Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

здобувача Чукута Артема Олександровича
(ПІБ)

академічної групи 123-22ск-1
(шифр)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою Комп'ютерна інженерія
(офіційна назва)

на тему «Програмний застосунок для оптимізації маршрутизації мережевого трафіку в розподілених системах»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Нікулін С.Л.			
спеціальної частини	проф. Нікулін С.Л.			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)

«___» _____ 2025 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавр

здобувача Чукута А.О. **академічної групи** 123-22ск-1
(прізвище та ініціали) (шифр)

спеціальності 123 «Комп'ютерна інженерія»

за освітньо-професійною програмою «Комп'ютерна інженерія»
(офіційна назва)

на тему «Програмний застосунок для оптимізації маршрутизації мережевого трафіку в розподілених системах»

затверджену наказом ректора НТУ «Дніпровська політехніка» від 05.05.2025 № 336-с

Розділ	Зміст	Термін виконання
Стан питання і постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел показати актуальність завдання, сформулювати мету та задачі виконання кваліфікаційної роботи	10.02.2025
Технічні вимоги до об'єкту вивчення	Сформулювати найменування й призначення комп'ютерної системи, висунути технічні вимоги до неї	15.03.2025
Розробка програмного забезпечення	Розробити програму для оптимізації маршрутизації мережевого трафіку в розподілених системах	31.05.2025

Завдання видано _____ проф. Нікулін С.Л.
(підпис керівника) (прізвище, ініціали)

Дата видачі 25.02.2025

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ Чукут А.О.

РЕФЕРАТ

Пояснювальна записка: 66 с., 13 рис., 18 джерел, 1 додаток.

МЕРЕЖА, РОЗПОДІЛЕНА СИСТЕМА, ТРАФІК, КОНТРОЛЬ, МАРШРУТИЗАЦІЯ, ОПТИМІЗАЦІЯ, СИСТЕМА

Об’єкт вивчення – мережевий трафік в розподілених системах.

Мета роботи − створення програмного застосунку для оптимізації маршрутизації мережевого трафіку в розподілених системах.

Здійснено розробку програмного застосунку для оптимізації маршрутизації мережевого трафіку в розподілених системах. З метою вивчення актуальних методів та інструментів системного адміністрування розподілених комп'ютерних мереж. Основний акцент зроблено на розробці та вдосконаленні процедур управління мережевими ресурсами з метою забезпечення ефективної передачі даних.

Створений програмний застосунок призначений для системного адміністрування телекомунікаційних систем, сприяючи підвищенню ефективності використання мережі та оптимізації управління ресурсами в різноманітних умовах експлуатації, включаючи нештатні ситуації та збої.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – Програмне забезпечення;

API – Application Programming Interface (інтерфейс прикладного програмування);

DNS – Domain Name Service;

FTP – File Transfer Protocol;

GUI – Graphical User Interface;

LAN – Local Area Network;

ПК – Персональний комп’ютер;

КМ – Корпоративна (комп’ютерна) мережа;

ЛКМ – Локальна комп’ютерна мережа;

ОС – Операційна система;

IP-адреса – унікальний ідентифікатор комп'ютера локальної мережі;

ВСТУП

Актуальність дослідження. Сьогодні методи оптимізації маршрутизації спрямовані на підвищення ефективності мережі шляхом визначення та впровадження найкращих способів розподілу трафіку без зміни існуючої структури мережі. Ці методи стають особливо важливими при зростанні навантажень або тимчасових коливаннях трафіку, що можуть призвести до перевантаження каналів зв'язку, дозволяючи повністю або частково вирішити проблеми з продуктивністю. Суть оптимізації полягає в адаптації маршрутизації до поточного навантаження для кращого використання мережевих ресурсів, що, в свою чергу, покращує якість послуг (QoS). Завдання оптимізації маршрутизації полягає в тому, щоб для заданої структури мережі та існуючого попиту на трафік знайти таке рішення для його маршрутизації, яке забезпечить оптимальну QoS. Мірою QoS можуть виступати різні параметри продуктивності мережі, проте більшість визначень у літературі базуються на утилізації каналів зв'язку через її прямий вплив на затримку та втрату пакетів між маршрутизаторами. Кінцевою метою оптимізації є мінімізація максимального рівня утилізації каналів зв'язку в мережі, [1].

Сьогодні, поряд з традиційними підходами до трафік-інжинірингу в MPLS-TE, активно розвиваються технології Software-Defined Networking (SDN) та Network Function Virtualization (NFV), які пропонують більш гнучкі та динамічні механізми управління мережевими ресурсами та трафіком. SDN дозволяє централізовано контролювати поведінку мережевих пристроїв, що спрощує впровадження складних стратегій маршрутизації та оптимізації трафіку, [2]. NFV, у свою чергу, дозволяє віртуалізувати мережеві функції, що підвищує гнучкість та масштабованість мережі, [3].

У контексті багатошляхової маршрутизації, активно досліджуються та впроваджуються інтелектуальні алгоритми, що враховують не лише статичні параметри мережі, але й поточний стан завантаження, затримки та інші динамічні характеристики трафіку. Методи машинного навчання та штучного інтелекту знаходять все більше застосування для прогнозування трафіку,

виявлення аномалій та оптимізації маршрутів в режимі реального часу, що дозволяє значно підвищити якість обслуговування та ефективність використання мережевих ресурсів в умовах нестаціонарного трафіку та можливих збоїв. Також розвиваються протоколи та механізми, спрямовані на покращення QoS в IP-мережах, такі як DiffServ та IntServ, які дозволяють надавати різним типам трафіку різні рівні обслуговування.

Метою дослідження є створення програмного застосунку для оптимізації маршрутизації мережевого трафіку в розподілених системах.

Практичне значення цієї роботи полягає у створені програмного застосунку для оптимізації маршрутизації мережевого трафіку для збільшення ефективності застосування мережевих ресурсів.

1 СТАН ПИТАННЯ І ПОСТАНОВКА ЗАВДАННЯ

Вибір методів та об'єктів для моніторингу комп'ютерної мережі є комплексним процесом, що залежить від багатьох факторів, включаючи топологію та розмір мережі, наявні сервіси та служби, конфігурацію серверів та інше. На базовому рівні до найпоширеніших елементів моніторингу належать перевірка фізичної готовності обладнання, контроль стану (працездатності) запущених в мережі служб та сервісів, аналіз завантаженості апаратних ресурсів системи, детальне дослідження параметрів мережі (ефективності функціонування, продуктивності, завантаження) та перевірка специфічних параметрів, характерних для конкретних сервісів та служб (наприклад, наявності необхідних даних у таблицях баз даних, вмісту файлів журналів).

Початковим етапом будь-якого моніторингу є тестування фізичної доступності обладнання, оскільки відсутність підключення або блокування каналів зв'язку унеможливлює його використання. Така перевірка зазвичай включає відправлення ICMP-запитів (ping) та аналіз не лише факту отримання відповіді, але й часу проходження сигналу та кількості втрачених пакетів. Аномальні значення цих показників часто свідчать про серйозні проблеми в конфігурації мережі. Для виявлення деяких з цих проблем може бути використане трасування маршруту (traceroute), процес якого також підлягає автоматизації за наявності еталонних маршрутів.

Наступним важливим кроком є перевірка принципової працездатності критичних служб. Як правило, це передбачає встановлення TCP-з'єднання з відповідним портом сервера, на якому має функціонувати служба, та, можливо, виконання тестового запиту (наприклад, аутентифікації на поштовому сервері за протоколами SMTP або POP3, або запиту тестової веб-сторінки). У багатьох випадках бажано контролювати не лише факт відповіді служби, але й час її відгуку, що вже відноситься до наступного важливого завдання – перевірки навантаження. Окрім часу відгуку пристроїв та служб, для різних типів серверів існують інші ключові параметри для моніторингу,

такі як використання пам'яті та завантаженість центрального процесора (для веб-серверів, серверів баз даних), обсяг вільного місця на дискових накопичувачах (для файлових серверів) та більш специфічні показники, наприклад, статус принтерів на сервері друку.

Методи перевірки цих параметрів можуть варіюватися, проте одним з основних та широкодоступних є використання протоколу SNMP (Simple Network Management Protocol). Крім того, можуть застосовуватися специфічні інструменти, що надаються операційними системами обладнання. Наприклад, сучасні серверні версії Windows на системному рівні надають так звані лічильники продуктивності (performance counters), які дозволяють отримувати детальну інформацію про стан комп'ютера.

1.1 Огляд методів моніторингу мереж

1.1.1 Моніторинг мережевих процесів та стану апаратного блоку

Ефективне адміністрування складних комп'ютерних мереж сьогодні значною мірою залежить від використання автоматизованих систем, які забезпечують облік апаратного та програмного забезпечення, централізоване розгортання програмних продуктів та інструменти віддаленого аналізу продуктивності й діагностики проблем.

Автоматизований облік апаратних і програмних засобів дозволяє адміністраторам отримувати актуальну інформацію про всі комп'ютери в мережі, їхню конфігурацію та встановлене програмне забезпечення. Створення детальної бази даних ресурсів значно спрощує такі завдання, як інвентаризація, планування оновлень, виявлення невідповідностей та оптимізація використання наявних ресурсів (наприклад, визначення ПК з недостатнім обсягом пам'яті або дискового простору).

Централізоване розгортання та адміністрування програмного забезпечення є ключовим для зниження витрат часу та зусиль на встановлення та оновлення програм на великій кількості комп'ютерів. Системи управління можуть створювати пакети для автоматизованої розсилки програмного

забезпечення, а також забезпечувати централізоване адміністрування додатків, що запускаються з файлових серверів, надаючи користувачам зручний доступ до необхідного інструментарію з будь-якої точки мережі, [4].

Можливість віддаленого аналізу продуктивності та виникаючих проблем значно підвищує оперативність реагування на інциденти. Адміністратори можуть дистанційно керувати робочими станціями користувачів (передавати керування мишею та клавіатурою, переглядати екран) для діагностики та усунення несправностей без необхідності фізичної присутності. Детальна інформація про конфігурацію кожного комп'ютера, що зберігається в централізованій базі даних системи управління, є основою для ефективного віддаленого аналізу.

Прикладами традиційних засобів управління розподіленими системами є LANDesk Manager та Microsoft System Center Configuration Manager (раніше відомий як Systems Management Server), а серед класичних систем управління мережами можна назвати HP OpenView (тепер HPE iLO), SunNet Manager (застарілий) та IBM NetView (перейшов у лінійку продуктів Tivoli).

В області систем управління спостерігаються стійкі тенденції до конвергенції функцій управління мережами та IT-інфраструктурою в єдиних продуктах, а також до архітектур розподіленого управління, де для збору інформації та виконання керуючих дій використовуються численні агенти, встановлені на керованих пристроях.

Ефективне управління гетерогенними корпоративними мережами неможливе без орієнтації на відкриті стандарти, оскільки мережеве обладнання, програмне забезпечення управління та агенти для них розробляються великою кількістю різних виробників, [5].

Найбільш поширеним протоколом управління мережами залишається SNMP (Simple Network Management Protocol), який підтримується широким спектром обладнання. Його ключовими перевагами є простота, доступність та незалежність від виробника. Популярність SNMP свого часу стримала ширше впровадження CMIP (Common Management Information Protocol), протоколу

управління за моделлю OSI. SNMP розроблений для управління мережевими пристроями в IP-мережах та є частиною стека протоколів TCP/IP.

SNMP використовується для отримання інформації про стан, продуктивність та конфігурацію мережевих пристроїв, яка зберігається в MIB (Management Information Base), [5]. Існують стандарти, що визначають структуру MIB, типи змінних (об'єктів), їхні імена та права доступу. MIB містить різноманітну інформацію, включаючи IP- та MAC-адреси пристроїв, лічильники трафіку та помилок, стан інтерфейсів тощо. MIB має ієрархічну деревоподібну структуру, що складається зі стандартних та приватних (специфічних для виробників) піддерев.

Агент SNMP - це програмний компонент, що працює на керованому пристрої та забезпечує доступ менеджерів на керуючих станціях до значень змінних MIB для здійснення функцій управління та моніторингу, [5].

Основні операції управління ініціюються з керуючої станції, що мінімізує навантаження на керовані пристрої. Агент відповідає за збір даних про стан пристрою та їхню передачу менеджеру у відповідь на запити. SNMP є протоколом типу «запит-відповідь» і включає обмежений набір простих команд, [5].

Основні команди SNMP:

- Get-request, це запит менеджера до агента на отримання значення конкретного об'єкта за його ідентифікатором;
- GetNext-request, це запит менеджера до агента на отримання значення наступного об'єкта в таблиці об'єктів для послідовного перегляду;
- Get-response, це відповідь агента менеджеру на команди Get-request або GetNext-request, що містить значення запитуваного об'єкта;
- Trap, це асинхронне повідомлення агента менеджеру про виникнення важливої події (наприклад, помилки, зміни стану);
- Set, це запит менеджера до агента на встановлення значення певного об'єкта або умови, при виконанні якої агент повинен надіслати повідомлення (trap).

Версія SNMPv2c додала команду GetBulk, що дозволяє менеджеру ефективно отримувати великі обсяги даних (наприклад, вміст таблиць MIB) за один запит.

У сучасному ландшафті управління мережами, поряд з традиційним SNMP, все більшу популярність набувають більш сучасні протоколи та підходи, такі як NetConf та RESTCONF, які використовують структуровані дані (XML, JSON) та забезпечують кращу масштабованість та гнучкість. Також активно розвиваються програмно-визначені мережі (SDN) та мережеві функції віртуалізації (NFV), які пропонують централізоване управління мережевою інфраструктурою та віртуалізацію мережевих функцій, що значно спрощує адміністрування та автоматизацію. Інструменти управління стають більш інтелектуальними, використовуючи штучний інтелект (AI) та машинне навчання (ML) для прогнозування проблем, автоматизації рутинних завдань та оптимізації роботи мережі. Прикладами сучасних платформ управління мережами є Cisco DNA Center, SolarWinds Network Performance Monitor, PRTG Network Monitor та інші, які інтегрують широкий спектр функцій, включаючи моніторинг, управління конфігурацією, автоматизацію та аналітику.

1.1.2 Інтегровані системи управління та аналізу

1. Агенти SNMP та бази даних керуючої інформації (MIB).

В екосистемі протоколу SNMP (Simple Network Management Protocol) агенти відіграють ключову роль, забезпечуючи доступ менеджерів до інформації про стан та конфігурацію керованих мережевих пристроїв. Ця інформація організована в спеціалізованих базах даних, відомих як MIB (Management Information Base). На сьогодні існує кілька стандартів для MIB, серед яких основними є MIB-I та MIB-II, а також розширення для віддаленого моніторингу RMONMIB (Remote Network Monitoring MIB). Крім загальних стандартів, існують також специфічні MIB для пристроїв конкретних типів (наприклад, для комутаторів, маршрутизаторів, бездротових точок доступу) та приватні MIB, розроблені окремими виробниками обладнання, [2].

Перша базова специфікація MIB-I (визначена в RFC 1156) передбачала лише операції читання значень змінних, що відображали стан керованих пристроїв. Можливість зміни або встановлення значень об'єктів була введена пізніше, у специфікаціях MIB-II. MIB-I визначала до 114 об'єктів, об'єднаних у вісім основних груп, [4]:

- System, який містить загальну інформацію про фізичний пристрій, таку як ідентифікатор виробника, час останньої ініціалізації системи тощо.
- Interfaces надає параметри мережевих інтерфейсів пристрою, включаючи їхню кількість, типи, швидкість передачі даних, максимальний розмір пакету (MTU) та іншу статистику.
- AddressTranslationTable містить інформацію про відповідність між мережевими (наприклад, IP) та фізичними (наприклад, MAC) адресами, зокрема дані протоколу ARP (Address Resolution Protocol).
- InternetProtocol включає дані, пов'язані з протоколом IP (Internet Protocol), такі як адреси IP-шлюзів, хостів, а також статистику щодо IP-пакетів (кількість переданих, отриманих, перенаправлених, помилкових).
- ICMP містить дані, що стосуються протоколу обміну керуючими повідомленнями ICMP (Internet Control Message Protocol), включаючи статистику щодо різних типів ICMP-повідомлень.
- TCP надає інформацію, пов'язану з протоколом TCP (Transmission Control Protocol), наприклад, дані про TCP-з'єднання (кількість активних, пасивних, встановлених, закритих з'єднань).
- UDP містить дані, що стосуються протоколу UDP (User Datagram Protocol), включаючи кількість переданих, отриманих та помилкових UDP-дейтаграм.
- EGP надає інформацію, пов'язану з протоколом обміну маршрутною інформацією EGP (Exterior Gateway Protocol), який використовувався в мережі Інтернет для обміну маршрутами між автономними

системами (кількість отриманих повідомлень з помилками та без них).

Зі складу цих груп змінних стає очевидним, що стандарт MIB-I був розроблений з акцентом на управління маршрутизаторами, що підтримують протоколи стека TCP/IP. У версії MIB-II (визначена в RFC 1213), прийнятій у 1992 році, набір стандартних об'єктів був значно розширений (до 185), а кількість груп збільшилася до десяти, додавши, зокрема, групи SNMP (інформація про роботу самого протоколу SNMP) та System (перенесена з MIB-I з розширеннями). MIB-II стала більш комплексною та універсальною, охоплюючи ширший спектр мережевих пристроїв та параметрів.

З часом з'явилися нові версії протоколу SNMP (SNMPv2c, SNMPv3) та відповідні розширення MIB. SNMPv2c (RFC 1901-1908) ввела покращення в продуктивності, безпеці та додала нові типи даних. SNMPv3 (RFC 3410-3418) зосередилася на значному підвищенні безпеки за рахунок впровадження аутентифікації та шифрування.

Стандарт RMONMIB (Remote Network Monitoring MIB), визначений у серії RFC 2819, дозволяє здійснювати більш глибокий моніторинг мережевого трафіку на рівні каналів передачі даних, надаючи статистику за різними рівнями моделі OSI та полегшуючи віддалену діагностику мережевих проблем.

Сучасні MIB продовжують розвиватися, відображаючи нові технології та типи мережевих пристроїв. Існують MIB для бездротових мереж (наприклад, IEEE 802.11 MIB), оптичних мереж, віртуальних машин, систем зберігання даних та багатьох інших. Виробники обладнання також активно розробляють власні приватні MIB, що надають доступ до специфічних для їхніх пристроїв параметрів та функцій.

Управління MIB та робота з SNMP-агентами залишаються важливими аспектами адміністрування сучасних мереж, забезпечуючи централізований збір інформації для моніторингу, діагностики та управління мережевою інфраструктурою.

2. Агенти RMON, це розширені можливості віддаленого моніторингу мережі.

Специфікація RMON (Remote Network Monitoring) є потужним розширенням функціональності протоколу SNMP, спеціально розробленим для забезпечення ефективної віддаленої взаємодії з базою керуючої інформації (MIB). До появи RMON можливості SNMP для віддаленого управління мережевими пристроями були обмежені, оскільки протокол переважно використовувався для локального адміністрування. База RMONMIB (RFC 2819 та пов'язані RFC) надає значно поліпшений набір властивостей для віддаленого моніторингу, оскільки містить агреговану інформацію про керовані пристрої, що дозволяє уникнути передачі великих обсягів даних мережею.

Об'єкти RMONMIB включають розширені лічильники помилок у пакетах, гнучкіші інструменти для аналізу графічних трендів та статистики, більш потужні засоби фільтрації для захоплення та детального аналізу окремих пакетів, а також складніші умови для встановлення сигналів попередження (тригерів). Агенти RMONMIB є більш інтелектуальними порівняно з агентами MIB-I або MIB-II, оскільки вони виконують значну частину роботи з обробки інформації про пристрій безпосередньо на місці, що раніше покладалося на менеджери SNMP. Ці агенти можуть бути інтегровані в різноманітні комунікаційні пристрої (комутатори, маршрутизатори тощо), а також реалізовані як окремі програмні модулі, що працюють на універсальних комп'ютерах та ноутбуках (раннім прикладом може слугувати LANalyzer від Novell).

В ієрархії об'єктів MIB об'єкту RMON присвоєно ідентифікатор 16. Сам об'єкт RMON об'єднує десять основних груп об'єктів, [4]:

– Statistics містить поточні накопичені статистичні дані про характеристики мережевих пакетів, такі як кількість переданих та отриманих пакетів, кількість колізій (в Ethernet-мережах) та інші базові показники;

- History забезпечує збір та зберігання статистичних даних через певні проміжки часу для подальшого аналізу тенденцій їхніх змін та виявлення закономірностей;
- Alarms дозволяє встановлювати порогові значення для різних статистичних показників. У разі перевищення встановленого порогу агент RMON автоматично надсилає повідомлення (trap) менеджеру SNMP;
- Host містить інформацію про хости, активні в мережі, включаючи їхні MAC-адреси, кількість переданих та отриманих пакетів і байтів;
- HostTopN надає таблиці найбільш завантажених хостів мережі за різними критеріями (наприклад, за обсягом трафіку, кількістю помилок) за певний період часу;
- TrafficMatrix збирає статистику про інтенсивність трафіку між кожною парою хостів у мережі, представляючи ці дані у вигляді матриці, що полегшує аналіз комунікаційних потоків;
- Filter дозволяє визначати критерії фільтрації мережевих пакетів для подальшого аналізу або захоплення;
- PacketCapture надає засоби для захоплення певних мережевих пакетів на основі заданих фільтрів для детального дослідження трафіку;
- Event дозволяє визначати умови для реєстрації певних подій та генерації відповідних повідомлень (event log);
- TokenRing містить спеціальні об'єкти, специфічні для моніторингу мереж, що використовують протокол Token Ring.

Ці групи об'єктів мають чітку ієрархічну структуру, що відображається в їхніх числових ідентифікаторах в дереві MIB (наприклад, група Hosts має OID 1.3.6.1.2.1.16.4).

Загалом, стандарт RMONMIB визначає близько 200 об'єктів у цих десяти групах, детально описаних у двох основних документах - RFC 2819 (який замінив попередній RFC 1271 для мереж Ethernet) та RFC 1513 для мереж Token Ring.

Важливою відмінністю стандарту RMONMIB є його незалежність від протоколу мережевого рівня (на відміну від стандартів MIB-I та MIB-II, які були тісно орієнтовані на протоколи стека TCP/IP). Це робить RMON особливо зручним для використання в гетерогенних мережевих середовищах, де можуть одночасно застосовуватися різні протоколи мережевого рівня.

З часом були розроблені нові версії та розширення RMON, такі як RMON2 (RFC 2021, RFC 2022, RFC 2023, RFC 2024, RFC 2025), який розширив можливості моніторингу на рівень мережі та застосунків, дозволяючи аналізувати трафік за протоколами вищих рівнів (наприклад, HTTP, DNS).

Сучасні системи управління мережами часто інтегрують розширені можливості, що виходять за рамки стандартів RMON та RMON2, використовуючи власні механізми для глибокого аналізу трафіку, продуктивності застосунків та поведінки користувачів. Проте, базові принципи віддаленого моніторингу та концепція агентів, які обробляють дані на місці та надають агреговану інформацію керуючим станціям, залишаються актуальними в сучасних рішеннях для управління мережами. Технології аналізу мережевого трафіку продовжують розвиватися, використовуючи методи машинного навчання та штучного інтелекту для виявлення аномалій, прогнозування проблем та оптимізації роботи мережі.

1.1.3 Спеціалізовані методи моніторингу

Аналізатори протоколів у дослідженні та діагностиці мереж.

Одним з найефективніших та найпоширеніших інструментів для глибокого дослідження комп'ютерних мереж є аналізатори протоколів (також відомі як мережеві сніфери або пакетні аналізатори). Процес аналізу протоколів полягає в захопленні мережевих пакетів, що циркулюють у мережі та реалізують певні мережеві протоколи, з подальшим детальним вивченням їхнього вмісту. Результати такого аналізу є основою для обґрунтованої оцінки будь-яких компонентів мережі, оптимізації її продуктивності, а також ефективного пошуку та усунення різноманітних несправностей. Для

отримання репрезентативних даних, що відображають реальний вплив змін на мережу, аналіз протоколів необхідно проводити протягом достатнього періоду часу, [6].

Аналізатор протоколів може бути представлений як окремим спеціалізованим апаратним пристроєм, так і персональним комп'ютером, оснащеним спеціальною мережевою картою та відповідним програмним забезпеченням. Використовані мережева карта та програмне забезпечення повинні бути сумісними з топологією досліджуваної мережі. Аналізатор підключається до мережі як звичайний вузол, проте його ключова відмінність полягає в здатності приймати всі пакети даних, що передаються мережею, тоді як звичайна робоча станція обробляє лише пакети, адресовані безпосередньо їй.

Програмне забезпечення аналізатора зазвичай включає ядро, що забезпечує взаємодію з мережевим адаптером та декодування отриманих даних, а також додаткове програмне забезпечення, специфічне для типу досліджуваної мережі. Крім того, до складу аналізатора входить набір процедур декодування, орієнтованих на різні мережеві протоколи (наприклад, TCP/IP, Ethernet, Wi-Fi, Bluetooth). Деякі аналізатори можуть також містити експертні системи, які надають користувачеві рекомендації щодо подальших дій, інтерпретації результатів вимірювань та способів усунення певних мережевих несправностей.

Незважаючи на різноманітність аналізаторів протоколів, представлених на ринку, можна виділити ряд загальних характеристик, властивих більшості з них, [6]:

Користувацький інтерфейс: Сучасні аналізатори протоколів мають розвинений та інтуїтивно зрозумілий графічний інтерфейс користувача, часто на базі Windows або кросплатформних фреймворків. Цей інтерфейс дозволяє користувачеві візуалізувати результати аналізу інтенсивності трафіку, отримувати миттєві та усереднені статистичні оцінки продуктивності мережі, моделювати різні події та критичні ситуації для відстеження їхнього впливу, а

також декодувати протоколи різних рівнів моделі OSI та представляти вміст пакетів у зручному для аналізу форматі.

Буфер захоплення: Обсяг буфера захоплення може варіюватися залежно від моделі аналізатора. Буфер може бути розташований як на спеціалізованій мережевій карті, так і в оперативній пам'яті комп'ютера, на якому запущено аналізатор. Використання буфера на мережевій карті може забезпечити вищу швидкість захоплення завдяки апаратній реалізації управління, але зазвичай призводить до збільшення вартості пристрою. Недостатня продуктивність процедури захоплення може призвести до втрати частини пакетів та унеможливити коректний аналіз. Обсяг буфера визначає тривалість та репрезентативність вибірки захоплених даних. Незалежно від розміру буфера, рано чи пізно він заповнюється, після чого захоплення може припинитися або початися перезапис з початку буфера. Сучасні аналізатори часто пропонують різні режими керування буфером (наприклад, циклічний буфер).

Фільтри: Механізми фільтрації дозволяють користувачеві контролювати процес захоплення даних, значно заощаджуючи простір буфера та спрощуючи подальший аналіз. На основі заданих критеріїв фільтрації (значень певних полів у заголовках пакетів), пакети можуть бути проігноровані або записані в буфер захоплення. Використання фільтрів значно прискорює процес аналізу, оскільки виключає необхідність перегляду нерелевантних пакетів. Сучасні аналізатори пропонують складні та гнучкі можливості фільтрації за різними протоколами та полями.

Тригери (перемикачі): Тригери визначаються оператором як умови початку та завершення процесу захоплення даних. Ці умови можуть включати ручні команди запуску та зупинки, заданий час доби, тривалість захоплення, появу певних значень у кадрах даних або комбінації цих параметрів. Використання тригерів разом з фільтрами дозволяє проводити більш детальний та точний аналіз, а також ефективніше використовувати обмежений обсяг буфера захоплення, зосереджуючись на конкретних періодах часу або подіях.

Пошук: Більшість сучасних аналізаторів протоколів надають функцію пошуку необхідної інформації у вже захоплених даних, що знаходяться в буфері. Це дозволяє автоматизувати процес перегляду вмісту пакетів та знаходити дані за заданими критеріями (наприклад, за IP-адресою, портом, певним текстом у payload). На відміну від фільтрів, які застосовуються до вхідного потоку даних, функції пошуку використовуються для аналізу вже накопичених даних.

Методологія проведення аналізу протоколів зазвичай включає наступні етапи, [4]:

1. Захоплення даних – це збір мережевих пакетів протягом певного періоду часу або за певних умов.
2. Перегляд захоплених даних – це візуальне ознайомлення зі списком захоплених пакетів, їхніми основними характеристиками (час, джерело, призначення, протокол, розмір).
3. Дослідження інформації – це детальне вивчення вмісту окремих пакетів, декодування протоколів різних рівнів для розуміння обміну даними між мережевими пристроями.
4. Відстеження помилок – більшість аналізаторів надають можливості виявлення та ідентифікації різних типів мережевих помилок, а також визначення станції, що надіслала пакет з помилкою. Сучасні інструменти можуть автоматично виявляти поширені мережеві проблеми.
5. Аналіз продуктивності – це розрахунок ключових метрик продуктивності мережі, таких як коефіцієнт використання пропускної здатності, середній час реакції на запити, затримки, втрати пакетів.
6. Ретельне дослідження окремих ділянок мережі або конкретних комунікаційних сесій. Цей етап є більш цілеспрямованим і деталізація його залежить від конкретних цілей аналізу та виявлених проблем.

Зазвичай базовий процес аналізу протоколів може займати від кількох годин до одного-двох робочих днів, залежно від складності мережі та мети дослідження.

Сучасні аналізатори протоколів стали ще більш потужними та функціональними. Вони підтримують широкий спектр сучасних мережевих протоколів, включаючи новітні версії TCP/IP (IPv6), бездротові протоколи (Wi-Fi 6/6E/7, Bluetooth), протоколи Інтернету речей (IoT), хмарні протоколи та багато іншого. З'явилися аналізатори з розширеними можливостями візуалізації даних, автоматичного аналізу та виявлення аномалій з використанням машинного навчання. Інтеграція з іншими інструментами мережевого моніторингу та управління також стала більш тісною. Прикладами сучасних популярних аналізаторів протоколів є Wireshark (безкоштовний та з відкритим кодом), SolarWinds Network Packet Analyzer, tcpdump (консольний інструмент для Linux та macOS), Fiddler (спеціалізований для веб-трафіку) та інші комерційні рішення. Розвиток хмарних технологій також призвів до появи хмарних аналізаторів протоколів, які дозволяють аналізувати трафік у віртуальних мережах та хмарних середовищах.

Сертифікація та тестування кабельних мереж.

Для забезпечення надійної роботи комп'ютерних мереж важливим етапом є сертифікація та тестування кабельної інфраструктури. До обладнання, призначеного для цих цілей, належать мережеві аналізатори, кабельні сертифікатори, кабельні сканери та кабельні тестери. Перед розглядом цих пристроїв варто ознайомитися з основними електромагнітними характеристиками кабельних систем, які безпосередньо впливають на якість передачі даних. До таких характеристик належать загасання сигналу, імпеданс (хвильовий опір), перехресні наведення між парами провідників (NEXT - Near-End Crosstalk) та рівень зовнішнього електромагнітного випромінювання.

Мережеві аналізатори є високоточними еталонними вимірювальними приладами, призначеними для комплексної діагностики та сертифікації кабелів і кабельних систем. Вони включають високостабільний частотний

генератор та чутливий вузькосмуговий приймач. Шляхом передачі сигналів різних частот в одну пару провідників та вимірювання сигналу в іншій парі, мережеві аналізатори здатні точно визначати такі параметри, як загасання сигналу та рівень перехресних наведень (NEXT). Це складні, великогабаритні та дорогі прилади, що вимагають спеціалізованої підготовки технічного персоналу для їхнього використання, [4].

Кабельні сертифікатори (які часто об'єднують функціональність кабельних сканерів, але з розширеними можливостями) є більш універсальними приладами, що дозволяють визначати довжину кабелю, рівень перехресних наведень (NEXT), загасання сигналу, схему розведення провідників, імпеданс, рівень електричних шумів, а також проводити оцінку отриманих результатів на відповідність існуючим стандартам (наприклад, категоріям Cat 5e, Cat 6, Cat 6a тощо). Вартість цих приладів може варіюватися в широкому діапазоні. На ринку представлено багато виробників кабельних сертифікаторів, серед яких Fluke Networks є одним з лідерів. Сучасні сертифікатори є більш зручними у використанні порівняно з мережевими аналізаторами і можуть застосовуватися не лише спеціалістами, але й кваліфікованими мережевими адміністраторами.

Для локалізації місця та визначення причини несправності в кабельній системі (обриву, короткого замикання, неправильно обжатого роз'єму тощо) використовується метод кабельного радара, або рефлектометрії в часовій області (TDR - Time Domain Reflectometry). Суть методу полягає у випромінюванні в кабель короткого електричного імпульсу та вимірюванні часу затримки до повернення відбитого сигналу. Аналіз полярності відбитого імпульсу дозволяє визначити характер пошкодження (коротке замикання або обрив). У правильно прокладеному та підключеному кабелі значне відбиття сигналу відсутнє.

Точність визначення відстані до пошкодження залежить від точності відомої швидкості поширення електромагнітних хвиль у конкретному типі кабелю. Ця швидкість (NVP - Nominal Velocity of Propagation) зазвичай

вказується виробником у відсотках від швидкості світла у вакуумі. Сучасні кабельні сертифікатори та сканери часто містять вбудовані електронні таблиці значень NVP для різних типів кабелів, а також надають користувачеві можливість самостійного введення або калібрування цього параметра для підвищення точності вимірювань.

Кабельні тестери є найбільш простими та доступними приладами для базової діагностики кабельних з'єднань. Вони дозволяють перевірити цілісність провідників (безперервність кабелю) та правильність їхньої розводки (відповідність пінауту). Однак, на відміну від кабельних сертифікаторів та сканерів, вони не надають інформації про такі параметри, як загасання, перехресні наведення або місцезнаходження несправності вздовж кабелю. Сучасні кабельні тестери можуть мати додаткові функції, такі як виявлення коротких замикань або обривів, але їхні можливості обмежені порівняно з сертифікаторами та сканерами.

Сучасні тенденції в галузі тестування кабельних мереж включають розробку більш компактних, швидких та простих у використанні приладів. З'являються сертифікатори з сенсорними екранами, бездротовим підключенням та інтеграцією з хмарними сервісами для зберігання та обміну результатами тестування. Розвиваються стандарти для тестування нових категорій кабелів (наприклад, Cat 8) для підтримки вищих швидкостей передачі даних. Також зростає важливість тестування оптоволоконних кабельних систем, для чого використовуються спеціалізовані оптичні рефлектометри (OTDR - Optical Time Domain Reflectometer) та оптичні тестери. Виробники продовжують вдосконалювати програмне забезпечення для аналізу результатів тестування та створення професійних звітів про сертифікацію кабельних систем.

1.2 Огляд інструментів моніторингу стану розподілених систем

Моніторинг стану сервісів є однією з найважливіших функцій системного адміністрування, поряд з резервним копіюванням. На ринку

представлено безліч інструментів для виконання цієї задачі. Нижче наведено огляд основних систем моніторингу з коротким описом, порівняльним аналізом, перевагами, недоліками та особливостями.

Monit - одна з ранніх розробок, що позиціонується як інструмент для «гавкання на демонів». Це локальний, самостійний демон, написаний на мові C. Його основне призначення - відстежувати роботу фонових процесів (демонів) та автоматично перезапускати ті, що завершилися аварійно, зависли або перевищили встановлені квоти ресурсів. Monit має своєрідний формат конфігурації та не підтримує плагіни для розширення функціональності. Проте, базова функціональність включає більшість необхідних перевірок, таких як перевірка наявності процесу, використання ресурсів, можливість підключення до процесу (через мережу або сокет), перевірка відповіді від сервера (на факт наявності та вміст), а також перевірка відповідності певним протоколам (HTTP, FTP, POP3/IMAP/SMTP тощо). Monit підтримує роботу з SSL та має вбудований веб-інтерфейс. Перевірки виконуються за розкладом, через задані інтервали часу. За результатами перевірок система може автоматично приймати рішення (зупинити процес, перезапустити, повідомити адміністратора). Компанія-розробник також пропонує платну багатосервісну систему M/Monit для централізованого управління групами серверів.

Munin - проект, спочатку орієнтований на візуалізацію даних у вигляді графіків, але також може використовуватися як система моніторингу завдяки вбудованій системі оповіщень. Munin має строго клієнт-серверну архітектуру. На цільових вузлах запускається процес-клієнт, який збирає числові метрики. Процес-сервер періодично підключається до клієнтів, збирає ці дані, зберігає їх та генерує графіки. Сервер також відповідає за відправку повідомлень. Munin є легко розширюваною системою, оскільки всі метрики збираються за допомогою плагінів (без плагінів збір даних неможливий). Плагіни легко розробляти на будь-якій мові програмування, оскільки Munin очікує лише числові дані метрик у визначеному форматі та код повернення. Через свою

архітектуру Munin не має можливості виконувати дії на клієнтських вузлах (наприклад, перезапускати завислі процеси), [4].

Ganglia - досить зрілий інструмент, розроблений у CERN, основним призначенням якого є збір даних про продуктивність великої кількості однотипних машин, зокрема обчислювальних кластерів. Ganglia відрізняється високою продуктивністю та масштабованістю (здатність обробляти дані з десятків тисяч вузлів на середньому сервері), але має певну затримку в відображенні даних (збір відбувається періодично, тому для моніторингу в реальному часі підходить менше). Система має клієнт-серверну архітектуру: клієнт збирає дані, акумулює їх та періодично відправляє на сервер. У випадку перевищення порогових значень метрик, повідомлення відправляється негайно. Сервер збирає дані від усіх клієнтів, обробляє їх та зберігає в базі даних. Веб-інтерфейс є окремим компонентом, який взаємодіє з сервером за внутрішнім протоколом та відображає графіки. Інтерфейс Ganglia є специфічним, але добре підходить для порівняльного аналізу даних з груп серверів. Ganglia є розширюваною системою, проте розробка плагінів є нетривіальною задачею, оскільки плагін представляється як бібліотека, що накладає обмеження на мову реалізації (C або Python) та структуру коду. Основні демони написані на C, веб-інтерфейс - на PHP. Ganglia має базову систему оповіщень і часто використовується в комбінації з Icinga для розширення функціональності сповіщень, [5].

Nagios / Icinga - ще один зрілий інструмент, спочатку розроблений Етаном Галстадтом як Nagios. Згодом з'явився форк Icinga, і наразі обидві системи розвиваються незалежно, хоча мають багато спільного. Nagios доступний у безкоштовній (core) та платній редакціях, тоді як Icinga є open source проектом. Обидві системи мають серверно-орієнтований демон, написаний на мові C, а управління здійснюється через конфігураційні файли у C-подібному синтаксисі. Всі перевірки виконуються послідовно, що може обмежувати масштабованість при великій кількості перевірок. Для локального виконання перевірок на клієнтських вузлах (коли віддалена перевірка

неможлива) існує опціональний клієнт NRPE, але перевірки через клієнта ініціюються сервером, що не вирішує проблеми масштабованості. Nagios відрізняється потужною та гнучкою системою оповіщень (підтримує email, SMS, пейджери та інші плагіни для сповіщень, включаючи голосові дзвінки та світлофори). Icinga пропонує два веб-інтерфейси на вибір - класичний CGI та сучасніший на PHP (попередня версія Nagios Core має лише CGI-інтерфейс). Nagios Core не має вбудованих функцій збору та аналізу статистичної інформації. Для інтеграції з системами графіків, такими як Grafana, часто використовуються додаткові плагіни або інтеграція з Ganglia (у випадку Icinga). У цьому тандемі Icinga відповідає за карту мережі, огляд хостів та оповіщення, [5].

Cacti - інструмент, спочатку розроблений для швидкого та простого збору статистики через SNMP, але з використанням плагінів (наприклад, для сповіщень та встановлення порогових значень) набув функціональності системи моніторингу. Це PHP-додаток, конфігурація та налаштування зберігаються в базі даних MySQL, а статистичні дані - у RRD (Round-Robin Database) для підвищення продуктивності. Cacti відрізняється простотою встановлення та базового налаштування через веб-інтерфейс. Система добре масштабується завдяки використанню багатопотокового опитувальника spine для SNMP. Cacti має велику кількість готових шаблонів для моніторингу різних типів мережевих пристроїв. Основним недоліком є його SNMP-центричність. Cacti добре підходить для збору даних з активного мережевого обладнання, середньо - для збору типової статистики операційних систем (завантаження процесора, використання дискового простору тощо) і менш ефективний для збору складних та нестандартних метрик. Можливості кастомізації обмежені, [4].

OpenNMS - складна, багатокомпонентна система, написана на Java. Має власний сервер застосунків та орієнтована на великі інфраструктури з передачею статистики через SNMP. Через свою складність та гнучкість налаштування OpenNMS може бути досить складним для невеликих

інфраструктур. Подібно до Cacti, OpenNMS в основному призначена для моніторингу мережевих пристроїв через SNMP. Система має проблеми з документацією та розширюваністю (являє собою монолітний Java-додаток, а конфігурація здійснюється через складні XML-файли).

Graphite + CollectD + Whisper - набір окремих компонентів, кожен з яких самостійно не вирішує завдання моніторингу, але в комбінації утворюють потужну систему. CollectD - це демон для збору системних та застосункових метрик. Graphite - це система для зберігання та візуалізації часових рядів даних. Whisper - це база даних часових рядів, яка використовується Graphite. На відміну від попередніх продуктів, це не готові рішення, а набір інструментів, що вимагають значної ручної конфігурації. Незручні файли конфігурації, потенційна нестабільність деяких компонентів (особливо альфа-версій) та часто відсутня документація компенсуються високою швидкістю роботи та масштабованістю. Цей набір інструментів добре підходить для моніторингу великих інфраструктур з потребою в відображенні даних у режимі, близькому до реального часу.

Zabbix - потужна система моніторингу enterprise-класу з відкритим вихідним кодом. Має трирівневу архітектуру (сервер - проксі (опціонально) - агент), дані зберігаються в SQL базі даних, а веб-інтерфейс є самостійним PHP-додатком. Демони моніторингу написані на C/C++. Всі налаштування зберігаються в базі даних та змінюються через веб-інтерфейс. Zabbix має надзвичайно широкий функціонал, включаючи безліч вбудованих перевірок, гнучку систему графіків, ескалацію проблем та відстеження SLA, моніторинг на рівні застосунків, імітацію поведінки користувачів (з підтримкою JavaScript, cookie, GET/POST/PUT запитів), карти та схеми, настроювані панелі (dashboard) з будь-якими метриками. Система має власний API, розподілену модель прав доступу, підтримує кластеризацію для зниження навантаження та має велику кількість готових шаблонів перевірок. На відміну від багатьох інших систем, Zabbix може виконувати дії на клієнтських вузлах (наприклад, перезапускати процеси). Zabbix є надзвичайно гнучкою та

розширюваною системою. В SQL базі даних зберігається вся інформація - налаштування, статистика, метрики, вузли. Активна робота Zabbix може призводити до високого навантаження на дискову підсистему (IOPS). Зниження кількості метрик та частоти їхнього збору може зменшити це навантаження, але також знижує інформативність моніторингу. Zabbix є складною системою в установці та налаштуванні, незважаючи на добре написану документацію. Впровадження Zabbix з нуля може зайняти значний час, а його оптимізація під потреби конкретного бізнесу - ще більше, [6].

З часом з'явилися нові системи моніторингу та еволюціонували існуючі. Серед сучасних тенденцій можна відзначити зростання популярності хмарних сервісів моніторингу (наприклад, AWS CloudWatch, Azure Monitor, Google Cloud Monitoring), систем моніторингу продуктивності застосунків (APM) (наприклад, Datadog, New Relic, Dynatrace), а також інструментів, що активно використовують машинне навчання та штучний інтелект для виявлення аномалій, прогнозування проблем та автоматизації реагування на інциденти. Контейнеризація та мікросервісна архітектура також породили нові потреби в моніторингу, що призвело до розвитку спеціалізованих інструментів для моніторингу контейнерів (наприклад, Prometheus, Grafana (часто використовується в екосистемі Kubernetes), cAdvisor). Інтеграція систем моніторингу з платформами оркестрації контейнерів (наприклад, Kubernetes) стає все більш поширеною.

1.3 Маршрутизація

Маршрутизація є фундаментальним процесом пересилання пакетів даних між різними комп'ютерними мережами або підмережами. Цей процес здійснюється мережевими пристроями третього (мережевого) рівня моделі OSI/ISO, які називаються маршрутизаторами. Для визначення оптимального шляху передачі даних маршрутизатори використовують спеціальні таблиці маршрутизації та протоколи маршрутизації, що реалізують різноманітні алгоритми, [7].

Маршрутизатор підтримує таблицю маршрутизації, яка містить перелік відомих мережевих адрес, а також інформацію про наступний пункт призначення (next hop) для досягнення цих мереж. Завдяки цим записам маршрутизатор визначає, чи може він доставити пакет безпосередньо до адресата, чи необхідно переслати його через інший маршрутизатор. Таблиця маршрутизації може містити записи двох основних типів, [7]:

1. Статичні маршрути. Інформація про маршрути вводиться в таблицю адміністратором вручну. Цей підхід є простим у налаштуванні для невеликих мереж, але стає неефективним та схильним до помилок у великих або динамічно змінюваних мережах. Будь-які зміни в топології мережі або відмови на окремих ділянках вимагають ручного оновлення статичних маршрутів, що може призвести до проблем з доставкою трафіку.
2. Динамічні маршрути. Таблиця маршрутизації заповнюється автоматично шляхом обміну інформацією про маршрути між маршрутизаторами. Цей обмін відбувається за допомогою протоколів маршрутизації, які дозволяють маршрутизаторам обмінюватися повідомленнями про оновлення щодо стану мережі та доступних маршрутів. Залежно від конкретного протоколу, оновлення можуть надходити періодично або лише при виявленні змін у топології мережі, [4]. Динамічна маршрутизація забезпечує більшу гнучкість та стійкість мережі до змін і відмов, оскільки маршрути автоматично перераховуються при зміні мережевої топології.

У складних мережах часто існує кілька можливих шляхів для передачі даних від джерела до призначення. Кожен протокол маршрутизації використовує власні метрики для визначення найкращого шляху (наприклад, кількість переходів, пропускна здатність каналу, затримка, навантаження). У випадках, коли в мережі використовуються різні протоколи маршрутизації, вибір оптимального шляху здійснюється на основі адміністративної відстані (Administrative Distance - AD). Адміністративна відстань є числовим

значенням від 0 до 255 (чим менше значення, тим вищий пріоритет протоколу), яке присвоюється кожному протоколу маршрутизації для визначення його надійності та переваги при виборі маршруту, [7]. Маршрути з меншою адміністративною відстанню вважаються більш надійними та використовуються для пересилання трафіку.

Сучасні мережі стають все більш складними та динамічними, що зумовлює розвиток більш інтелектуальних та гнучких протоколів маршрутизації. Серед актуальних тенденцій можна виділити:

1. Програмно-визначені мережі (SDN). SDN відокремлює рівень керування мережею від рівня передачі даних, дозволяючи централізовано програмувати маршрутизацію та інші мережеві функції. Це забезпечує більшу гнучкість та автоматизацію управління маршрутами.
2. Протоколи маршрутизації для великих та складних мереж. Протоколи, такі як BGP (Border Gateway Protocol), відіграють ключову роль в маршрутизації між автономними системами в Інтернеті та великих корпоративних мережах, забезпечуючи стабільний обмін маршрутною інформацією в умовах постійних змін.
3. Трафік-інжиніринг (Traffic Engineering). Методи трафік-інжинірингу використовуються для оптимізації використання мережевих ресурсів та покращення якості обслуговування (QoS) шляхом впливу на шляхи маршрутизації трафіку. Технології, такі як MPLS-TE (Multiprotocol Label Switching Traffic Engineering) та SDN, надають розширені можливості для управління трафіком.
4. Сегментна маршрутизація (Segment Routing), ця технологія дозволяє спростити маршрутизацію в складних мережах шляхом використання сегментів (ідентифікаторів), які визначають шлях проходження пакетів. Це спрощує управління та масштабування мережі.
5. Маршрутизація з урахуванням якості обслуговування (QoS-aware Routing) – сучасні протоколи та механізми маршрутизації можуть

враховувати вимоги до якості обслуговування (наприклад, затримку, джиттер, втрату пакетів) при виборі оптимального шляху для різних типів трафіку.

6. Використання машинного навчання та штучного інтелекту. У перспективі методи машинного навчання та штучного інтелекту можуть бути використані для оптимізації алгоритмів маршрутизації, прогнозування трафіку та адаптації маршрутів у реальному часі для підвищення ефективності та стійкості мережі.

Таким чином, маршрутизація залишається ключовим елементом сучасної мережевої інфраструктури, а її методи та протоколи постійно розвиваються для задоволення зростаючих вимог до швидкості, надійності та гнучкості передачі даних.

Протокол маршрутизації являє собою набір чітко визначених правил, які використовуються маршрутизаторами для обміну інформацією один з одним з метою визначення оптимальних шляхів до віддалених мереж та підтримки актуальних записів про ці мережі в їхніх таблицях маршрутизації, [3]. Важливо розрізняти два схожих, але принципово різних поняття: маршрутизуючий протокол та протокол маршрутизації, [5].

Маршрутизуючий протокол - це будь-який протокол, що має адресу мережевого рівня та відповідає за пересилання пакетів даних між хостами. Характерною особливістю таких протоколів є відсутність повної інформації про весь шлях проходження пакета від джерела до призначення. Прикладом маршрутизуючого протоколу є IP (Internet Protocol).

На противагу цьому, протокол маршрутизації спеціально розроблений для забезпечення обміну даними про маршрути між різними мережами, що дозволяє маршрутизаторам автоматично будувати та оновлювати динамічні таблиці маршрутизації. Маршрутизатору необхідно знати лише наступний пункт призначення (next hop) для пересилання пакета, але не весь його подальший шлях через інші мережеві пристрої.

Протоколи маршрутизації класифікуються на основі типу взаємодії між мережами, і ця класифікація тісно пов'язана з концепцією автономної системи, рис. 1.1.

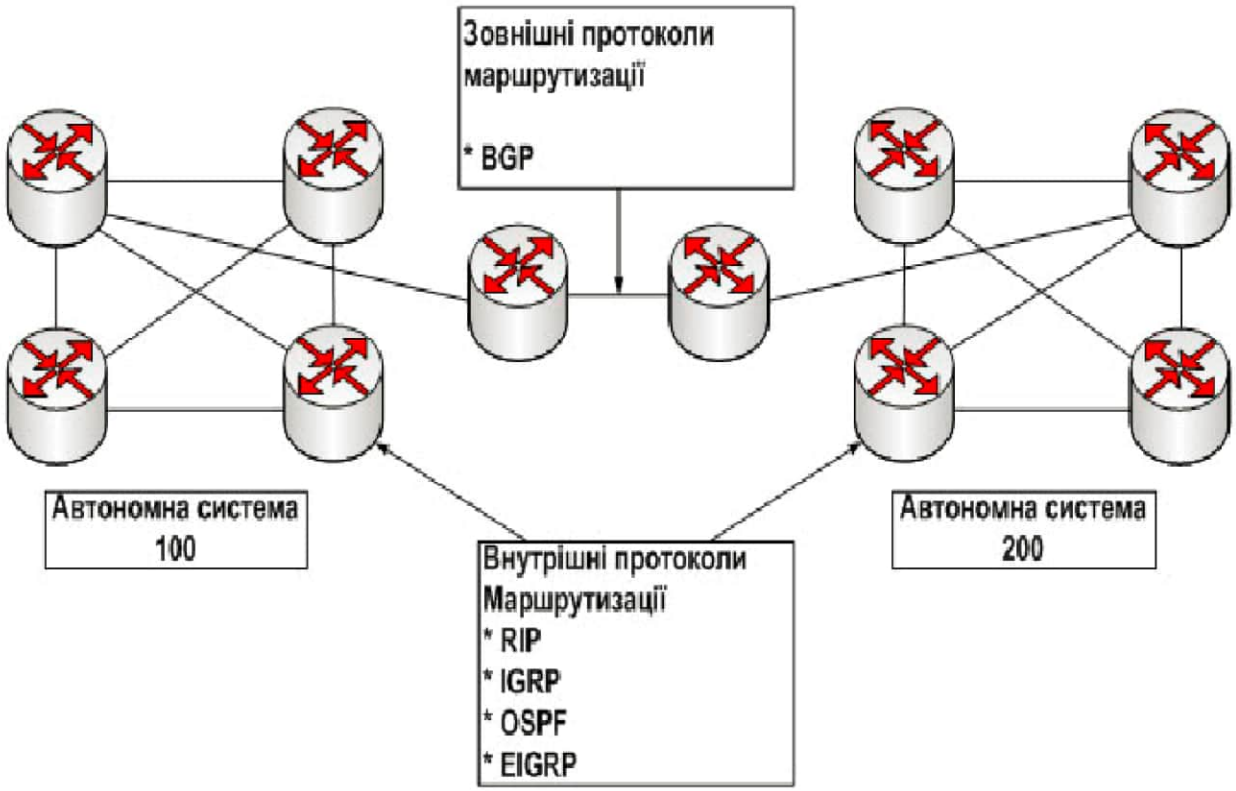


Рисунок 1.1 – Класифікація протоколів по типу взаємодії між мережами

В контексті комп'ютерних мереж, автономна система (АС) являє собою сукупність мереж, що знаходяться під єдиним адміністративним управлінням, де всі маршрутизатори використовують узгоджені правила маршрутизації. Відповідно до цієї концепції, протоколи маршрутизації поділяються на два основних типи:

Внутрішні протоколи маршрутизації (Interior Gateway Protocols - IGP): Ці протоколи використовуються для обміну інформацією про маршрути між маршрутизаторами, що знаходяться в межах однієї автономної системи. До найпоширеніших IGP належать RIP (Routing Information Protocol), OSPF (Open Shortest Path First), EIGRP (Enhanced Interior Gateway Routing Protocol) та інші, [2, 4].

Зовнішні протоколи маршрутизації (Exterior Gateway Protocols - EGP): Ці протоколи призначені для обміну інформацією про досяжність мереж між різними автономними системами. Основним прикладом EGP є BGP (Border Gateway Protocol), [4].

Такий поділ протоколів відображає ієрархічний підхід до маршрутизації в глобальних мережах, таких як Інтернет.

Протоколи маршрутизації також можуть бути класифіковані за типом використовуваного ними алгоритму маршрутизації, який визначає оптимальний шлях для передачі пакетів даних від відправника до отримувача. Ефективні алгоритми маршрутизації повинні відповідати ряду важливих вимог, [1]:

Оптимальність. Здатність алгоритму вибирати найкращий доступний шлях на основі визначених критеріїв (метрик).

Простота. Алгоритм не повинен бути надмірно складним для програмної реалізації та обробки маршрутизаторами.

Живучість (Надійність). Здатність алгоритму продовжувати функціонувати та забезпечувати маршрутизацію в умовах непередбачених подій, таких як відмова обладнання або перевантаження мережі.

Швидка збіжність. Процес узгодження інформації про найкращі маршрути між усіма маршрутизаторами в мережі повинен відбуватися швидко. У випадку зміни топології (наприклад, відмови маршрутизатора), оновлення маршрутної інформації повинні оперативно досягати всіх інших маршрутизаторів, щоб вони могли перерахувати шляхи та вибрати нові оптимальні маршрути. Повільна збіжність може призвести до нестабільності мережі та проблем з маршрутизацією.

Гнучкість (Адаптивність). Алгоритм повинен швидко та точно адаптуватися до змін у мережі, таких як зміни в топології, пропускній здатності каналів зв'язку, затримках тощо.

Основні типи алгоритмів маршрутизації включають, [4]:

Статичні алгоритми. Маршрути в таблиці маршрутизації визначаються та вводяться вручну системним адміністратором. Цей метод не підходить для великих мереж та складний в адмініструванні при змінах у топології.

Динамічні алгоритми. Ці алгоритми автоматично враховують зміни в мережі на основі отриманих повідомлень про оновлення маршрутної інформації. При зміні топології відбувається перерахунок маршрутів та розсилка оновлень іншим маршрутизаторам, [3].

Протоколи внутрішньої маршрутизації (IGP) можуть бути додатково класифіковані за типом використовуваного динамічного алгоритму маршрутизації:

Алгоритми на основі вектора відстаней (Distance Vector): Ці алгоритми визначають напрямок (вектор) та відстань (наприклад, кількість переходів) до кожної відомої мережі шляхом періодичного обміну векторами відстаней зі сусідніми маршрутизаторами. При отриманні вектора від сусіда, маршрутизатор оновлює свою таблицю маршрутизації, збільшуючи відстань на одиницю (вартість переходу) та додаючи інформацію про нові мережі, після чого розсилає оновлений вектор своїм сусідам. Недоліком цього методу у великих мережах може бути значне широкомовне трафік, пов'язаний з обміном векторами.

Алгоритми на основі стану каналу (Link-State): У цьому методі маршрутизатори обмінюються інформацією про стан своїх безпосередніх з'єднань (каналів) зі своїми сусідами. На основі отриманих повідомлень кожен маршрутизатор будує власну повну топологічну базу даних мережі. Потім, використовуючи алгоритм найкоротшого шляху (наприклад, алгоритм Дейкстри), кожен маршрутизатор самостійно обчислює оптимальне дерево найкоротших шляхів до всіх відомих мереж, [4].

Важливо зазначити, що ідеального алгоритму маршрутизації, який би оптимально підходив для будь-якої мережі, не існує.

У процесі вибору найкращого маршруту алгоритми маршрутизації використовують різні метрики - кількісні показники, що характеризують

якість або вартість шляху. Зазвичай, менше значення метрики вказує на кращий маршрут. Складні алгоритми можуть враховувати комбінацію кількох метрик при прийнятті рішення про маршрут. До найчастіше використовуваних метрик належать, [2]:

Кількість переходів (Hop Count): Кількість маршрутизаторів, через які необхідно пройти пакету на шляху до призначення.

Швидкість передачі даних (Пропускна здатність - Bandwidth): Максимальна швидкість передачі даних на каналі зв'язку.

Затримка (Delay), це час, необхідний для передачі пакета від джерела до призначення, який може залежати від завантаження мережі та пропускної здатності каналів.

Завантаження (Load), це рівень використання мережевого ресурсу (маршрутизатора, каналу тощо).

Надійність (Reliability) , це показник стабільності та безвідмовності каналу зв'язку.

Вартість (Cost), це адміністративно визначене значення, що присвоюється каналу для впливу на вибір маршруту.

1.4 Технології маршрутизації розподілених комп’ютерних мереж

Для визначення оптимальних шляхів у мережі використовуються різноманітні аналітичні алгоритми, серед яких особливу популярність та поширеність здобули алгоритми Беллмана-Форда та Дейкстри.

Традиційні протоколи маршрутизації зазвичай підтримують лише принцип «найкращого зусилля» (best-effort), не враховуючи вимог до якості обслуговування (QoS). На противагу цьому, алгоритми маршрутизації з урахуванням якості обслуговування (QoS) при виборі маршруту аналізують доступність мережевих ресурсів та потреби трафіку. Згідно з дослідженнями [8], QoS-маршрутизація визначається як метод вибору шляху, при якому маршрути для передачі потоків даних формуються на основі інформації про

наявні ресурси мережі та специфічні вимоги до якості обслуговування цих потоків.

У роботі [8] детально розглядаються одноколійні підходи до реалізації QoS-маршрутизації. Проте, більшість таких алгоритмів виходять з припущення про наявність хоча б одного шляху, що повністю відповідає вимогам трафіку. Така ситуація є типовою для мереж з низьким рівнем завантаженості. Однак, зі збільшенням навантаження може виникнути сценарій, коли знайти єдиний маршрут, здатний забезпечити необхідну якість обслуговування, стає неможливим. У таких випадках доцільним є перехід до багатоколійної маршрутизації та використання менш оптимальних, але незадіяних маршрутів. Таким чином, застосування одноколійних алгоритмів у перевантажених мережах може бути неефективним, проте їх комбінація з технологіями багатоколійної маршрутизації може знайти практичне застосування.

Окремий напрямок розвитку представляють алгоритми маршрутизації з урахуванням системних політик, які з'явилися відносно недавно. Ці алгоритми забезпечують гнучкий вибір маршруту, враховуючи не лише технічні параметри мережі, але й правила, прийняті в даній мережі, а також угоди про рівень обслуговування (SLA).

Як було зазначено раніше, використання маршрутизації лише за найкоротшим шляхом може призвести до нерівномірного розподілу трафіку, коли канали, що входять до найкоротших маршрутів, стають перевантаженими, а інші мережеві шляхи залишаються недовикористаними. В якості альтернативного підходу до маршрутизації через єдиний найкоротший шлях була запропонована мережева топологія з багатоколійною маршрутизацією (рис. 1.2). Метою такого підходу є розподіл трафіку між різними доступними шляхами та зниження ризику перевантаження окремих ділянок мережі.

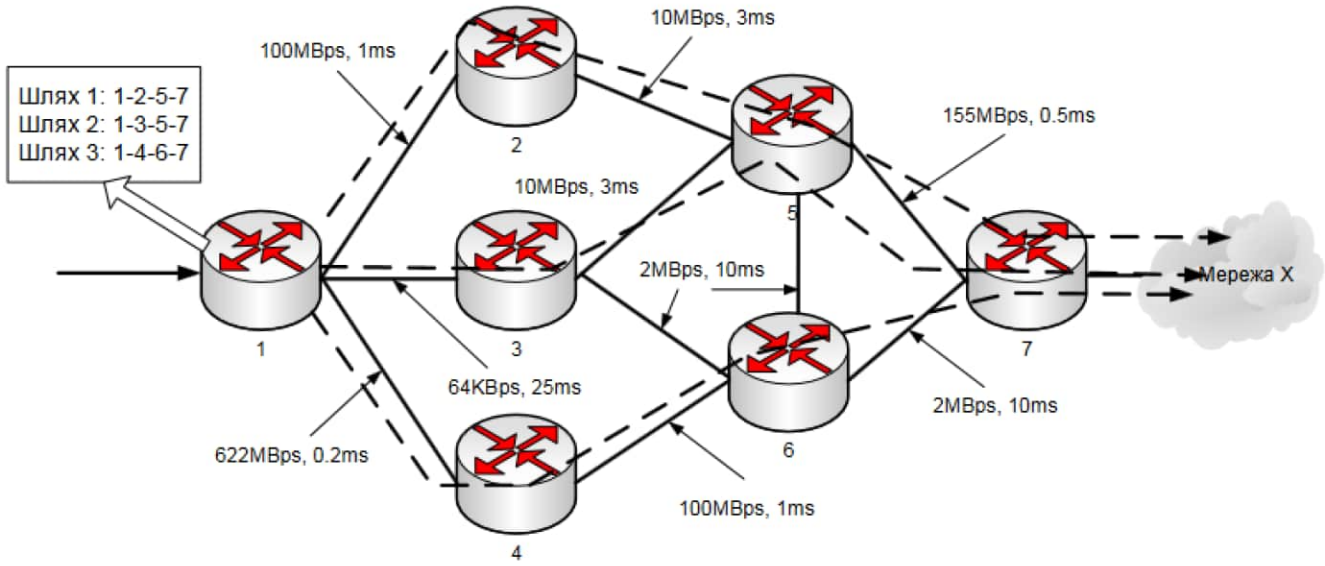


Рисунок 1.2 – Реалізація багатоколійної маршрутизації

Для реалізації багатоколійної маршрутизації з урахуванням якості обслуговування (QoS) необхідно вирішити три ключові завдання: збір даних про трафік та визначення вимог до обслуговування, вибір одного або декількох маршрутів для передачі трафіку, та забезпечення актуальності інформації про ресурси на обраних маршрутах.

Перше завдання спирається на теорію телетрафіку, яка дозволяє обґрунтувати вибір показників трафіку. У багатьох випадках параметри трафіку можуть бути описані відомими законами розподілу, такими як рівномірний або експоненціальний.

Ефективність багатоколійної маршрутизації залежить від «якості» обраних маршрутів. Зменшення кількості використовуваних маршрутів є бажаним через значні витрати на встановлення, підтримку та видалення маршрутів, зростання складності розподілу трафіку зі збільшенням їхньої кількості, а також можливі обмеження на кількість маршрутів між вузлами (наприклад, в MPLS). Тому важливо використовувати методи для визначення оптимальної кількості маршрутів.

Для ефективного вибору маршрутів необхідна актуальна інформація про стан мережі, включаючи QoS та доступність ресурсів. Отримання та аналіз цієї

інформації, що постійно змінюється, вимагає значних обчислювальних ресурсів та активного обміну даними між маршрутизаторами. Тому розробка стійкої маршрутизаційної схеми, незалежної від нерегулярності оновлень, є критично важливою.

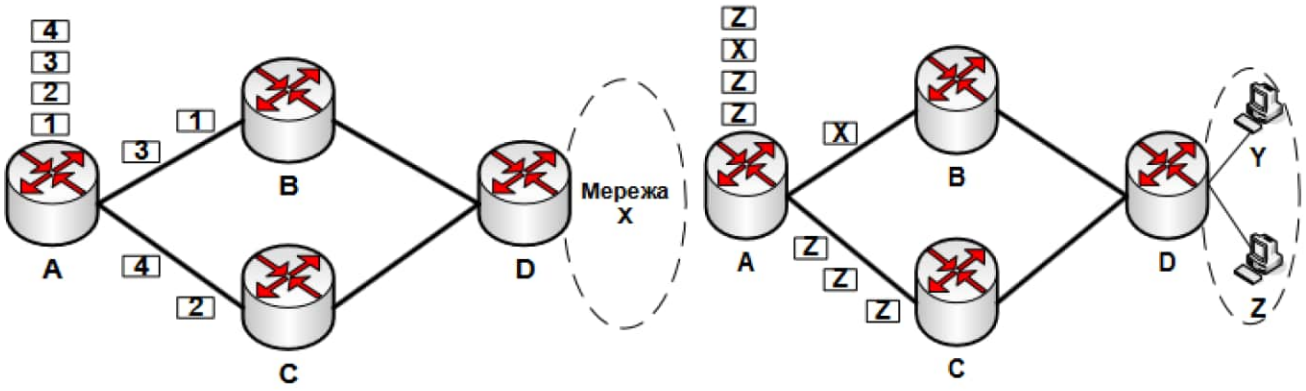
Різні багатошляхові методи маршрутизації, такі як ECMP (Equal Cost Multipath) та OMP (Optimized Multipath), [8], використовуються для оптимізації розподілу навантаження на рівні мережі. ECMP, реалізований у протоколі OSPF, рівномірно розподіляє трафік між декількома шляхами з однаковою вартістю, яка в OSPF обчислюється на основі пропускної здатності каналів:

$$\cos t = \frac{10^8}{Bw}, \tag{1.1}$$

де *Bw* – пропускна здатність каналу. У наведній реалізації ECMP (Equal Cost Multipath), розподіл мережевого трафіку здійснюється між шляхами, що мають ідентичну пропускну здатність каналів. Однак, ці маршрути конфігуруються статично і не враховують поточний стан мережі, що може призводити до неефективного використання ресурсів при зміні навантаження. Більш того, розподіл трафіку на рівні окремих пакетів (рис. 1.3а) вимагає значних обчислювальних потужностей від мережевих пристроїв, особливо у високошвидкісних магістральних мережах, де потоковий (flow-based) підхід до розподілу трафіку (рис. 1.3б) вважається більш ефективним.

Важливим обмеженням алгоритму ECMP є його здатність розподіляти навантаження лише між шляхами з однаковою вартістю, ігноруючи динаміку зміни завантаження мережі в реальному часі. Це може стати проблемою, оскільки управління навіть двома каналами з однаковою пропускною здатністю може вимагати значних ресурсів. На практиці часто організації використовують основний високошвидкісний канал зв'язку разом з одним або кількома резервними каналами з нижчою пропускною здатністю. У таких сценаріях резервні канали залишаються незадіяними до моменту виходу з ладу

основного каналу, що є економічно невигідним та неоптимальним використанням наявних мережевих ресурсів.



а) б)

Рисунок 1.3 – Розподілення навантаження, а) пакетами б) потоком

Ефективним підходом до оптимізації розподілу мережевого навантаження є використання схем багатоколійної маршрутизації, які здатні розподіляти трафік між маршрутами з різною вартістю. Одним з найпростіших методів реалізації такої схеми є пропорційний розподіл ресурсів, що базується на метриках маршрутів. Відомими прикладами протоколів, які використовують цей підхід, є EIGRP (Enhanced Interior Gateway Routing Protocol) та IGRP (Interior Gateway Routing Protocol) - обидва розроблені компанією Cisco та є її пропрієтарними алгоритмами. Якість маршруту в цих протоколах визначається за допомогою складної (композитної) метрики, яка враховує п'ять основних параметрів: пропускну здатність (Bw), надійність (R), затримку (dl), завантаження (L) та максимальний розмір пакету (MTU), який може бути переданий через даний маршрут без необхідності фрагментації:

$$metric = [k1 \cdot Bw + \frac{k2 \cdot Bw}{256 - L} + k3 \cdot dl] \cdot [\frac{k5}{R + k4}] \qquad (1.2)$$

Використання композитної метрики в протоколах маршрутизації, таких як EIGRP та IGRP (розробки Cisco), дозволяє точніше враховувати характеристики маршрутів, залежні від транспортної технології. Трафік

розподіляється обернено пропорційно до значень метрики, тобто маршрути з меншою метрикою отримують більший обсяг трафіку. Однак, пакетний підхід цих протоколів є ресурсомістким. Оскільки IGRP та EIGRP є пропрієтарними, їх застосування обмежене обладнанням Cisco. Сучасні відкриті протоколи, такі як RIP, OSPF або IS-IS, не мають вбудованої підтримки розподілу навантаження між маршрутами з різною вартістю. Це створює потребу в розробці нової методики багатошляхової маршрутизації, здатної розподіляти навантаження між маршрутами з різною вартістю та інтегруватися в існуючі або нові протоколи.

З появою технології MPLS (Multiprotocol Label Switching) виник клас алгоритмів явної маршрутизації, нетиповий для дейтаграмних IP-мереж. MPLS дозволила створювати віртуальні шляхи з гарантованою якістю обслуговування. Ранні спроби емуляції віртуальних каналів в IP-мережах, такі як протокол RSVP, виявилися не масштабованими через резервування ресурсів для окремих потоків. MPLS створює віртуальні шляхи для агрегованих потоків.

У традиційних IP-мережах кожен маршрутизатор на шляху пакета аналізує його заголовок для визначення потоку та наступного напрямку. В MPLS відповідність між потоком і пакетом встановлюється лише на вході до мережі MPLS через клас еквівалентності пересилання (FEC). Пакети одного FEC мають однаковий маршрут. Кожному пакету присвоюється коротка мітка-ідентифікатор, що визначає його FEC (рис. 1.4). Мітка є локальною та дійсна лише між сусідніми маршрутизаторами. При передачі пакета наступний маршрутизатор призначає нову мітку для того ж FEC на наступній ділянці шляху. Таким чином, для кожного FEC формується власна система міток.

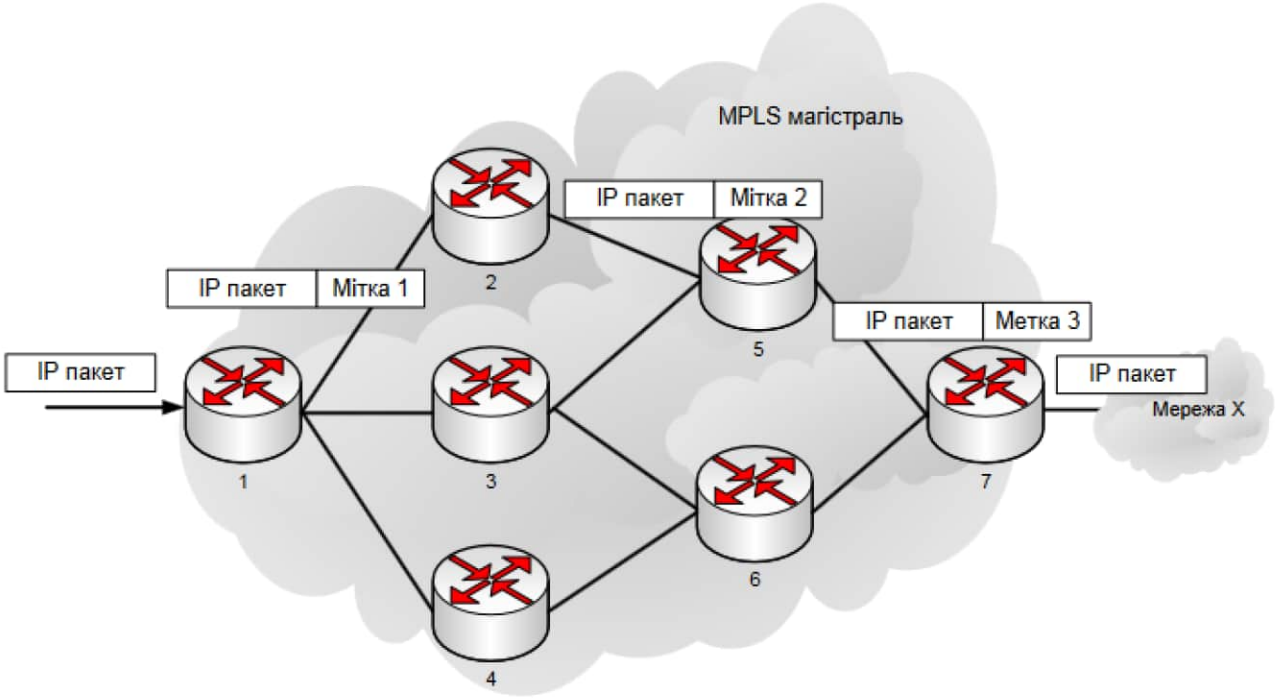


Рисунок 1.4 – Структура MPLS мережі

Таким чином, кожному класу еквівалентності пересилання (FEC) відповідає власна система міток. Використання міток значно спрощує процес пересилання пакетів, оскільки маршрутизатори обробляють лише коротку мітку замість повного IP-заголовка, що істотно зменшує час обробки.

На вхідному маршрутизаторі LER (Label Edge Router) відбувається аналіз заголовка вхідного пакета для визначення його FEC. Потім пакету присвоюється відповідна мітка, і він пересилається до наступного маршрутизатора LSR (Label Switching Router). Пройшовши через послідовність LSR, пакет досягає вихідного LER, який видаляє мітку, аналізує заголовок IP-пакета та пересилає його кінцевому одержувачу за межами MPLS-мережі. Послідовність маршрутизаторів, через які проходять пакети конкретного FEC, утворює віртуальний комутований шлях LSP (Label Switched Path), організований на основі міток.

Система MPLS підтримує так званий стек міток. При переході потоку пакетів до іншого FEC, мітка нового FEC розміщується поверх поточної мітки та використовується для комутації, а попередня мітка зберігається нижче та стає активною при поверненні до попереднього FEC.

Отже, мітка в пакеті ідентифікує його FEC. Зазвичай, віднесення пакета до певного FEC відбувається на основі мережевої адреси отримувача. Мітка може вставлятися в пакет різними способами, наприклад, між заголовками канального та мережевого рівнів або в адресне поле заголовка канального рівня. У будь-якому випадку, спеціальний заголовок містить поле для значення мітки (рис. 1.5) та службові поля, включаючи поле QoS (3 біти, що дозволяє визначити до 8 класів якості обслуговування), яке є важливим для розгляду в контексті даного розділу.

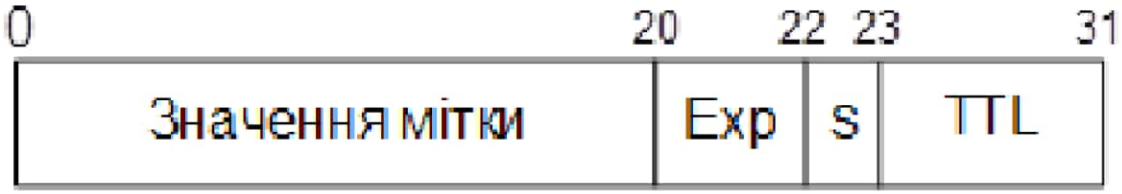


Рисунок 1.5 – Структура MPLS мітки

Унікальність мітки в MPLS потрібна лише в межах пари сусідніх LSR, тому одна й та сама мітка може асоціюватися з різними FEC, якщо пакети надходять від різних сусідів, що дозволяє однозначно визначити джерело пакета. Це мінімізує ризик вичерпання простору міток.

Кожен LSR використовує таблицю, яка зіставляє пару «вхідна мітка, вхідний інтерфейс» з парою «вихідна мітка, вихідний інтерфейс». При отриманні пакета LSR визначає вихідний інтерфейс на основі вхідної мітки та інтерфейсу, замінює вхідну мітку на вихідну з таблиці та пересилає пакет далі. Ця процедура одноразової ідентифікації значень у таблиці відбувається значно швидше, ніж зіставлення IP-адреси з префіксом у класичній маршрутизації.

MPLS підтримує два способи пересилання: незалежний вибір наступного маршрутизатора кожним LSR (як в IP) та пересилання згідно з інструкціями від одного з LSR на LSP (часто попереднього), що дозволяє організовувати віртуальні шляхи для IP-мереж.

Оскільки належність пакета до FEC визначається не лише IP-адресою, а й іншими параметрами, створення різних LSP для потоків з унікальними

вимогами до QoS є простим. Усі FEC обробляються ізольовано, включаючи створення окремих LSP та доступ до спільних ресурсів (пропускна здатність, буферний простір), що забезпечує ефективну підтримку QoS та дотримання гарантій. Однак ключовим напрямком розвитку MPLS є механізми трафік-інжинірингу.

Трафік-інжиніринг спрямований на оцінку та оптимізацію продуктивності IP-мереж шляхом застосування технологій та наукових принципів до вимірювання, моделювання та управління міжмережевим трафіком. Однією з його функцій є управління та оптимізація маршрутизації для найефективнішої передачі трафіку.

Оптимізація в трафік-інжинірингу досягається через управління пропускною здатністю (планування, управління маршрутизацією та ресурсами) та управління трафіком (формування трафіку, управління чергами, їхнє планування). Завдання оптимізації розглядається як безперервний процес для поліпшення продуктивності мережі.

Завдання оптимізації в трафік-інжинірингу є динамічними та залежать від нових вимог і технологій, а також від бізнес-моделей та обмежень конкретних мереж. Управління оптимізацією може бути проактивним (запобігання небажаним станам) або реактивним (адаптація до подій, що вже відбулися), включаючи керування пропускною здатністю, маршрутизацією, трафіком та ресурсами, а також сервісними політиками. Вхідні дані для прийняття рішень включають стан мережі та політики.

Важливими аспектами трафік-інжинірингу є автоматизоване управління для швидкої адаптації до змін у мережі та оцінка її продуктивності, що впливає на подальші рішення щодо оптимізації та прогнозування проблем.

З точки зору реалізації трафік-інжинірингу, виділяють три основні проблеми: співвіднесення вхідних пакетів з FEC, співвіднесення FEC з групами трафіку та співвіднесення груп трафіку з фізичною топологією в MPLS-мережі.

Основною метою трафік-інжинірингу є ефективний розподіл мережевих ресурсів шляхом точного керування маршрутизацією з урахуванням вимог та обмежень. Завдання трафік-інжинірингу можуть бути орієнтовані на трафік (QoS) або ресурси (оптимізація використання, запобігання перевантаженням). Рішення, орієнтовані на трафік, включають управління чергами для забезпечення QoS (втрати пакетів, затримки, пропускна здатність), а ефективність політик оцінюється щодо виконання вимог трафіку (особливо в SLA). Рішення, орієнтовані на ресурси, включають алгоритми багатошляхової маршрутизації, протоколи сигналізації та управління для оптимізації використання ресурсів та боротьби з перевантаженнями.

1.5 Мета і задачі роботи

Головною метою даної роботи є створення програмного застосунку для оптимізації маршрутизації мережевого трафіку в розподілених системах.

Зростання популярності сервіс-орієнтованих мереж спричинило значне збільшення обсягів трафіку та виникнення перевантажень на окремих ділянках мережі внаслідок неефективного використання наявних ресурсів. Для підвищення ефективності функціонування таких мереж виникає необхідність у вдосконаленні існуючих механізмів маршрутизації та розподілу ресурсів, що є важливим завданням для системного адміністрування розподілених комп'ютерних мереж.

Основними завданнями для реалізації роботи є:

- провести комплексний аналіз проблеми оптимізації маршрутизації мережевого трафіку;
- розробити архітектуру програмного застосунку, що базується на модульному принципі;
- здійснити вибір та обґрунтування технологій розробки;
- реалізувати ключові функціональні можливості програмного застосунку;

– розробити інтуїтивно зрозумілий користувацький інтерфейс. Графічний інтерфейс забезпечує легку взаємодію з програмою, дозволяючи користувачеві обирати режими роботи та вводити необхідні дані.

2 РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ ДЛЯ ОПТИМІЗАЦІЇ МАРШРУТИЗАЦІЇ МЕРЕЖЕВОГО ТРАФІКУ В РОЗПОДІЛЕНИХ СИСТЕМАХ

2.1 Технічні вимоги до об’єкту професійної діяльності

2.1.1 Найменування і призначення об’єкту професійної діяльності

Програмний застосунок призначений для аналізу, візуалізації та оптимізації маршрутизації мережевого трафіку в існуючих та проектованих розподілених системах. Він дозволяє ідентифікувати джерела та призначення трафіку, кластеризувати IP-адреси за географічним принципом, моделювати мережеву топологію на основі даних пакетного аналізу (PCAP) та знаходити оптимальні шляхи передачі даних з урахуванням заданих метрик. Це сприяє підвищенню ефективності використання мережевих ресурсів, зниженню затримок та покращенню загальної продуктивності розподілених систем, [9].

2.1.2 Перелік підсистем, їхнє призначення й основні характеристики, вимоги до числа рівнів ієрархії

Програмний застосунок для оптимізації маршрутизації мережевого трафіку в розподілених системах побудований за модульним принципом, що забезпечує гнучкість, розширюваність та полегшує супровід. Він складається з декількох ключових підсистем, кожна з яких виконує специфічні функції.

Підсистема вводу та попередньої обробки даних забезпечує інтерфейс для завантаження файлів мережевого трафіку у форматі PCAP та первинний розбір їх вмісту. Вона підтримує послідовне читання пакетів даних, виконує парсинг Ethernet та IP заголовків для вилучення ключової інформації, такої як IP-адреси джерела та призначення, а також буферує прочитані дані в пам'яті для багаторазового доступу без повторного зчитування файлу. Ця підсистема має дворівневу ієрархію, де верхній рівень представлений модулем завантаження файлу (через графічний інтерфейс користувача), а нижній рівень — модулем парсингу та буферизації з використанням бібліотеки dpkt.

Підсистема географічного аналізу та візуалізації відповідає за визначення географічного розташування IP-адрес та їх наочне представлення на картах. Вона використовує базу даних GeoLiteCity.dat для перетворення IP-адрес на географічні координати (широту, довготу, місто, країну). На основі отриманих даних генеруються KML-файли для перегляду у відповідних картографічних програмах, а також інтерактивні HTML-карти за допомогою бібліотеки folium, які відображають розташування IP-адрес за допомогою маркерів та ліній. Ця підсистема є трирівневою: верхній рівень – елементи GUI для вибору режиму візуалізації, середній рівень – логіка геокодування та підготовка даних, а нижній рівень – безпосередня генерація KML/HTML-файлів.

Підсистема кластеризації IP-адрес призначена для групування географічно близьких IP-адрес з метою виявлення логічних сегментів мережі або вузлів, які належать до однієї фізичної локації. Вона застосовує алгоритм кластеризації K-Means до географічних координат IP-адрес, автоматично визначаючи оптимальну або мінімальну кількість кластерів залежно від наявних даних. Результати кластеризації візуалізуються на інтерактивній HTML-карті, де кластери позначаються різними кольорами, а їхні центри також відображаються. Структура підсистеми є трирівневою, включаючи виклик з GUI на верхньому рівні, алгоритм K-Means (sklearn.cluster.KMeans) на середньому, та підготовку вхідних/обробку вихідних даних на нижньому рівні, [10].

Підсистема моделювання та оптимізації мережі відповідає за побудову абстрактної моделі мережі у вигляді графа та пошук оптимальних маршрутів між заданими вузлами на основі певних мережевих метрик, таких як затримка або пропускна здатність. Вона використовує бібліотеку networkx для представлення мережі, де IP-адреси є вузлами, а зв'язки – ребрами. Вузли графа можуть бути збагачені географічними координатами. Зв'язки між вузлами генеруються випадковим чином із асоційованими метриками для цілей симуляції. Підсистема реалізує пошук найкоротшого шляху, наприклад,

за алгоритмом Дейкстри, та візуалізує побудований граф разом зі знайденими шляхами на інтерактивній HTML-карті. Ця підсистема має три рівні ієрархії: верхній рівень – клас NetworkOptimizer, середній рівень – функціонал networkx для операцій з графами, а нижній рівень – генерація та присвоєння метрик мережевим зв'язкам.

Нарешті, Підсистема користувацького інтерфейсу забезпечує інтуїтивно зрозумілу графічну взаємодію користувача з програмним застосунком. Розроблена за допомогою бібліотеки tkinter, вона надає елементи управління для вибору режиму роботи (CLI, KML, Cluster, Optimize), включає поля вводу для IP-адрес джерела та призначення в режимі оптимізації, кнопку для завантаження PCAP-файлів та текстове поле для виведення результатів обробки та повідомлень. Додатково використовуються стандартні діалогові вікна для вибору файлів та відображення помилок. Ієрархія цієї підсистеми є дворівневою: верхній рівень – це головне вікно та елементи управління, а нижній рівень – обробники подій, такі як функція open_file.

2.1.3 Вимоги до способів і засобів зв'язку для обміну інформації між підсистемами

Взаємодія між підсистемами програмного застосунку для оптимізації маршрутизації мережевого трафіку реалізована за допомогою внутрішніх механізмів обміну даними в межах одного процесу. Основним способом передачі інформації є безпосередній виклик функцій та методів однієї підсистеми іншою, а також передача даних через параметри функцій та повернення результатів. Такий підхід зумовлений архітектурою застосунку, що працює як єдиний десктопний додаток, де всі підсистеми функціонують в одному адресному просторі.

Основним засобом зв'язку є передача об'єктів та структур даних у пам'яті. Зокрема, після завантаження PCAP-файлу, розпарсені дані пакетів зберігаються у вигляді списку об'єктів або кортежів (pcap_data_list), який потім передається як аргумент до інших підсистем (наприклад, підсистемі

географічного аналізу, кластеризації або моделювання мережі). Словники abc та dest_abc, що зберігають географічні координати IP-адрес, виступають як централізовані сховища даних, доступ до яких мають відповідні підсистеми для їх наповнення та використання. Результати обробки, такі як згенеровані KML/HTML-файли, або текстові звіти про знайдені маршрути, передаються до підсистеми користувацького інтерфейсу для відображення або збереження на диску.

Вимоги до швидкості обміну інформацією між підсистемами є високими, оскільки обробка даних відбувається синхронно і впливає на загальний час відгуку застосунку. Забезпечення обміну даними в межах пам'яті одного процесу мінімізує накладні витрати на передачу та серіалізацію/десеріалізацію даних, що відповідає цим вимогам. Механізми синхронізації між підсистемами не вимагаються, оскільки виконання операцій є послідовним та синхронним, і багатопотоковість для обміну даними не використовується. Це спрощує архітектуру та зменшує ризики виникнення умов перегонів або взаємних блокувань, [11].

2.1.4 Вимоги до задач, які виконуються у комп’ютерній системі

Програмний застосунок для оптимізації маршрутизації мережевого трафіку в розподілених системах призначений для виконання ряду аналітичних та оптимізаційних задач у комп'ютерній системі, що дозволяють підвищити ефективність функціонування мереж. Основні задачі, які виконуються за стосунком. Збір та попередню обробку даних мережевого трафіку. Застосунок повинен забезпечувати можливість завантаження та парсингу файлів формату PCAP, вилучення з них необхідної інформації про IP-адреси джерела та призначення, а також часових міток для подальшого аналізу. Географічну локалізацію мережевих вузлів. Однією з ключових задач є визначення географічного розташування IP-адрес, що зустрічаються в аналізованому трафіку. Це дозволяє формувати візуальне представлення мережі на географічній карті. Кластеризацію IP-адрес за географічною

ознакою. Застосунок повинен виконувати групування IP-адрес, які знаходяться географічно близько одна до одної, для ідентифікації логічних або фізичних кластерів мережевих вузлів.

Побудову логічної моделі мережі у вигляді графа. На основі виявлених IP-адрес та їх взаємодій (зв'язків джерело-призначення) застосунок повинен формувати граф мережі, де вузли відповідають IP-адресам, а ребра – мережевим зв'язкам, з можливістю додавання метрик (наприклад, симульованої затримки або пропускної здатності).

Пошук оптимальних маршрутів у модельованій мережі. Застосунок має виконувати пошук найкоротших шляхів між двома заданими IP-адресами-вузлами в побудованому графі, враховуючи задані метрики (наприклад, мінімізація затримки або максимізація пропускної здатності).

Візуалізацію результатів аналізу та оптимізації. Важливою задачею є наочне представлення отриманих даних, включаючи відображення географічного розташування IP-адрес, кластерів, а також візуалізацію побудованого графа мережі та знайдених оптимальних маршрутів на інтерактивних картах (HTML, KML).

Генерацію звітів та результатів. Застосунок повинен надавати можливість виведення текстових звітів про виконаний аналіз, знайдені маршрути та інші релевантні дані для подальшого використання або документування.

2.1.5 Вимоги до обчислювальної системи

Для ефективного парсингу великих PCAP-файлів та виконання ресурсомістких алгоритмів кластеризації та пошуку шляхів, необхідний сучасний багатоядерний процесор з тактовою частотою не менше 2.0 ГГц.

Обсяг оперативної пам'яті повинен бути достатнім для завантаження та обробки великих PCAP-файлів, а також для зберігання побудованих графів мережі. Рекомендований мінімум – 8 ГБ RAM, для великих файлів – 16 ГБ і більше.

Для зберігання програмних компонентів, бази даних GeoLiteCity.dat, а також згенерованих KML/HTML-файлів та звітів, необхідно не менше 500 МБ вільного дискового простору.

Застосунок повинен коректно функціонувати на поширених настільних операційних системах (наприклад, Windows 10/11, Ubuntu 20.04+, macOS).

Для завантаження оновлень бібліотек або баз даних (GeoLiteCity.dat), а також для доступу до картографічних сервісів (якщо використовуються онлайн-ресурси для карт, хоча в поточній реалізації карти генеруються локально), бажана наявність стабільного інтернет-з'єднання.

Вимоги до однорідності комп'ютерних систем, на яких виконуються ці задачі, зводяться до наявності встановленого інтерпретатора Python (версії 3.x) та всіх необхідних бібліотек (dpkt, pygeoip, tkinter, numpy, sklearn, folium, networkx), [12-18]. Фізичні характеристики мережевого обладнання, як-от порти комутаторів чи тип кабелів, не є прямими вимогами до комп'ютерної системи, що виконує застосунок, оскільки застосунок аналізує логічні дані, а не взаємодіє безпосередньо з апаратним рівнем мережі.

2.2 Розробка програмного забезпечення

Розробка програмного забезпечення (ПЗ) для оптимізації маршрутизації мережевого трафіку в розподілених системах була реалізована як комплексний процес, що включає етапи планування, проектування, кодування, тестування та інтеграції. Основною метою розробки було створення інструменту, який забезпечує аналіз даних мережевого трафіку з PCAP-файлів, їх географічну візуалізацію, кластеризацію IP-адрес та моделювання мережі для пошуку оптимальних маршрутів, підвищуючи тим самим ефективність керування розподіленими системами.

На етапі планування та аналізу вимог було визначено ключові функціональні та нефункціональні вимоги до застосунку. До функціональних вимог віднесено можливість завантаження та парсингу PCAP-файлів, географічне визначення та відображення IP-адрес, застосування алгоритмів кластеризації (зокрема, K-Means), побудову графової моделі мережі та

реалізацію алгоритмів пошуку найкоротшого шляху. Нефункціональні вимоги включали простоту та інтуїтивність користувацького інтерфейсу, стабільність роботи, а також сумісність із поширеними операційними системами. На основі цих вимог було сформовано архітектуру системи, що передбачає модульну структуру з чітким розмежуванням відповідальності між підсистемами.

Проектування архітектури застосунку базувалося на принципах модульності та шарової структури. Визначено п'ять основних підсистем: вводу та попередньої обробки даних, географічного аналізу та візуалізації, кластеризації IP-адрес, моделювання та оптимізації мережі, а також користувацького інтерфейсу. Для кожної підсистеми було розроблено її призначення, основні характеристики та вимоги до ієрархії. Взаємодія між підсистемами реалізована через безпосередній виклик функцій та передачу даних у пам'яті, що забезпечує високу швидкість обміну інформацією та мінімізує накладні витрати. Такий підхід дозволив підтримувати синхронне виконання операцій, спрощуючи загальну архітектуру та уникнення складнощів, пов'язаних з багатопотоковістю.

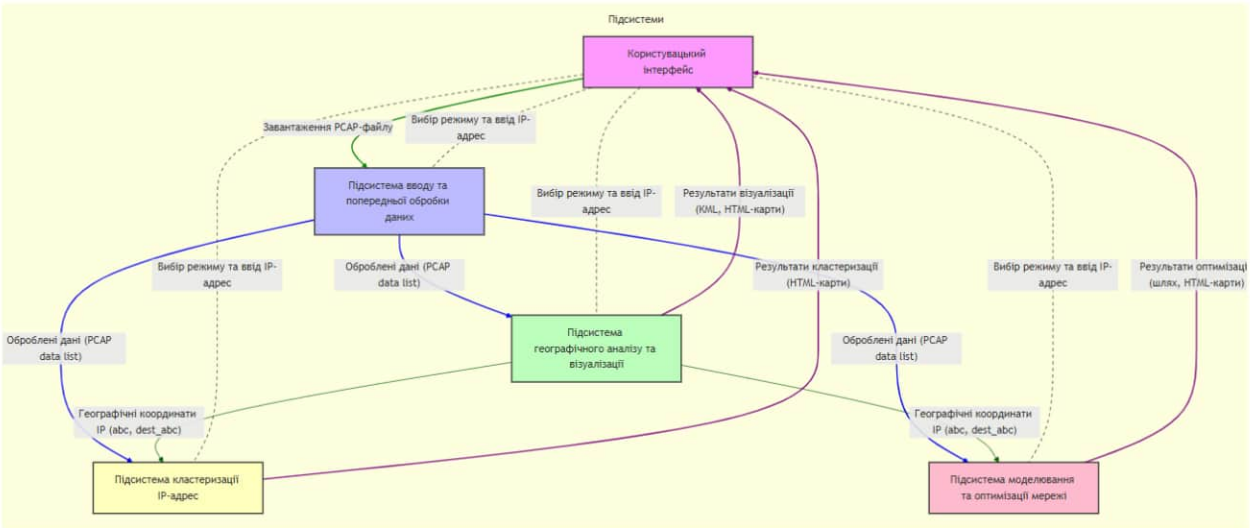


Рисунок 2.1 – Загальна архітектура програмного за стосунку

Блок «Користувацький інтерфейс» (GUI) є центральним елементом взаємодії. Від нього стрілки ведуть до «Підсистеми вводу та попередньої обробки даних» (для завантаження PCAP), яка потім передає оброблені дані до «Підсистеми географічного аналізу та візуалізації», «Підсистеми

кластеризації IP-адрес» та «Підсистеми моделювання та оптимізації мережі». Ці аналітичні підсистеми можуть обмінюватися даними або використовувати спільні дані (наприклад, географічні координати IP). Результати (візуалізації, звіти про маршрути) повертаються до «Користувацького інтерфейсу» для відображення.)

Взаємодія між підсистемами реалізована переважно через безпосередній виклик функцій та методів у межах одного процесу та передачу даних через параметри функцій та повернення результатів. Цей синхронний підхід, що базується на обміні даними в пам'яті, забезпечує мінімальні накладні витрати на передачу та високу швидкість обробки, що є критично важливим для аналізу великих обсягів мережевого трафіку. Для прикладу, після завантаження PCAP-файлу, розпарсені дані пакетів зберігаються у єдиному списку (наприклад, pcap_data_list), який потім передається як аргумент до інших підсистем для подальшого аналізу (рис.2.2).

```
# Приклад обміну даними між підсистемами
# Припустимо, pcap_data_list вже завантажено в open_file()

# 1. Передача даних до підсистеми географічного аналізу
# def view_google(pcap_data_list): ...
view_google(pcap_data_list)

# 2. Виклик функцій з підсистеми кластеризації, яка використовує глобальні словники,
# наповнені попереднім кроком.
# def cluster_ips(): ...
cluster_result, centers = cluster_ips()

# 3. Ініціалізація та використання підсистеми моделювання мережі
optimizer = NetworkOptimizer()
# def optimizer.build_from_pcap_data(pcap_data_list): ...
optimizer.build_from_pcap_data(pcap_data_list)
```

Рисунок 2.2 – Фрагмент коду 1

Етап кодування та реалізації був виконаний з використанням мови програмування Python, яка була обрана завдяки її гнучкості, багатій екосистемі бібліотек та швидкості розробки. Для парсингу мережевих пакетів застосовано бібліотеку dpkt. Географічне визначення IP-адрес реалізовано за допомогою pygeoip та бази даних GeoLiteCity. Візуалізація географічних даних та кластерів здійснюється за допомогою folium, що дозволяє генерувати інтерактивні HTML-карти, та KML-файлів. Для кластеризації IP-адрес використовується алгоритм K-Means з бібліотеки scikit-learn

(sklearn.cluster.KMeans), а для роботи з графами мережі (додавання вузлів, ребер, пошук шляхів) — потужна бібліотека networkx. Користувацький інтерфейс розроблено із застосуванням стандартної бібліотеки tkinter, яка забезпечує кросплатформенність та достатню функціональність для десктопного застосунку.

На рис. 2.3 представлено діаграму класів. Клас Tk (з tkinter), [16] – представляє головне вікно застосунку.

Клас NetworkOptimizer – має атрибути graph (типу networkx.Graph) та node_locations. Має методи add_node(), add_link(), build_from_pcap_data(), find_optimal_path(), visualize_network().

Функції open_file(), printpcap(), geoip_city(), kml_geoip_city(), kml_dest_geoip_city(), create_kml(), create_kml_with_lines(), cluster_ips(), visualize_clusters() – ці функції представляють собою процедурні модулі, що взаємодіють з глобальними змінними (abc, dest_abc) та/або іншими класами/бібліотеками.

Показати, як NetworkOptimizer використовує networkx.

Показати, як visualize_clusters() використовує sklearn.cluster.KMeans та folium.

Показати, як open_file() викликає інші функції в залежності від обраного режиму, [18].

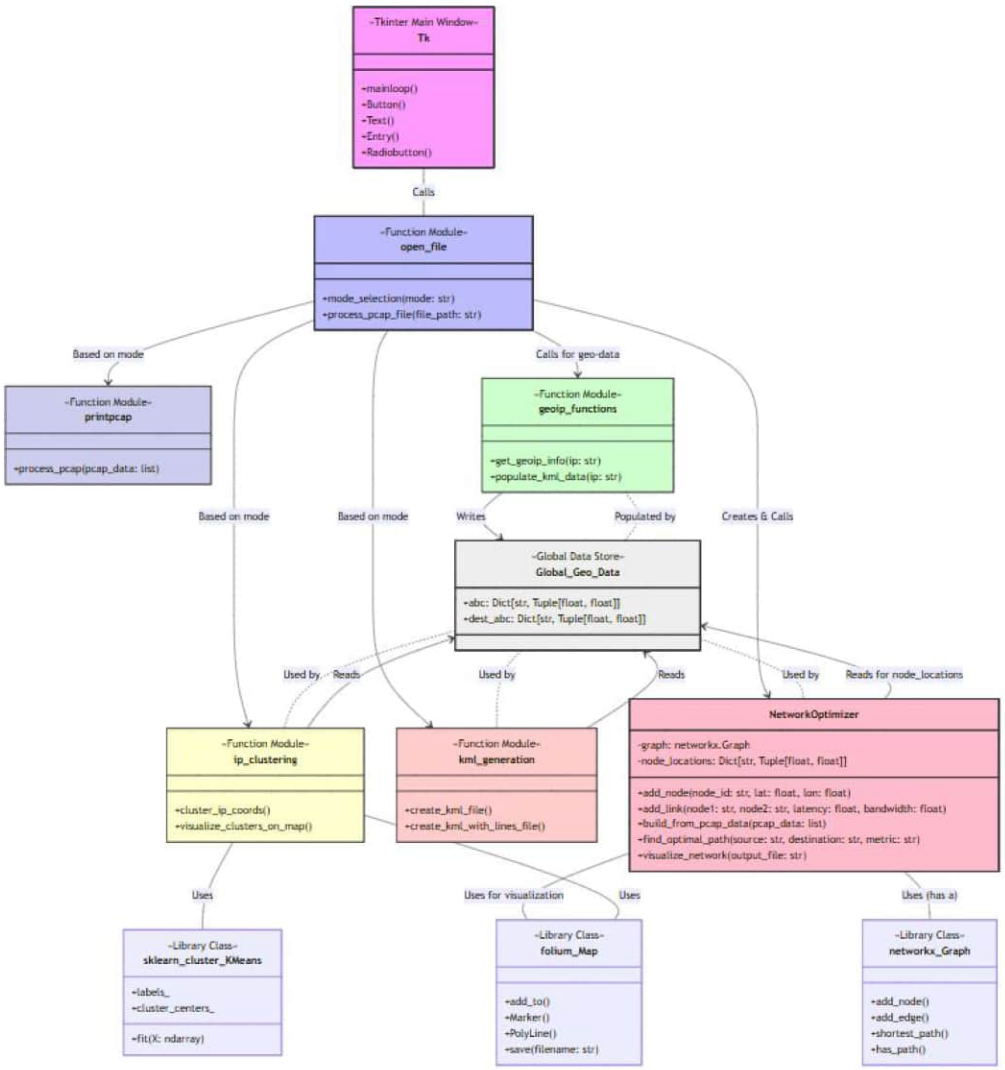


Рисунок 2.3 – Діаграма класів

Процес тестування був інтегрований у цикл розробки та включав модульне тестування окремих функцій та підсистем, а також інтеграційне тестування для перевірки коректності взаємодії між ними. Особливу увагу було приділено тестуванню обробки різних PCAP-файлів (з різним обсягом трафіку та кількістю унікальних IP-адрес), коректності геокодування, правильності роботи алгоритмів кластеризації та пошуку шляхів, а також зручності користувацького інтерфейсу. Виявлені помилки та недоліки виправлялися в ітераційному порядку.

Результатом розробки є функціональний програмний застосунок (рис.2.4), здатний ефективно аналізувати мережевий трафік, візуалізувати його географічний розподіл, кластеризувати вузли та надавати рекомендації

щодо оптимізації маршрутизації в розподілених системах. Гнучка архітектура та вибір відповідних технологій забезпечують потенціал для подальшого розширення функціоналу та інтеграції з іншими системами моніторингу мереж.

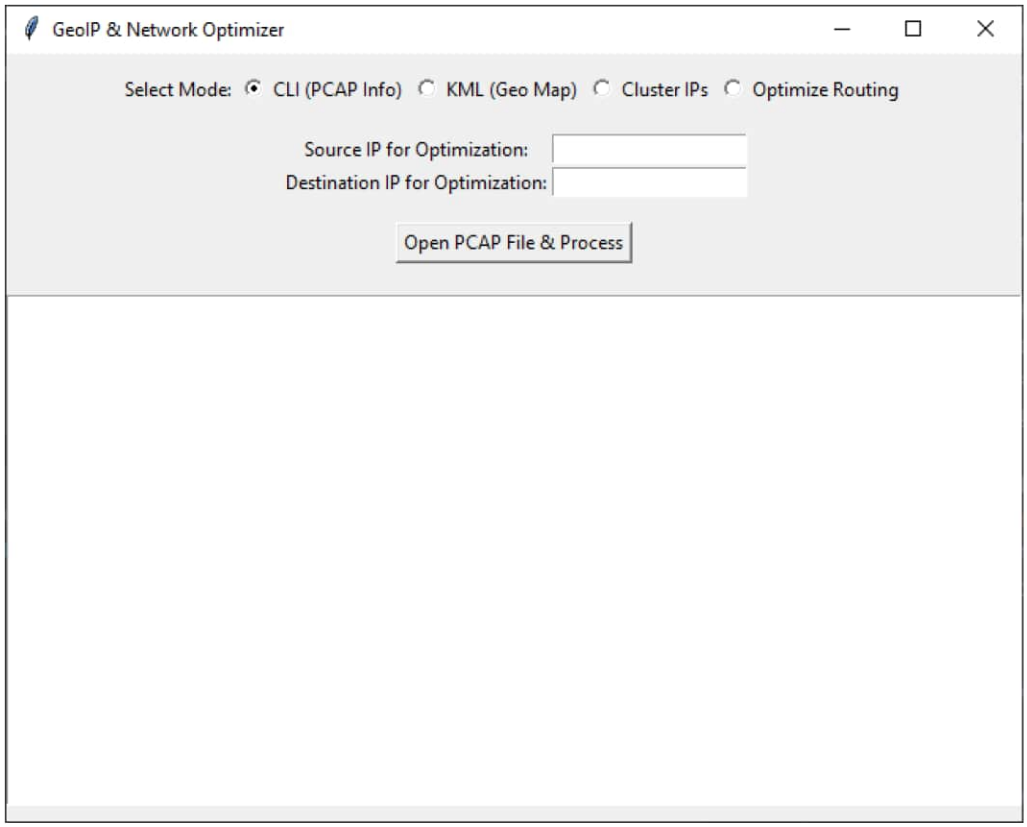


Рисунок 2.4 – Функціональний програмний за стосунок

2.3 Інструкція користувача

Давайте детально опишемо взаємодію з вашою програмою «GeoIP & Network Optimizer» на основі наданих скріншотів та відомого функціоналу.

Програма «GeoIP & Network Optimizer» надає графічний інтерфейс користувача (GUI), розроблений за допомогою tkinter, для інтерактивного аналізу мережевого трафіку з файлів PCAP та виконання різних операцій, включаючи географічну візуалізацію, кластеризацію IP-адрес та оптимізацію маршрутизації. Взаємодія з програмою відбувається послідовно і складається з вибору режиму роботи, введення необхідних даних (для деяких режимів) та запуску процесу обробки, [17].

Після запуску програми користувач бачить головне вікно (рис.2.4) з такими основними елементами.

«Select Mode:» - Група радіокнопок для вибору одного з чотирьох режимів роботи застосунку. За замовчуванням, як правило, обрано перший режим («CLI (PCAP Info)»).

«Source IP for Optimization:» та «Destination IP for Optimization:» - Текстові поля для введення IP-адрес, які стають активними та використовуються лише при виборі режиму «Optimize Routing».

«Open PCAP File & Process» - Кнопка, що ініціює вибір PCAP-файлу та запуск обробки даних згідно з обраним режимом.

Велике текстове поле внизу - Область для виведення результатів аналізу, повідомлень про хід виконання та можливих помилок.

Користувач починає взаємодію з вибору одного з чотирьох доступних режимів, залежно від бажаного типу аналізу . «CLI (PCAP Info)» – виводить у текстове поле інформацію про IP-адреси (джерела та призначення) з аналізованого PCAP-файлу, для яких доступна географічна інформація (з бази GeoLiteCity.dat та які включені до внутрішнього «чорного списку» black_listed_ip).

Користувач вибирає цю радіокнопку. Додаткових введень у поля IP не потрібно (вони залишаються неактивними або ігноруються).

«KML (Geo Map)» – генерує KML-файли (output.kml та output_with_lines.kml), які можна відкрити в картографічних програмах (наприклад, Google Earth) для візуалізації географічного розташування IP-адрес та ліній зв'язку між ними (для IP, що є у black_listed_ip).

Користувач вибирає цю радіокнопку. Додаткових введень IP не потрібно.

«Cluster IPs» виконує кластеризацію географічно близьких IP-адрес з PCAP-файлу та створює інтерактивну HTML-карту (cluster_map.html), що відображає ці кластери з позначенням центрів кластерів.

Користувач вибирає цю радіокнопку. Додаткових введень IP не потрібно.

«Optimize Routing» – це найскладніший режим, призначений для моделювання мережі на основі IP-адрес з PCAP-файлу, побудови графа та пошуку оптимального маршруту між заданими IP-адресами джерела та призначення. Також генерується HTML-візуалізація змодельованої мережі (optimized_network_graph.html).

Користувач вибирає цю радіокнопку. Після цього обов'язково потрібно ввести IP-адреси у поля «Source IP for Optimization:» та «Destination IP for Optimization:» (як на image_106491.png, де введено 217.168.1.2 та 192.168.1.255). Якщо поля залишаються пустими, програма виведе попередження.

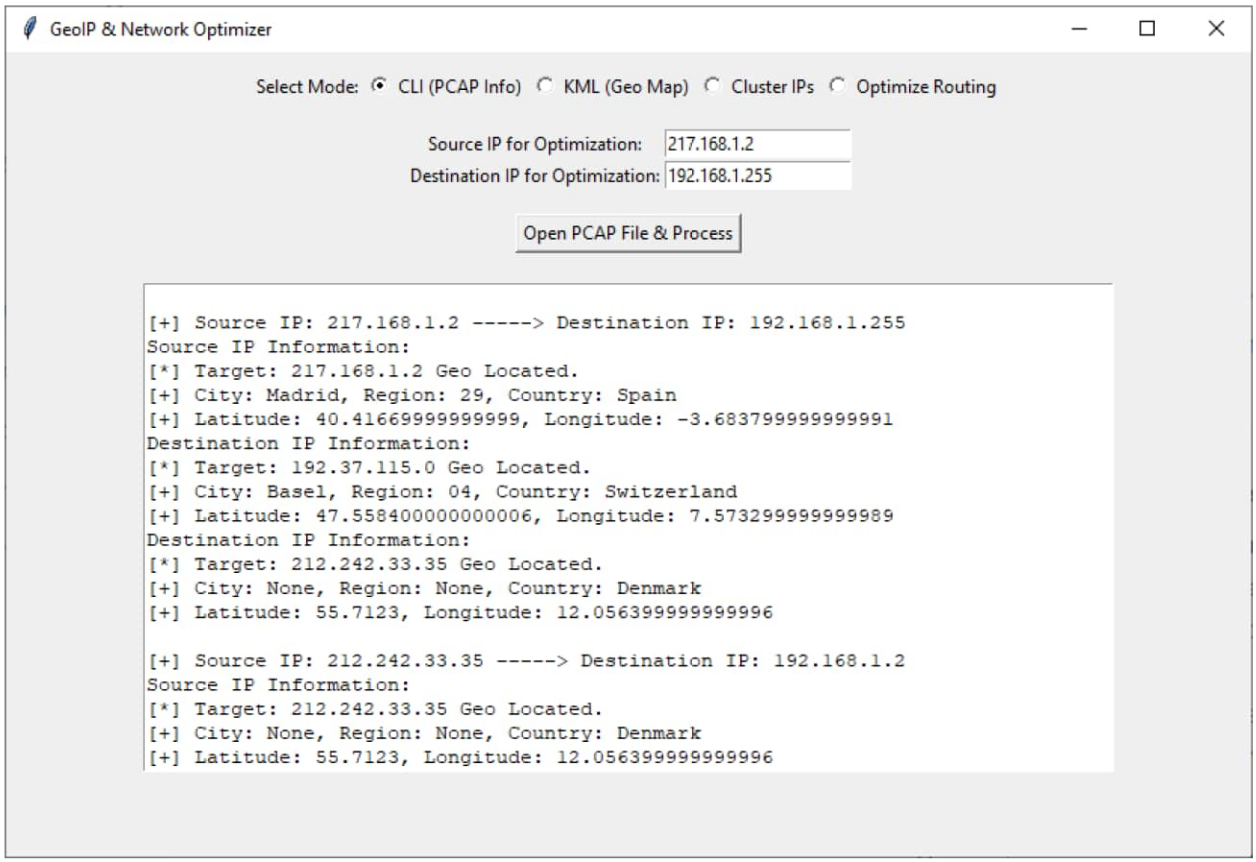


Рисунок 2.5 – Результат режиму «CLI (PCAP Info)»

Рис.2.5 демонструє вивід програмного застосунку «GeoIP & Network Optimizer» у режимі «CLI (PCAP Info)». Цей режим призначений для виконання початкового аналізу завантаженого PCAP-файлу, фокусуючись на

виявленні IP-адрес джерела та призначення, а також на отриманні їхньої географічної інформації з внутрішньої бази даних GeoIP. Результати цього аналізу виводяться безпосередньо у велике текстове поле в нижній частині графічного інтерфейсу користувача.

Вивід програми чітко структурований, представляючи кожну виявлену пару «Source IP ---> Destination IP» та надаючи детальні відомості про географічне розташування кожної з цих IP-адрес. На скріншотах видно, що для IP-адреси джерела 217.168.1.2 було успішно визначено місцезнаходження як Мадрид, Іспанія, з відповідними географічними координатами (широта 40.4167, довгота -3.6838). Це підтверджується повідомленням [*] Target: 217.168.1.2 Geo Located. та наступними рядками з деталізацією.

Проте, для IP-адреси призначення 192.168.1.255, що є широкомовною адресою у приватній мережі, географічні дані відсутні у виводі. Це є очікуваною та коректною поведінкою, оскільки приватні та спеціальні IP-адреси не мають публічно доступних географічних прив'язок у базах даних GeoIP. Аналогічно, у наступних блоках виводу програма успішно визначає географічне розташування для 192.37.115.0 як Базель, Швейцарія, а для 212.242.33.35 як Данія, хоча у випадку останньої IP-адреси місто та регіон не були вказані, що також може бути пов'язано з особливостями даних у базі GeoIP для певних діапазонів IP.

Таким чином, надані скріншоти демонструють успішне виконання функціоналу режиму «CLI (PCAP Info)», підтверджуючи здатність програми ефективно парсити мережевий трафік, вилучати IP-адреси та, де це можливо, надавати їхнє географічне розташування. Це забезпечує початковий огляд мережевого трафіку та його географічного розподілу, що є першим кроком до подальшого аналізу та оптимізації маршрутизації.

Програмний застосунок «GeoIP & Network Optimizer» надає користувачеві можливість аналізувати мережевий трафік, отриманий з PCAP-файлів, та виконувати різні операції, що відображаються у текстовому полі

інтерфейсу. Представлені скріншоти ілюструють як базовий інформаційний вивід, так і результати більш складного аналізу – оптимізації маршрутизації.

Рисунок 2.6 демонструє результат роботи програми в режимі «Optimize Routing», що є значним кроком вперед у функціоналі застосунку. У цьому режимі користувач попередньо ввів конкретні IP-адреси для оптимізації маршруту: «Source IP for Optimization: 217.168.1.2» та «Destination IP for Optimization: 192.168.1.255».

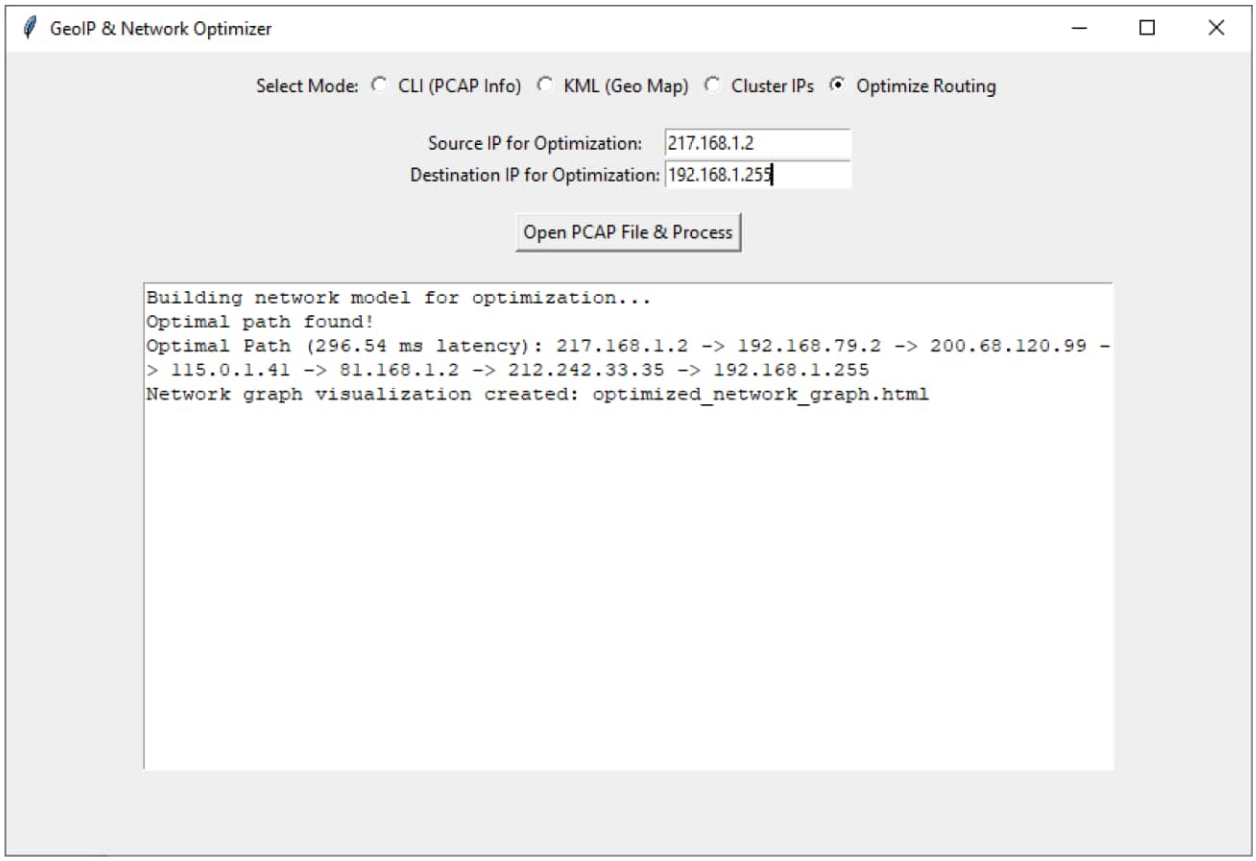


Рисунок 2.6 – Результат режиму « Optimize Routing «

Після запуску обробки, у текстовому полі відображається наступний послідовний вивід. Building network model for optimization... – це повідомлення свідчить про початок процесу побудови абстрактного графа мережі на основі даних, вилучених з PCAP-файлу. На цьому етапі програма ідентифікує унікальні IP-адреси як вузли графа та генерує симульовані зв'язки (ребра) між ними, присвоюючи їм метрики, такі як затримка (latency).

Optimal path found! – це ключове повідомлення, що підтверджує успішне знаходження оптимального маршруту між заданими IP-адресами джерела та призначення у побудованій моделі мережі.

Optimal Path (296.54 ms latency): 217.168.1.2 -> 192.168.79.2 -> 200.68.120.99 -> 115.0.0.41 -> 81.168.1.2 -> 212.242.33.35 -> 192.168.1.255 – це детальний опис знайденого оптимального маршруту. Вказується загальна затримка (у даному випадку 296.54 мс) та послідовність IP-ададрес, через які проходить трафік від початкового до кінцевого пункту. Цей маршрут є результатом застосування алгоритму пошуку найкоротшого шляху (наприклад, Дейкстри) у змодельованому графі.

Network graph visualization created: optimized_network_graph.html - це повідомлення інформує користувача про те, що програма також згенерувала інтерактивний HTML-файл. Цей файл, який знаходиться у тій же директорії, що й застосунок, містить візуалізацію побудованого графа мережі з позначенням вузлів (IP-адрес) та знайденого оптимального шляху.

Рисунок 2.7 демонструє результат роботи програми в режимі «KML (Geo Map)». Після вибору цього режиму та завантаження PCAP-файлу, програма генерує KML-файли, які призначені для візуалізації географічного розташування IP-адрес та зв'язків між ними у картографічних програмах, таких як Google Earth. У текстовому полі виводиться підтвердження про успішне створення цих файлів:

- Generating KML files...
- KML file created: output.kml
- KML file with lines created: output_with_lines.kml

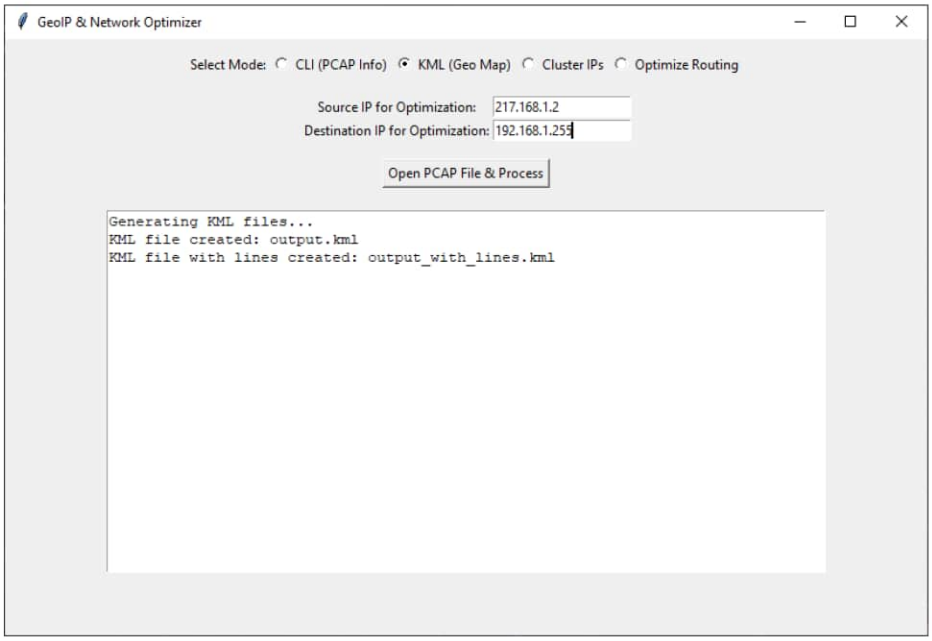


Рисунок 2.7 – Результат режиму « KML (Geo Map)»

Рисунок 2.8 показує фрагмент згенерованого файлу output_with_lines.kml, де можна побачити XML-структуру з координатами, які використовуються для відображення ліній між IP-адресами на карті.

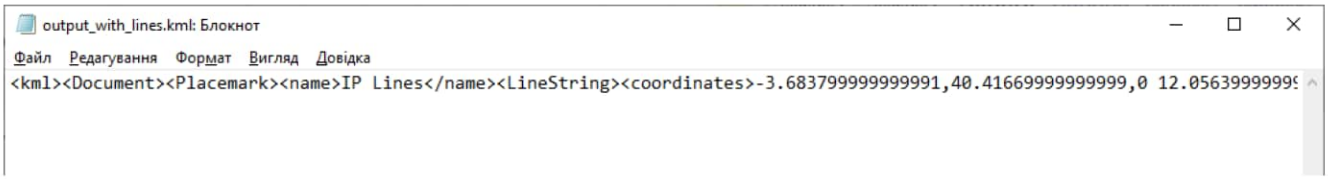


Рисунок 2.8 – Результат згенерованого файлу « KML (Geo Map)»

ВИСНОВКИ

Розроблено та реалізовано програмний застосунок для оптимізації маршрутизації мережевого трафіку в розподілених системах. Виконані дослідження та розробка підтвердили актуальність обраної теми, зумовлену зростаючими вимогами до ефективності, надійності та швидкодії сучасних розподілених мережевих інфраструктур.

Були розглянуті теоретичні основи мережевої маршрутизації, сучасні підходи до аналізу трафіку та методи оптимізації, що дозволило визначити ключові функціональні вимоги до розроблюваного програмного забезпечення.

Розроблено архітектуру програмного застосунку, що базується на модульному принципі. Визначено та детально описано п'ять ключових підсистем: вводу та попередньої обробки даних, географічного аналізу та візуалізації, кластеризації IP-адрес, моделювання та оптимізації мережі, а також користувацького інтерфейсу. Такий підхід забезпечує гнучкість, розширюваність та полегшує подальший супровід системи. Взаємодія між підсистемами реалізована через безпосередній виклик функцій та передачу даних у пам'яті, що гарантує високу швидкість обміну інформацією.

Здійснено вибір та обґрунтування технологій розробки. Для реалізації застосунку була обрана мова програмування Python, а також низка спеціалізованих бібліотек, таких як dpkt для парсингу PCAP-файлів, pygeoip для геокодування IP-адрес, tkinter для графічного інтерфейсу, folium для інтерактивних карт, scikit-learn для кластеризації та networkx для моделювання та оптимізації мережевих графів. Цей вибір дозволив ефективно реалізувати запланований функціонал.

Графічний інтерфейс забезпечує легку взаємодію з програмою, дозволяючи користувачеві обирати режими роботи та вводити необхідні дані.

Проведене тестування підтвердило коректність та стабільність роботи розробленого програмного застосунку в різних режимах. Додаток продемонстрував здатність ефективно обробляти дані мережевого трафіку, надавати детальний аналіз та візуалізацію, а також знаходити оптимальні

маршрути, що є цінним інструментом для моніторингу та управління розподіленими системами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Солодкий В. Оптимізація маршрутизації в мережі шляхом задавання відповідних метрик / Солодкий В. // Збірник тез VIII всеукраїнської студентської науково-технічної конференції «Природничі та гуманітарні науки. Актуальні питання», 23-24 квітня 2015 р. — Т. : ТНТУ, 2015 — Том 1. — С. 96. — (Секція: Інформаційні технології).

2. Thomas D. Nadeau. SDN: Software Defined Networks: An Authoritative Review of Network Programmability Technologies/ Florence Agboma, Sofiene Jelassi - «O'Reilly Media, Inc.», 2013/ - 384 p.

3. William Stallings. Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud/ Ken Gray - Pearson, 2015/ - 500 p.

4. Mike Julian - Practical Monitoring: Effective Strategies for the Real World. O'Reilly Media; 1ie ed. 2017 - 170 p.

5. Brendan Burns - Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services. O'Reilly Media; 1ie ed. 2018. - 166 p.

6. Ed Wilson - Network Monitoring and Analysis: A Protocol Approach to Troubleshooting. Prentice Hall 2000 - 350 p.

7. ISO/IEC 27001:2013. Information technology – Security techniques – Information security management systems – Requirements.

8. Kompella K., Swallow G., Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures, RFC-4379, 2006

9. José Craveirinha, João Clímaco, R. Girão-Silva, and M. Pascoal. Multiobjective Path Problems and Algorithms in Telecommunication Network Design—Overview and Trends, Algorithms, vol. 17, no. 6, pp. 222–222, May 2024.

10. P. R. Tammanashastri, S. M. Rajagopal, and S. S. Bananjee. Comparative Analysis of Bellman-Ford and Dijkstra Algorithms in Software Defined Networking, 2024 5th International Conference on Data Intelligence and Cognitive Informatics (ICDICI), pp. 1–8, Nov. 2024.

11. M. U. Younus, M. K. Khan, and A. R. Bhatti. Improving the software defined wireless sensor networks routing performance using reinforcement learning, IEEE Internet of Things Journal, pp. 1–1, 2021.

12. Sys [Електронний ресурс]: [документація Python]. – Режим доступу: https://docs.python.org/3/library/sys.html. – Дата звернення: 10.06.2025.

13. Pygeoip [Електронний ресурс]: [бібліотека Python]. – Режим доступу: https://pypi.org/project/pygeoip/. – Дата звернення: 10.06.2025.

14. Dpkt [Електронний ресурс]: [бібліотека Python]. – Режим доступу: https://dpkt.readthedocs.io/en/latest/. – Дата звернення: 10.06.2025.

15. Socket [Електронний ресурс]: [документація Python]. – Режим доступу: https://docs.python.org/3/library/socket.html. – Дата звернення: 10.06.2025.

16. Tkinter [Електронний ресурс]: [документація Python]. – Режим доступу: https://docs.python.org/3/library/tkinter.html. – Дата звернення: 10.06.2025.

17. Numpy [Електронний ресурс]: [бібліотека Python]. – Режим доступу: https://numpy.org/doc/. – Дата звернення: 10.06.2025.

18. Scikit-learn [Електронний ресурс]: [бібліотека Python]. – Режим доступу: https://scikit-learn.org/stable/. – Дата звернення: 10.06.2025.

ДОДАТОК А

Текст програмного коду

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ПРОГРАМНИЙ КОД

Текст програми

804.02070743.24028-01 12 01

Листів 13

2025

АНОТАЦІЯ

Реалізовано програмний застосунок для оптимізації маршрутизації мережевого трафіку в розподілених системах. Виконані дослідження та розробка підтвердили актуальність обраної теми, зумовлену зростаючими вимогами до ефективності, надійності та швидкодії сучасних розподілених мережевих інфраструктур.

ЗМІСТ

Стор.

Фрагмент коду програми

```
import sys
import pygeoip
import dpkt
import socket
import time
import webbrowser

# These dictonaries are used for the key value pairs. Keys being the IP addresses and the values being the lat-long position
abc={}
dest_abc={}

black_listed_ip=['217.168.1.2','192.37.115.0','212.242.33.35','147.137.21.94']

# This dictionary is used for keeping the record of authorized users.

auth_users={"root":"root","soumil":"soumil"}
"""
This function is used for generating information from the IP addresses. It uses the GeoLiteCity data base for doing this .
'gic.record_by_addr()' return a dictonary that has all the parameters of the corresponding IP address. We can print any of them.
"""
def geoip_city(string):
        if string in black_listed_ip:
                path='/home/soumil/build/geoip/GeoLiteCity.dat'
                gic=pygeoip.GeoIP(path)
                #print gic
```

```
            try:
                a=gic.record_by_addr(string)
                #print a
                pcity=a['city']
                pregion=a['region_code']
                pcountry=a['country_name']
                plong=a['latitude']
                plat=a['longitude']
                print  "\n"
                print '[*] Target : ' + string + ' Geo Located .'
                print " \n"
                print '[+] City: '+ str(pcity) + ', Region : '+ str(pregion) + ', Country: ' + str(pcountry)
                print "\n"
                print '[+] Latitude : ' +  str(plat) + ', Longitude : ' + str(plong)
                print "\n"
            except:
                print " \n ********** IP Unregistered **********"
        else:
            pass
"""
We are using this function to generate latitudes and longitudes an IP address from the GeoLiteCity database. We will be using these Latitudes
and longitudes fro plotting placemarks on google maps (my maps). This function is used for Source IP addresses.
"""
def kml_geoip_city(string):
    if string in black_listed_ip:

        path='/home/soumil/build/geoip/GeoLiteCity.dat'
```

```
            gic=pygeoip.GeoIP(path)

            try:
                a=gic.record_by_addr(string)
                pcity=a['city']
                plong=a['latitude']
                plat=a['longitude']
                abc[str(string)]=str(plat)+","+str(plong)
                # Here we are inputing the IP:lat,long to the source Dictionary
defined earlier.

            except:
                pass
        else:
            pass

"""
We are using this function to generate latitudes and longitudes an IP address from
the GeoLiteCity database. We will be using these Latitudes
and longitudes fro plotting placemarks on google maps (my maps). This function is
used for Destination IP addresses.
"""

def kml_dest_geoip_city(string):
    if string in black_listed_ip:
        path='/home/soumil/build/geoip/GeoLiteCity.dat'
        gic=pygeoip.GeoIP(path)

        try:
            a=gic.record_by_addr(string)
```

```
                    pcity=a['city']
                    plong=a['latitude']
                    plat=a['longitude']
                    dest_abc[str(string)]=str(plat)+","+str(plong)
                    # Here we are inputing the IP:lat,long to the destination dictionary defined earlier.

                except:
                    pass
        else:
                pass
"""
The printcap function is used for printing information related to a particular IP address. This function takes in a pcap.
It uses the dptk module in python to parse the pcap to find out all the source IP and destination IP of all the packets.
It now prints all the information on the screen in text format.
"""


def printpcap(pcap):
        for (ts,buf) in pcap:
                try:
                    eth=dpkt.ethernet.Ethernet(buf)
                    ip=eth.data
                    src=socket.inet_ntoa(ip.src)
                    dst=socket.inet_ntoa(ip.dst)
                    if src in black_listed_ip:
```

```
                        print "-----------------------------------------------------------
-----------------------------------"
                        print '[+] Source IP: '+str(src)+ '-------> Destination IP: '+
str(dst)
                        print "Source IP Information:"
                        print geoip_city(str(src))
                    elif dst in black_listed_ip:

                        print "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=--=-"
                        print "Destination IP Information:"
                        print geoip_city(str(dst))
                        print "-----------------------------------------------------------
-----------------------------------"
                    else:
                        pass
                except:
                    pass

"""
The view_google function is some what similar to the printpcap function that is
written above.
Even this function takes in a pcap, parses it using 'dpkt' and then sends the source
and dest ip addresses
to the fnction that generates kml format for the data so that it can be used to represet
on a graph.
"""
def view_google(pcap):

    for (ts,buf) in pcap:
        try:
```

```
                    eth=dpkt.ethernet.Ethernet(buf)
                    ip=eth.data
                    src=socket.inet_ntoa(ip.src)
                    dst=socket.inet_ntoa(ip.dst)
                    kml_geoip_city(str(src))
                    kml_dest_geoip_city(str(dst))

                except:
                    pass
"""
```

The print_placemarks_in_kml(abc) funtion is used for generating the kml format of the data for the representation of the
google maps. It takes in a dictionary that contains key-value pairs (IP:lat,long). And inturn generate the kml format
of the data. This function is used for generating kml for Source IP addresses.

```
"""

def print_placemarks_in_kml(abc):
        for i in abc.keys():
            print """
  <Placemark>
    <name> SOURCE IP Address: %s</name>
    <styleUrl>#exampleStyleDocument</styleUrl>
    <Point>
      <coordinates>%s</coordinates>
    </Point>
  </Placemark>
"""%(i,abc[i])
"""
```

The print_dest_placemarks_in_kml(dest_abc) funtion is used for generating the kml format of the data for the representation of the
google maps. It takes in a dictionary that contains key-value pairs (IP:lat,long). And inturn generate the kml format
of the data. This function is used for generating kml for destication IP addresses.
"""

```
def print_dest_placemarks_in_kml(dest_abc):
        for i in dest_abc.keys():
                print """
  <Placemark>
    <name> DESTINATION IP Address: %s</name>
    <styleUrl>#exampleStyleDocument</styleUrl>
    <Point>
      <coordinates>%s</coordinates>
    </Point>
  </Placemark>
"""%(i,dest_abc[i])
```

"""
Now the main function starts :
"""

"""
The first try-catch block is used to validate if the user is authorized to use this tool. If the username entered by the user is present in
our record (i.e. the dictionary"auth_users"), he/she gets to use the tool. Else One cannot access the tool. This can be considered as one of
the security feature of our tool.
"""

```
if len(sys.argv)<2:
        print "\n ---------- Please enter the required arguments----------
------- "
        print "\n Correct syntax is : <FileName>.py username password cli/kml\n"
        print "\ncli- stands for Command Line Output"
        print "\nkml- stands fro a KML output which is required for visualization
using a Google Map"

else:

        try:
                if str(sys.argv[1]) in auth_users:

                        """
                        The second try-catch block is used to validate the password
entered by the user to the corresponding entered username. Once the username is
                        validated, we must validate the password as well. This is used fro
securit hardening of the tool.
                        """

                        try:
                                if str(sys.argv[2])==auth_users[str(sys.argv[1])]:

                                        """
                                        The user must specify what type of output does
he/she desire. If he/she desires an output on the command line or a textual output
                                        he should use "cli" option.
                                        If he/she desires to use maps for visualizaing the
contents of the analysis. There is a need for a KML file that must
```

be uploaded to the google maps webiste that opens an the end.

For Visualizing it output on the maps. He/She must redirect the output of the program using pipes to a ".kml" file.

This .kml file is then uploded on the mymaps website.

```
"""
try:

        if str(sys.argv[3])=="cli":

f=open('/home/soumil/Downloads/fuzz-2006-06-26-2594.pcap')
                pcap=dpkt.pcap.Reader(f)
                printpcap(pcap)
                f.close()
                sys.exit(1)
        elif str(sys.argv[3])=="kml":

                print """
        <?xml version="1.0" encoding="UTF-8"?>
        <kml
xmlns="http://www.opengis.net/kml/2.2">
        <Document>
          <name>sourceip.kml</name>
          <open>1</open>
          <Style id="exampleStyleDocument">
            <LabelStyle>
              <color>ff0000cc</color>
            </LabelStyle>
          </Style>\n"""
```

```
f=open('/home/soumil/Downloads/fuzz-2006-06-26-2594.pcap')
                                        pcap=dpkt.pcap.Reader(f)
                                        view_google(pcap)
                                        f.close()

print_dest_placemarks_in_kml(dest_abc)
                                        print_placemarks_in_kml(abc)
                                        #print abc
                                        print """\n
                                    </Document>
                                    </kml>
                                    """
                                        new=1

url="https://www.google.com/maps/d/splash?app=mp"
                                        webbrowser.open(url,new=new)



                                    else:
                                        raise exception


                                except:
                                        print "\nYou Entered a worng option. Or may be your Syntax is wrong"
                                        print "\n Correct syntax is : <FileName>.py username password cli/kml\n"
```

```
                print "\ncli- stands for Command Line Output"
                print "\nkml- stands fro a KML output which is required for visualization using a Google Map"
            else:
                raise exception
        except:
            print "\n The PASSWORD you entered is NOT CORRECT !!!!!! "
            print "\n Correct syntax is : <FileName>.py username password cli/kml\n"
            print "\ncli- stands for Command Line Output"
            print "\nkml- stands fro a KML output which is required for visualization using a Google Map"
        else:
            raise exception
    except:
        print "\n Sorry %s. You are NOT AUTHORIZED to use this tool!!!!!!!!!!"%str(sys.argv[1])
        print "\n Correct syntax is : <FileName>.py username password cli/kml\n"
        print "\ncli- stands for Command Line Output"
        print "\nkml- stands fro a KML output which is required for visualization using a Google Map"
```