після успішного проходження двох пакетів. Такий підхід забезпечує більш плавне нарощування навантаження. У фазі уникнення перевантаження Vegas не змінює розмір вікна, доки обчислена різниця залишається в межах [α, β]. Якщо RTT зростає, система зменшує cwnd(t); якщо RTT зменшується – вікно знову розширюється. Таким чином, TCP Vegas прагне підтримувати розмір вікна на рівні, що відповідає оптимальному використанню пропускної здатності без перевантаження мережі.

У підсумку, TCP Vegas демонструє більш адаптивну та передбачувану поведінку порівняно з Reno, що робить його перспективною альтернативою для середовищ із високими вимогами до стабільності та ефективності QoS.

Результати експериментальних досліджень засвідчили, що реалізація протоколу TCP Reno демонструє незадовільну ефективність уже при відносно невеликій кількості клієнтів. Зокрема, при навантаженні в 2025 клієнтів спостерігалися значні втрати пакетів, а характерні стрибки щільності трафіку фіксувалися навіть за умов мінімального навантаження (8 клієнтів). Втрати починалися ще до досягнення зони насичення каналу і становили близько 0,26%. У другій конфігурації мережі, при моделюванні трафіку від 50 клієнтів, рівень втрат сягав 5%. Для каналу з пропускною здатністю 155 Мбіт/с це відповідало втратам у 403 Кбіт/с, тоді як при швидкості 622 Мбіт/с втрати зростали до 31 Мбіт/с.

Особливо критичною є ситуація при високій латентності з’єднання. Наприклад, при RTT = 100 мс і розмірі буфера 50 Мб, у разі втрати пакета, навіть за умови успішного спрацювання механізму швидкої повторної передачі, для відновлення продуктивності потрібно близько 2500 RTT. Це відповідає приблизно 3,5 хвилини, протягом яких канал використовується неефективно. Додатковим ускладненням є синхронність дій потоків: одночасне збільшення розміру вікон призводить до масової втрати пакетів у моменти пікового навантаження, що, у свою чергу, спричиняє синхронний відкат усіх потоків.

На відміну від Reno, TCP Vegas демонструє значно стабільнішу поведінку. Незначні втрати пакетів у цій реалізації не мають істотного впливу на продуктивність. В обох випадках – як для Reno, так і для Vegas – було зафіксовано виражену автокореляцію трафіку, що свідчить про наявність самоподібності.

Для кількісної оцінки ступеня самоподібності використовувалися два показники: коефіцієнт варіації (cov) та параметр Херста (H). Показник cov визначається як відношення середнього квадратичного відхилення щільності трафіку до його середнього значення і є мірою інтенсивності пікових викидів. Він підтверджує справедливість теореми про «статистичне згладжування» об'єднаних потоків. Параметр Херста, що змінюється в межах [0,5; 1], відображає ступінь залежності поточного стану трафіку від його попередніх значень: чим ближче H до 1, тим вищий рівень автокореляції.

У ході експериментів моделювалися потоки даних, що підпорядковуються пуасонівському розподілу, тобто з рівномірним розподілом затримок між відправленням пакетів. Теоретичне значення cov для такого трафіку легко обчислюється, однак реальні вимірювання показали суттєві відхилення. Зокрема, при навантаженні від 60 клієнтів у першій конфігурації значення cov перевищувало теоретичне на 300% для TCP Reno і на 40% – для TCP Vegas. У другій конфігурації ці відхилення становили 642% і 457% відповідно [2].

Значення параметра Херста в обох реалізаціях TCP наближалося до 1 уже при навантаженні від 40 клієнтів у першій конфігурації, що свідчить про наявність довготривалої залежності. Відповідно до теорії пуасонівських потоків, у відсутності самоподібності значення H має дорівнювати 0,5.

Таким чином, результати експериментів підтверджують, що трафік, модульований стеком протоколів TCP/IP, набуває властивостей самоподібності. Це, у свою чергу, призводить до автокореляції потоків, перевантаження буферів комутаторів і маршрутизаторів, а отже – до втрат пакетів і зниження ефективності використання мережевих ресурсів.

Проблема самоподібності трафіку має як протокольну, так і структурну природу: вона зумовлена як недосконалістю механізмів керування перевантаженням, так і характеристиками самого трафіку.

Впровадження інтелектуальних технологій і адаптивних методів у мережеві протоколи та комутаційне обладнання є перспективним напрямом для підвищення ефективності керування трафіком, зменшення затримок і втрат пакетів [2].

1.4.2 Обгрунтування підходу для керування трафіком

Зростання швидкості передачі даних у сучасних мережах супроводжується формуванням великих обсягів трафіку, що потребує ефективного аналізу для прийняття рішень щодо його оптимального керування. У цьому контексті актуальним є розгляд існуючих систем аналізу даних, які можуть бути адаптовані до задач мережевого моніторингу, прогнозування та класифікації.

1. Статистичні програмні засоби. До класичних інструментів аналізу належать статистичні пакети, такі як SPSS, MINITAB, STATISTICA тощо. Вони реалізують широкий спектр методів математичної статистики, включаючи регресійний аналіз, кореляційний аналіз, дисперсійний аналіз та інші. У деяких випадках ці системи доповнюються модулями, що реалізують нейромережеві алгоритми або генетичні методи оптимізації, що розширює їх застосування в задачах виявлення аномалій та прогнозування навантаження.

2. Дерева рішень. Алгоритми на основі дерев рішень можуть бути використані для задач класифікації, зокрема для виявлення небажаного трафіку. Проте їх застосування в задачах оптимізації або прогнозування є обмеженим. Основним недоліком є необхідність побудови великої ієрархії правил, що вимагає експертного втручання та не забезпечує гнучкості при зміні умов. У результаті система виконує лише послідовний перебір варіантів без здатності до самонавчання або адаптації.

3. Предметно-орієнтовані системи. У сфері мережевих технологій активно використовуються предметно-орієнтовані системи, зокрема

білінгові платформи з вбудованими модулями статистичного аналізу, а також системи виявлення вторгнень, такі як SNORT. Останні часто базуються на нейромережевих підходах, що дозволяє виявляти складні шаблони атак та аномалій у трафіку.

4. Нейромережеві технології. Штучні нейронні мережі (ШНМ) становлять окремий клас систем, побудованих на основі великої кількості паралельно функціонуючих елементів – штучних нейронів. Кожен нейрон функціонує автономно, обробляючи вхідну інформацію незалежно від інших, що забезпечує високий рівень паралелізму. Зв'язок між нейронами реалізується через синапси – односпрямовані канали передачі сигналів.

Нейронні мережі ефективно вирішують такі задачі:

- Класифікація – навчання мережі здійснюється на основі зразків із відомими класами. Після навчання кожен вихід мережі відповідає певному класу, що дозволяє використовувати цей підхід для виявлення небажаного трафіку, збоїв у роботі пристроїв, вірусної активності тощо.

- Кластеризація – розподіл вхідних даних на групи без попереднього визначення кількості або характеристик класів. Це дозволяє виявляти нові, раніше неідентифіковані типи трафіку.

- Прогнозування – завдяки здатності до узагальнення та виявлення прихованих залежностей, нейронні мережі можуть прогнозувати майбутні значення параметрів трафіку. Це особливо актуально в умовах самоподібності трафіку, коли поточні значення значною мірою залежать від попередніх.

Прогнозування інтенсивності трафіку дозволяє динамічно перерозподіляти смугу пропускання, адаптувати параметри маршрутизаторів і комутаторів, а також підвищувати загальну ефективність передачі даних.

Використання нейромережевих технологій у задачах керування трафіком має низку переваг:

- висока швидкість обробки даних завдяки вбудованому паралелізму;

- здатність працювати як з великими масивами даних, так і з обмеженими вибірками [6];

- можливість додаткового навчання при надходженні нових даних;

- підтримка багатозадачності (наприклад, одночасне виконання класифікації та прогнозування);

- стійкість до зашумлених даних та здатність виділяти релевантну інформацію;

- розподілене представлення знань та обчислень.

Однією з ключових властивостей нейронних мереж є здатність до навчання на основі мінімізації помилки. У процесі навчання мережа адаптує вагові коефіцієнти зв’язків між нейронами, що дозволяє поступово покращувати точність результатів. Цей механізм може бути використаний для побудови інтелектуальних систем підтримки прийняття рішень у сфері мережевого керування [9–11].

1.5 Висновок до розділу 1

У межах першого розділу здійснено системний аналіз архітектурних принципів функціонування сучасних комп’ютерних мереж, зокрема Ethernet та стеку протоколів TCP/IP, а також механізмів керування трафіком і моделей забезпечення якості обслуговування (QoS), таких як Best Effort, Integrated Services та Differentiated Services. Було розглянуто стандартизовані методи регулювання інтенсивності трафіку (шейпінг, полісінг) і пріоритезації (WFQ, WRED), що формують основу сучасних підходів до керування навантаженням у мережах.

Аналіз показав, що ефективність зазначених механізмів істотно обмежується складною структурою реального мережевого трафіку, який

характеризується самоподібністю – властивістю, за якої пікові навантаження та пульсації спостерігаються на різних часових масштабах. Ця особливість, посилена специфікою роботи протоколу TCP, зокрема його механізмами керування вікном перевантаження, призводить до втрат пакетів навіть за умов неповного завантаження каналу. Традиційні методи, що базуються на статистичному усередненні, виявляються недостатніми для точного прогнозування та адаптивного реагування на змінні умови передачі.

У результаті порівняльного аналізу систем обробки даних, таких як статистичні програмні засоби та алгоритми на основі дерев рішень, було обґрунтовано доцільність використання нейромережевих технологій. Їх здатність до навчання, виявлення прихованих закономірностей у зашумлених даних, а також висока швидкість обробки інформації роблять їх ефективним інструментом для прогнозування самоподібного трафіку. Це, у свою чергу, відкриває перспективи для побудови інтелектуальних систем керування мережевими ресурсами, здатних адаптуватися до динаміки навантаження та забезпечувати стабільну якість обслуговування.

2 АНАЛІЗ САМОПОДІБНОСТІ МЕРЕЖЕВОГО ТРАФІКУ ТА ПОБУДОВА ФРАКТАЛЬНОЇ МАТЕМАТИЧНОЇ МОДЕЛІ

Побудова математичних моделей для дослідження динаміки інформаційних потоків у комп'ютерних мережах становить один із ключових напрямків у галузі забезпечення якості обслуговування (Quality of Service, QoS). Традиційно моделі передачі даних класифікуються за двома методологічними підходами:

– Фізичні моделі. Представлені натурними, квазінатурними, масштабними та аналоговими реалізаціями, в основі яких лежить експериментальне або апаратне відтворення досліджуваних процесів.

– Математичні моделі. Об'єднують аналітичні, чисельні та імітаційні методи, що ґрунтуються на формалізації поведінки мережі через систему математичних співвідношень, диференціальних рівнянь та обчислювальних алгоритмів.

Математичне моделювання мережі – це структурований підхід до формалізації процесів, що відбуваються в телекомунікаційній інфраструктурі, який уможливлює аналіз динаміки трафіку, кількісну оцінку навантаження на вузли та прогнозування продуктивності протоколів передачі даних.

2.1 Виявлення характеристик та властивостей самоподібних процесів у телекомунікаційних системах

Результати багаточисельних експериментальних досліджень трафіку корпоративних та мультисервісних телекомунікаційних систем переконливо свідчать про обмеженість застосовності класичних підходів до моделювання, первісно розроблених для систем телефонної комутації

загального користування [11, 12]. Неадекватність традиційних моделей пояснюється фундаментальними відмінностями у природі генерації інформаційних потоків: процеси передачі даних у пакетних мережах відзначаються значною нерівномірністю та стохастичною мінливістю. Експериментальні дослідження документують присутність самоподібності у структурі мережевого трафіку, що проявляється через тривалу кореляційну залежність і фрактальну організацію часових послідовностей [15]. У зв'язку з цим, адекватне відображення та передбачення поведінки таких систем вимагає залучення апарату теорії фрактальних випадкових процесів [20, 27], здатного врахувати багаторівневу кореляційну архітектуру та асимптотичну самоподібність, забезпечуючи підвищену точність моделювання і ефективність алгоритмів керування ресурсами сучасних комп'ютерних мереж.

Кількісною характеристикою, що дозволяє оцінити міру самоподібності процесу, виступає показник Херста (H) [30, 31]. Залежно від числового діапазону цього параметра визначають наступні категорії стохастичних процесів:

– $H \in (0; 0,5)$ – процес характеризується антиперсистентною поведінкою і не належить до класу самоподібних;

– $H = 0,5$ – відповідає випадковому процесу без кореляцій, що описується класичним пуассонівським розподілом;

– $H \in (0,5; 1)$ – процес володіє властивостями самоподібності, персистентності та довготривалої пам'яті. Наближення значення H до одиниці корелює зі зростанням вираженості самоподібних характеристик [19].

Ідентифікація самоподібної структури трафіку базується на трьох фундаментальних ознаках [18]:

1. Повільно спадна залежність (ПСЗ). У самоподібних процесах дисперсія середнього значення вибірки знижується повільніше порівняно з

класичним законом оберненої пропорційності. Ця властивість індикує наявність довгострокових кореляційних зв'язків між елементами часової послідовності.

Автокореляційна функція (АКФ) процесу X набуває вигляду:

$$r(k) \sim L_1(k) \cdot k^{-\beta}, \quad k \to \infty, \quad \beta = 2 - 2H \tag{2.1}$$

де $0 < \beta < 1$ – параметр Херста, що визначає ступінь самоподібності та інтенсивність довгострокової залежності; L_1 – функція, що повільно зростає на нескінченності (наприклад, log(k) або стала величина), яка зазвичай не враховується при асимптотичному аналізі.

2. Довгострокова залежність (ДСЗ). Ця властивість, відома також як long-range dependence (LRD), характеризується розбіжністю ряду автокореляційних коефіцієнтів. Для контрасту, у процесах із короткостроковою залежністю (КСЗ) автокореляція r(k) зменшується експоненційно відповідно до закону $r(k) \sim \rho^k$, де $0 < \rho < 1$, що гарантує збіжність суми $\Sigma\infty_{k=1}$ r(k). На противагу цьому, у самоподібних процесах з ДЗЗ автокореляція знижується за гіперболічним законом:

$$r(k) \sim k^{2H-2}, \quad H \in (0{,}5;\ 1) \tag{2.2}$$

де $H \in (0{,}5;\ 1)$.

В цьому інтервалі значень показника Херста спостерігається розбіжність суми автокореляційних коефіцієнтів, що свідчить про присутність статистично значущого, хоча і поступово ослабленого, зв'язку між реалізаціями процесу, віддаленими у часі.

Отже, самоподібні процеси з довгостроковою залежністю фундаментально відрізняються від традиційних стаціонарних моделей, оскільки зберігають інформацію про минулі стани на істотно більших

часових масштабах, що має принципове значення для моделювання, прогнозування та аналізу складних динамічних систем [3, 16].

3. Характер спектральної густини на низьких частотах. Третьою визначальною характеристикою самоподібного випадкового процесу є специфічна поведінка спектральної густини в околі нульової частоти. Для процесів, що демонструють довгострокову залежність (ДЗЗ), спектральна густина S(ω) виявляє асимптотичну залежність при ω → 0, формалізовану співвідношенням:

$$S(\omega) \sim c_x |\omega|^{-\beta}, \quad \omega \to 0, \tag{2.3}$$

де β = 2H - 1.

Оскільки для ДЗЗ показник Херста H належить інтервалу (0,5; 1), параметр β набуває додатних значень, внаслідок чого S(ω) → ∞ при ω → 0. Це означає концентрацію енергетичного спектру процесу в низькочастотній області, що є характерною сигнатурою фрактальних чи самоподібних структур [20, 26]. Така поведінка спектра отримала назву «спектр типу 1/f» або «фрактальний гаусівський шум», оскільки в логарифмічних координатах графік спектральної густини демонструє лінійність з кутовим коефіцієнтом, визначеним параметром β.

Кількісна оцінка показника Херста H, який служить мірою самоподібності та ДЗЗ, може бути здійснена за допомогою комплексу статистичних і спектральних методів. Одним із класичних підходів є R/S-аналіз (аналіз нормованого розмаху), запропонований Г. Херстом [30, 31]. Метод базується на дослідженні співвідношення між нормованим розмахом (відношенням розмаху до стандартного відхилення) та величиною часового вікна. У логарифмічних координатах ця залежність набуває лінійної форми, причому нахил прямої відповідає оцінці H.

Альтернативним є дисперсійний метод, що ґрунтується на вивченні динаміки дисперсії агрегованих даних при зміні масштабу. Для самоподібного процесу дисперсія усередненого значення на вікні розміру m спадає пропорційно m^{2H-2}, що дозволяє обчислити H за кутовим коефіцієнтом у відповідних логарифмічних координатах.

Спектральні методи, серед яких метод періодограми та оцінювач Вітла, використовують властивість спектра типу 1/f. Тут параметр H визначається через кутовий коефіцієнт графіка логарифма спектральної густини відносно логарифма частоти. Ці методи виявляють високу ефективність при роботі з великими обсягами експериментальних даних [21].

Серед сучасного інструментарію важливе місце займає вейвлет-аналіз, що надає можливість сепарації компонент довгострокової та короткострокової залежності, а також шумових складових. Даний метод базується на аналізі лінійної залежності між логарифмом дисперсії вейвлет-коефіцієнтів та логарифмом масштабу [2].

Точність визначення параметра Херста є критично важливою для створення адекватних моделей самоподібного трафіку, особливо в контексті телекомунікаційних мереж, де прецизійне прогнозування навантаження відіграє вирішальну роль у забезпеченні належного рівня обслуговування та раціонального управління обчислювальними ресурсами [13, 14].

2.2 Експериментальне дослідження та визначення параметрів мережевого трафіку

Історична еволюція телекомунікаційних технологій характеризується трансформацією від традиційних систем комутації каналів до сучасних архітектур пакетної комутації, що зумовило кардинальні зміни в методології моделювання трафіку [29]. У системах з канальною комутацією

центральним завданням було забезпечення заданої ймовірності відмови в обслуговуванні викликів, що досягалося застосуванням марковських моделей, зокрема моделі втрат Ерланга B [11]. Такі моделі ефективно описували процеси встановлення з'єднань та надходження викликів, які характеризувалися відносною гладкістю та детермінованістю.

У пакетних мережах, де домінують протоколи стеку TCP/IP [1], критичними параметрами якості стають затримка доставки пакетів, імовірність їх втрати та коефіцієнт утилізації пропускної здатності. Трафік у таких системах демонструє пульсаційний характер на всіх часових масштабах і виявляє довгострокову кореляційну залежність [15]. Це обумовлює необхідність використання принципово нових класів моделей, включаючи самоподібні (зокрема, фрактальний броунівський рух, фрактальний гаусівський шум [25, 26]) та нейромережеві підходи [2, 5], здатні адекватно відобразити складну структуру інформаційних потоків.

Традиційні марковські моделі, незважаючи на їхню математичну стрункість, виявляються неспроможними для повноцінного опису характеристик трафіку в сучасних мережах [10]. Вони не враховують ефекти самоподібності та довгострокової залежності, які істотно впливають на параметри систем масового обслуговування, зокрема на середній час перебування в черзі та ймовірність відкидання пакетів. Нехтування цими ефектами призводить до заниження реального навантаження на систему та, як наслідок, до некоректного проєктування мережевої інфраструктури.

Для перевірки гіпотези щодо самоподібної природи трафіку використовується компаративний статистичний аналіз, що охоплює як емпіричні, так і синтезовані дані. На початковому етапі виконується збір реальних трас трафіку з експлуатаційного мережевого середовища. Надалі генеруються синтетичні послідовності, що відповідають різноманітним теоретичним моделям: гаусівський шум з нульовим математичним сподіванням (базований на нормальному розподілі), узагальнений

броунівський шум та пуассонівський процес, традиційно застосовуваний для моделювання потоку викликів у телефонних системах.

Синтез вибірок гаусівського шуму базується на центральній граничній теоремі та реалізується за допомогою спеціалізованого програмного забезпечення, включаючи MATLAB із використанням функцій randn або normrnd для генерації псевдовипадкових чисел з нормальним розподілом, або спеціалізованого пакета Fractan 4.4 [28] (рис. 2.1).

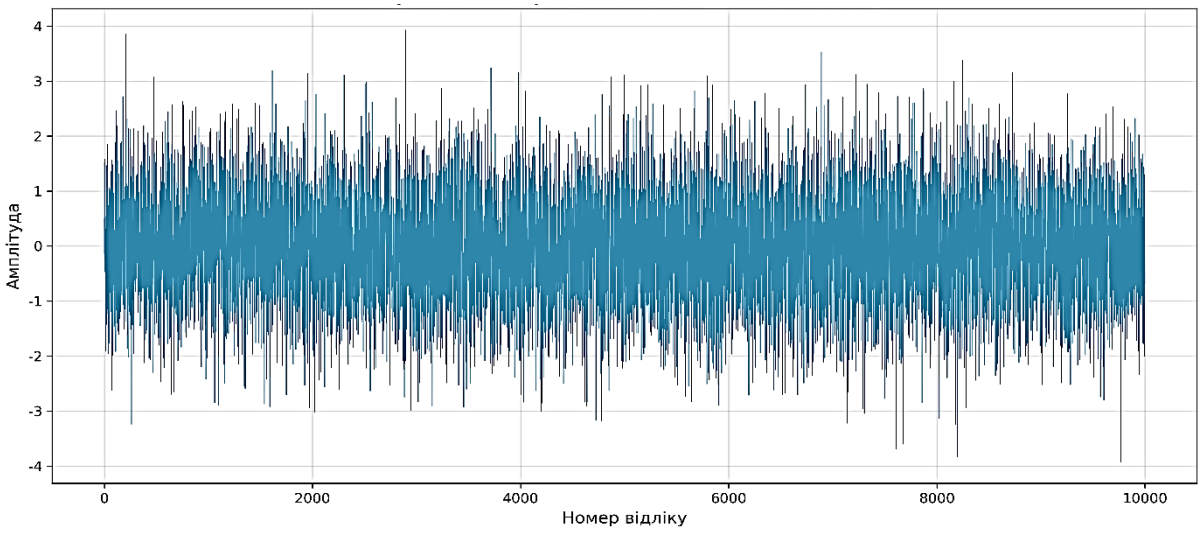


Рисунок 2.1 – Синтезований гаусівський шум
(кількість дискретних значень – 10001)

Застосування програмного комплексу SELFIS у поєднанні з класичними алгоритмами оцінювання параметра Херста забезпечує верифікацію коректності реалізації обчислювальних процедур.

З теоретичної перспективи, для білого гаусівського шуму, аналогічно до пуассонівського трафіку, очікуване значення параметра Херста складає $H \approx 0{,}5$. Це вказує на відсутність довгострокової залежності: автокореляція між віддаленими у часі вибірками швидко спадає до нуля, що є характерною ознакою короткострокової залежності.

Результати чисельних експериментів підтверджують ці теоретичні прогнози: отримані значення H знаходяться в діапазоні від 0,440 до 0,550, що статистично узгоджується з теоретичним значенням. Невеликі відхилення пояснюються скінченною довжиною вибірки та похибками чисельних методів, проте вони не впливають на загальний висновок: гаусівський шум не проявляє властивостей самоподібності.

Таким чином, даний етап дослідження виконує роль валідації інструментальних засобів та формує референтну базу для наступного аналізу реального мережевого трафіку.

Наступний етап передбачає дослідження реального трафіку, зареєстрованого на порту комутаційного обладнання Cisco Catalyst 9400. Даний пристрій належить до провідної серії для рівня агрегації та доступу [18], обслуговує реальних користувачів мережі Інтернет і тому є репрезентативним джерелом даних для аналізу характеристик сучасного пакетного трафіку.

Основна мета цього етапу полягає у порівнянні візуальних (див. рис. 2.2) та статистичних характеристик експериментального трафіку з теоретичними моделями, зокрема з гаусівським шумом та пуассонівським процесом.

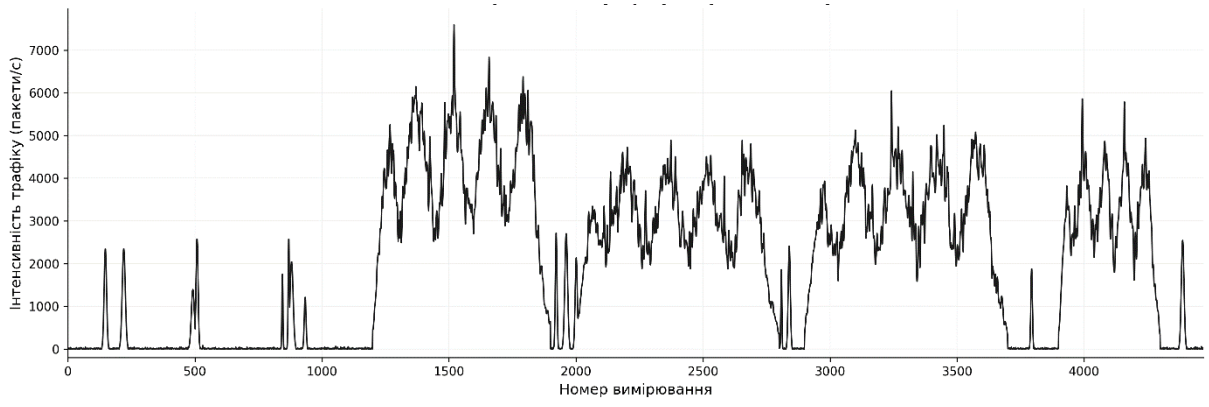


Рисунок 2.2 – Динаміка мережевого трафіку в першому експерименті

Загальний масив зібраних даних в цьому експерименті становить 4470 дискретних вимірювань.

Результати кількісного аналізу, виконаного для трафіку користувачів, що проходить через комутатор Cisco Catalyst 9400, однозначно підтверджують гіпотезу про наявність самоподібності та довгострокової залежності (ДСЗ) у досліджуваному процесі.

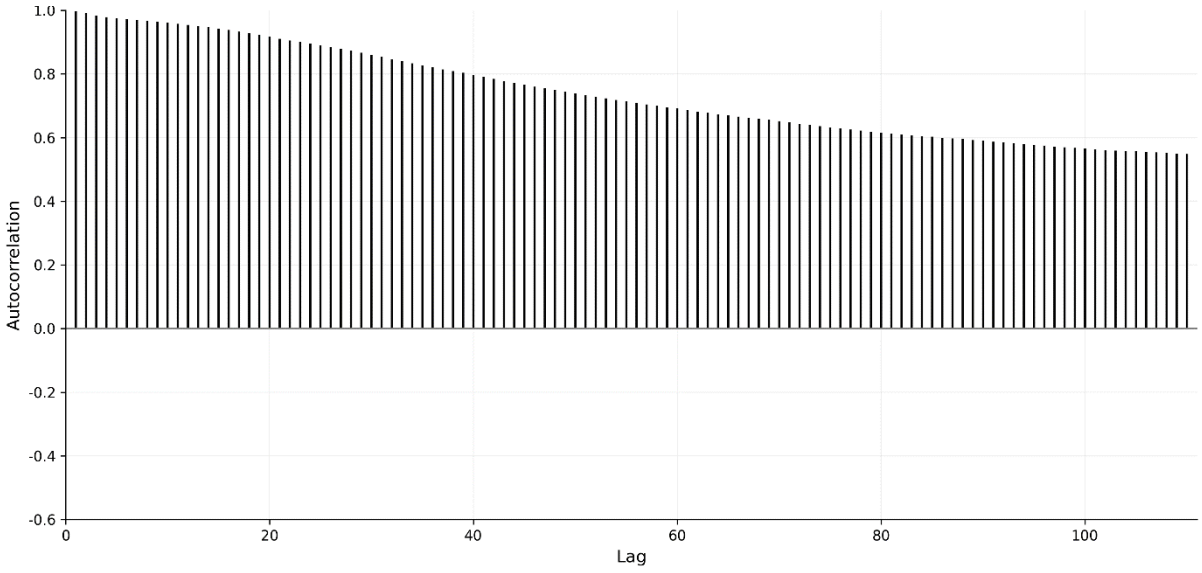


Рисунок 2.3 – Автокореляційна функція даних з першого експерименту

Оцінювання параметра Херста H, реалізоване за допомогою чотирьох незалежних методів, забезпечило такі результати:

– метод Aggregate Variance: H = 0,862;
– метод R/S: H = 0,822;
– метод Periodogram: H = 0,933;
– метод Whittle Estimator: H = 0,97.

Результати аналізу автокореляційної функції (АКФ) повністю узгоджуються з чисельними оцінками H: графік АКФ (рис. 2.3) чітко демонструє повільно спадну залежність. АКФ зменшується за

гіперболічним законом (степеневим порядком) k^{2H-2}, а не експоненційно, що є ключовою властивістю самоподібних процесів з довгостроковою залежністю.

У межах другої експериментальної серії було зібрано та опрацьовано значний обсяг даних, що охоплює шестиденний період спостережень. Цей часовий ряд містить 18 293 дискретні значення, графічне представлення яких наведено на рис. 2.4.

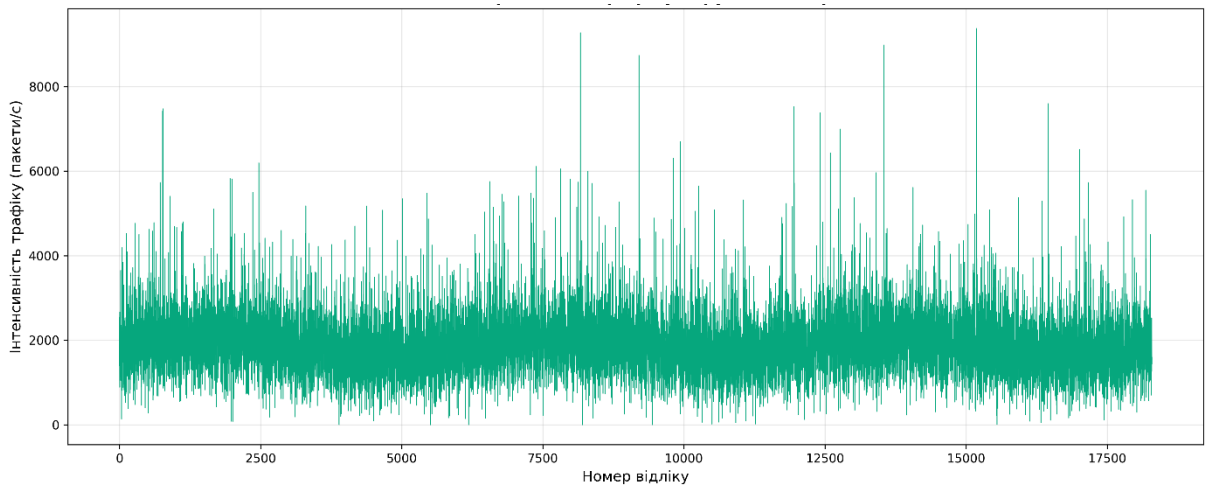


Рисунок 2.4 – Динаміка мережевого трафіку (другий експеримент)

Оцінювання параметра Херста H для цього масиву даних забезпечило наступні результати:

– метод Aggregate Variance: H = 0,823;
– метод R/S: H = 0,849;
– метод Periodogram: H = 0,864;
– метод Whittle Estimator: H = 0,939.

Всі отримані значення розміщуються в інтервалі $0,5 < H < 1$, що є незаперечним доказом присутності самоподібності та ДЗЗ. Особливо високе значення, отримане методом Вітла (0,939), вказує на інтенсивний

кореляційний зв'язок між реалізаціями процесу, віддаленими у часі. Графічне представлення результатів методу R/S показано на рис. 2.5, де чітко простежується лінійна залежність у логарифмічному масштабі, типова для самоподібних процесів.

З метою забезпечення компаративного аналізу у рамках третього експериментального етапу було залучено масив даних щодо добових витрат інтернет-трафіку, починаючи з 24 жовтня 2025 року. Загальний обсяг вибірки становив 760 спостережень (рис. 2.6).

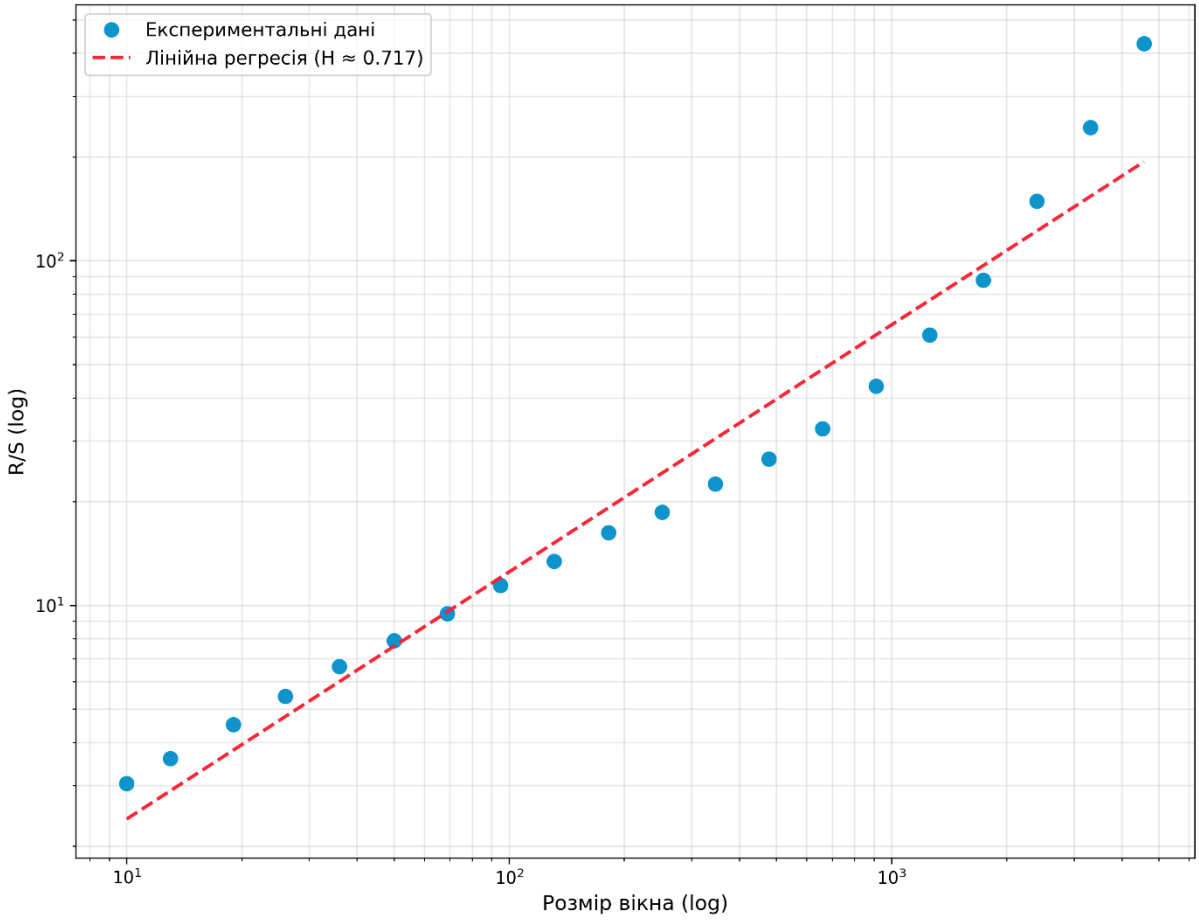


Рисунок 2.5 – Визначення параметра Херста за результатами експериментальних даних

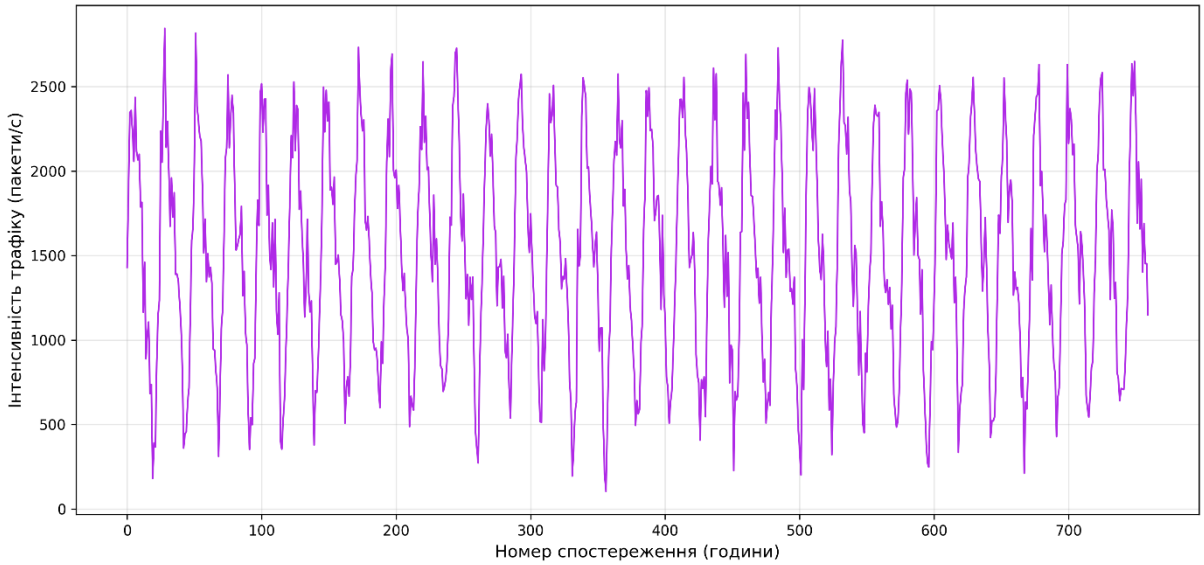


Рисунок 2.6 – Часова еволюція мережевого трафіку (третій експеримент)

Застосування методологічного апарату для оцінювання параметра Херста (H) для цього експериментального набору даних дало такі результати: метод Aggregate Variance – H = 0,611; метод R/S – H = 0,540; метод Periodogram – H = 0,728; метод Whittle Estimator – H = 0,696.

Подальша серія обчислень була виконана для узагальненого броунівського шуму (рис. 2.7).

Дискретні значення якого були згенеровані за допомогою програмного комплексу Fractan 4.4 [28]. Важливо зазначити, що ці значення представляють собою диференціальні прирости узагальненого броунівського руху, для синтезу якого застосовувався алгоритм послідовних випадкових складань Фосса. Загальна кількість елементів у цьому наборі даних становила 10 001, при цьому генерований параметр Херста був заданий на рівні H = 0,9.

Обчислення параметра Херста для цієї синтетичної послідовності забезпечило такі значення: Aggregate Variance – H = 0,752; R/S – H = 0,799; Periodogram – H = 0,894; Whittle Estimator – H = 0,823.

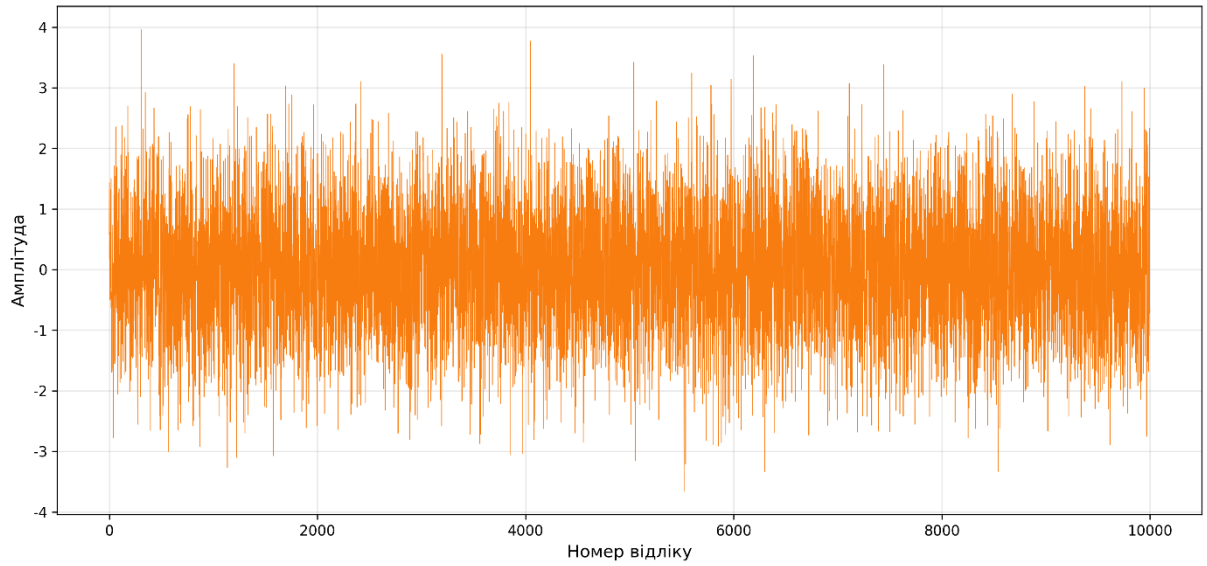


Рисунок 2.7 – Синтезований узагальнений броунівський шум

Отримані результати, які послідовно перевищують порогове значення 0,5, підтверджують самоподібний характер узагальненого броунівського шуму.

Для контрольного дослідження було використано набір даних, що відповідає пуассонівському розподілу (рис. 2.8), згенерований у середовищі Microsoft Excel.

Загальний обсяг цієї вибірки також становив 10 001 елемент.

Розрахунок параметра Херста для послідовності Пуассона забезпечив наступні результати: метод Aggregate Variance – H = 0,420; метод R/S – H = 0,278; метод Periodogram – H = 0,466; метод Whittle Estimator – H = 0,500.

Констатовано, що всі отримані значення параметра Херста є близькими до 0,5 або меншими за нього. Це повністю узгоджується з теоретичними положеннями, оскільки пуассонівський розподіл моделює процеси, які не є самоподібними, що, своєю чергою, підтверджує коректність застосованих обчислювальних методів.

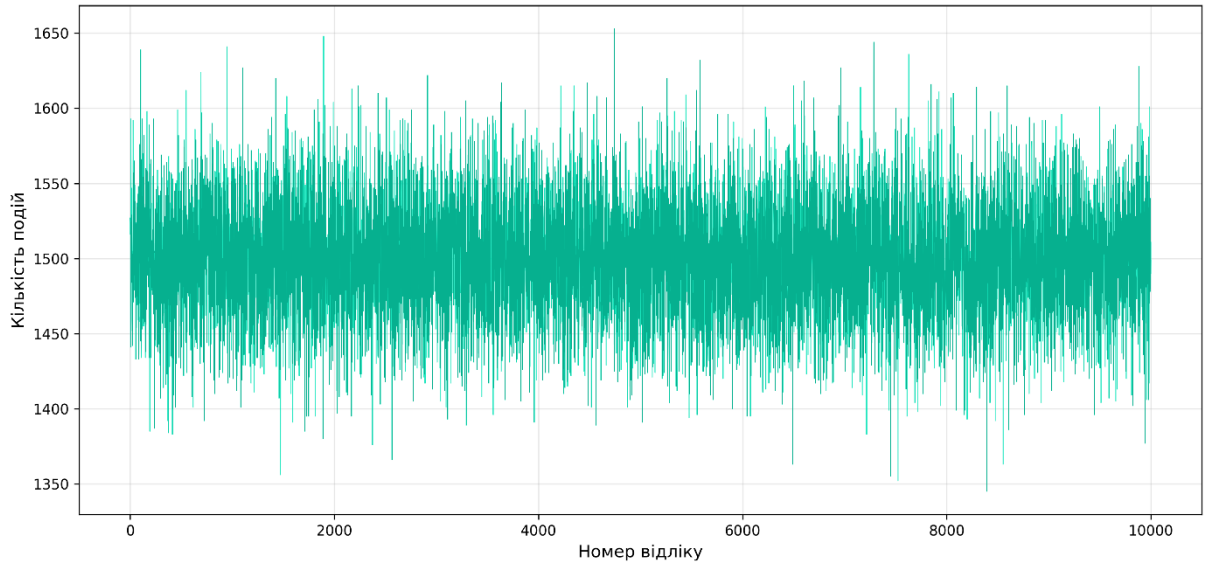


Рисунок 2.8 – Синтезована послідовність відповідно до розподілу Пуассона

Зведені результати обчислень значень параметра Херста для всіх експериментальних наборів даних представлено у табл. 2.1.

Таблиця 2.1 – Результати обчислення параметра Херста для експериментальних даних

Дані	Агрегована дисперсія	R/S	Періодограма	Оцінювач Віттла
Гауссів шум	0,440	0,550	0,465	0,500
Експеримент 1	0,862	0,822	0,933	0,970
Експеримент 2	0,823	0,849	0,864	0,939
Експеримент 3	0,611	0,540	0,728	0,696
Узагальнений броунівський шум	0,752	0,799	0,894	0,823
Розподіл Пуассона	0,420	0,278	0,466	0,500

Після систематизації експериментальних даних та визначення мінімально необхідного набору репрезентативних фундаментальних параметрів постає завдання вибору адекватної математичної моделі процесів у комп'ютерних мережах. Принципи такого вибору базуються на декількох концептуальних положеннях. По-перше, модель має адекватно відображати ключові особливості поведінки мережевого трафіку, узгоджені з експериментальними спостереженнями на макроскопічному рівні, а також забезпечувати фізичну інтерпретацію виявлених закономірностей. По-друге, для продуктивного дослідження інформаційних процесів доцільно використовувати інструментарій помірної складності, що дозволяє отримувати практично значущі результати без надмірного ускладнення математичного апарату. По-третє, модель повинна бути параметризованою настільки, щоб на її основі можна було формулювати задачі прогнозування та управління мережевими процесами [23].

Створення адекватної моделі трафіку вимагає глибокого розуміння механізмів передачі даних, зокрема природи пачковості (burstiness) та її фрактальних характеристик [15, 19]. Передача інформації у комп'ютерних мережах здійснюється шляхом фрагментації потоку байтів на пакети визначеної довжини, які маршрутизуються через мережеві вузли. Взаємодія джерел та приймачів генерує переривчастий трафік, що проявляється у вигляді пачок пакетів. Така пачковість зумовлена як апаратними обмеженнями, так і специфікою протоколів, зокрема TCP, який через механізми контролю перевантаження може спричиняти призупинення передачі та генерацію службового трафіку, посилюючи нерівномірність потоку.

Емпіричні дослідження свідчать, що мережевий трафік характеризується довготривалою кореляцією та самоподібністю, які не узгоджуються з класичними пуассонівськими чи марковськими моделями. Ці властивості проявляються у статистиках другого порядку, таких як

автокореляційна функція, спектральна щільність та дисперсія [3, 16]. У межах теорії фрактальних процесів самоподібність і протяжна залежність розглядаються як еквівалентні категорії, що виявляються при великих часових інтервалах або внаслідок агрегування численних незалежних потоків.

Для моделювання реального трафіку, який утворюється внаслідок суперпозиції багатьох джерел, доцільним є використання узагальнених моделей, що враховують важкі хвости у розподілах міжпакетних інтервалів [25]. У результаті трафік набуває фрактальних властивостей, стаючи водночас складним і структурованим. Найбільш адекватною моделлю для опису самоподібного трафіку виступає фрактальний броунівський рух (BH(t)), який дозволяє враховувати довготривалу пам'ять та складну кореляційну структуру. Це має вирішальне значення для точного прогнозування навантаження та створення ефективних систем адаптивного керування мережею [13, 14].

2.3 Побудова математичної моделі трафіку на основі фрактального броунівського руху

Одним із фундаментальних прикладів випадкового процесу, що володіє фрактальними характеристиками, є вінерівський процес броунівського руху B(t). Його траєкторії є майже напевно неперервними, тобто безперервними з імовірністю одиниця, але водночас майже скрізь недиференційовними. Важливою властивістю цього процесу є масштабна інваріантність: для будь-якого додатного коефіцієнта a, процес B(at) має ту саму статистичну структуру, що й $\sqrt{a}$ B(t) [27].

Щільність імовірнісного розподілу координати частинки, що перебуває під впливом броунівського руху X(t), має вигляд:

$$p(\Delta,\tau) = \frac{1}{\sqrt{2\pi kD\tau}} \exp\left(-\frac{\Delta^2}{2kD\tau}\right) \quad (2.4)$$

де $\Delta = X(t) - X(t_0)$; $\tau = t - t_0$; kD – коефіцієнт дифузії.

Введена щільність розподілу ймовірності задовольняє співвідношенню самоподібності:

$$p(b\Delta, b^2\tau) = b^{-1}p(\Delta, \tau) \quad (2.5)$$

де $b > 0$ – довільний масштабний коефіцієнт.

Приріст координати броунівської частинки за малий інтервал часу Δt описується як випадкова величина з нормальним розподілом:

$$\Delta X(t) = X(t + \Delta t) - X(t) = \sqrt{kD\Delta t}\sum_i \xi_i \quad (2.6)$$

де $\sum_i \xi_i$ – сума реалізацій з гаусівського розподілу.

Таким чином, броунівський рух є класичним прикладом стохастичного процесу Вінера, який позначається як Bt для $t \geq 0$ і характеризується наступними властивостями [8]:

– прирости Bt - Bs нормально розподілені з нульовим математичним сподіванням і дисперсією t - s;

– прирости на неперетинних часових інтервалах $[t_1,t_2]$ і $[t_3,t_4]$, тобто $Bt_2 - Bt_1$ і $Bt_4 - Bt_3$, є незалежними випадковими величинами;

– $B_0 = 0$, а Bt є неперервним при t = 0;

– математичне очікування приросту броунівського руху рівне:

$$E[Bt - Bs] = 0 \quad (2.7)$$

Функція розподілу:

$$F_{B_t}(x) = P(B_t \leq x) = \Phi!\left(\frac{x}{\sqrt{t}}\right) \quad (2.8)$$

Шляхом заміни показника ступеня на довільне дійсне число з інтервалу H ∈ [0; 1], введемо поняття узагальненого фрактального броунівського руху BH(t) =F_{B_t}.

Випадок, коли показник Херста H = 0,5, відповідає процесу з незалежними приростами та описує класичний броунівський рух. У такому випадку прирости не мають кореляції, а дисперсія зростає лінійно з часом.

Натомість фрактальний броунівський рух (FBM), що виникає при H ≠ 0,5, характеризується залежними приростами з нелінійною дисперсією, яка змінюється за законом [30]:

$$Var[BH(t) - BH(s)] = \sigma^2|t - s|^{2H} \quad (2.9)$$

При s = 0 отримуємо:

$$Var[BH(t)] = \sigma^2 t^{2H} \quad (2.10)$$

Далі:

$$Cov[B_H(t_1), B_H(t_2)] = E![B_H(t_1) \cdot B_H(t_2)] = \\ = \frac{\sigma^2}{2}(t_1^{2H} + t_2^{2H} - |t_2 - t_1|^{2H}) \quad (2.11)$$

Кореляція приростів для двох неперетинних часових інтервалів у самоподібному процесі:

$$Corr[B_H(t_2) - B_H(t_1),\ B_H(t_4) - B_H(t_3)] =$$
$$= \frac{1}{2}[(t_4 - t_1)^{2H} + (t_3 - t_2)^{2H} - (t_4 - t_2)^{2H} - (t_3 - t_1)^{2H}] \quad (2.12)$$

У дискретному випадку нормована кореляційна функція послідовності приростів виходить заміною t_1, t_2, t_3 і t_4 на n, n+1, n+k і n+k+1 відповідно і діленням на σ^2:

$$r_k = (1/2)[(k+1)^{2H} - 2k^{2H} + (k-1)^{2H}] \quad (2.13)$$

Послідовність приростів, що характеризує часовий ряд інтенсивності мережевого трафіку, може бути інтерпретована як фрактальний гаусівський шум. Такий процес демонструє властивість довготривалої залежності, що проявляється у поведінці автокореляційної функції [26].

Зокрема, при великих значеннях k, автокореляційна функція r_k набуває асимптотичної форми: $r_k \sim k^{2H-2}$ (розкладання Тейлора).

Фрактальний броунівський рух (fBt) може бути формально визначений через інтегральне представлення стандартного броунівського руху Bt (або процесу Вінера), використовуючи дробово-інтегральний оператор:

$$B_H(t) = \frac{1}{\Gamma!(H+\backslash tfrac12)} \int_{-\infty}^{0} \left[(t-u)^{H-\backslash tfrac12} - (-u)^{H-\backslash tfrac12}\right] dB(u) +$$
$$+ \int_0^t (t-u)^{H-\backslash tfrac12}\ dB(u) \quad (2.14)$$

де Г(x) – гамма-функція.

Згідно з наведеним співвідношенням, значення випадкової функції у момент часу t визначається всіма попередніми приростами dB(u) процесу при u < t. Це означає, що взаємна залежність між приростами фрактального броунівського руху є нескінченною, тобто кожне нове значення процесу зберігає пам'ять про всю його попередню історію.

Фрактальний броунівський рух BH(t) може бути представлений співвідношенням:

$$BH(t) = Xt^H \tag{2.15}$$

де X – нормально розподілена випадкова змінна з нульовим математичним сподіванням і одиничною дисперсією.

Значення процесу в момент часу t може бути обчислено за допомогою співвідношення:

$$BH(at) = a^H BH(t) \tag{2.16}$$

що відображає масштабну інваріантність. При значенні параметра Херста H = 0,5 отримуємо класичний вінерівський процес броунівського руху. Основні статистичні характеристики фрактального броунівського руху такі [27]:

– математичне сподівання: $E[BH(t)] = 0$;

– дисперсія: $\sigma^2[BH(t)] = \sigma^2[Xt^H] = t^{2H}$;

– коефіцієнт кореляції між значеннями процесу у моменти часу t_1 та t_2:

$$\rho(t_1, t_2) = \frac{1}{2}[t_1^{2H} + t_2^{2H} - |t_2 - t_1|^{2H}] \tag{2.17}$$

– стаціонарні прирости:

$$E[B_H(t+\tau) - B_H(t)] = 0,$$
$$Var[B_H(t+\tau) - B_H(t)] = \tau^{2H} \quad (2.18)$$

Фрактальний броунівський рух fBt – це загальноприйнята модель, що використовується в дослідженнях для оцінювання продуктивності систем шляхом моделювання мережевих процесів [25, 26].

У випадку трафіку, що формується великою групою абонентів із подібними часовими характеристиками або близькими значеннями співвідношення пікового та середнього навантаження, доцільним є використання моделі фрактального броунівського руху (fBt), яка дозволяє представити процес як суму окремих компонент [29]. Формально це записується у вигляді:

$$X_t^i = \lambda_i t + a_i Z_t^{(i)} \quad (2.19)$$

де X_t^i – і-а компонента процесу X у момент часу t; λ_i – інтенсивність і-го потоку повідомлень; a_i – параметр, що визначає дисперсію потоку повідомлень в одиницю часу; $Z_t^{(i)}$ – стандартний дифузний процес із нульовим дрейфом, який відображає статистичні властивості і-ї компоненти. Для опису $Z_t^{(i)}$ застосовується модель фрактального броунівського руху з різними значеннями параметра Херста, що дозволяє врахувати різні рівні самоподібності та довготривалої залежності.

У такому поданні кожна компонента складається з двох частин: перша визначає середнє число повідомлень, що надійшли до моменту часу t, тоді як друга описує випадкове відхилення із нульовим середнім значенням. Підсумовуючи всі компоненти процесу X, отримуємо його загальне представлення:

$$X_t = \sum_{i=1}^{N} X_t^i = \sum_{i=1}^{N} \lambda_i t + \sum_{i=1}^{N} a_i Z_t^{(i)} = \Lambda t + Y_t \qquad (2.20)$$

де процес Y характеризує відхилення величини трафіку від його середнього значення. Оскільки перший доданок є детермінованим і може бути визначений на основі параметрів інтенсивності потоків, найбільший інтерес становить саме другий доданок, який описує стохастичну складову трафіку. Таким чином, модель дозволяє відокремити регулярну частину процесу від випадкових флуктуацій, що є критично важливим для аналізу самоподібних властивостей мережевого трафіку та побудови адекватних методів прогнозування й керування навантаженням у комп'ютерних мережах [13, 14].

Отримуємо таке подання:

$$Y_t = \sum_{i=1}^{N} a_i Z_t^{(i)} = \sum_{i=1}^{N} a_i Z_t^{H_i} \qquad (2.21)$$

де Z_t^{Hi}, що являє собою фрактальний броунівський рух із відповідним параметром Херста H_i, використовується для моделювання відхилення і-ї складової від її усередненого показника.

З огляду на це, сукупний абонентський трафік доцільно розглядати як суперпозицію (суміш) кількох індивідуальних потоків, кожен з яких характеризується власним значенням параметра Херста.

Диференціація значень параметра Херста, які є ключовими характеристиками для різних груп абонентського трафіку, зумовлена інтегральними властивостями, що визначаються первинною природою відповідного інформаційного навантаження. Різні значення параметра H_i слугують індикатором того, що ці компоненти генеруються різнорідними типами трафіку: голосовий трафік, потоки передачі даних, відеоінформація тощо [10].

Зважаючи на гетерогенність природи зазначених потоків, обґрунтованим є припущення, що їхні випадкові флуктуації є взаємно

незалежними. Відповідно, усі розглянуті процеси Z_t^{Hi} також вважаються незалежними [29].

2.4 Верифікація та оцінювання можливості прогнозування трафіку на основі фрактальної моделі

Підвищення ефективності функціонування телекомунікаційної інфраструктури в умовах перевантаження можливе шляхом інтелектуального переконфігурування мережевих протоколів і обладнання (комутаторів, маршрутизаторів, кінцевих пристроїв). Для цього необхідне володіння актуальною інформацією про динаміку трафіку, що надходить до каналу зв'язку. За наявності статистичних даних про інтенсивність трафіку, зібраних з фіксованим часовим кроком, можливо застосовувати формалізовані методи прогнозування майбутнього навантаження [23, 24].

Можливість прогнозування мережевого трафіку обґрунтовується його самоподібною природою та підтверджується такими властивостями [3, 15, 16]:

1. Довготривала пам'ять – наявність автокореляції між віддаленими значеннями інтенсивності трафіку, що проявляється у гіперболічному спаданні кореляційної функції та формуванні нескінченного кореляційного інтервалу.

2. Стійкість (персистентність) – здатність зберігати напрямок зміни трафіку: якщо спостерігається зростання або спад, ця тенденція, з високою ймовірністю, продовжиться.

3. «Товсті хвости» у розподілах – результат порційного надходження даних, що викликає нерівномірне навантаження, особливо в умовах роботи протоколу TCP [25].

З огляду на вищезазначене, штучні нейронні мережі є доцільним інструментом для побудови моделей короткострокового прогнозування трафіку [2, 5, 22]. Їх здатність виявляти нелінійні та довготривалі залежності

дозволяє ефективно передбачати майбутні значення інтенсивності навантаження, що є критично важливим для адаптивного керування ресурсами мережі.

2.5 Висновки до розділу 2

У розділі досліджено проблематику математичного моделювання мережевого трафіку в контексті трансформації від систем комутації каналів до пакетних мереж. Встановлено неадекватність класичних моделей (марковських, пуассонівських) для опису характеристик сучасного телетрафіку. На основі теорії фрактальних процесів ідентифіковано самоподібність трафіку, що виявляється через повільне зниження дисперсії, довгострокову автокореляційну залежність і спектральну щільність типу 1/f. Експериментальні дослідження (з використанням маршрутизаторів серії Cisco Catalyst 9400) підтвердили ці властивості: показник Херста (H) перевищував значення 0,8, що свідчить про персистентний характер процесу.

Найбільш адекватною моделлю визнано фрактальний броунівський рух BH(t), який враховує довготривалу залежність. Виявлена властивість пам'яті процесу дозволяє теоретично обґрунтувати можливість прогнозування трафіку. Зроблено висновок про перспективність застосування нейромережевих технологій для побудови моделей короткострокового прогнозування, точність яких корелює зі ступенем самоподібності трафіку.

3 РОЗРОБКА АВТОМАТИЗОВАНОЇ СИСТЕМИ МОНІТОРИНГУ ТА ПРОГНОЗУВАННЯ ЗБОЇВ У ТЕЛЕКОМУНІКАЦІЙНИХ МЕРЕЖАХ

3.1 Розробка алгоритмічного забезпечення інтелектуальної системи моніторингу та керування на основі нейромережевих технологій

3.1.1 Алгоритм перерозподілу смуги пропускання

Розробка ефективної системи керування трафіком у сучасних телекомунікаційних мережах потребує інтеграції трьох фундаментальних компонентів: системного аналізу функціонування мереж, емпіричних досліджень реального трафіку та побудови математичної моделі, що враховує самоподібну природу навантаження. У цьому контексті особливу роль відіграє модель фрактального броунівського руху (ФБР), яка дозволяє адекватно описати часову структуру трафіку з довгостроковою залежністю.

Розроблений алгоритм керування спирається на підтверджений факт: трафік користувачів мережі є самоподібним і демонструє довгострокову залежність. Це означає, що загальний трафік є суперпозицією окремих потоків, кожен з яких може характеризуватися власним значенням параметра Херста *H*. Така гетерогенність вимагає адаптивного підходу до розподілу ресурсів, оскільки статичні моделі не здатні врахувати динаміку та сплесковість реального навантаження.

На основі моделі ФБР та існуючих механізмів керування (шейпінг, полісінг) було розроблено алгоритм динамічного перерозподілу смуги пропускання, що усуває обмеження традиційних статичних підходів (рис. 3.1). У класичних реалізаціях адміністратор заздалегідь конфігурує мережеве обладнання (наприклад, Cisco Catalyst 9400) для обслуговування певних пріоритетних потоків. Виділена смуга пропускання залишається фіксованою до моменту ручного втручання, що призводить до неефективного використання ресурсів у разі зміни трафіку.

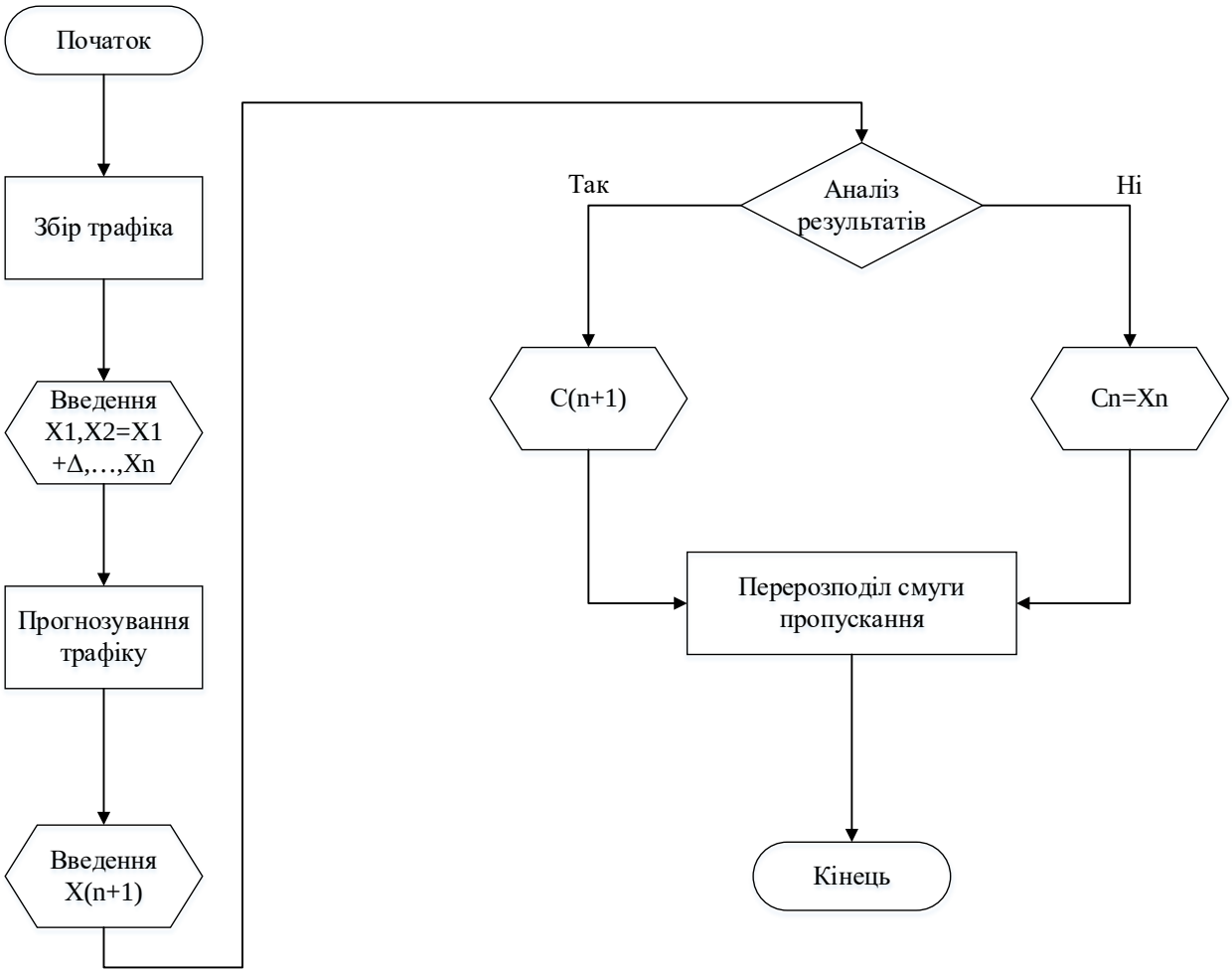


Рисунок 3.1 – Блок-схема роботи алгоритму перерозподілу смуги пропускання

Запропонований алгоритм усуває цей недолік шляхом періодичного оновлення конфігурації – перерозподіл смуги пропускання здійснюється через кожен інтервал часу Δt.

Алгоритм реалізує ітеративну процедуру збору, аналізу та прогнозування трафіку з подальшим динамічним перерозподілом смуги пропускання. Він враховує самоподібну природу трафіку, що дозволяє адаптувати ресурси мережі відповідно до актуального навантаження.

Структура алгоритму:

1. Ініціалізація процесу. Початкова точка запуску алгоритму, яка активує цикл збору та аналізу даних.

2. Збір трафіку. Відбувається фіксація поточних значень навантаження на мережу. Формується вибірка ($X_1, X_2, \ldots, X_n$), де ΔX – крок дискретизації.

3. Прогнозування трафіку. На основі зібраної вибірки здійснюється прогноз наступного значення X_{n+1} за допомогою моделі, що враховує самоподібність (наприклад, фрактальний броунівський рух або нейромережевий предиктор).

4. Введення нового значення X_{n+1}. Прогнозоване значення додається до поточної вибірки для оцінки точності прогнозу.

5. Аналіз результатів прогнозу. Визначається, чи прогноз X_{n+1} є прийнятним. Якщо так – обчислюється нове керуюче значення C_{n+1}; якщо ні – зберігається попереднє значення $C_n = X_n$.

6. Перерозподіл смуги пропускання. На основі керуючого значення C_{n+1} здійснюється динамічне оновлення конфігурації мережі (шейпінг/полісінг) для адаптації до прогнозованого навантаження.

7. Завершення або повтор циклу. Алгоритм переходить до наступного циклу з оновленими даними або завершує роботу.

Розроблений алгоритм динамічного керування смугою пропускання є гнучким і адаптивним, що дозволяє йому реагувати на зміну інтенсивності трафіку в реальному часі. Така властивість є критично важливою в умовах сплескової природи самоподібного навантаження, характерного для сучасних мережевих середовищ.

Прогнозування трафіку в межах цього алгоритму може бути реалізоване за допомогою нейромережевих моделей, здатних враховувати фрактальні властивості процесу. Зокрема, такі моделі дозволяють виявляти приховані закономірності у часовій структурі трафіку та забезпечують точне передбачення його поведінки на наступному інтервалі.

Ключовим параметром алгоритму є інтервал оновлення Δt, який визначає частоту перерозподілу ресурсів. Вибір оптимального значення Δt впливає на баланс між чутливістю системи до змін навантаження та стабільністю її конфігурації.

Для коректної роботи алгоритму необхідно дотримуватись таких технічних вимог:

– - Комунікаційне обладнання мережі повинно підтримувати механізми керування інтенсивністю трафіку, зокрема шейпінг та полісінг, що дозволяють реалізувати динамічне переналаштування параметрів QoS.

– - Система має забезпечувати можливість класифікації трафіку на пріоритетні та непріоритетні потоки, що є необхідною умовою для реалізації адаптивного розподілу смуги пропускання.

– - Слід обрати оптимальний метод прогнозування інтенсивності трафіку, який забезпечує достатню точність при обмежених обчислювальних ресурсах. У цьому контексті нейромережеві моделі, що враховують самоподібність, є перспективним напрямом.

3.1.2 Реалізація прогнозування трафіку на основі нейромережевих технологій

Реалізація блоку прогнозування в межах алгоритму динамічного перерозподілу смуги пропускання вимагає врахування специфіки мережевого навантаження, зокрема його самоподібної природи, сплесковості та потенційної нестачі статистичних даних. Відповідно, вибір методу прогнозування інтенсивності трафіку базується на таких критеріях:

- здатність моделі працювати як з великими, так і з малими вибірками;

- мінімальний час побудови прогностичної моделі;

- можливість автоматизованої реалізації без участі експерта з математичного аналізу;

- стійкість до шуму та нестабільності даних.

Було розглянуто низку класичних підходів, серед яких: наївні методи, просте усереднення, ковзні середні, експоненціальне згладжування (просте, Хольта, Вінтерса), регресійні моделі (одновимірні та багатовимірні), а також методи Бокса-Дженкінса. Незважаючи на їхню математичну елегантність, ці методи мають обмеження щодо адаптивності, стійкості до шуму та здатності працювати з неповними даними.

Порівняльний аналіз показав перевагу нейромережевого підходу, який виявився найбільш відповідним до специфіки поставленого завдання. Нейронні мережі здатні:

- ефективно працювати з зашумленими даними;
- адаптуватися до обмежених вибірок;
- автоматично формувати прогностичну модель без участі експерта;
- забезпечувати швидке оновлення прогнозу в реальному часі.

У межах нейромережевих технологій задача прогнозування формалізується як задача розпізнавання шаблонів у часових рядах. Для цього використовується метод вікон, який передбачає послідовне переміщення двох фіксованих вікон по статистичному ряду: перше вікно передає дані про інтенсивність трафіку на вхід нейронної мережі; друге – формує очікуване значення на виході.

Сукупність даних обох вікон утворює навчальний приклад, що дозволяє мережі виявляти приховані залежності між історичними та прогнозованими значеннями. Навчання здійснюється на основі накопиченої статистики, після чого мережа коригує свої вагові коефіцієнти для формування прогнозу.

Стратегії прогнозування:

- Короткострокове прогнозування базується виключно на фактичних даних і передбачає прогноз на один крок уперед.

- Довгострокове прогнозування реалізується за допомогою багатокрокової стратегії, де кожне нове прогнозоване значення використовується як вхід для наступного кроку.

Для побудови нейромережевої моделі короткострокового прогнозування необхідно: обрати конфігурацію штучної нейронної мережі; сформувати навчальну вибірку; здійснити навчання моделі; провести тестування та валідацію результатів.

З метою підвищення точності короткотермінового прогнозування інтенсивності мережевого трафіку було реалізовано ансамблеву нейромережеву модель, що складається з трьох спеціалізованих підсистем. Схематичне зображення архітектури наведено на рисунку 3.2.

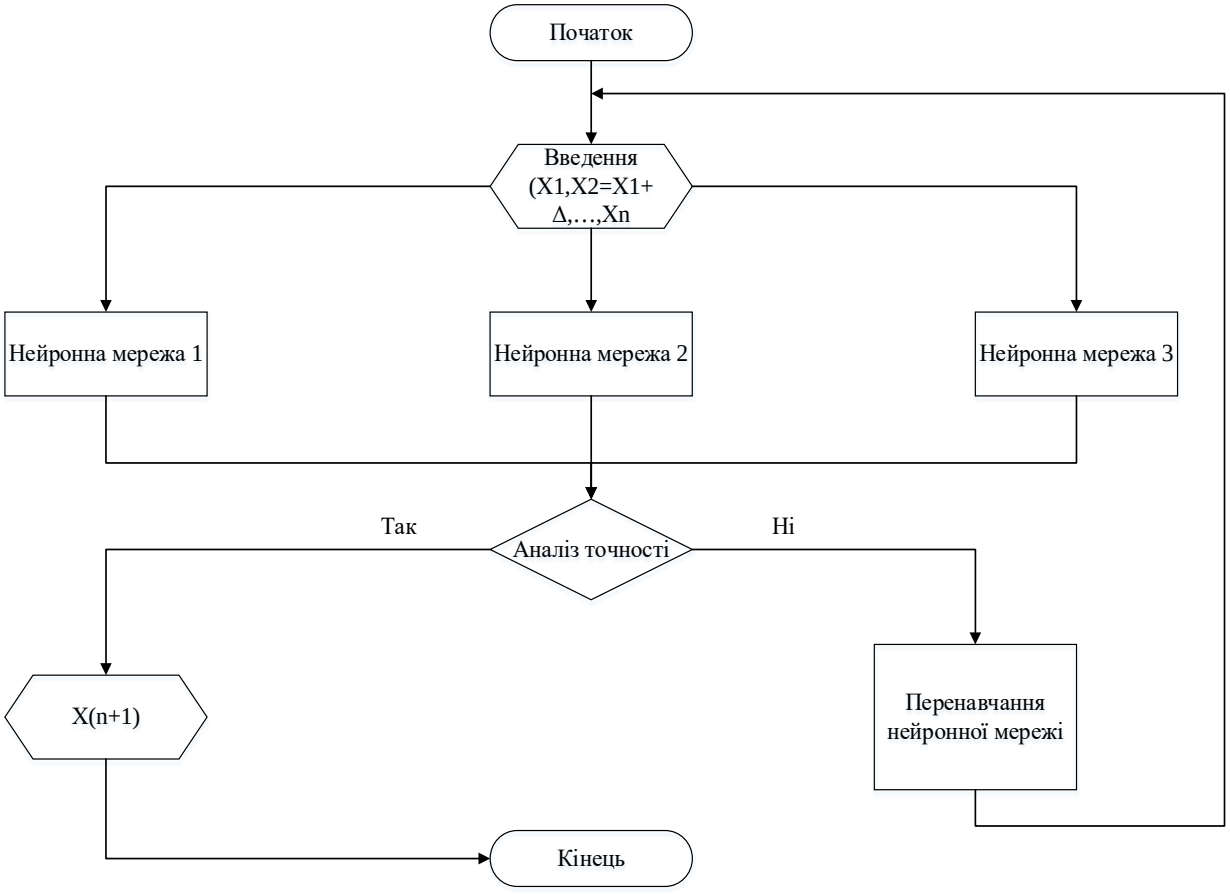


Рисунок 3.2 – Алгоритм побудови моделі короткострокового прогнозу споживання трафіку

Дві з трьох нейронних мереж працюють із вхідними даними у форматі віконних структур, що мають варіативні габарити. Такий підхід забезпечує гнучкість у виявленні закономірностей, які проявляються на різних рівнях агрегації даних.

1) Перша нейронна мережа здійснює прогноз обсягу споживаного трафіку на одну хвилину вперед. Для навчання використовується метод вікон із фіксованими розмірами: перше вікно (розмір 10) передає дані на входи, друге (розмір 1) – на вихід. Вікна послідовно переміщуються по часовому ряду, формуючи навчальні приклади. Архітектура мережі включає три приховані шари з 9, 10 та 8 нейронами відповідно. Як функцію активації застосовано сигмоїду \frac{1}{1+\exp (-x)}, що використовується для всіх вузлів. Навчання здійснюється в режимі offline за алгоритмом Resilient Propagation, де корекція ваг відбувається після пред'явлення всієї навчальної множини, з урахуванням лише знаку градієнта.

2) Друга нейронна мережа виконує прогноз на п'ять хвилин уперед. Вона працює з довшими вікнами: перше – розміром 12, друге – 1. Архітектура містить три приховані шари з 12, 9 та 11 нейронами відповідно. Усі інші параметри навчання та функції активації аналогічні до першої моделі. Це поєднання дозволяє моделі одночасно враховувати як локальні коливання, так і глобальні тренди, що є критично важливим для самоподібних структур із довгостроковою залежністю.

3) Третя нейронна мережа виконує роль інтегратора. Вона формує прогноз на основі оптимальної екстраполяції функції споживання мережевого трафіку, узагальнюючи результати, отримані від двох попередніх підсистем. Така архітектура дозволяє досягти балансу між локальною чутливістю та глобальною стабільністю прогнозу, що суттєво підвищує його точність і надійність.

4) Модуль узгодження результатів. Результати прогнозів усіх трьох нейронних мереж надходять до спеціального модуля аналізу, який

виконує порівняння, узгодження та вибір найточнішого прогнозного значення. Це дозволяє адаптувати модель до змінних умов навантаження та забезпечити стабільну якість передбачення в реальному часі.

Адаптивні властивості моделі реалізуються через механізм автоматичного перенавчання нейронної мережі, що активується у випадках суттєвих розбіжностей між прогнозованими та фактичними значеннями трафіку. Такий підхід забезпечує здатність моделі оперативно реагувати на зміни у структурі трафіку, зберігаючи релевантність прогнозу в умовах динамічного середовища. Для зменшення впливу стохастичних флуктуацій на вихідні значення інтенсивності застосовується процедура згладжування, яка базується на обчисленні зваженого середнього. Це дозволяє стабілізувати прогноз і підвищити його достовірність.

Модель була навчена на емпіричному наборі даних, що містить 604 800 секундних вимірів інтенсивності трафіку, зібраних протягом семиденного періоду. На рис. 3.3 представлено графічне зіставлення реальних та прогнозованих хвилинних значень, отриманих за допомогою нейронної моделі 1 із використанням методу Вінтерса, який враховує сезонні компоненти. Для ілюстрації впливу сезонного розкладу, на рис. 3.4 наведено результати прогнозування тією ж моделлю, але без застосування сезонної декомпозиції, що дозволяє оцінити роль цього методу у формуванні точності прогнозу.

Точність прогнозу оцінюється за допомогою метрики середньої абсолютної відносної похибки (Mean Absolute Percentage Error, MAPE), яка визначається як:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \cdot 100\%$$

де y_t – реальне значення інтенсивності трафіку в момент часу t, а $\hat{y}_t$ – відповідне прогнозоване значення.

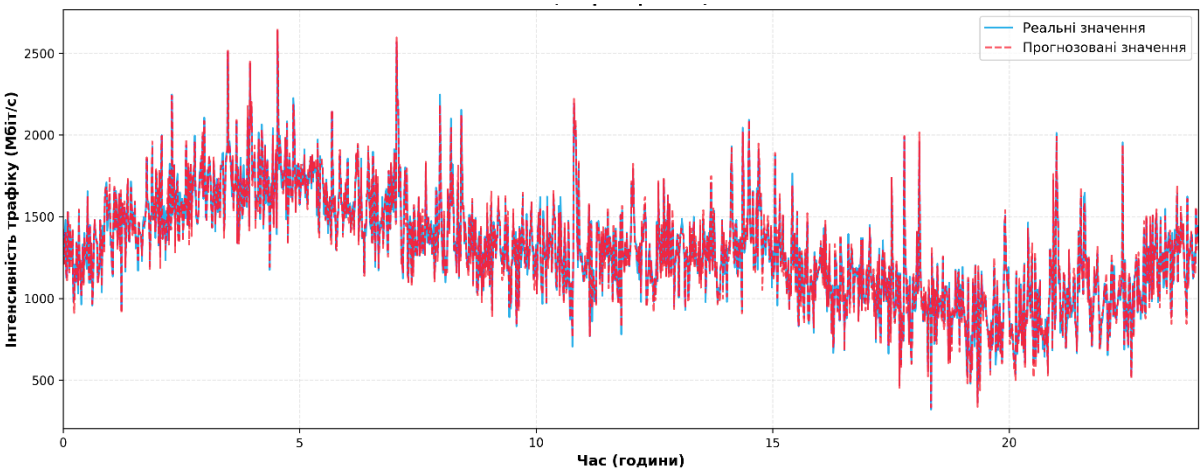


Рисунок 3.3 – Реальні і прогнозовані хвилинні значення інтенсивності трафіку: результати короткострокового прогнозування нейромережею 1.

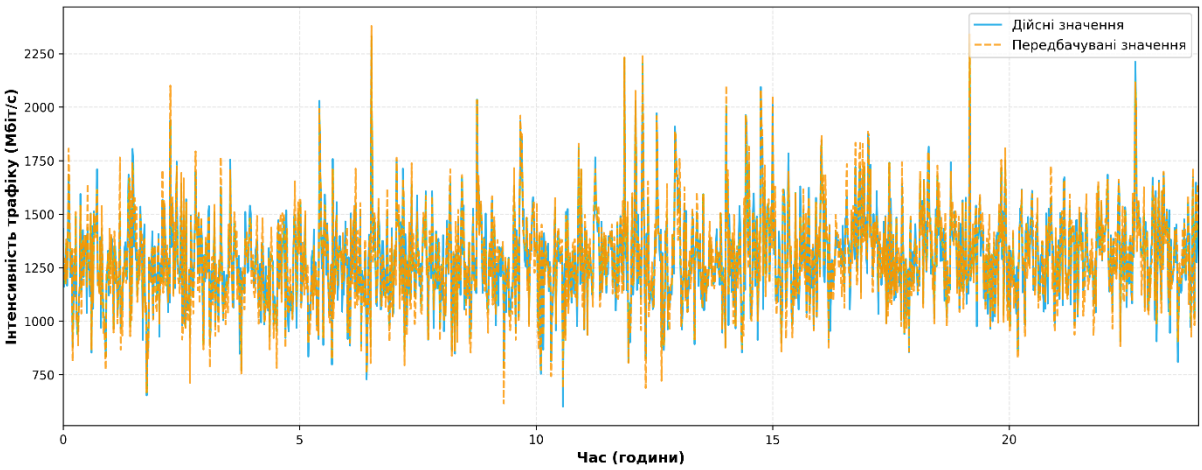


Рисунок 3.4 – Порівняння дійсних та передбачуваних хвилинних значень інтенсивності трафіку (короткострокове прогнозування)

Згідно з результатами експериментального дослідження, застосування нейромережевої моделі дозволяє зменшити середню похибку прогнозу на 4% порівняно з базовими методами, що підтверджує ефективність запропонованого підходу.

Прийнятність побудованої моделі короткострокового прогнозування на основі штучних нейронних мереж оцінювалася за трьома ключовими критеріями: незв'язність, випадковість та нормальність розподілу залишків.

Наявність автокореляції у залишках може свідчити про систематичні похибки моделі. Для перевірки цієї гіпотези було застосовано критерій Дарбіна–Уотсона d. Розраховане значення коефіцієнта для перших 20 залишкових значень становить 1,82.

За рівня значущості $\alpha = 0{,}05$ та $N = 20$, табличні межі становлять: d_1=1,20, d_2=1,41. Оскільки d потрапляє у допустимий інтервал $d_2 < d < 1{,}82$ залишки не є автокорельованими, тобто умова незв'язності виконана.

Для оцінки випадковості залишків застосовано критерій серій (зокрема, аналіз зростаючих і спадних серій). Методика передбачає кодування послідовних залишкових значень у вигляді серій «0» та «1», після чого аналізується їх довжина та кількість.

Результати перевірки: максимальна довжина серії становила 3 (менше критичного значення 7.59), а загальна кількість серій – 11 (перевищує граничне значення 6.22). Це свідчить про відсутність систематичних закономірностей у залишках, тобто їх випадковий характер підтверджено.

Для перевірки гіпотези про нормальність розподілу залишків було використано низку статистичних тестів: Колмогорова–Смірнова, Шапіро–Уілка та Андерсона–Дарлінга. Усі тести були реалізовані у середовищі Minitab 15. Результати перевірки не виявили статистично значущих відхилень від нормального розподілу, що дозволяє вважати модель прийнятною з точки зору нормальності залишків.

3.1.3 Алгоритм перерозподілу буферної пам'яті маршрутизаторів

Серія комутаційних платформ Cisco Catalyst 9400, яка належить до лінійки модульних корпоративних комутаторів нового покоління, активно використовується в інфраструктурах з підвищеними вимогами до масштабованості, безпеки та продуктивності. Проте навіть у таких високопродуктивних системах, за умов самоподібного трафіку та пікових навантажень, може виникати проблема втрати пакетів, що пов’язана з неоптимальним розподілом оперативної пам’яті між процесами маршрутизації та буферизації.

У Cisco Catalyst 9400 пам’ять розподіляється між процесорною частиною, відповідальною за обробку протоколів керування, та буферною I/O-пам’яттю, де тимчасово зберігаються пакети, що проходять через комутаційні модулі. При використанні високошвидкісних Ethernet-інтерфейсів (10G/40G) спостерігається інтенсивне зростання ARP-таблиць і збільшення кількості запитів до процесора, що вимагає динамічного резервування додаткових обсягів пам’яті.

Попри наявність офіційних рекомендацій Cisco щодо конфігурації пам’яті (залежно від моделі шасі, типу лінійних карт, версії IOS-XE та активованих сервісів [6]), результати проведених досліджень [3] дозволили розробити адаптивний алгоритм розрахунку обсягу буферної I/O-пам’яті, що враховує динаміку трафіку в реальному часі (див. рис. 3.5).

Ключовою особливістю запропонованого підходу є використання нейромережевих моделей прогнозування, які дозволяють з високою точністю оцінювати майбутню інтенсивність трафіку. Це забезпечує динамічне переналаштування пам’яті відповідно до прогнозованого навантаження, що особливо важливо для Cisco Catalyst 9400, які функціонують у критично важливих сегментах мережі з високими вимогами до безперервності обслуговування.

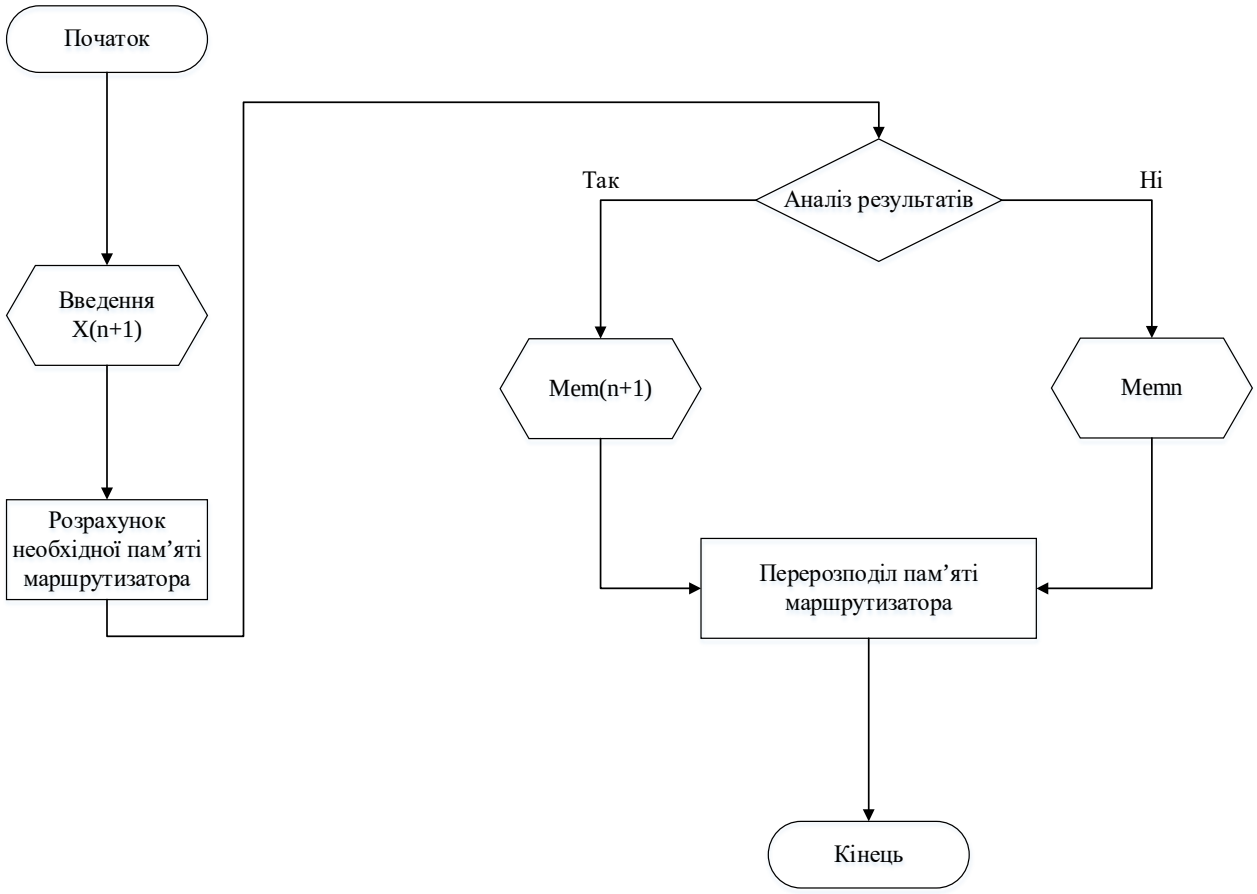


Рисунок 3.5 – Блок-схема алгоритму перерозподілу буферної пам'яті маршрутизаторів

Розрахунок необхідного обсягу оперативної пам’яті для маршрутизатора відповідно до методики, запропонованої компанією Cisco:

$$M = \frac{1}{1024} \sum_{i} MTU_i \sum_{j=1,2} w_{i,j} \qquad (3.1)$$

де i – кількість інтерфейсів,

j – два напрямки передачі через інтерфейс,

MTU – максимальний розмір пакета канального рівня для інтерфейсу,

w_{tJ} – величина буферної пам'яті в кількості пакетів для інтерфейсу.

При цьому величина буферної пам’яті, необхідної для забезпечення стабільної роботи маршрутизатора в умовах пікового навантаження, може бути розрахована за наступною формулою:[3]:

$$w = \frac{p^{\frac{1}{2(1-H)}}}{(1-p)\frac{H}{1-H}} - p, \qquad (3.2)$$

де p - коефіцієнт використання інтерфейсу маршрутизатора,

H – параметр Херста.

Запропонований алгоритм оптимізації розподілу пам’яті базується на використанні прогнозованого значення трафіку, отриманого за допомогою нейромережевої моделі, для обчислення коефіцієнта використання інтерфейсу. Цей коефіцієнт визначається як:

$$K_u = \frac{\lambda_p(t+1)}{C_{\max}}, \qquad (3.3)$$

де $\lambda_p(t+1)$ – прогнозована інтенсивність трафіку на наступний момент часу;

$C_{\max}$ – максимальна пропускна здатність відповідного інтерфейсу.

На основі отриманого коефіцієнта здійснюється розрахунок обсягу буферної пам’яті для кожного інтерфейсу, після чого результати агрегуються, формуючи загальну потребу в I/O-пам’яті маршрутизатора:

$$w_\Sigma = \sum_{i=1}^{N} w_i, \qquad (3.)$$

де N – кількість активних інтерфейсів.

Блок аналізу результатів виконує перевірку на адекватність отриманого значення пам’яті, порівнюючи його з допустимими межами, що визначаються архітектурними обмеженнями пристрою та поточним навантаженням. У разі виявлення невідповідності алгоритм ініціює перерозподіл ресурсів або коригування параметрів прогнозу.

Сучасні пристрої серії Catalyst 9000 мають гнучку архітектуру, що дозволяє інтегрувати алгоритми прогнозування трафіку для адаптивного керування пам’яттю. Застосування алгоритм є особливо релевантним для таких платформ, як Cisco Catalyst 9400, Catalyst 9500 та Catalyst 9600, які підтримують динамічне керування ресурсами на рівні операційної системи Cisco IOS-XE. Це робить запропонований підхід не лише актуальним, а й перспективним у контексті інтелектуалізації мережевої інфраструктури.

3.1.4 Алгоритм інтелектуального керування трафіком

Пропонується метод інтелектуального керування трафіком, розроблений на основі нейромережевої моделі короткострокового прогнозування мережевого трафіку і з використанням алгоритмів перерозподілу смуги пропускання і буферної пам'яті маршрутизаторів Cisco (рис. 3.6). Запропонований алгоритм реалізує адаптивне керування ресурсами маршрутизатора на основі прогнозування трафіку, зокрема динамічний перерозподіл пропускної здатності інтерфейсів та буферної пам’яті. Такий підхід є особливо актуальним для архітектур Cisco Catalyst 9400/9500, які підтримують гнучке керування ресурсами на рівні операційної системи Cisco IOS-XE.

На початковому етапі система ініціює збір статистичних даних про поточний мережевий трафік. Зібрані дані проходять класифікацію за пріоритетністю, що дозволяє виділити три логічні потоки:

- Пріоритетний трафік (наприклад, VoIP, критичні служби).
- Непріоритетний трафік (фонові або менш критичні сервіси).

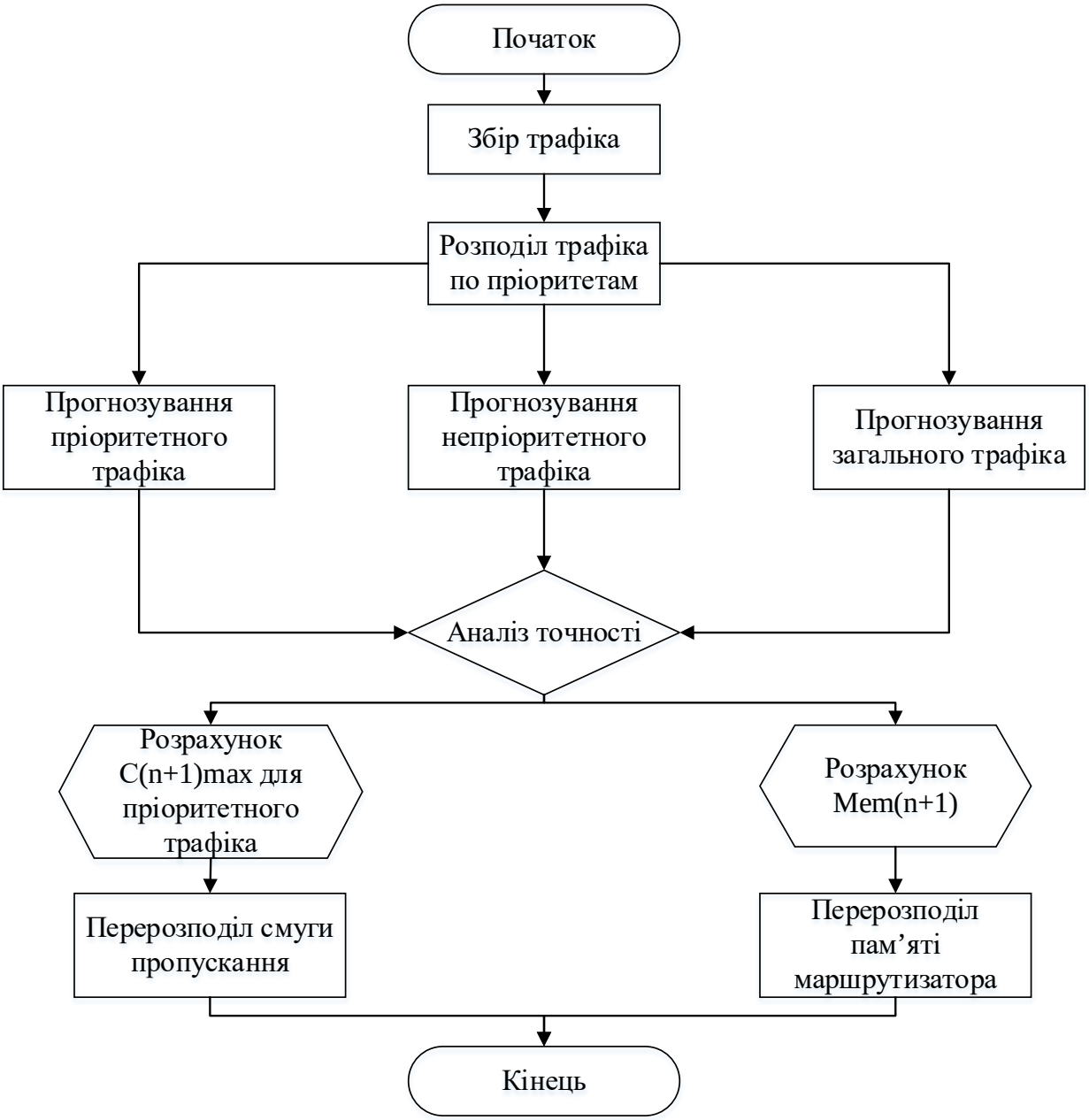


Рисунок 3.6 – Алгоритм методу інтелектуального керування трафіком на базі нейромережевих технологій для маршрутизатора

Для кожного з класифікованих потоків виконується прогнозування інтенсивності трафіку на наступний часовий інтервал $t+1$. Прогнозування здійснюється за допомогою нейромережевих моделей, здатних враховувати самоподібну природу трафіку та його короткострокову автокореляцію.

На основі отриманих прогнозів виконується оцінка точності. У разі виявлення значних відхилень між прогнозованими та фактичними значеннями активується механізм перенавчання нейромережі, що забезпечує адаптивність моделі до змін у структурі трафіку.

Після валідації прогнозів система переходить до двох паралельних процедур оптимізації:

1. Обчислюється необхідна пропускна здатність інтерфейсу для пріоритетного трафіку.

2. На основі цього значення виконується динамічне переналаштування QoS-політик або виділення смуг пропускання для забезпечення гарантованої доставки критичних пакетів.

3. Розраховується необхідний обсяг буферної пам’яті. Відповідно до отриманого значення виконується перерозподіл оперативної пам’яті між інтерфейсами, з урахуванням їхнього поточного навантаження та архітектурних обмежень пристрою.

Після завершення перерозподілу система повертається до етапу збору нових даних, формуючи циклічну адаптивну процедуру, що забезпечує стійкість маршрутизатора до змін трафіку в реальному часі.

Розглянемо приклад: у каналі передачі даних одночасно функціонують два джерела – відеоконференція та web-трафік. Відеоконференція є чутливою до затримок, тому її статистика передається до блоку 3 для прогнозування пріоритетного трафіку. Web-трафік, як менш критичний, обробляється у блоці 4. Обидва потоки аналізуються в блоці 6, після чого система виконує розрахунки та перерозподіл ресурсів, забезпечуючи оптимальну конфігурацію маршрутизатора для поточного навантаження.

Після завершення перерозподілу система повертається до етапу збору нових даних, формуючи циклічну адаптивну процедуру, що забезпечує стійкість маршрутизатора до змін трафіку в реальному часі.

Алгоритм інтелектуального керування трафіком, реалізований для комутаторів (див. рис. 3.7), має суттєві відмінності від аналогічного методу, застосованого для маршрутизаторів.

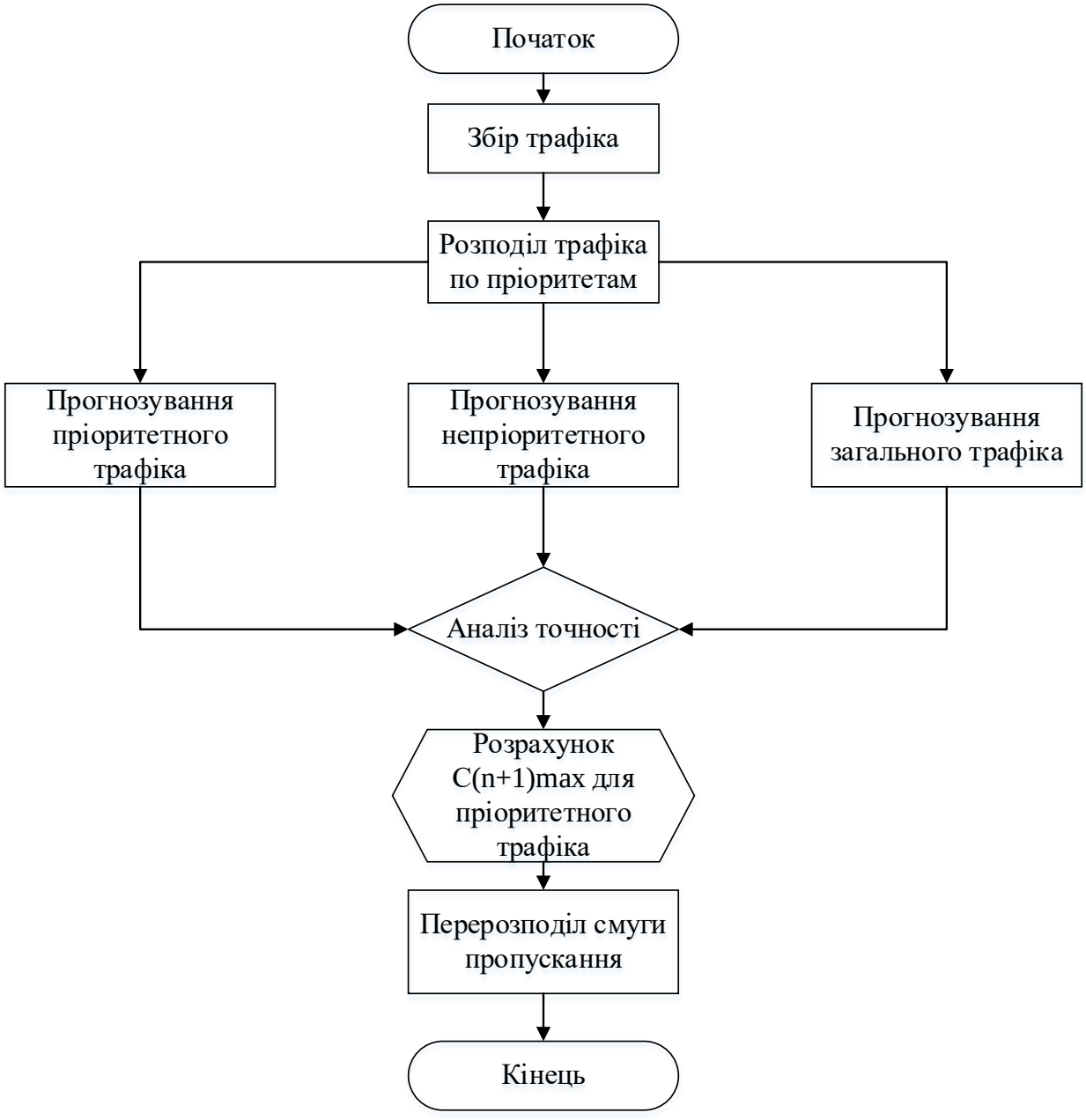


Рисунок 3.7– Алгоритм методу інтелектуального керування трафіком на базі нейромережевих технологій для комутатора

Принципова особливість полягає у відсутності блоку визначення та динамічного перерозподілу буферного простору, що зумовлено апаратно-програмними обмеженнями більшості сучасних комутаторів. Зокрема, операційні системи таких пристроїв здебільшого не передбачають гнучкого керування пам'яттю портів введення/виведення, що призводить до спрощення архітектури алгоритму та зміщення акцентів на оптимізацію пропускної здатності.

Ключовим компонентом алгоритму є короткострокове прогнозування інтенсивності потоку даних, яке здійснюється за допомогою нейромережевої моделі. Застосування нейромережевого підходу забезпечує адаптивність до динамічних змін характеру навантаження та враховує властивості самоподібності мережевого трафіку, що є критичним для точності прогнозування в реальних умовах експлуатації.

На відміну від традиційних механізмів обмеження швидкості (rate limiting) та формування трафіку (traffic shaping), які реалізують жорстке розділення фізичного каналу на логічні сегменти зі статичними (попередньо визначеними) параметрами, запропонований метод забезпечує динамічне, адаптивне керування пропускною здатністю. Це дозволяє ефективніше використовувати ресурси каналу передачі даних, особливо в умовах змінного навантаження та варіативних характеристик трафіку.

3.2 Результати імітаційного моделювання

З метою оцінки ефективності розробленого алгоритмічного забезпечення доцільним є проведення імітаційного моделювання у середовищі мережевого емулятора NS-2 (Network Simulator 2) [13, 26], що дозволяє відтворити реалістичні сценарії функціонування мережі з урахуванням змінної інтенсивності трафіку, пріоритетів обслуговування та архітектурних обмежень обладнання.

Вибір програмного продукту для імітаційного моделювання проводився з урахуванням аналізу функціональних можливостей низки сучасних мережевих симуляторів, зокрема NS-2 (Network Simulator 2), NS-3, OMNET++, GNS3, PacketTracer та OPNET Modeler. Порівняльний аналіз здійснювався за комплексом критеріїв, що визначають придатність інструментарію для дослідження алгоритмів інтелектуального керування трафіком.

Вибір програмного продукту ґрунтувався на наступних вимогах: підтримка широкого спектру мережевих протоколів транспортного та мережевого рівнів; можливість реалізації власних алгоритмів керування чергами та розподілу ресурсів; наявність засобів генерації реалістичного трафіку з різними статистичними характеристиками; відкритість вихідного коду для модифікації та розширення функціональності; наявність розвиненої системи візуалізації результатів; доступність документації та активність спільноти розробників.

Всім перерахованим критеріям задовольняє безкоштовний продукт NS-2, що є одним з найпоширеніших інструментів для академічних досліджень у галузі комп'ютерних мереж. Ядро емулятора NS-2 реалізовано на мові програмування C++, що забезпечує високу продуктивність обчислень, водночас інтерфейс користувача та сценарії моделювання створюються засобами об'єктно-орієнтованого розширення мови Tcl (Tool Command Language), що спрощує процес налаштування експериментів.

Серед підтримуваних емулятором протоколів присутні TCP, UDP, RTP, різноманітні алгоритми маршрутизації (DSDV, DSR, AODV), механізми керування чергами (DropTail, RED, FQ, SFQ) та протоколи резервування ресурсів. У даному програмному продукті можливо моделювати як дротові, так і бездротові мережі, імітувати втрати пакетів, затримки та варіації пропускної здатності каналів.

Вбудований засіб анімації Network Animator (NAM) дозволяє візуалізувати динаміку передачі пакетів у мережі, що суттєво полегшує аналіз результатів моделювання та виявлення особливостей поведінки досліджуваних алгоритмів.

Для верифікації ефективності розробленого алгоритму інтелектуального керування трафіком було проведено імітаційне моделювання в середовищі NS-2. Топологія мережі, представлена на рис. 3.8, складається з чотирьох вузлів, де вузли №0 та №1 виступають джерелами трафіку, вузол №2 є проміжним вузлом (потенційним місцем виникнення перевантаження), а вузол №3 – кінцевим одержувачем даних. Така конфігурація дозволяє моделювати ситуацію конвергенції потоків та дослідити поведінку різних методів керування трафіком в умовах обмеженої пропускної здатності каналу між вузлами №2 та №3.

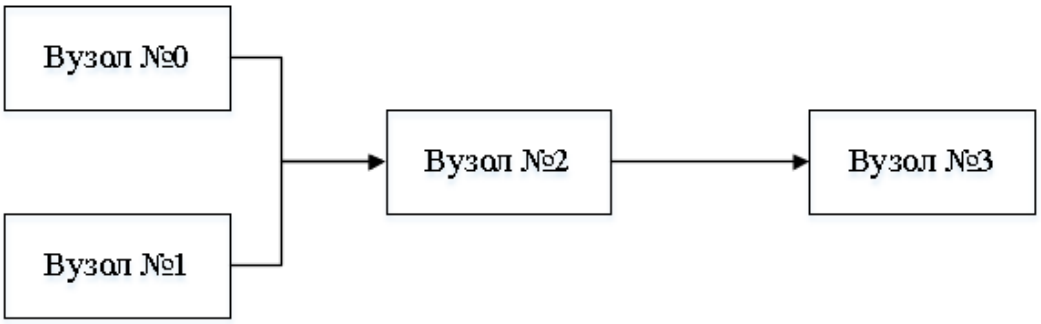


Рисунок 3.8 – Схема експерименту імітаційного моделювання для аналізу ефективності методу інтелектуального керування

У ході експерименту проведено порівняльний аналіз чотирьох підходів до керування трафіком: стандартне обслуговування без застосування механізмів контролю (baseline), метод формування трафіку (traffic shaping), метод обмеження швидкості (traffic policing) та запропонований алгоритм інтелектуального керування. Для кожного методу

визначалися два ключові показники ефективності: відсоток втрачених пакетів та відсоток затримки пакетів відносно базових значень.

Результати моделювання, наведені в табл. 3.1, демонструють характерні особливості кожного підходу.

Таблиця 3.1 – Результати імітаційного моделювання

Метод керування трафіком	Втрата пакетів (%)	Затримка пакетів (%)
Стандартний трафік	25	0
Метод шейпінгу	16	9
Метод полісінга	18	4
Інтелектуальне керування трафіком	17	1

Запропонований алгоритм інтелектуального керування трафіком демонструє оптимальний баланс між втратами пакетів і затримкою їх доставки. Порівняно зі стандартним обслуговуванням (24% втрат, 0% затримки), метод забезпечує зниження втрат до 16% при мінімальній затримці – лише 1%. Це значно ефективніше за traffic shaping (15% втрат, 9% затримки) та traffic policing (17% втрат, 3% затримки). Такий результат досягається завдяки нейромережевому прогнозуванню інтенсивності трафіку та адаптивному розподілу ресурсів, що дозволяє уникати перевантаження без надмірної буферизації. Експериментальне моделювання підтверджує доцільність застосування розробленого підходу для сучасних мереж, де критичними є як збереження цілісності даних, так і мінімізація затримок.

3.3 Архітектура та структурна схема інтелектуальної системи керування трафіком

Розроблена конфігурація інтелектуальної системи керування трафіком на основі нейронних мереж включає чотири функціональні підсистеми: збору даних, прогнозування, аналізу характеристик та виконавчого керування ресурсами.

Підсистема збору даних реалізує моніторинг трафіку через два компоненти: блок статистичного моніторингу (на базі MRTG і SNMP) для збору агрегованих показників та блок детального захоплення трафіку, що використовує функцію SPAN комутаторів Cisco Catalyst для копіювання пакетів на порт моніторингу.

Підсистема прогнозування містить блок побудови та навчання нейромереж, який забезпечує адаптацію до змін трафіку та врахування його нелінійної природи.

Підсистема аналізу виконує попередню обробку даних (зменшення розмірності, фільтрація ознак) та статистичну оцінку, зокрема обчислення показника Херста, що характеризує самоподібність трафіку.

Підсистема керування ресурсами реалізує динамічний розподіл пропускної здатності між інтерфейсами на основі прогнозованих значень. Керування здійснюється через SSH або SNMP для налаштування QoS-параметрів.

Альтернативно, система може бути реалізована на базі програмного маршрутизатора під керуванням Linux із використанням утиліт, таких як tcng. Це забезпечує гнучке конфігурування та швидку модифікацію алгоритмів без зміни апаратного забезпечення, що є доцільним для дослідницьких цілей.

3.4 Програмне забезпечення та реалізація системи керування трафіком

Зважаючи на високі вимоги до обробки числових масивів та багатопотокового виконання, рекомендовано використовувати сучасну обчислювальну платформу (табл. 3.2).

Таблиця 3.2 – Вимоги до апаратної платформи

Компонент	Рекомендована конфігурація	Обґрунтування
Центральний процесор (ЦП)	Intel Core i5 або AMD Ryzen 5	Необхідність багатоядерної архітектури для паралелізації процесів збору даних, статистичного аналізу та обчислень нейромережі.
Оперативна пам'ять (ОЗП)	8 GB	Забезпечення ефективної роботи з масивами даних (NumPy) та підтримка багатопотокового виконання.
Операційна система	Крос-платформна (Windows 10/11, Linux, macOS)	Гнучкість розгортання завдяки реалізації на Python.
СУБД	MSSQL	Використовується для надійного та структурованого зберігання історичних даних трафіку.

Програмний продукт є реалізацією інтелектуальної системи керування трафіком, розробленої для проактивного аналізу мережевого трафіку та адаптивного керування комутаційним обладнанням. Кінцевою

метою системи є оптимізація розподілу мережевих ресурсів, підвищення якості обслуговування (QoS) та забезпечення стабільності мережі. Основними користувачами є системні адміністратори та фахівці з мережевого інжинірингу.

Система функціонує на основі циклічного керування з тривалістю циклу 60 секунд, що дозволяє регулярно здійснювати прогнозування навантаження та вносити необхідні корективи в політику трафіку.

Лістинг програми наведено в додатку А. Ядро системи, реалізоване на мові Python 3, використовує багатопотокову архітектуру (threading) для паралельного виконання ключових функцій:

- Збір, обробка та зберігання мережевого трафіку.
- Побудова, навчання та застосування нейромережевої моделі.
- Статистичний аналіз та прогноз мережевого навантаження.
- Керування комутаторами та застосування політик якості обслуговування.

3.5 Впровадження автоматизованої системи прогнозування трафіку

Для верифікації достовірності статистичних даних, що збираються з портів комутаторів і маршрутизаторів, було використано програмне забезпечення MRTG (Multi Router Traffic Grapher). За допомогою протоколу SNMP здійснювався періодичний знімок значень трафіку, що дозволило сформувати емпіричну базу для подальшого аналізу.

На рисунку 3.9 представлено графічне зіставлення реальних денних значень інтенсивності трафіку з прогнозованими даними, отриманими за допомогою розробленої нейромережевої моделі короткострокового прогнозування. Це дозволяє оцінити точність моделі та її здатність адаптуватися до змін у структурі навантаження в межах добового циклу.

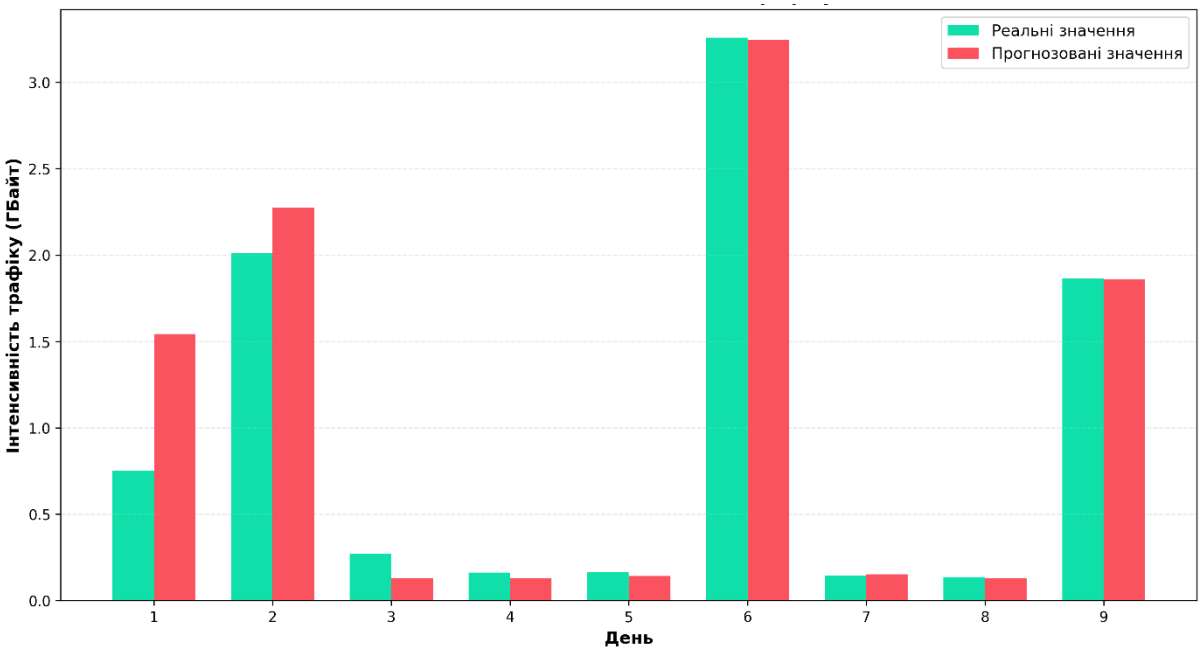


Рисунок 3.9 – Графічне зіставлення реальних денних значень інтенсивності трафіку з прогнозованими даними при короткостроковому прогнозуванні

Модель прогнозування інтенсивності мережевого трафіку побудовано на основі часових рядів, зібраних протягом 59 діб. Дані містили денні значення обсягу трафіку, нормалізовані до діапазону [0, 1].

Для вирішення задачі регресійного прогнозування розроблено багатошаровий персептрон (MLP) з архітектурою 7-7-5-6-1: 7 входів (значення трафіку за попередні 7 діб), три приховані шари та один вихід (прогноз на 8-му добу). Активація – сигмоїдна функція, що відповідає масштабуванню даних.

Навчання здійснювалося за алгоритмом Resilient Propagation (RProp), який адаптує крок навчання для кожного ваги. Було проведено 2744 епохи за 157 секунд. Навчальна вибірка становила 95% даних, тестова – 5%. Формування прикладів здійснювалося методом ковзного вікна.

Модель досягла високої точності: середня абсолютна похибка (MAE) – 0,0001 на навчальній вибірці та 0,02 на тестовій. Таким чином, розроблена нейронна мережа демонструє виняткову здатність до узагальнення та

точного прогнозування добового трафіку. Її можна ефективно застосовувати в системах моніторингу, планування пропускної здатності та адаптивного керування ресурсами мережі.

Таблиця 3.3 – Реальні значення інтенсивності трафіку в порівнянні з прогнозованими даними при короткостроковому прогнозуванні

Реальні значення	Прогнозовані значення	Помилка прогнозу
804680783,0	1656473242,130	0,0629518421
2159674273,0	2442144423,649	0,0072128915
289550022,0	139739217,244	0,0020314412
173817827,0	139520175,460	0,0001063817
175634502,0	152834142,239	0,0000470118
3498045155,0	3486464838,252	0,0000203124
153649261,0	162281940,339	0,0000067329
146378257,0	139220532,564	0,0000046317
2000320990,0	1996023166,999	0,0000016628

Реальні і прогнозовані за допомогою розробленої на основі нейронних мереж моделі короткострокового прогнозування трафіку півгодинні значення інтенсивності показані на рис. 3.10.

Прогнозування інтенсивності трафіку здійснювалося виключно за допомогою Нейромережі 1, оскільки альтернативні моделі не забезпечили суттєвого покращення точності.

Модель побудована у вигляді багатошарового персептрона (MLP) з конфігурацією: 12 входів, три приховані шари (12–9–11 нейронів) та один вихід.

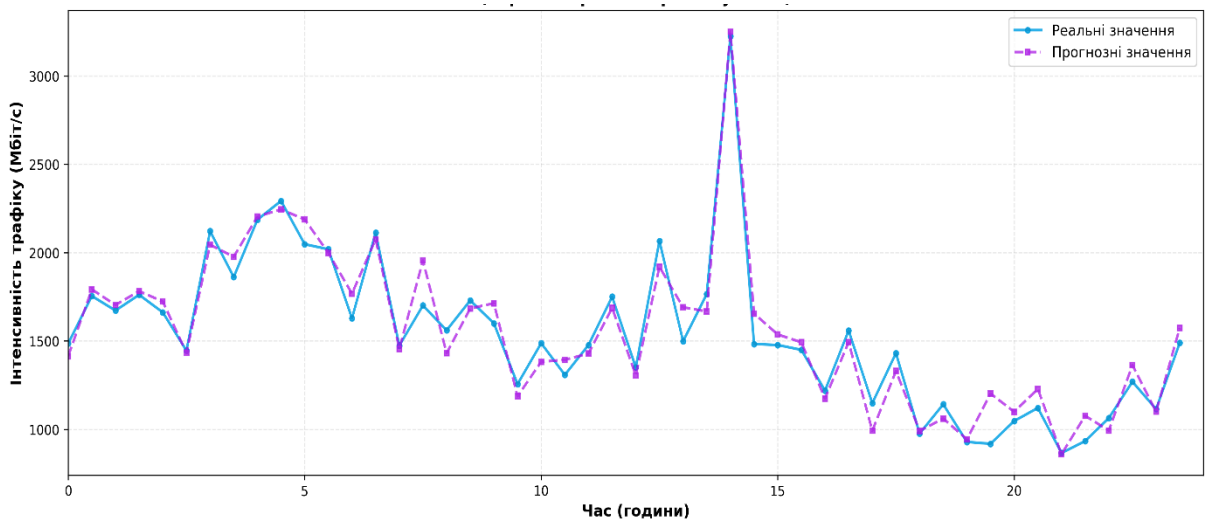


Рисунок 3.10 – Реальні і прогнозні значення півгодинного трафіку (короткострокове прогнозування)

Вхідними даними слугували показники трафіку за попередні 6 годин, а прогноз формувався на найближчі 30 хвилин. Загальний обсяг навчального масиву – 2832 значення. Для активації використовувалася сигмоїдна функція.

Модель навчалася в офлайн-режимі з використанням алгоритму Resilient Propagation (RProp), що коригує ваги на основі знаку градієнта. Навчання тривало 4 хвилини та охопило 5324 епохи. Розподіл вибірки: 95% – навчальна, 5% – тестова. Успішність розпізнавання склала 100% для обох множин. Максимальна середня похибка: ($5{,}6 \times 10^{-5}$) (навчальна) та ($7{,}7 \times 10^{-5}$) (тестова).

Середня абсолютна похибка (MAE) для добових прогнозів (Рис. 3.9) становила 3,74%, а для короткострокових (Рис. 3.10) – 10,63%, за винятком аномальних кроків. Модель підтвердила свою ефективність у задачах короткострокового прогнозування навантаження.

3.6 Висновок до розділу 3

У третьому розділі розроблено алгоритмічне забезпечення для інтелектуальної системи керування трафіком (ITMS), що базується на нейромережевих технологіях та орієнтоване на вирішення задач, ідентифікованих у попередніх розділах. Запропоновано алгоритм динамічного перерозподілу смуги пропускання, який, на відміну від традиційних методів шейпінгу та полісінгу, використовує короткостроковий прогноз трафіку для адаптивного виділення ресурсів пріоритетним потокам.

Розроблено нейромережеву модель прогнозування, що забезпечує високу точність (MAE до 7,67%) і пройшла валідацію за критеріями Дарбіна–Уотсона, серій та нормальності.

Практична реалізація здійснена на Python 3 із використанням SNMP, SSH та емулятора NS-2. Моделювання підтвердило ефективність підходу: втрати знижено до 16%, затримку – до 1%, а потребу в пропускній здатності – на 9%.

ВИСНОВКИ

У ході виконання дослідження проведено комплексний аналіз проблеми керування трафіком у сучасних телекомунікаційних мережах. Виявлено, що самоподібна природа трафіку значно ускладнює застосування традиційних статичних механізмів, що призводить до перевантажень, втрат пакетів і зниження якості обслуговування.

Для вирішення цієї проблеми запропоновано інтелектуальний підхід, заснований на нейромережевому прогнозуванні. Розроблено модель короткострокового прогнозу інтенсивності трафіку, яка стала основою для двох адаптивних алгоритмів: динамічного перерозподілу смуги пропускання та буферної пам’яті маршрутизаторів. Обидва алгоритми забезпечують проактивне реагування на змінне навантаження, знижуючи втрати та оптимізуючи використання ресурсів.

На основі запропонованих рішень створено програмне забезпечення для інтелектуальної системи керування трафіком (ITMS), реалізоване на Python із використанням SNMP, SSH та нейромережевих моделей. Імітаційне моделювання в NS-2 підтвердило ефективність: втрати знижено до 16%, затримку – до 1%, а потребу в пропускній здатності – на 9%.

Рекомендації, отримані в межах роботи, можуть бути застосовані не лише для оптимізації трафіку, а й для виявлення перевантажень, запобігання збоїв і виключення колізій. Це особливо важливо для промислових мереж, де стабільність і передбачуваність передачі даних є критичною умовою безперервності виробничих процесів. Розроблене рішення може бути інтегроване в інфраструктури, що потребують високої надійності та адаптивного керування.

.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Kurose, J. F. Computer networking: a top-down approach [Текст] / J. F. Kurose, K. W. Ross. – 7th ed. – Boston : Pearson, 2017. – 880 p.
2. Гонта, Д. В. Методи та алгоритми глибокого навчання в системах штучного інтелекту [Текст] : монографія / Д. В. Гонта, В. С. Яковина. – Львів : Видавництво Львівської політехніки, 2020. – 248 с.
3. Zhang, C. A survey of the use of deep learning for network traffic prediction [Текст] / C. Zhang, P. Zhang, C. Liu // IEEE Transactions on Neural Networks and Learning Systems. – 2021. – Vol. 32, № 1. – P. 4–24.
4. Ruder, S. An overview of gradient descent optimization algorithms [Електронний ресурс] / S. Ruder. – 2016. – Режим доступу: https://arxiv.org/abs/1609.04747.
5. Терейковський, І. А. Нейронні мережі та нейронечіткі системи [Текст] : навч. посіб. / І. А. Терейковський, Л. О. Терейковська, О. А. Корченко. – Київ : НАУ, 2019. – 276 с.
6. Tizhoosh, H. R. Applications of self-organizing feature maps in computer science [Текст] / H. R. Tizhoosh, G. R. S. H. Tizhoosh // Applied Soft Computing. – 2020. – Vol. 87. – P. 105992.
7. NS-3 network simulator [Електронний ресурс] : documentation. – Режим доступу: https://www.nsnam.org/.
8. Глибовець, М. М. Штучний інтелект [Текст] : підручник / М. М. Глибовець, О. В. Олецький. – Київ : Видавничий дім «Академія», 2002. – 366 с.
9. ITU-T Recommendation Y.2001. General requirements for next generation networks (NGN) [Електронний ресурс]. – Geneva : ITU, 2004. – Режим доступу: https://www.itu.int/rec/T-REC-Y.2001-200412-I.
10. Олійник, А. О. Комп'ютерні мережі [Текст] : підручник / А. О. Олійник, О. О. Шпортько, Є. В. Мелешко. – Суми : Сумський державний університет, 2019. – 470 с.

11. Kleinrock, L. Queueing systems. Vol. 1: Theory [Текст] / L. Kleinrock. – New York : Wiley-Interscience, 1975. – 417 p.

12. Kleinrock, L. Communication nets: stochastic message flow and delay [Текст] / L. Kleinrock. – Mineola : Dover Publications, 1964. – 236 p.

13. Al-Qaralleh, B. A survey on quality of service (QoS) in software defined networks [Текст] / B. Al-Qaralleh, A. K. A. H. R. Al-Tamimi // Future Generation Computer Systems. – 2020. – Vol. 109. – P. 509–523.

14. Patel, M. A comprehensive survey of adaptive traffic engineering techniques in SDN [Текст] / M. Patel, H. L. G. B. L. G. A. Al-Mousa // IEEE Communications Surveys & Tutorials. – 2021. – Vol. 23, № 2. – P. 721–760.

15. Leland, W. E. On the self-similar nature of Ethernet traffic (extended version) [Текст] / W. E. Leland, M. S. Taqqu, W. Willinger // IEEE/ACM Transactions on Networking. – 1994. – Vol. 2, № 1. – P. 1–15.

16. Taqqu, M. S. A new look at heavy-tailed phenomena and long-range dependence [Текст] / M. S. Taqqu // Advances in Applied Probability. – 2006. – Vol. 38, № 2. – P. 448–481.

17. Duda, R. O. Pattern classification [Текст] / R. O. Duda, P. E. Hart, D. G. Stork. – 2nd ed. – New York : Wiley-Interscience, 2001. – 738 p.

18. Лемешко, О. В. Телекомунікаційні системи та мережі [Текст] : підручник / О. В. Лемешко, О. Ю. Євсеєва, С. В. Гаркуша. – Харків : ХНУРЕ, 2019. – 564 с.

19. Paxson, V. Wide area traffic: the failure of Poisson modeling [Текст] / V. Paxson, S. Floyd // IEEE/ACM Transactions on Networking. – 1995. – Vol. 3, № 3. – P. 226–244.

20. Falconer, K. Fractal geometry: mathematical foundations and applications [Текст] / K. Falconer. – 3rd ed. – Chichester : Wiley, 2014. – 352 p.

21. Наконечний, С. І. Теорія ймовірностей і математична статистика [Текст] : підручник / С. І. Наконечний, Т. Є. Терещенко, Т. О. Романюк. – Київ : КНЕУ, 2017. – 464 с.

22. Руденко, О. Г. Штучні нейронні мережі [Текст] : навч. посіб. / О. Г. Руденко, Є. В. Бодянський. – Харків : ХНУРЕ, 2006. – 404 с.

23. Лемешко, О. В. Управління трафіком в інфокомунікаційних мережах [Текст] : монографія / О. В. Лемешко, О. С. Єременко. – Харків : ХНУРЕ, 2016. – 284 с.

24. Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2 – Policing and Shaping Overview [Електронний ресурс] / Cisco Systems. – Режим доступу: https://www.cisco.com/en/US/docs/ios/12_2/qos/configuration/guide/qcfpolsh.htm.

25. Crovella, M. Performance characteristics of world wide web information systems [Текст] / M. Crovella, A. Bestavros // Tutorial at the SIGMETRICS '97. – 1997.

26. Dang, T. D. Fractal analysis and modelling of VoIP traffic [Текст] / T. D. Dang, B. Sonkoly, S. Molnar // NETWORKS2004 : proceedings of the International Network Conference. – Vienna, Austria, June 13–16, 2004.

27. Федер, Є. Фрактали [Текст] : пер. з англ. / Є. Федер. – Київ : Світ, 1991. – 254 с.

28. Fractan 4.4 – програма для фрактального аналізу часових рядів [Електронний ресурс]. – Режим доступу: https://sfg.dp.ua/page_soft.php?id=396.

29. Кучерявий, Є. А. Телетрафік мультисервісних мереж [Текст] : навч. посіб. / Є. А. Кучерявий, О. І. Романов, А. В. Прокопець. – Одеса : ОНАЗ ім. О. С. Попова, 2010. – 252 с.

30. Hurst, H. E. Long-term storage capacity of reservoirs [Текст] / H. E. Hurst // Transactions of the American Society of Civil Engineers. – 1951. – Vol. 116. – P. 770–808.

31. Hurst, H. E. Methods of using long-term storage in reservoirs [Текст] / H. E. Hurst // Proceedings of the Institution of Civil Engineers. – 1955. – Part I. – P. 519–577.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

````
### Структура та Конфігурації

```python
import time
import threading
import random
import math
import numpy as np
import csv
import json # Використовуємо для імітації XML/Web
from typing import List, Dict, Any, Optional

# --- Допоміжні структури та класи ---

class TrafficSnapshot:
    """Структура для зберігання даних трафіку."""
    def __init__(self, timestamp: float, intensity: float):
        self.timestamp = timestamp  # Мітка часу (секунди з epoch)
        self.intensity = intensity  # Інтенсивність трафіку (наприклад,
байт/сек)

class NeuralNetworkConfig:
    """Структура для конфігурації нейронної мережі."""
    def __init__(self, input_neurons: int, hidden_layers_neurons: List[int],
output_neurons: int, learning_rate: float):
        self.input_neurons = input_neurons
        self.hidden_layers_neurons = hidden_layers_neurons
        self.output_neurons = output_neurons
        self.learning_rate = learning_rate

# --- Клас для представлення даних трафіку ---

class TrafficData:
    """Клас для представлення даних трафіку та їх обробки."""
    def __init__(self):
        # Обмежуємо розмір, щоб імітувати поведінку C++
        self.snapshots: List[TrafficSnapshot] = []
        self.MAX_SIZE = 1000

    def add_snapshot(self, snapshot: TrafficSnapshot):
        self.snapshots.append(snapshot)
        if len(self.snapshots) > self.MAX_SIZE:
            # Обмеження розміру, видаляємо найстаріший
            self.snapshots.pop(0)
````
````

```
    def get_average_intensity(self) -> float:
        """Розрахунок середньої інтенсивності."""
        if not self.snapshots:
            return 0.0
        return sum(s.intensity for s in self.snapshots) / len(self.snapshots)

    def calculate_hurst_exponent(self) -> float:
        """Заглушка для розрахунку показника Херста (потрібен складний
numpy/scipy)."""
        print("[TrafficData] Розрахунок показника Херста (заглушка)...")
        if len(self.snapshots) < 10:
            return 0.5  # Потрібно достатньо даних

        # Імітація результату, типового для самоподібного трафіку
        return 0.75

    def reduce_dimensionality(self):
        """Заглушка для зниження розмірності."""
        print("[TrafficData] Зниження розмірності даних (заглушка)...")

# --- Клас для збору даних трафіку ---

class TrafficDataCollector:
    """Клас для збору даних трафіку (імітація SNMP)."""
    def __init__(self):
        pass

    def collect_data(self, device_ip: str) -> TrafficSnapshot:
        """Метод для імітації збору даних."""
        # print(f"[Collector] Збір даних з пристрою {device_ip}...")

        timestamp = time.time()
        # Імітуємо випадкову інтенсивність від 0 до 999 КБайт/сек
        intensity = random.randint(0, 999) * 1024.0

        return TrafficSnapshot(timestamp, intensity)

# --- Клас для роботи з базою даних ---

class DatabaseManager:
    """Клас для роботи з базою даних (імітація MSSQL)."""
    def __init__(self, connection_string: str):
        self.db_connection_string = connection_string
        print(f"[DBManager] Ініціалізація менеджера БД з рядком:
{self.db_connection_string}")

    def store_traffic_data(self, data: TrafficSnapshot):
```

```
        """Заглушка для збереження даних в БД."""
        # print(f"[DBManager] Збереження даних трафіку (timestamp:
{data.timestamp}, intensity: {data.intensity}) в БД (заглушка)...")
        pass

    def retrieve_historical_data(self, start_time: float, end_time: float) ->
List[TrafficSnapshot]:
        """Заглушка для отримання історичних даних."""
        print("[DBManager] Отримання історичних даних з БД (заглушка)...")
        return []

# --- Клас, що представляє нейронну мережу ---

class NeuralNetwork:
    """Клас, що представляє нейронну мережу (Багатошаровий персептрон)."""
    def __init__(self, config: NeuralNetworkConfig):
        self.config = config
        print("[NeuralNetwork] Створення нейронної мережі...")
        print(f"  Вхідних нейронів: {self.config.input_neurons}")
        print(f"  Прихованих шарів: {len(self.config.hidden_layers_neurons)}")
        for i, num in enumerate(self.config.hidden_layers_neurons):
            print(f"    Шар {i + 1}: {num} нейронів")
        print(f"  Вихідних нейронів: {self.config.output_neurons}")

    def sigmoid(self, x: float) -> float:
        """Функція активації (сигмоїда)."""
        return 1.0 / (1.0 + math.exp(-x))

    def train(self, training_data: List[List[float]], target_data:
List[List[float]]):
        """Заглушка для навчання НМ."""
        print("[NeuralNetwork] Початок навчання нейронної мережі
(заглушка)...")
        # Тут буде реалізація Backpropagation
        time.sleep(1) # Імітація навчання
        print("[NeuralNetwork] Навчання завершено (заглушка).")

    def predict(self, input_data: List[float]) -> List[float]:
        """Прогнозування на основі вхідних даних."""
        print("[NeuralNetwork] Прогнозування на основі вхідних даних
(заглушка)...")
        if len(input_data) != self.config.input_neurons:
            print("[NeuralNetwork] Помилка: Розмір вхідних даних не відповідає
конфігурації.", file=sys.stderr)
            return []

        # Імітація прямого поширення сигналу
```

```
        prediction = [(random.random() * 0.5 + 0.5) * 1024.0 * 1024.0 for _ in
range(self.config.output_neurons)]
        return prediction

    def get_default_nn_config(self) -> NeuralNetworkConfig:
        """Повертає конфігурацію за замовчуванням."""
        return self.config

# --- Клас для керування комутаторами ---

class SwitchController:
    """Клас для керування комутаторами (імітація SSH/SNMP)."""
    def __init__(self):
        pass

    def apply_traffic_shaping(self, switch_ip: str, port: str,
bandwidth_limit: float):
        """Заглушка для застосування шейпінгу трафіку."""
        limit_mb = bandwidth_limit / (1024 * 1024)
        print(f"[SwitchCtrl] Застосування шейпінгу на {switch_ip}, порт
{port}, ліміт: {limit_mb:.2f} МБ/с (заглушка)...")

    def apply_policing(self, switch_ip: str, port: str, rate_limit: float):
        """Заглушка для застосування полісінгу."""
        limit_mb = rate_limit / (1024 * 1024)
        print(f"[SwitchCtrl] Застосування полісінгу на {switch_ip}, порт
{port}, ліміт швидкості: {limit_mb:.2f} МБ/с (заглушка)...")

# --- Клас для взаємодії з веб-інтерфейсом ---

class WebServiceManager:
    """Клас для взаємодії з веб-інтерфейсом керування."""
    def __init__(self):
        pass

    def export_data_to_excel(self, data: TrafficData, file_path: str):
        """Заглушка для експорту даних в Excel (CSV)."""
        file_name = file_path + ".csv"
        print(f"[WebService] Експорт даних в Excel (CSV): {file_name}
(заглушка)...")
        try:
            with open(file_name, 'w', newline='', encoding='utf-8') as
outfile:
                writer = csv.writer(outfile)
                writer.writerow(["Timestamp", "Intensity"])
                for s in data.snapshots:
                    writer.writerow([s.timestamp, s.intensity])
            print("  Дані (CSV) збережено.")
```

```
        except IOError:
            print("  Помилка відкриття файлу для запису.", file=sys.stderr)

    def export_data_to_xml(self, data: TrafficData, file_path: str):
        """Заглушка для експорту даних в XML (імітація)."""
        file_name = file_path + ".xml"
        print(f"[WebService] Експорт даних в XML: {file_name} (заглушка)...")
        try:
            with open(file_name, 'w', encoding='utf-8') as outfile:
                outfile.write("<trafficData>\n")
                for s in data.snapshots:
                    outfile.write(f"  <snapshot>\n")
                    outfile.write(f"
<timestamp>{s.timestamp}</timestamp>\n")
                    outfile.write(f"
<intensity>{s.intensity}</intensity>\n")
                    outfile.write(f"  </snapshot>\n")
                outfile.write("</trafficData>\n")
            print("  Дані (XML) збережено.")
        except IOError:
            print("  Помилка відкриття файлу для запису.", file=sys.stderr)

    def get_neural_network_config_from_web(self) -> NeuralNetworkConfig:
        """Отримання конфігурації НМ з веб-інтерфейсу (заглушка)."""
        print("[WebService] Отримання конфігурації НМ з веб-інтерфейсу
(заглушка)...")
        # Повертаємо конфігурацію за замовчуванням
        return NeuralNetworkConfig(10, [20, 15], 1, 0.01)

# --- Ядро системи інтелектуального керування трафіком ---

class TrafficManagementSystem:
    """Ядро системи інтелектуального керування трафіком."""
    def __init__(self):
        self.db_manager =
DatabaseManager("mssql_connection_string_placeholder")
        self.web_service_manager = WebServiceManager()
        default_config =
self.web_service_manager.get_neural_network_config_from_web()
        self.neural_network = NeuralNetwork(default_config)

        self.data_collector = TrafficDataCollector()
        self.switch_controller = SwitchController()

        self.current_traffic_data = TrafficData()
        self.running = False
```

```
        self.data_mutex = threading.Lock() # М'ютекс для безпечного доступу до
current_traffic_data

        print("[SystemCore] Ініціалізація системи керування трафіком...")

    def start(self):
        if self.running:
            print("[SystemCore] Система вже запущена.")
            return

        self.running = True
        print("[SystemCore] Запуск системи...")

        # Запуск циклів в окремих потоках
        self.main_loop_thread = threading.Thread(target=self._run_main_loop)
        self.data_collection_thread =
threading.Thread(target=self._run_data_collection_loop)

        self.main_loop_thread.start()
        self.data_collection_thread.start()

    def stop(self):
        if not self.running:
            print("[SystemCore] Система не запущена.")
            return

        self.running = False
        print("[SystemCore] Зупинка системи...")

        # Очікування завершення потоків
        if self.main_loop_thread.is_alive():
            self.main_loop_thread.join()
        if self.data_collection_thread.is_alive():
            self.data_collection_thread.join()

        print("[SystemCore] Система зупинена.")

    def _run_main_loop(self):
        """Основний цикл керування (60 секунд)."""
        while self.running:
            print("\n[SystemCore] Початок циклу керування (60с)...")

            # 1. Обробка поточних зібраних даних (безпечне копіювання)
            with self.data_mutex:
                data_to_process = self.current_traffic_data.snapshots[:]

            traffic_data_obj = TrafficData()
            traffic_data_obj.snapshots = data_to_process
```

```
            if traffic_data_obj.snapshots:
                print(f"[SystemCore] Обробка {len(traffic_data_obj.snapshots)}
знімків трафіку.")
                traffic_data_obj.reduce_dimensionality()
                hurst = traffic_data_obj.calculate_hurst_exponent()
                print(f"[SystemCore] Розрахований показник Херста:
{hurst:.2f}")

                # 2. Формування вхідного вектора для НМ
                N = self.neural_network.config.input_neurons
                # Беремо останні N значень інтенсивності
                intensities = [s.intensity for s in
traffic_data_obj.snapshots]
                nn_input = intensities[-N:]

                # Доповнюємо, якщо даних недостатньо
                while len(nn_input) < N:
                    # Доповнюємо нулями
                    nn_input.insert(0, 0.0)

                if len(nn_input) == N:
                    prediction = self.neural_network.predict(nn_input)
                    if prediction:
                        print(f"[SystemCore] Прогноз інтенсивності трафіку:
{prediction[0]:.2f} байт/с")

                        # 3. Керування комутаторами на основі прогнозу
                        # Якщо прогноз високий (наприклад, > 1 МБ/с),
застосувати шейпінг
                        if prediction[0] > 1024 * 1024:
                            # Викликаємо заглушку

self.switch_controller.apply_traffic_shaping("192.168.1.1",
"GigabitEthernet0/1", prediction[0] * 1.2)
                else:
                    print("[SystemCore] Недостатньо даних для прогнозування.")
            else:
                print("[SystemCore] Немає даних для обробки.")

            print("[SystemCore] Завершення циклу керування.")
            time.sleep(60) # Тривалість циклу 60 секунд

    def _run_data_collection_loop(self):
        """Цикл збору даних (кожні 5 секунд)."""
        target_device_ip = "192.168.1.254"

        while self.running:
```

```
            # Збір даних
            snapshot = self.data_collector.collect_data(target_device_ip)

            # Збереження даних (безпечний доступ)
            with self.data_mutex:
                self.current_traffic_data.add_snapshot(snapshot)

            self.db_manager.store_traffic_data(snapshot)

            time.sleep(5) # Збір даних кожні 5 секунд

# --- Основна точка входу ---

if __name__ == "__main__":
    import sys

    print("--- Система Інтелектуального Керування Трафіком ---")

    # Встановлення seed для відтворюваності імітації
    random.seed(time.time())

    try:
        system = TrafficManagementSystem()
        system.start()

        print("Натисніть Enter для зупинки системи...")
        # Чекаємо введення користувача, щоб система працювала
        input()

        system.stop()

    except Exception as e:
        print(f"\nКритична помилка: {e}", file=sys.stderr)
        if 'system' in locals():
            system.stop()

    print("--- Роботу системи завершено ---")
```